Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues

Ehab Nabiel Alkhanak[a], Sai Peck Lee[a], Reza Rezaei[a], Reza Meimandi Parizi[b]

[a] *Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia.*
[b] *School of Engineering and Computing Sciences, New York Institute of Technology, Nanjing campus.*
{*ehabsoa@siswa.um.edu.my, saipeck@um.edu.my, rezarezaei@siswa.um.edu.my, rparizi@nyit.edu*}

Abstract

Workflow scheduling in scientific computing systems is one of the most challenging problems that focuses on satisfying user-defined quality of service requirements while minimizing the workflow execution cost. Several cost optimization approaches have been proposed to improve the economic aspect of Scientific Workflow Scheduling (SWFS) in cloud and grid computing. To date, the literature has not yet seen a comprehensive review that focuses on approaches for supporting cost optimization in the context of SWFS in cloud and grid computing. Furthermore, providing valuable guidelines and analysis to understand the cost optimization of SWFS approaches is not well-explored in the current literature. This paper aims to analyze the problem of cost optimization in SWFS by extensively surveying existing SWFS approaches in cloud and grid computing and provide a classification of cost optimization aspects and parameters of SWFS. Moreover, it provides a classification of cost based metrics that are categorized into monetary and temporal cost parameters based on various scheduling stages. We believe that our findings would help researchers and practitioners in selecting the most appropriate cost optimization approach considering identified aspects and parameters. In addition, we highlight potential future research directions in this on-going area of research.

Keywords:
Scientific workflow, Scheduling, Cloud computing, Grid computing, Cost benefit analysis, Monetary cost, Temporal cost.

1. Introduction

Efficient resources utilization remains a key issue in parallel and distributed computing environments. To resolve this issue, an organization needs to focus on finding the most suitable allocation of an application's tasks to available computational resources. This notion is generally referred as scheduling [1, 2]. Optimal scheduling problem is known to be an NP-complete problem [3–5]. There is no proposed scheduling approach that can achieve an optimal solution within the polynomial time, especially in the case of scheduling large-size tasks [6, 7]. Users can employ different available computational resources to execute the tasks in an efficient manner. However, current limited computational resources lack in accomplishing users' demands (e.g., strict service completion deadline, and vast amount of required storage) due to the tremendous increase in complexity and size of today's applications. Consequently, users need to determine an appropriate computational environment that provides the required storage space and computational resources for processing large-scale complex applications.

Grid computing and cloud computing resources can provide an optimal solution that can meet the user's requirements by providing scalable and flexible solutions for considered applications [3]. The cloud computing based task scheduling differs from the grid computing based scheduling in the following two ways:

- Resource sharing: cloud computing offers advanced services by sharing resources using the virtualization notion with the help of internet technologies. Consequently, it supports real-time allocation to fully utilize the available resources while improving elasticity of cloud services. Thus, the scheduler in a cloud workflow system needs to consider the virtualization infrastructure (e.g., virtual services and virtual machines) to efficiently facilitate the computational processes. In contrast, grid computing allows allocating a large cluster of resources in a shared mode. Therefore, it supports batch processing and resources will be available once they are released by other users.
- Cost of resource usage: cloud computing provides a flexible costing mechanism in considering the user's requirements (i.e. pay-as-you-go and on-demands services). On the other hand, grid computing follows a quota strategy to determine the accumulated cost of requested services [8]. Therefore, grid computing has no flexible costing mechanism as in cloud computing.

In the literature, researchers have categorized task-scheduling strategies into two main categories: (i) job-based, and (ii) workflow-based [9–13]. Job-based scheduling usually focuses on scheduling a set of independent tasks to be executed in a sequence or parallel manner [14, 15]. In contrast, workflow-based scheduling (or global task scheduling) aims at mapping and managing the execution of inter-dependent (i.e. precedence constraints) tasks on shared resources for applications with higher complexity [16]. The workflow can be defined as multiple steps or activities, which are necessary to complete a submitted task. The components of these activities can be any executable instances (e.g. load sets, report sets, programs, and data) with different structures (e.g. process, pipeline, data distribution, data aggregation, and data redistribution). The workflow scheduling attained more attention of researchers compared to job scheduling, since workflow-based scheduling is able to efficiently determine an optimal solution for large and complex applications by considering precedence constraints between potential tasks. Motivated by this, we focused on reviewing workflow-based scheduling in cloud and grid computing. Workflow-based scheduling is commonly represented using a Directed Acyclic Graph (DAG) model [3, 17–20]. The DAG is usually represented by:

$$DAG = \{T, E\} \tag{1}$$

where T (vertex) is a set of tasks (a task can be any program that the user would like to execute in a workflow application) and E is a set of directed edges between the vertices.

$$T = \{t_0, ..., t_n\} \tag{2}$$

$$E = \{e_1, ..., e_m\} \tag{3}$$

Note that there is a data dependency between edges in E. For instance, if there is a directed edge e (*i.e.* $e \in E$) connecting t_i and t_j (denoted as $t_i \rightarrow t_j$), then t_i is considered as a parent and t_j as a child. The input data of task j depends on the produced data by the parent task i. Similarly, the complete path from t_0 to t_n can be represented as:

$$(t_0 \rightarrow t_1), (t_1 \rightarrow t_2), ..., (t_{n-2} \rightarrow t_{n-1}), (t_{n-1} \rightarrow t_n) \tag{4}$$

In order to execute workflow tasks in cloud and grid computing, it requires tasks mapping to the set of heterogeneous resources, which are commonly used in cloud as a set of Virtual Machines (VMs):

$$VM = \{vm_0, ..., vm_k\} \tag{5}$$

Furthermore, it is crucial to consider the computational cost (in terms of time) of executing the workflow tasks on available heterogeneous VMs along with the communication cost between these VMs.

Traditionally, the information technology staff manually executes workflow tasks, which requires knowledge about resource availability and the estimated starting time for each workflow task [1, 21, 22]. It is necessary to automate and optimize the workflow scheduling process in order to achieve an efficient Workflow Management System (WfMS). A WfMS defines, manages, and executes workflows on available computing resources, where the workflow execution order is driven by a computer representation of the workflow logic. The WfMS can be implemented for different purposes including process management, process redesign/optimization, system integration, achieving flexibility, and improving maintainability. The main stages of any WfMS are modeling stage, instantiation stage and execution stage (as depicted in Figure 1)[23]. In the modeling stage, scientific processes are redesigned based on cloud workflow specifications which should contain the task definitions, tasks structural representation (e.g. DAG), and user-defined QoS requirements. The cloud workflow service provider will negotiate with the service consumer to finalize Service Level Agreement (SLA). In the instantiation stage, the WfMS selects and reserves the suitable cloud services (from private cloud, public cloud, and hybrid cloud) based on the SLA in order to execute workflow activities as well as satisfy the defined QoS requirements. Finally, at the execution stage, the cloud workflow execution scheduler coordinates the data and control flows according to the workflow specifications obtained from the modeling stage, and employs the candidate software services reserved at the instantiation stage to execute all the workflow activities. The workflow scheduler (i.e. workflow engine) plays a crucial role in scheduling and allocating the given tasks to the available resources by considering their dependencies as modeled using a DAG.

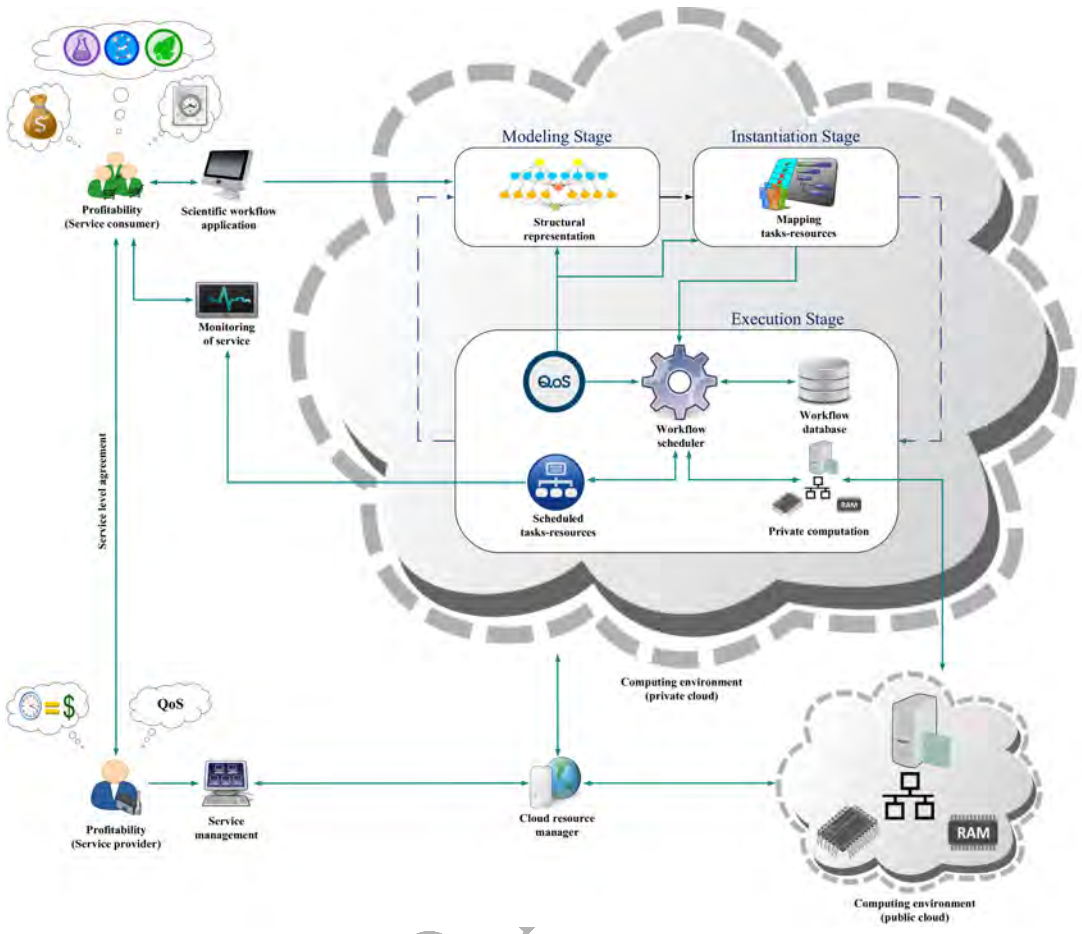
Modeling Stage
Instantiation Stage
Structural representation
Mapping tasks-resources
Execution Stage
QoS
Workflow scheduler
Workflow database
Scheduled tasks-resources
Private computation
RAM
Profitability (Service consumer)
Scientific workflow application
Monitoring of service
Service level agreement
QoS
Profitability (Service provider)
Service management
Cloud resource manager
Computing environment (private cloud)
Computing environment (public cloud)

Figure 1: Process flow of scientific workflow scheduling.

WfMS in cloud and grid computing must have the ability to handle the requests from different application domains such as business workflow applications and scientific workflow applications. The business workflow application (also referred as transaction intensive workflow) has been defined by Workflow Management Coalition (WfMC) as the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules (e.g. bank transactions and insurance claim applications) [24–28]. Conversely, Scientific Workflow Application (SWFA) (also known as data and computational intensive scientific workflow) mostly implies data flows together with the tasks execution [12, 24, 29, 30], including input scripts (scientific program or data), which can be used to produce, analyze and visualize output results. It can provide interactive tools to help scientists better execute their own workflows and view results in real time. In addition, the SWFA simplifies the process for scientists to reuse the same workflows and provide them with an easy-to-use environment to track and share the output results virtually. Thus, SWFAs have been used in different scientific applications including weather forecasting, bioinformatics, geoinformatics, chem-informatics, biomedical informatics, and astrophysics [3, 31]. To execute SWFA data, high performance resources, such as supercomputers, need to be delivered by the service provider (i.e. infrastructure as a service)[3, 31–34]. Therefore, WfMSs using cloud and grid services enable scientists to define multi-stage computational and data processing pipelines that can be executed as resources with predefined quality of service. Consequently, the scheduling process can automate complex analyses, improve application performance, and reduce the time required to obtain the desired results [11, 14, 35–37]. Inspired by this, we surveyed the studies that focused on Scientific Workflow Scheduling (SWFS) in cloud and grid computing.

One of the most challenging problems with SWFS in cloud and grid computing is to optimize the cost of workflow execution [6, 19, 38]. The cost optimization challenge of SWFS in cloud computing is a multi-objective cost-aware problem that requires consideration of three main aspects: (i) different users which usually compete for resources within the cloud or grid computing to satisfy QoS constraints, (ii) the inter-dependencies among workflow tasks, and (iii) high communication cost due to the inter-dependencies between the tasks (i.e. data needs to be transferred from one resource to another). However, considering all cost optimization problem related aspects makes the SWFS process more complicated and also requires a high amount of computational resources in terms of computational time [12, 39–41]. Inspired by this, a significant number of SWFS approaches have been proposed in the literature, focusing on reducing the overall execution cost of SWFS [3, 42–44].

The main aim of this paper is to analyze the cost optimization problem in SWFS by extensively surveying the state-of-the-art SWFS approaches in cloud and grid computing. To achieve this aim, we targeted three main objectives: (1) to classify cost optimization approaches based on the relevant aspects; (2) to classify cost parameters into monetary cost [45–48] and temporal cost [9, 20, 49, 50] parameters based on scheduling stages (i.e. pre-scheduling, during scheduling, and post-scheduling); and (3) to identify the correlation between the cost parameters and their profitability to service consumers and service providers. Therefore, classification is used as the survey method to identify and analyze the cost aspects and parameters of SWFS. To achieve the aforementioned objectives, the following research questions were formulated:

RQ 1: What are the relevant cost optimization approaches for scientific workflow scheduling problem? Answering RQ1 helps researchers to identify the relevant cost optimization approaches for SWFS problem. It also provides a clearer understanding of strengths of the underlying optimization, and limitations for all reviewed approaches (Section 2).

RQ 2: What are the main classifications affecting the cost optimization of scientific workflow scheduling? Answering RQ2 helps researchers to understand the overall classification of the cost optimization aspect. The first classification emphasizes the scheduling aspects by focusing on cost optimization approaches (Section 3). The second classification categorizes the cost optimization parameters (cost metrics) into two groups, namely monetary and temporal cost parameters. Moreover, the second classification is extended to divide the reviewed approaches into two groups: (i) profitability for service consumers, and (ii) profitability for service providers (Section 4).

RQ 3: What are the main aspects for cost optimization SWFS approaches? Answering RQ3 would help researchers to identify important cost optimization aspects of SWFS problem in cloud and grid computing (Section 3).

RQ 4: What are the main cost optimization parameters of SWFS in cloud and grid computing and how do monetary and temporal cost parameters affect the profitability of cost optimization of SWFS? Answering RQ4 could help researchers to identify the relevant cost optimization parameters based on the purpose of the model, which might be beneficial to service consumers and/or service providers. Additionally, the results of applying classifications on the reviewed cost optimization models will be obtained, providing a complete relation between monetary and temporal costs based on the scheduling stage (Section 4 and Section 5).

1.1. Motivation and related work

The SWFS challenges have gained more attention after the emergence of cloud computing area. A significant number of SWFS approaches [3, 4, 13, 51–55] have been proposed that focused on the cost optimization challenge due to the direct impact on the profitability of service consumers and service providers for business and scientific workflows. Cost optimization plays an important role in different aspects within the proposed cost optimization based SWFS approaches, such as computing environment, optimization method, structural representation, scheduling technique, and workload type.

Many review studies addressed the SWFS challenge in grid and cloud computing. Yu and Buyya [56] reviewed a number of grid computing workflow management systems. In contrast, Prodan and Wieczorek [24, 57] devised a classification of multi-criteria problems for SWFS. Their devised taxonomy classifies the multi-criteria into four different aspects based on the workflow structure (i.e. cost aggregation method, intra-dependence, optimization direction, and interdependence). On the other hand, some review studies focused on optimizing multi-objective criteria from the cost aspect while considering other constraints such as network bandwidth, storage requirements, energy efficiency, robustness, and fault-tolerance. To the best of our knowledge, no cost-specific review has been conducted that completely covers the cost optimization SWFS approaches in cloud and grid computing. Conversely, other works have emphasized the SWFS problem in cloud computing and grid computing [58–60]. Singh and Singh [61] reviewed various SWFS algorithms and compared the algorithms according to their type, objective criteria, and environment. In contrast, some reviews [62–64] only focused on workflow scheduling in cloud computing.

In order to fully cover the body of knowledge of cost optimization problem of workflow scheduling in cloud and grid computing, it is necessary to provide a complete review regarding SWFS challenges, aspects, and parameters. Our prior work focused on cost-aware SWFS challenges [65]. We have classified the challenges in cloud workflow scheduling by focusing more on scheduling objectives and functionalities of workflow system architecture as well as the QoS challenges. The current work aims at reviewing important aspects and parameters compared to challenges presented in our previous work. As a result, it provides a complete body of knowledge of cost optimization of SWFS in cloud computing. The following are significant differences in our conducted works:

- Our prior work reviewed approaches on a broader workflow scheduling domain for different types of workflow applications. However, the current work is specifically focused on reviewing approaches for scientific workflow applications.
- Both works focused on cost optimization in the area of workflow scheduling in the cloud computing. However, in our previous paper, we focused only on the cost challenges (i.e. system performance, system functionality and QoS). In contrast, the current work determines more precisely the cost optimization aspects and parameters specifically for SWFS.
- In our previous paper, we classified the profitability of cost challenges based on two viewpoints: (i) service consumers, and (ii) service providers. However, the current work classifies the profitability perspective based on the cost parameters of SWFS in cloud and grid computing.
- The previous work explicitly reported QoS constraints of cost-aware workflow scheduling. In contrast, the current work focuses on classifying the QoS constraints based on two levels of consideration: (i) activity level, and (ii) workflow level.

To the best of our knowledge, no work has focused on providing a comprehensive classification of cost optimization aspects of SWFS. Moreover, none of the researchers has focused thoroughly on comparing cost optimization SWFS approaches in cloud and grid computing environments. A comprehensive taxonomy is required to provide an in-depth understanding of cost optimization aspects, parameters, constraints, and opportunities that can be useful for future researchers.

Motivated by this, we extensively review cost aspects and parameters in SWFS, which would help researchers and scientists by providing a thorough overview of current state-of-the-art works that is beneficial in optimizing the cloud services of SWFS. The following are newly targeted objectives in the current work:

- Focusing on cost optimization approaches (not challenges as provided in our prior work) specifically in SWFS domain in cloud and grid computing.
- Comparing existing cost optimization SWFS approaches.
- Classifying SWFS aspects and cost parameters in cloud and grid computing.
- Highlighting potential future research issues in the context of SWFS aspects and cost parameters.

1.2. Research methodology

It is important to note that this paper extensively reviews cost optimization SWFS approaches in cloud and grid computing. We have followed a systematic literature review methodology to select most suitable papers in this area of research. Moreover, the adopted methodology helped us to avoid missing any important paper(s), which otherwise would probably be missed out if we used a simple survey strategy for selection of papers. First of all, we formulated an initial set of research questions based on our research experience and discussions with field experts. Then the formulated research questions were refined throughout the literature review to be presented in an unambiguous manner. The systematic literature review methodology was adopted by following the guidelines suggested by Kitchenham [66] to ensure the selection of suitable papers.

1.2.1. Sources of information

We have widely searched various digital library sources to obtain a large pool of relevant potential papers. The main goal of using the selected digital libraries was to make sure that we do not miss out any of the relevant papers (as recommended by Dieste et al. [67]) rather than just focusing on workshop proceedings, conference proceedings and journals. The following are the selected digital libraries that have been covered:

- ScienceDirect - Elsevier (https://www.sciencedirect.com)
- IEEE eXplore (https://ieeexplore.ieee.org)
- ACM Digital Library (https://www.portal.acm.org)
- Google Scholar (https://www.scholar.google.com)
- Springer LNCS (https://www.springer.com/lncs)
- Web of Science (https://www.isiknowledge.com)

1.2.2. Search criteria

The initial search criteria were devised by formulating a search query (as shown below) and picking important terms, keywords and their synonyms based on our formulated research objectives (i.e. papers should focus on cost optimization aspects and parameters for SWFS). We joined the terms using logical operators, including (AND) and (OR), to formulate our search query and searched on the title field. Note that we refined the query based on the searching facility and conditions provided by the selected digital libraries.

[(workflow scheduling) AND ((cost optimization) AND (parameters) OR (metrics) OR (aspects)) AND ((cloud computing) OR (grid computing)) AND ((approaches) OR (models) OR (algorithms)) AND ((profitability) AND (service consumer) OR (service provider) OR (Utility provider))]

Based on the suggestions of Kitchenham [66], we only considered and included the papers written in English. The earliest selected primary study was published in 2004. Therefore, we set the start year to 2003 in order to confirm that related studies within this area of research would be included, and the last date was set to 2015. The initial search resulted in collecting 1,043 potential papers.

1.2.3. Inclusion and exclusion criteria

This section presents the inclusion/exclusion criteria, which have been used to select most relevant papers. Based on the objectives of this review, we formulated the following inclusion criteria:

- The papers targeting the main focus of our review including cost-optimization problems, parameters, and aspects in SWFS were selected for initial evaluation.
- The papers written in English were considered. It is mainly due to the fact that English is regarded as a standard language in the research community. Moreover, to the best of our knowledge, majority of reputed journals accept papers only written in English language.
- The publication period starting from the year 2004 to year 2015 was considered since scheduling of cloud and grid computing area emerged in the year 2004. The objective was to provide the most up-to-date view in this field of research.
- The papers published in peer-reviewed journals, conferences, and workshop proceedings were selected since they have gone through quality evaluation process (i.e. peer-reviewed by field experts).
- If a paper is published in two different venues with more and less similar contributions, then the latest and complete version of the paper is included. For instance, a paper published in a conference whose extended version is later published in a journal. In this case, we only included the journal version of the paper.

Similarly, we formulated and adopted the following exclusion criteria in order to exclude irrelevant papers:

- The Grey literature (e.g., work in progress, workshop reports, and technical reports) were excluded due to the lack of technical details. Moreover, there could be a threat associated with Grey literature that no peer-reviewed process might have been adopted.
- The papers which do not cover cost optimization problems, parameters, and aspects in SWFS in grid and cloud computing were excluded. This is because they do not focus on the defined objectives of this review.
- The papers written in non-English languages were excluded.
- Duplicate papers found from different selected digital libraries were manually excluded to avoid reporting similar results.
- Papers published pre 2004 were excluded since they lack in covering grid and cloud computing.
- Surveys, systematic literature reviews, and mapping studies were excluded since they lack in presenting any new approach focusing on cost optimization problems in SWFS.
- Extended abstracts and short papers were excluded due to the lack of technical details.

1.2.4. Selection strategy

In order to critically investigate potential papers, we involved three researchers by adopting a three-stage paper selection strategy, as shown in Figure 2.

Stage 1: First of all, the potential papers (1043) were checked based on the title of the collected papers to remove the duplication. We observed that there was a large number of irrelevant papers due to conflict between the topics. For instance, scheduling term is related to project management or could be relevant to other computational environments (e.g. utility computing, parallel computing). Similarly, the term workflow is related to different types of workflow applications, which are out of the scope of this survey (e.g. business workflow applications). Finally, after Stage 1, we considered 317 papers.

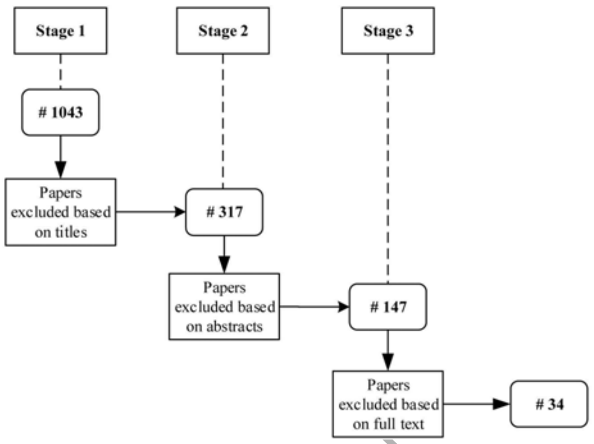
Stage 1
Stage 2
Stage 3
# 1043
Papers excluded based on titles
# 317
Papers excluded based on abstracts
# 147
Papers excluded based on full text
# 34

Figure 2: Flowchart of research methodology.

Stage 2: In this stage, the abstracts of the considered papers were checked based on the formulated research objectives. Therefore, we have classified the papers of various types (e.g. original work, survey, systematic literature review, mapping study, and empirical study) into two dimensions based on their research focus including cost optimization aspects and cost optimization parameters. As a result, this left us with 147 papers, which target on the focused dimensions.

Stage 3: In this stage, participating researchers read the full contents of the 147 papers. Finally, they selected 34 papers, which satisfy the defined inclusion/exclusion criteria. We used the final selected papers to create a classification based on the considered aspects and parameters of cost optimization approaches.

This paper is organized as follows: Section 2 presents the relevant reviewed cost optimization SWFS approaches. Section 3 presents the main aspects of cost optimization SWFS approaches in cloud and grid computing. Furthermore, it provides an in-depth discussion for each of these aspects. Section 4 categorizes the cost parameters into monetary and temporal cost parameters. Special emphasis is given to the mathematical models used to calculate cost, which may possibly affect a particular parameter of scientific workflow scheduling system. The obtained results and findings are discussed in Section 5. Finally, Section 6 concludes the review and presents useful open issues for future research.

2. Reviewed cost optimization SWFS approaches

This section reviews relevant cost optimization Scientific Workflow Scheduling (SWFS) approaches in cloud and grid computing. Table 1 presents the existing approaches, strengths of the underlying optimization, and limitations for all considered approaches.

Table 1: Cost optimization SWFS approaches.

Approach	Strength of underlying optimization	Limitation
Market-oriented hierarchical scheduling strategy [3]	This strategy offers provisional exploration in market-based cloud workflow structures. It assures suitable QoS requirements are imposed by users while decreasing the overall running cost of the workflow system.	Provisional exploration is offered only in market-based cloud workflow structures. There is a need to consider utility structure.
Dynamic Constraint Algorithm (DCA) [57]	User-friendly and is used to meet the requirements of the bi-criteria scheduling issue in grid. DCA uses a two-phase heuristic algorithm (primary and secondary), which has an extension for the knapsack problem for multiple-optimal solution.	The authors concluded that this algorithm requires a longer running time to satisfy the specified criteria.
SaaS Cloud Partial Critical Paths (SC-PCP) [68]	The cost of workflow execution is reduced while the user-determined deadline for the Software-as-a-Service (SaaS) framework is met. Schedules workflow based on QoS using a PCP.	The IaaS and pricing for cloud processing frameworks are not supported by this technique and data transaction cost is unaffected.
Workflow Orchestrator for Distributed Systems Architecture [5]	Extracts the disparities and encapsulates the QoS features offered by the cloud and grid structure. Allows competent organizations at moderate cost for batch queue with or without public resource management.	The authors found that this model is deficient in enhancing sequential resources possibly requiring higher cost.
Partial Critical Path Scheduling (PCP) [69]	The normalized workflow cost is decreased by creating a schedule that minimizes the total workflow execution cost, while satisfying a user-defined deadline with the total execution time for SLAs in utility grids.	The authors reported that PCP exceeds the user-defined deadline for small and medium-size workflow problems.
Compatibility of Hybrid Processor Scheduler [70]	Execution cost is lower since resources are chosen based on their energy levels and various forms of applications concerning high processing and storage space demands are characterized.	This method is only applicable for hybrid systems and is based on resources. However, other performance features, such as throughput, are not considered.
Ant Colony System (ACS) algorithm [71]	Algorithm focuses on solving the problems of large-scale WFS computational grids. Seven heuristics for the Ant Colony Optimization (ACO) approach have been designed based on the characteristics of WFS in a Grid system.	It has been concluded that this algorithm does not consider the service providers' performance while optimizing the user-preferred QoS parameters.
Critical Path-based Priority Scheduling (CPPS) [72]	The approach outperforms the traditional fair-share scheduling policy commonly adopted in real systems. Deploys and executes a test bed network by adopting an on-node scheduling policy, while improving the workflow performance of the mapping scheme.	It was found that the CPPS model fails to achieve optimized interaction between task mapping and the scheduling scheme.
Dynamic resource provisioning techniques [73]	Achieved a cost-effective execution of scientific workflows by implementing a just-in-time algorithm. This technique uses a combination of grid and cloud resources via dynamic provisioning of cloud resources when grid resources are unavailable.	Each task is scheduled when it becomes ready by implementing a just-in-time algorithm.
Time-cost tradeoff workflow scheduling algorithm [74]	Less completion time and lower cost meet requirements in practical applications, which can effectively meet users' needs. Adopts dynamic shared and autonomous resources in grid.	The authors concluded that the proposed algorithm lacks support of some user-defined QoS constraints, such as deadlines and budget.
Ant Colony Optimization (ACO) [75]	Improves cloud service performance in terms of reliability, response time, cost, and security. ACO utilizes a default rate to explain the ratio of cloud system service providers that break the SLAs of the WFS.	The authors did not consider the effect of QoS constraints in SLA from the users' perspective.
Genetic Algorithmic (GA) [7]	Monetary cost is reduced, and at the same time, user budget constraints and execution time are reduced simultaneous with meeting the user deadline constraint. Provides a dynamic search method by offering high-quality solutions for a vast search area in polynomial time by using the evolutionary principle.	The authors found that GA requires a longer running time than other heuristic techniques.
Low-cost rescheduling policy [76]	Rescheduling is taken into consideration at a few, carefully selected points during execution. This policy yields equally good performance as policies that consider rescheduling for every DAG task at lower cost.	It was reported that this policy focuses only on minimizing computation and communication time of Grid application in WFS without considering users' budget.
Budget constraint based workflow scheduling [19]	Presents a cost-based scheduling heuristic to minimize execution cost and time while meeting the user's budget. This method adopts GA to minimize grid workflow execution cost within a certain deadline.	It was observed that this model supports only specific types of services and does not consider duplication of critical tasks to meet users' QoS requirements.
Mixed-Integer Non-Linear Programming (MINLP) [77]	The main goal of this method is to satisfy users' deadlines as well as to improve failure rates and turnaround times. Focuses on the global cost optimization problem for the entire grid workload of WFS while minimizing the cost of utilizing grid resources.	It was observed through the conducted study that this model is not efficient for large WFS problems.
Transaction intensive cost-constraint algorithm [78]	Aimed at minimizing cost while meeting user-determined deadlines. This algorithm offers a graph of just-in-time cost relations during workflow execution by utilizing intensive transaction settings as well as considering the specified budget.	The authors mentioned that the service provider's performance is not considered in this algorithm.
Multi-cost job routing and scheduling [79]	An algorithm for polynomial complexity is provided. Resource reservations are enhanced and the start time for data transmission is identified as well as for task completion.	The communication and computational parameters of monetary cost are not completely considered.
DAG-LOSS & DAG-GAIN [80]	DAG-LOSS and DAG-GAIN use DAG scheduling as a fundamental framework for WFS in grid applications. Satisfy budget constraints while optimizing execution time.	It has been observed that this model necessitates additional time to achieve the required budget.
Integer Linear Programming (ILP) technique [81]	ILP utilizes two heuristic methods that are competent when deadlines are substantial. Reduces the monetary cost while addressing the deadline determined by the cloud users in the SLA contract.	For manifold workflow scheduling, the authors mentioned that this technique does not consider the mechanism's fault tolerance in a similar group of resources.

Continued on next page

Table 1 – continued from previous page

Approach	Strength of underlying optimization	Limitation
Multiple QoS of Multi-Workflows (MQMW) [82]	This model is used to minimize execution cost while prominently enhancing scheduling success rates. Scheduling success rates are prominently enhanced. It is designed for several workflows with various QoS requirements.	According to the authors, MQMW is not applicable for other QoS constraints such as execution time, availability, etc.
Time and cost-constraint scheduling strategy [21]	This strategy is able to control the complexity of large-scale, contemporary computing, network services, and the integration of grid computing towards a service-oriented technique. Applicable for different types of workflow applications, as it minimizes the overall execution cost and time.	It is necessary to consider other performance criteria for the service providers.
IC-PCPD2 and IC-PCP [6]	The aim is to minimize the workflow execution cost while meeting the deadline-constrained workflow. These algorithms present a commercial cloud that involves the on-demand resource, homogeneous network, and framework of pay-as-you-go pricing.	IC-PCPD2 and IC-PCP ignore the privacy requirements while considering the workflow's deadline.
Deadline-Markov Decision Process (MDP) [83]	Execution costs are reduced as deadlines are met. Task partitions and overall deadline assignments are prescribed for optimized planning of execution and efficient run-time rescheduling.	The authors reported that MDP lacks in achieving efficient run-time rescheduling.
Scalable-Heterogeneous-Earliest-Finish-Time algorithm [84]	Execution time optimization is achieved while elastic runtime scalability is attained. The authors mentioned that SHEFT is more flexible than other cost-aware WFS models as it efficiently schedules on-demand size of a workflow along with allocated resources.	Apparently this algorithm is deficient in indicating accurate runtime prediction, which acts as a pre-requisite of scheduling.
Multi-objective Differential Evolution [17]	Provides flexibility of trade-off while calculating preference requirements. Furthermore, it selects WFS according to the requirements of user-specified grid system QoS.	The authors considered only the distance criteria to examine the algorithm's performance.
Particle Swarm Optimization (PSO) [18]	Is for designing a heuristic that utilizes PSO by formulating a framework for task-resource mapping to reduce the overall completion cost. PSO incurs large amounts of data and execution cost.	The authors mentioned that the proposed algorithm ignores the temporal cost of service providers.
Multi-Constraint Graph Partitioning (MCGP)[85]	Reduces the cost of communication while minimizing the workflow execution time. MCGP utilizes the partitioning DAG graph that is applied for a complicated Cloud workflow. The time elapsed for file size access from remote nodes is decreased as well.	The MCGP method does not consider the input/output files of each job, which ultimately affect the dynamic changes of workflow applications.
Hybrid Cloud Optimized Cost model (HCOC) [4]	Improves WFS by making use of multicore resources and minimizing the monetary cost within a deadline while offering affordable makespans to users. HCOC imparts sufficient processing power by determining which resource ought to be leased from the public cloud.	The authors mentioned that this model does not consider the workflow execution time required by a user in a single-level SLA contract.
Workflow-aware Pre-processing Provisioning Dynamic Scheduling (WPPDS) [86]	An elastic resource provisioning and task scheduling mechanism have been proposed to perform scientific workflows and submitted at unpredictable time in cloud. The aim is to finish as many high-priority workflows as possible by considering the budget and deadline aspects. The evaluation of this mechanism shows stable performance with inaccurate parameters and task failure.	The approach should be tested within a real scientific workflow environment by considering the data communication cost and storage cost for executing larger workflows in cloud.
SciCumulus [13]	A parallel technique to define the number and types of VMs and to design the parallel execution strategy for a scientific workflow. Model a cost based approach to satisfy the QoS and help determine an adequate configuration of the environment according to restrictions imposed by service consumers.	The approach is lacking in providing a comprehensive evaluation environment in order to test the total execution cost and execution time.
Dynamic Provisioning Dynamic Scheduling [31]	A series of adaptive scheduling algorithms for scientific workflows cloud-based have been proposed to provide a dynamic dimensioning and vertical scaling during workflow execution to comply with service consumers' constraints. This work represents good performance results by bringing opportunities for modifying the number of VMs.	Lacks in considering the optimization of the initial virtual machine allocation. Also, this approach is unable to handle unpredictable workloads.
Cost-Effective Virtual Machine Allocation Algorithm within Execution Time Bound [87]	A two-step heuristic workflow mapping (scheduler) has been used to maximize the resource utilization while minimizing the overhead of the cloud. The delay of workflow execution has also been considered to calculate makespan and guarantee the user required deadline, to reduce the overhead of start-up and shutdown of a virtual machine.	The authors have taken the VM launching overhead variation as a key variable for designing resource allocation algorithms only. However, there are several other constraints which should also be considered (e.g. the predication of workload, QoS requested by users)
Critical-Greedy (CG) [54]	An algorithm (MED-CC) has been designed for workflow scheduling problem to minimize end-to-end delay while meeting the budget constraint. Furthermore, the authors have proposed a Critical-Greedy algorithm using GAIN approach that only imposes rescheduling process on dynamic critical tasks.	The reported experimental results prove the performance benefits of the approach but not completely achieve the required cost optimization.
SO-Routine and MO-Routine algorithm [44]	Evolutionary approach based algorithm for multi-objective optimization has been proposed as a solution to scheduling of SWF applications. Also, the authors designed a novel representation of two independent chromosomes, one for mapping and one for ordering.	This method is more time consuming than other heuristic based approaches. The authors used only two Amazon EC2-based clouds which are not enough for evaluating a multi-cloud challenge.

3. Cost optimization aspects of SWFS

One of the major aims of SWFS is to reduce the overall cost of running the cloud workflow system [3] especially for complex jobs like scientific workflow. The execution cost must be taken into consideration when scheduling tasks into the resources [7, 19]. The running cost of an application is minimized through the cost optimization scheduling policies [88]. Therefore, several aspects need to be considered while scheduling the scientific workflow tasks. This section presents, a classification for aspects of cost optimization SWFS approaches in cloud and grid computing. Figure 3 presents our classification of cost optimization aspects of SWFS based on eight main classes: (a) computing environment, (b) optimization method, (c) structural representation, (d) profitability, (e) scheduling technique, (f) workload type, (g) optimization criteria, and (h) QoS constraints.

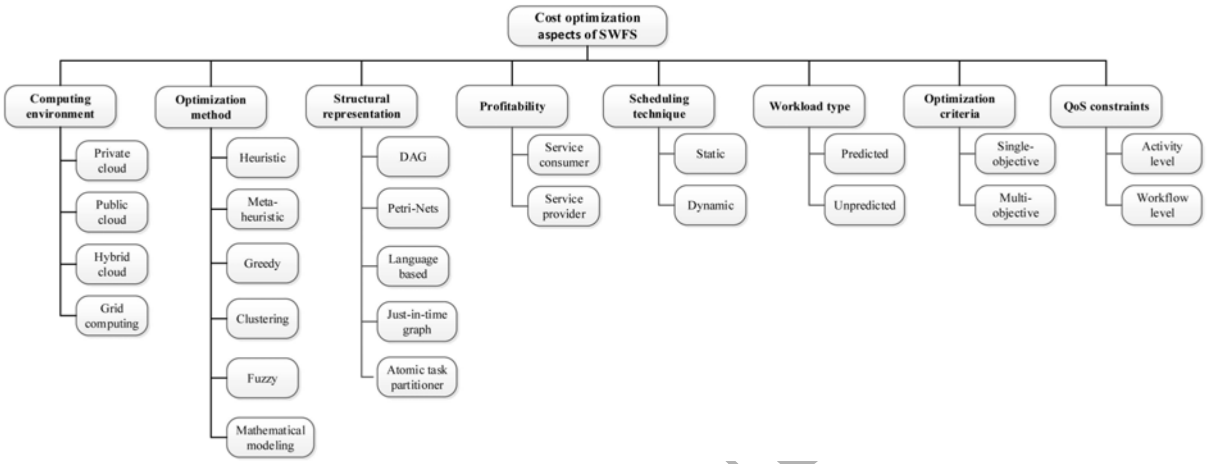

Figure 3: A classification of cost optimization aspects of SWFS.

In the following sub-sections, we briefly discuss each aspect of the presented classification.

3.1. Computing environment

The cost of executing a scientific workflow is usually affected by the computational environment due to the fact that it has direct impact on resource utilisation. For example, scientists should select the computing environment based on their requirement which could be related to the size of application, privacy of the used data and other QoS constraints (e.g. budget, and deadline). Therefore, each computing environment has a different specification, which ultimately affects the total cost of SWFS. In this paper, we consider four main computing environments: (i) private cloud, (ii) public cloud, (ii) hybrid cloud, and (v) grid computing.

Private cloud: due to data privacy limited budget constraints, most of the service consumers select private cloud. The operation cost is usually not taken into consideration and the resources' usage cost is also not measured [4, 40]. The total cost of SWFS in the private cloud model can normally be calculated by adding computation cost (i.e. computation time) to communication cost (i.e. communication time).

Public cloud: this computing type is usually selected when service consumers need to execute a large scientific workflow, which can not be executed locally. The total cost of SWFS includes the cost per time unit of using cloud resources [73]. The communication cost value of executing workflow in the public cloud is normally assumed to be zero, because of the assumption that all resources are built into the same computational infrastructure.

Hybrid cloud: this type of computing can give a flexible specification for the required resources by service consumers. The scheduler is required to consider the load balancing of submitted work in order to fully utilize the private and public cloud. However, this model adds more complexity to the cost of the consumed services. Hence, the cost of SWFS can be calculated by adding the SWFS cost in a private cloud to the SWFS cost in a public cloud. Another hybrid cloud scenario is where the total cost of SWFS is defined as data transfers from and to the cloud [5, 89]. The available bandwidth into the connected processing resources of the hybrid cloud affects the makespan [9, 20, 31, 90]. Therefore, the bandwidth cost for the hybrid cloud environment can be defined as the cost that the service provider charges to service consumers per the amount of data transferred ($/GB).

Grid computing: this computing type is very similar to the other cloud computational environments which is also used to provide an optimal solution to meet the service consumer requirements by providing scalable, and flexible solutions for the considered scientific applications [3]. The total cost of SWFS in grid computing can be calculated by summing all the used resources.

3.2. Optimization method

This aspect is considered one of the most important cost optimization aspects due to its direct impact on task-resource mapping processes. Several methods (i.e. rule-based, search-based, coverage-based) have been proposed in the literature to find an optimal solution for the total cost of executing the SWFS in cloud and grid computing environments. The heuristic methods have been widely used for the scheduling problem. The heuristic methods efficiently determine the tasks' order and schedule them according to the best performance (in terms of effectiveness and accuracy) [80, 91]. On the other hand, meta-heuristic methods (e.g. genetic algorithm) have also been effectively used to achieve improved performance compared to other heuristic methods, but with some compromise on the execution time [3, 68, 69].

3.3. Structural representation

Due to the complex nature of the scientific workflow application, the structural representation is the first stage of any WfMS that is used to simplify the submitted scientific activities to the scheduler. Several types of structural representation methods have been adopted in the literature to represent the tasks' dependency (precedence constraints) of SWFS: (i) Graph-based modeling methods (i.e. DAG [50, 52, 92–95], Petri Nets [96, 97]), and (ii) Language-based modeling tools (i.e. XML Process Definition Language (XPDL)) [98]. Notice that for each type of these methods, there are cost parameter representations (e.g. computation cost, and communication cost). For instance, the DAG based method is the most popular method used in the state-of-the-art approaches to estimate the execution cost of different available resources for every task, which represents the overall computational cost. In addition, the time to communicate data between resources is given, which represents communication cost (e.g. bytes to transmit).

3.4. Profitability

As shown in Figure 1, due to the importance of the cost optimization of SWFS in cloud and grid computing for different WfMS users, in order to have a deeper understanding on the proposed cost optimization approaches, we have classified the aim of the reviewed cost optimization SWFS state-of-the-art approaches into two groups with respect to profitability: (1) approaches whose main goal is the service consumers' profitability, and (2) approaches whose primary goal is the service providers' profitability [3, 31, 71, 74, 88, 99]. The service consumer represents a person or organization (which could be a scientist or researcher) that uses the cloud computing services (i.e. Infrastructure as a Service (IaaS), Software as a Service (SaaS) or Platform as a Service (PaaS)) in order to execute the scientific application [100]. Conversely, the service provider represents a company or organization (which could be any cloud service provider) that offers cloud computing services to service consumers (person or organization) with different QoS constraints and prices [3, 6, 68].

When a cloud service consumer requests a service from the service provider, a vital matter of reducing the leasing cost (for the service consumer) arises, while the contribution to lower the overall execution cost of workflow (for the service provider) increases [73, 101–103]. In contrast, for the service provider, the aim is to reduce the cost of leasing time of the cloud resources [2, 4, 18, 69, 71, 73, 78]. Therefore, time optimization is profitable for the service provider by reducing the cost of maintaining resources. Consequently, this gives an advantage to cloud service consumers since it will reduce the cost of workflow execution [104].

Recently, researchers have focused on the study of scientific workflow applications on multi-cloud service provider. The implementation of multi-cloud service provider is more complex which requires interoperability. For example, an application is hosted in Azure Cloud (database server) and deployed on a web server on Amazon Cloud. Thus, results and benefits of cloud adoption cannot be achieved, unless data and applications are integrated across clouds properly [105]. As a solution, interoperability approaches should be used to avoid lock-in of multi-cloud service provider. The interoperability perspective in technology focuses on the resulting mission of compatibility or incompatibility between systems and data collation partners. Therefore, several approaches such as mOSAIC [106] and RightScale [107] have proposed software and frameworks to help organizations to manage and utilize resources spanning on multi-cloud service provider. However, due to the above-mentioned challenges, there is no SWFS approach that provides a generic solution to monitor full stack challenge for multi-cloud service provider, from software development tools to run-time control [105, 106, 108, 109]. Therefore, future researchers need to focus on optimization of the cost-aware approaches of SWFS in cloud computing.

3.5. Scheduling technique

The scheduling technique represents the mechanism that the scheduler chooses to schedule scientific workflow tasks which is strongly related to the cost of the utilized resources in WfMS. Two types of techniques have been adopted in state-of-the-art cost optimization SWFS approaches: (i) static technique, and (ii) dynamic technique.

The static technique requires the scheduler to know in advance the characteristics of all the scientific tasks, including their sizes, service demands and estimated execution cost. Also, the static techniques are performed under two assumptions (i) the tasks arrive simultaneously to the resources, and (ii) the resource's available time is updated after each task is scheduled [70, 95]. The static technique can efficiently schedule large workflows on large data centers. Another advantage is that it is easier to adapt a static technique based on scheduler's perspective. Furthermore, it is more user friendly as the precomputed schedule allows quoting a price for the computation. In addition, it allows the service consumers to choose from multiple scheduling options according to the price and time constraints [110, 111].

The dynamic technique is more flexible than static scalability where scientific tasks are dynamically available (continuous stochastic stream) for scheduling. However, it is more complex than the static scheduling technique since it needs to update the system information on the fly [70]. The main advantage of dynamic strategy is that it can be adopted when a task set or a resource set is heterogeneous. For instance, not all tasks arrive simultaneously, or some resources are off-line at intervals [112]. The other advantage is that it considers only few required parameters in advance. Due to the aforementioned advantages, the dynamic scalability is more suitable for executing the on-demand workflow applications in cloud environments.

3.6. Workload type

Two types of workload methods, i.e. predicted mode and unpredicted mode, have been adopted in SWFS approaches based on their method of loading the tasks to the scheduler in cloud and grid computing. The workload type can affect estimation.

Predicted (batch mode): in predicted or batch mode (also referred as latter mode), the tasks are first collected as a group of problems that are examined for scheduling at prescheduled times (predefined moments). Thus, it is better to map the tasks for suitable resources depending on their characteristics [70]. This enables the predicted mode to determine about the actual execution time for a larger number of tasks [113]. One of the main advantages of this mode is to maximize the throughput while minimizing the turnaround time (the time between task submission and its completion) [114].

Unpredicted (on-line mode): unpredicted or on-line mode (also referred as former mode) where tasks are scheduled to a resource as soon as they arrive for execution and there is no waiting for the next time interval on available resources at that moment [70, 113]. In this mode, each task is scheduled only once and hence the scheduling result cannot be changed. Therefore, unpredicted mode is suitable for the scheduling scenarios where the arrival rate is low [54, 113, 114].

3.7. Optimization criteria

In order to propose an optimal solution for SWFS problem, the total cost of executing workflow tasks needs to be minimized.

The reviewed SWFS approaches can be classified into two main classes: (i) Single-objective optimization based approaches, and (ii) Multi-objective optimization based approaches. In the literature, researchers have mainly focused on optimizing cost parameters only. Thus, the approaches which only consider execution cost or time (but not both) are referred as *Single-objective optimization* based approaches, since they only target at optimizing the cost of SWFS problem. Some approaches have focused on minimizing the execution cost [6, 68, 69, 75, 81, 115, 116], while other approaches have focused on the execution time [5, 70, 79, 85, 117]. However, due to the rapid development of the provided computing services (e.g. pay-as-you-go, and on-demand), many other constraints (e.g. QoS constraints) should also be considered to optimize the cost of SWFS. Due to the above-mentioned reasons, the complexity of the proposed approaches has increased to a great extent, which ultimately demands to handle the trade-offs between the cost and other affected constraints. Nevertheless, a group of services may have the same requirements so they can complete similar tasks or activities with different execution cost and time [74]. These approaches are referred as *Multi-objective optimmization* based approaches. Therefore, majority of the recent approaches have adopted the hybrid and hyper techniques for the heuristics and meta-heuristics methods to obtain an optimal solution.

3.8. QoS constraints

QoS has a major impact on SWFS in cloud and grid computing, since the success of computational tasks heavily depends on the desired QoS levels [45, 69, 72, 118]. For multi-objective problem, such as SWFS, there are several QoS constraints that must be taken into consideration for a given service when designing an efficient WfMS in cloud and grid computing [4, 119–121]. The service providers must consider satisfying the service consumers' QoS requirements based on SLA. Therefore, for a scheduling process, the QoS has a direct affect on each stage of a typical workflow instance.

Interested readers may consult our previous work [65] for complete definitions and details about each of the QoS constraints in the content of cost-aware WFS. In this paper, we have identified the QoS constraints for each of the reviewed cost optimization approaches in Table 2 (as multi-objective optimization criteria). Furthermore, we have considered another important QoS aspect, which is the way of handling the QoS constraints in SWFS approaches. There are two methods to consider QoS for cost optimization SWFS approaches in cloud and grid computing (Figure 4).

The first method is to allow the users to assign activity-level QoS constraints, and then the overall QoS can be assessed by computing the QoS constraints of all individual activities based on the specific QoS model. For example, a workflow reduction algorithm can be employed to calculate the deadline for the entire workflow based on the desired execution time of individual workflow activities [23, 69].

The second method is to assign QoS constraints at workflow-level where users need to define the overall workflow QoS requirements, and then the workflow system uses automatic strategies to assign local and activity-level QoS constraints to the workflow segments and individual activities. For example, a deadline assignment approach such as Equal Slack and Equal Flexibility [82] can be applied to determine the expected execution time of individual activities based on the deadline for the entire workflow.

Table 2 provides a comparison of the proposed SWFS approaches based on the defined aspects: computing environment, optimization method, structural representation, profitability,

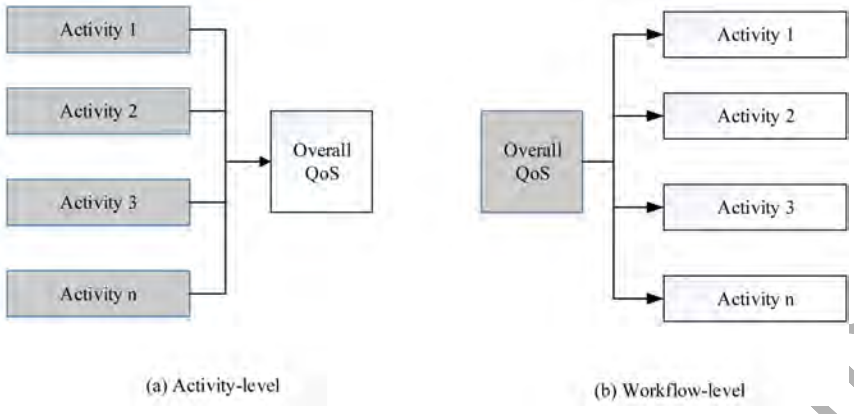

Figure 4: Methods for considering QoS constraints.

scheduling technique, workload type, optimization criteria and QoS constraints.

The cost optimization aspects of SWFS (Table 1) can be classified based on the computing environment aspects: (i) grid computing, (ii) private cloud, (iii) public cloud, and (iv) hybrid cloud. The cost optimization SWFS approaches such as [17, 57, 69, 71, 74, 83] considered a grid computing model to schedule the given workflow tasks. Alternatively, SWFS approaches, including [18, 78, 84] employed a private cloud environment to execute the given workflow tasks. Similarly, public cloud and hybrid cloud models have been considered by [3, 4, 68, 75, 81], approaches to optimally solve the SWFS problem respectively. In contrast, very few approaches including [5, 72, 73] focused on utilizing the strengths of both cloud and grid computing environments to provide a more cost-effective solution for SWFS problem. As compared to private clouds, public clouds are more expensive in terms of communication cost and execution time mainly due to the far-proximity of the resources. On the other hand, considering hybrid model offers a highly flexible scalability feature due to the efficient utilization of resources compared to standalone, public, and private cloud models.

The SWFS approaches can be categorized based on different aspects of optimization method such as heuristic [5, 69], clustering [84], critical path [72], fuzzy [73], extended critical activity [74], holt-winter's method [77], greedy [70], market-driven [78], meta-heuristic [3, 6], cloud WF exchange agent, mathematical modeling [83] and partitioning [85]. Majority of the SWFS approaches focused on employing heuristic and meta-heuristic as an optimization method. Meta-heuristic approaches achieved a better performance in terms of effectiveness and accuracy at the cost of extended execution time compared to heuristic methods. The structural representation parameter categorizes SWFS approaches into network routing [79], DAG [57, 72, 77, 81], atomic task partitioner [21], and just-in time graph [78]. Majority of the state-of-the-art SWFS approaches used DAG structural model to graphically visualize the dependency among the SWFS tasks, especially focusing on the cost parameters (execution time, and communication time).

Table 2: A comparison on cost optimization aspects of SWFS.

Ref.	Environment	Method	Structural	Profitability	Technique	Workload	Optimization criteria	QoS constraints
[3]	Hybrid cloud	Meta-heuristic	DAG	Service consumer	Static-Dynamic	Unpredicted	Multi-objective (M/S)	Activity-level
[57]	Grid	Heuristic	DAG	Service consumer	Static-Dynamic	Predicted	Multi-objective (R/A)	N/A
[68]	Public cloud	Heuristic	DAG	Service consumer	Dynamic	Predicted	Multi-objective (D/M)	Activity-level
[5]	Cloud & Grid	Heuristic	DAG	Service provider	Dynamic	Predicted	Multi-objective (D/R/A/M)	Workflow-level
[69]	Grid	Heuristic	DAG	Service provider	Dynamic	Predicted	Multi-objective (D/M)	Workflow-level
[70]	Private & Public cloud	Greedy	N/A	Service provider	Dynamic	Predicted	Single-objective	N/A
[71]	Grid	Heuristic	DAG	Service consumer	Dynamic	Predicted	Multi-objective (B/D/R/M)	Activity-level
[72]	Cloud & Grid	Critical-path method	DAG	Service provider	Dynamic	Predicted	Multi-objective (M)	N/A
[73]	Cloud & Grid	Fuzzy	DAG	Service provider	Dynamic	Predicted	Multi-objective (M)	N/A
[74]	Grid	Extended critical activity	DAG	Service provider	Dynamic	Predicted	Multi-objective (D/M)	Workflow-level
[75]	Hybrid cloud	Heuristic	DAG	Service provider	Static	Unpredicted	Multi-objective (B/R/S)	Workflow-level
[7]	Grid	Heuristic	DAG	Service consumer	Dynamic	Predicted	Multi-objective (B/D/A/M)	Workflow-level
[76]	Grid	Heuristic	DAG	Service provider	Dynamic	Predicted	Multi-objective (M)	Activity-level
[19]	Grid	Heuristic	DAG	Service provider	Dynamic	Predicted	Multi-objective (B/M)	Workflow-level
[77]	Grid	Holt-winter's method	DAG	Service provider	Static-Dynamic	Predicted	Multi-objective (D)	Activity-level
[78]	Private cloud	Market-Driven	Just-in-time graph	Service consumer	Dynamic	Unpredicted	Multi-objective (B/D)	Activity-level
[79]	Grid	Heuristic	Grid network routing	Service provider	Dynamic	Predicted	Single-objective	Activity-level
[80]	Grid	Heuristic	DAG	Service consumer	Static	Unpredicted	Multi-objective (B/M)	Workflow-level
[81]	Public cloud	Heuristic	DAG	Service consumer	Static	Predicted	Multi-objective (D/S)	Workflow-level
[82]	Private cloud	Heuristic	DAG	Service provider	Dynamic	Unpredicted	Multi-objective (M)	Activity-level
[21]	Grid	Heuristic	Atomic task partitioner	Service provider	Static-Dynamic	Predicted	Multi-objective (B/D)	Workflow-level
[6]	Private cloud	Meta-heuristic	DAG	Service provider	Dynamic	predicted	Multi-objective (D/M)	Activity-level
[83]	Grid	Mathematical modelling	DAG	Service consumer	Dynamic	Predicted	Multi-objective (D)	Activity-level
[84]	Private cloud	Clustering	DAG	Service provider	Dynamic	Unpredicted	Multi-objective (M)	Workflow-level
[17]	Grid	Heuristics	DAG	Service consumer	Static	Unpredicted	Multi-objective (B/D/M)	Workflow-level
[18]	Private cloud	Heuristic	DAG	Service provider	Dynamic	Unpredicted	Multi-objective (M)	Activity-level
[85]	Private cloud	Partitioning	DAG	Service provider	Dynamic	Unpredicted	Single-objective	Activity-level
[4]	Hybrid cloud	Heuristic	DAG	Service provider	Dynamic	Unpredicted	Multi-objective (D/M)	Workflow-level
[44]	Public cloud	Meta-euristic	DAG	Service consumer and Service provider	Dynamic	Predicted	Multi-objective (D/R/A)	Activity-level
[86]	Private cloud	Dynamic programming algorithm	DAG	Service provider	Dynamic	Predicted	Multi-objective (B/D)	Workflow-level
[13]	Public cloud	Heuristic	VisTrails	Service provider	Dynamic	Unpredicted	Multi-objective (R/A/S)	Activity-level
[31]	Public cloud	Heuristic	DAG	Service consumer	Static and Dynamic	Unpredicted	Multi-objective (B/D/R/M/S)	Activity-level
[87]	Private cloud	Heuristic	DAG	Service provider	Dynamic	Unpredicted	Multi-objective (D/R/A/M)	Workflow-level
[54]	Public cloud	Heuristic	DAG	Service consumer and service provider	Dynamic	Predicted	Multi-objective (B/D/A/M)	N/A

Legends: B = Budget; D = Deadline; R = Reliability; A = Availability; M = Makespan; S = Service level agreement

The existing scheduling techniques have been classified into three parameters including static [17, 75, 80] dynamic [54, 71, 72], and static-dynamic [3, 21, 77]. Dynamic scheduling is efficient for a cloud computing environment due to its ability to handle the arriving tasks. The selection of workload type mainly depends on the tasks arrival rate based on the defined suitability criterion between using predicted or unpredicted mode. The workload types of SWFS approaches have been categorized into two mode, predicted [7, 71–74, 76], and unpredicted [3, 18, 75, 80, 82, 84] modes.

4. Cost optimization parameters of SWFS

This section critically analyzes the devised classifications for cost optimization parameters of SWFS in cloud and grid computing. A complete discussion on the sub-classification of cost parameters including the monetary cost and temporal cost is presented in sub-sections 4.1.1 and 4.2.1. Finally, the section provides the correlations between the surveyed cost optimization SWFS approaches and the profitability by extracting their association with cost optimization parameters.

After analyzing the cost optimization parameters considered by researchers in the area of SWFS in cloud and grid computing, it is found that the classification of the cost optimization parameters is dependent on two types: (i) monetary cost parameters, and (ii) temporal cost parameters, as shown in Figure 5.

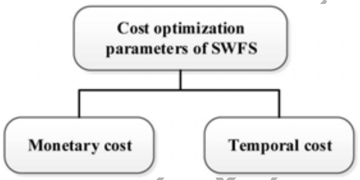

Figure 5: A classification of cost optimization parameters of SWFS.

The scheduling approaches are supposed to estimate in advance whether a workflow will be able to meet the requested constraints (e.g. deadline) or not [5, 122]. However, the estimation process may be compromised due to uncertainty in task estimations especially in the case of deadline-sensitive applications (e.g. weather forecasting). Also, the resource providers find it hard to ensure the resource availability due to the variability and complexity of the underlying resource characteristics and access policies. Researchers have considered three parameters to overcome the aforementioned challenges: (i) Estimated Execution Time (EET); (ii) Estimated data Transfer Time (ETT); and (iii) Estimated Finish Time (EFT) [2, 5, 69, 73, 85, 89, 123, 124]. The scheduler needs to include these parameters in the workflow definition to enhance the estimation process by considering the historical results.

Hence, it is crucial to recognize the parameters of the monetary cost and temporal cost, and the inter-dependent parameters. The specific breakdown (sub-classifications) and details on each of the monetary cost parameters are given in Section 4.1.1 and those of the temporal cost parameters are presented in Section 4.2.1.

4.1. Monetary cost parameters

This section presents two main sub-sections including: Classification of monetary cost parameters from the profitability aspect.

4.1.1. Classification of monetary cost parameters

This section provides details on the sub-classification of monetary cost optimization parameters as shown in Figure 6: (i) the estimated execution cost, (ii) the cost of service offered by the service provider, (iii) computation cost, (iii) communication cost, (iv) elasticity cost, and (v) cost of data storage.

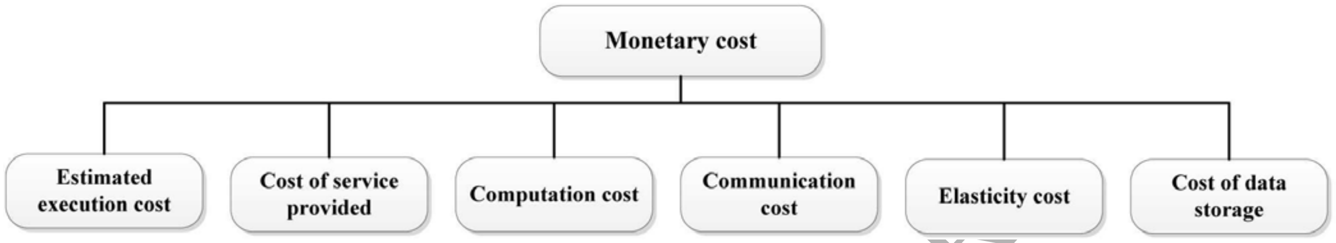

Figure 6: Sub-classification of monetary cost optimization parameters of SWFS.

4.1.1.1 Estimated execution cost

Estimated execution cost can be measured before the process of workflow scheduling, and it assists the algorithm with decision making for scheduling tasks and data with the help of the DAG graph partitioning of the workflow. The two main elements of estimated execution cost are: (i) estimation of computation cost, and (ii) estimation of communication cost [53, 85, 122]. The estimated execution cost represents the cost of processing a task at the resource. In contrast, the estimated communication cost refers to the cost of sending the required data along the edges of DAG from one resource to another based on the tasks' dependencies.

4.1.1.2 Cost of service provided

Cost of service provided represents the cost of the service offered by the service provider as an external cost to fulfill a service request to the service consumers, usually measured in dollars. Every service provider may have particular strategies for task-level scheduling to optimally use the system's running cost at its own data center [3, 33, 34, 46]. It is essential to reduce the total cost of application execution on the resources offered by the cloud service providers such as GoGrid and Amazon [18]. The total cost of service provided by the service provider can be calculated based on the cost of the services used by the workflow scheduler [57]. Therefore, workflow execution cost is the sum of the cost of each activity [74]. Additionally, the cost of an application is defined by the summation of the costs of all selected service instances [71].

4.1.1.3 Computation cost

Computation cost is defined as the cost of using computing resources and is usually measured in dollars per hour. The cost of computation is generally the user's main concern [14, 32, 45, 125]. The computation cost of tasks on the host computer is inversely proportional to the time spent on computing these tasks using the resources [3, 18]. For instance, the computation cost of the public cloud is represented by the amount of money to be paid for using the enterprise's computation resources, which can be categorized based on different computational specifications [18, 33, 46, 102].

The total cost of computing the workflow is also affected by the data size [18]. Thus, decreasing the execution cost of running a workflow application on the cloud system is one of the main reasons for lowering the total cost [3, 6, 81, 126].

4.1.1.4 Communication cost

Communication cost is defined as the cost of data transferred to/from a data-storage resource, and is usually measured in dollars per megabyte of data. Communication cost between resources as well as the dependency between tasks introduces high communication cost, as data needs to be transferred from one resource to another [15, 127]. The communication cost is only applicable when two tasks have data dependency [125]. This produces higher storage and transmission cost compared to the cost of running the data [18]. Nevertheless, the internal transfer of the data is free in many real clouds such as Amazon, so the cost of data transfer is said to be zero in this model [6, 68]. Hence, there is no charge for data transfers within the same service provider's region [125]. As such, there is a significant link between the cost of data communication and data allocation. It is essential to schedule the computational tasks near the data and comprehend the moving cost of the work in comparison to data movement (minimizing the cost of communication). Also, data must be distributed over many computers, and computations must be steered towards the best place for execution in order to minimize communication cost [8, 81].

4.1.1.5 Elasticity cost

Elasticity in resources provisioning to the service consumer's computing environment is one of the most important features of a cloud system. The cloud system is able to handle the execution of complex computational tasks, which require powerful resources (e.g. machines and storages). Thus, the cloud computing system is suitable to address the problems of large-scale SWFS applications [128, 129]. In contrast, the elasticity of SWFS is bounded by the number of resources requested by the scheduling algorithm [4]. Therefore, the SWFS approaches need to make full use of the resources' elasticity by providing an efficient resource allocation within the lowest cost when tasks are completed earlier than the predicted time. For instance, in a hybrid cloud system, the service consumer is required to efficiently utilize the usage of public cloud resources that can be aggregated to the private resources' pool as necessary [46, 81, 130].

4.1.1.6 Cost of data storage

Cloud Workflow Management System (WfMS) is required to deliver on-demand storage services and processing power, due to the fact that the cloud WfMS has to deal with data centers which can be clusters of commodity hardware [81, 131–134]. Executing a large size of scientific workflow applications usually needs high performance computing resources as well as massive storage [134]. The execution of a workflow task consists of three phases, downloading of input data from the storage system, running the task, and transferring output data to the storage system [46]. Therefore, for a cloud WfMS storing all the data generated during workflow executions may cause a high storage cost [132, 135]. In order to reduce the total cost of SWFS, the cloud workflow system requires a strategy that can reduce the cost of the cloud WfMS by automatically storing only appropriate data in the cloud storage [81, 132, 134, 136]. Several strategies have been reported for cloud storage. For example, BigTable includes Google File System (GFS), SimpleDB data cloud, and MapReduce infrastructure, Amazon's S3 storage cloud, EC2 compute cloud and Hadoop system.

4.1.2. Monetary cost parameters from profitability aspect

This section highlights the results related to the approaches, which represent the relationships between profitability for the service consumers and the monetary models' cost parameters of cost optimization scheduling as shown in Table 3.

Table 3: Monetary cost parameters from the service consumers' point of view.

Approach	**Estimated execution cost**	**Cost of service provided**	**Computation cost**	**Communication cost**	**Elasticity cost**	**Cost of data storage**
[78]	x					
[83]	x		x	x	x	x
[80]		x	x	x		
[81]		x	x	x		
[3]			x			
[68]			x	x		
[17]			x	x		
[7]	x		x	x		
[71]			x	x		
[69]	x		x	x		
[81]					x	
[19]	x		x	x		
[86]	x	x	x		x	x
[13]	x					x
[31]	x			x		
[87]		x	x	x		
[44]	x	x	x		x	

From the service consumers' profitability perspective (Table 3), some approaches have considered the estimated execution cost as a strategy for planning the scheduling before the scheduler allocates suitable resources based on their availability. Computation cost and communication cost are frequently used during the scheduling process stage. This shows that generally, the users of workflow applications are more concerned with the amount of money they need to pay for the service. However, only few models consider the cost of data storage in their approaches, which is potentially due to the need for using a private (locally) storage instead of using a storage that is provided by the service provider (remotely). Therefore, service provider needs to consider providing more flexible storage services by keeping only appropriate data in the cloud. For elasticity cost, there is a strong need for full use of the resource elasticity by providing an efficient resource allocation within the lowest cost.

Table 4 indicates the relationship between the profitability for service providers and the monetary cost parameters of cost optimization scheduling.

Table 4: Monetary cost parameters from the service providers' point of view.

Approach	Estimated execution cost	Cost of service provided	Computation cost	Communication cost	Elasticity cost	Cost of data storage
[18]			x	x	x	
[72]			x			
[73]		x				
[84]			x			
[76]			x	x		
[4]		x	x	x	x	
[5]			x	x		
[79]		x	x	x		
[6]			x			x
[85]	x		x	x		
[57]			x	x		
[69]	x		x	x		x
[19]	x		x	x		
[86]	x	x	x		x	x
[13]	x		x	x		x
[31]	x	x		x		
[54]	x		x	x		x
[44]	x	x	x		x	

From the point of view of service providers' profitability (Table 4), it is evident that many approaches take the estimated execution cost into consideration [19, 69, 85]. The purpose of this parameter is to measure the cost of the submitted workflow tasks, which is estimated by the scheduler before the process of SWFS execution starts. There is a small number of scheduling approaches dedicated to the cost of service provided as service profit. This signifies that customers find the cost of the service provided very important. In addition, the computation and communication cost parameters are widely applied to the service category as the principal parameters representative of monetary cost.

4.2. Temporal cost parameters

This section discusses temporal cost parameters in terms of the profitability for the service consumers and service providers.

The literature review points out that in order to achieve effective and efficient cost of application data, the schedulers should minimize the total time taken for execution (makespan) to reduce the total execution cost [7, 18]. However, the two aims (the cost of running a process on a resource and the time expected for execution) are contradictorily related [17, 19, 57, 83]. In addition, the time taken for execution and cost of execution are the two normal restrictions in the "pay-per-use" model of cloud computing [137]. Thus, faster resources are more costly and vice versa with slower resources. As a result, the scheduler has to face a time-cost tradeoff in choosing suitable resources [6]. Besides, the cost of execution rises during longer delays, as the scheduler switches the balanced tasks to more costly services to finish off the balance of execution within the subscribed deadline [14, 32, 39, 42, 83, 138].

Besides, to calculate the cost of SWFS, it is significant to take into consideration all the time intervals that each resource is utilizing for task processing and data transmission [4, 7, 57].

4.2.1. Classification of temporal cost parameters

This section categorizes the temporal cost according to the scheduling stages (i.e. pre-scheduling, during scheduling, and post-scheduling) as per needs of the scheduler.

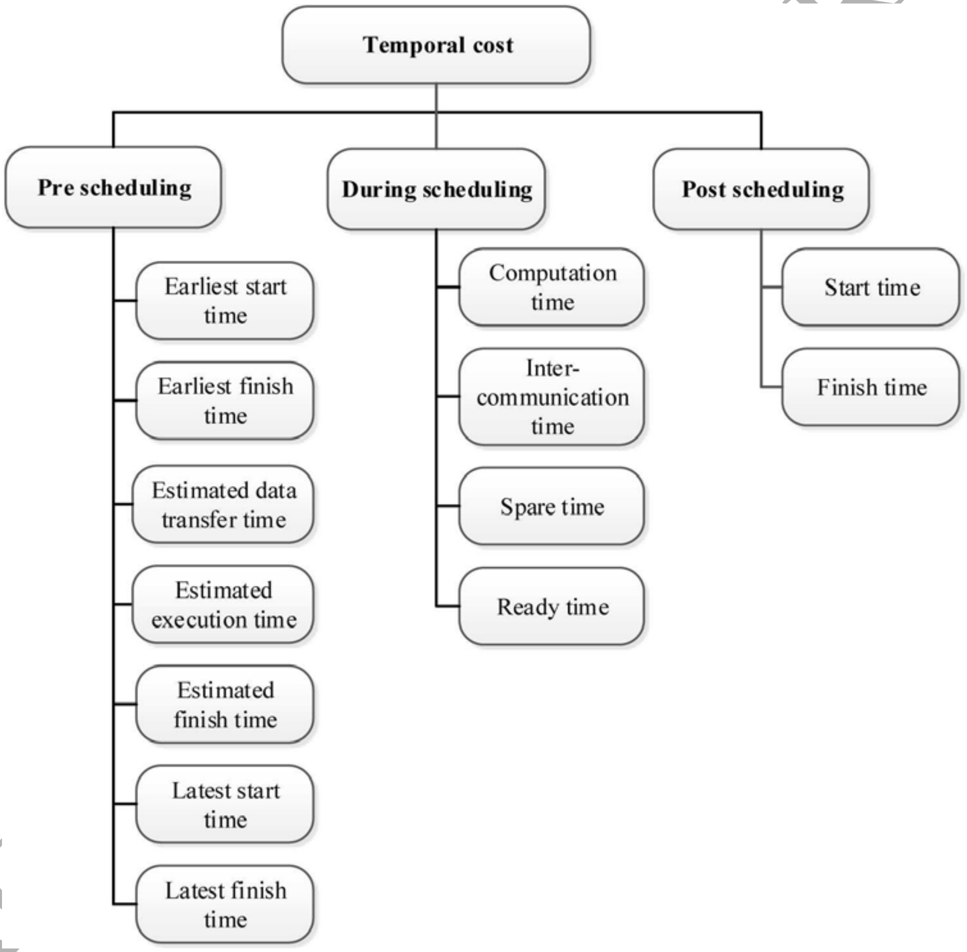

Figure 7: Sub-classification of temporal cost optimization parameters of SWFS.

From the sub-classification of cost optimization parameters of temporal cost as depicted in Figure 7, three scheduling stages are incorporated in the temporal cost: (i) pre-scheduling,

(ii) during scheduling, and (iii) post-scheduling [6, 68]. The pre-scheduling stage covers the earliest start time [4–6, 79, 89, 137], earliest finish time [6, 79], estimated data transfer time [69], estimated execution time [6, 21, 69], estimated finish time [4, 21, 89], latest start time, and latest finish time [2, 68, 69, 139] parameters. However, the during-scheduling stage includes the computation time, communication time, spare time, and ready time parameters. On the other hand, the post-scheduling stage consists of the actual start time as well as actual finish time.

4.2.1.1 Pre-scheduling stage

The following are the main parameters that need to be calculated before the scheduling process to ultimately help the scheduler with scheduling decisions by estimating the temporal cost.

Earliest start time (EST)

EST is defined as the earliest time to begin task computation, regardless of the actual resource to process the task that can be decided on while scheduling [69, 79]. Nevertheless, it is impossible to exactly measure EST in a heterogeneous environment, as a specific cloud's computation time of tasks differs within each resource [69]. Every task has a period and should not be scheduled earlier than EST, and must end latest by the Finish Time [5].

The EST of each unscheduled task is described in the following equation [6, 68, 69]:

$$EST(t_{entry}) = 0 \tag{6}$$

The task t_{entry} refers to the beginning of the workflow.

$$EST(t_i) = max_{t_p \in t_i Parents} \{EST(t_p) + MET(t_p) + TT(e_{p,i})\} \tag{7}$$

where the minimum execution time of a task t_i, $MET(t_i)$, refers to the task's execution time on a resource $r_j \epsilon R$ which has the minimum $ET(t_i, r_j)$ among all the available resources. ET denotes the estimated execution time of t_i, and the t_p is the parent task of t_i, and $e_{p,i}$ refers to the edge between the parent task node to the t_i task node in DAG. TT denotes the estimated data transfer time.

Earliest finish time (EFT)

EFT for each unscheduled task is the earliest time the task's computation can finish [6, 68]. Thus, it is essential to first calculate the EST, and then calculate the EFT for each task in the workflow prior to assigning it to the fastest resource [6]. EFT can be calculated with the following equation :

$$EFT(t_i) = EST(t_i) + MET(t_i) \tag{8}$$

where the $MET(t_i)$ denotes the minimum execution time of a task t_i.

Estimated data transfer time (TT)

TT can be defined as the amount of data that needs to be transmitted along with the data latency and bandwidth between services, which can be used to estimate the time it takes to transfer the required data. For instance, $TT(e_{i,j})$ is defined as the data transfer time of the selected resource for t_i and t_j. Thus, TT represents the transfer cost of sending the required data along $e_{i,j}$ from resource r (processing task t_i) to resource n (processing task t_j) [69].

Estimated execution time (ET)

ET refers to the computation time for every task in each resource according to the scheduler's estimation after initiating a job request execution [2, 69]. ET is usually influenced by a few parameters (budget, total number of atomic tasks, tested configuration, and deadlines) [21]. Additionally, the ET for every resource differs based on task size [21]. An application is able to provide an estimated execution time according to the available metadata of user requests, unlike resource services [7].

Estimated finish time (ESHT)

ESHT refers to the estimated completion time of task computation by a particular resource. Every task is scheduled to the resource with the lowest cost and earliest ESHT [70, 89]. ESHT can be calculated as follows [4]:

$$ESHT(t_i, r_k) = EST(t_i, r_k) + w(t_i, r_k) \tag{9}$$

ESHT (t_i, r_k) represents the estimated finish time of task i in resource k, and, $w(t_i, r_k)$ represents the execution time of task i in resource k.

Latest start time (LST)

LST represents the difference between the latest finish time and estimated computation time of the task [139].

$$LST(t_i) = LFT(t_i) - D_{ti} \tag{10}$$

where $LST(t_i)$ denotes the latest start time, $LFT(t_i)$ denotes the latest finish time, and D_{ti} is the estimated task duration.

Latest finish time (LFT)

LFT represents the latest time for finishing a computation task. LFT is beneficial in calculating the required time for completing a workflow with respect to the user's determined deadline of a specific set of tasks [2, 6, 68, 137]. Thus, the LFT is an important algorithm component as it receives a workflow as the input and attempts to seek the schedule that minimizes cost, reduces the total cost of workflow and completes the task before the LFT. LFT can be calculated using the following equation [6, 68]:

$$LFT(t_{exit}) = DD \tag{11}$$

where task t_{exit} refers to the ending of the workflow, and DD is the user-defined deadline.

$$LFT(t_i) = min_{t_c \in successors\, of\, t_i} \{LFT(t_i) - ET(t_c, SS(t_i)) - TT(e_{i,j})\} \tag{12}$$

$SS(t_i)$ is defined as the resource selected for processing t_i during scheduling. ET denotes the estimated execution time and TT denotes the estimated data transfer time.

4.2.1.2 During scheduling stage

The following are the main parameters to be calculated during the scheduling process.

Computation time

The computation time represents the time that is required by the computational resources to execute the workflow tasks. The main factor for every single resource is deciding the execution cost of the task's processing time [140]. Thus, it is up to the users to select the most appropriate processing budget and time [74]. Workflow execution comprises the running time of the tasks and data transfer in and out of the computation resource [140]. Note that, communication cost and computation cost are inversely proportional to communication time and execution time, respectively [7].

Communication time

Data transfer from one computer source to another is time consuming and the duration of time is dependent on the amount of data that needs to be transferred between the corresponding tasks. Moreover, it is not dependent on the services that execute them [6, 68, 141]. The time for data transmission is dependent on the selected services and the service provider's bandwidth [69]. Nevertheless, the time for data transfer between two arbitrary tasks is constant and not dependent on the selected services [68]. Therefore, if all workflow tasks are scheduled at the same instance, the time for data transfer between them becomes zero, but the time for data transfer outside of the tasks should be still taken into consideration [6].

Spare time

Spare time (also referred as Application Spare Time) represents the time difference between the expected finish time (makespan) of the initial schedule and the deadline defined by the user for the whole workflow [46, 114]. The distribution scheme of the spare time affects the overall cost [46]. In order to guarantee the feasibility of the workflow execution when the actual execution time of task changes to a certain extent from the predicted time, the spare time is assigned to each workflow task based on its deadline. In the literature, two main approaches, including critical-path-based allocation and recursive allocation have been proposed for spare time allocation [49, 142].

Ready time

Ready time is defined as the earliest time for the first task to be executed and this task is computed based on the parent tasks [83]. The following equation is used to calculate the ready time of task t_i [19]:

$$readyTime(t_i) = max_{t_j \in p_i} endTime(t_j) \tag{13}$$

where p_i is the set of parent tasks of t_i, and endTime (t_j) denotes the time required to end the execution of task t_j (deadline).

4.2.1.3 Post scheduling stage

The workflow manager examines the workflow by consulting the repository or database containing the information linked to cost and performance records. The repository may also contain historical information regarding the execution of past services requested by SaaS clients and all their resource performance [81]. Thus, the user's input on cost and time will be considered for scheduling the next time around to ensure user satisfaction [78].A major consideration in the execution of applications that are performance-driven is effective scheduling, such as cost-driven and dynamic workflow environments like a cloud [126]. Performance estimation for resource services is derived by utilizing current performance estimation techniques, for instance, historical [21, 143] and empirical data [68], and analytical modeling [6] to predict the time taken for task execution on each discovered resource service [4, 7, 19, 21, 68, 73, 77, 83]. In addition, the costs of communication and computation for workflows are calculated from historical data of past filtered executions [4].

Start time (actual)

Every task has four components: (i) *serviceID*, (ii) *taskID*, (iii) *endTime*, and (iv) *startTime*. *serviceID* and *taskID* identify where each task is assigned to which resources. *startTime* and *endTime* represent the allocated time frame on the resource for task execution [7, 76, 79]. The entire workflow completes based on parallel and serial constraints between the start and finish times [74].

Once all tasks are scheduled, each task has a start time that is measured using the deadlines of the parent tasks in the workflow [68]. There are two concepts of tasks' start times in the scheduling algorithms. The first concept is, supposing the start time is EST, which is calculated prior to the workflow being scheduled; however, the real start time concept is calculated after scheduling the tasks [6, 69]. It is helpful to compare the start time estimated statically and the minimal spare time saved to determine rescheduling in the future [76].

Finish time (actual)

This is the time actually used to complete task execution [72, 76, 80].

4.2.2. Temporal cost parameters from profitability aspect

This section presents the results related to the relationship between the service consumers of the cost optimization approaches and their temporal cost parameters (Table 5).

Table 5: Temporal cost parameters from service consumers' point of view.

	Pre-Scheduling							**During Scheduling**					**Post Scheduling**	
Approach	**Latest Start Time (LST)**	**Latest Finish Time (LFT)**	**Earliest Start Time (EST)**	**Earliest Finish time (EFT)**	**Estimated Finish Time (EFHT)**	**Estimated Execution Time (ET)**	**Estimated data transfer time (TT)**	**Ready Time (RT)**	**Spare Time**	**Resource Allocation time**	**Computation Time**	**Communication Time**	**Start Time**	**Finish Time**
[78]						x								
[83]			x			x		x			x	x	x	
[80]				x									x	x
[3]			x	x										
[68]		x	x	x								x	x	
[17]				x								x		
[7]						x					x	x	x	
[71]			x										x	
[69]		x	x			x	x				x	x	x	
[19]				x				x			x	x	x	
[86]	x					x	x				x	x	x	
[87]		x	x	x			x			x	x	x	x	x
[44]											x	x		x

From the point of view of service consumers' profitability (Table 5), several approaches focus on measuring pre-scheduling parameters due to the significance of determining the estimated execution time, which is required to fulfil the customer's QoS attributes (i.e., deadline and makespan). Similarly, the monetary cost category, with computation time and communication time, is extensively used during the scheduling process stage. Also, several approaches measure the pre-scheduling stage. This shows that workflow application users are more concerned about the waiting duration needed for the service to be accomplished by resources. Only few approaches have considered the ready time and spare time.

Table 6 shows the results related to the relationship between the service providers of cost optimization approaches and their temporal cost parameters.

From the point of view of service providers' profitability (Table 6), most of the approaches focus on calculating the pre-scheduling parameters. Therefore, it is crucial for the service providers to identify the required execution time to schedule tasks to the available resources.

During the scheduling process stage, researchers have given more attention to determine computation time and communication time parameters. Thus, the service consumer is concerned about the waiting period required for resources to accomplish the service. Similarly, in the service consumers' category, there are not many approaches that consider the ready time and spare time.

Table 6: Temporal cost parameters from the service providers' point of view.

	Pre-Scheduling							**During Scheduling**					**Post Scheduling**	
Approach	**Latest Start Time (LST)**	**Latest Finish Time (LFT)**	**Earliest Start Time (EST)**	**Earliest Finish time (EFT)**	**Estimated Finish Time (EFHT)**	**Estimated Execution Time (ET)**	**Estimated data transfer time (TT)**	**Ready Time (RT)**	**Spare Time**	**Resource Allocation time**	**Computation Time**	**Communication Time**	**Start Time**	**Finish Time**
[70]			x		x						x			
[21]						x					x	x	x	
[72]		x	x										x	x
[74]											x		x	
[84]			x	x				x						
[76]									x		x	x	x	x
[4]			x		x							x		
[5]		x	x							x	x	x	x	
[79]			x	x							x	x	x	
[6]		x	x	x		x						x	x	
[85]				x							x	x		
[77]												x		
[69]		x	x			x	x				x	x	x	
[19]				x				x			x	x	x	
[86]	x					x	x				x	x	x	
[13]											x	x		
[31]	x		x				x				x	x	x	x
[54]	x	x	x	x			x			x	x	x	x	x
[44]											x	x		x

5. Discussion and open issues

In this section, we discuss the presented results and findings from the plethora of the current state-of-the-art cost optimization SWFS approaches and devised cost optimization aspects, parameters and experimental assessment for SWFS approaches. The following sections explain the main observations that we have extracted from our analyses.

5.1. Cost aspects

Computing environment: Several cost optimization SWFS approaches (34%) used grid computing. Note that the percentage indicates that grid computing is still an active area of research. Furthermore, a large number of cost optimization SWFS systems (35%) implemented or developed in the private cloud are used to present Software as a Service (SaaS) issues. The advantage of utilizing the SaaS cloud model for experimentation purpose is that, SaaS does not require any details about the computational infrastructure (Infrastructure as a Service) where the requests are being processed. Surprisingly, public cloud has achieved less attention (22%) from researchers compared to other environments. Moreover, we found that a small number of models (9%) focus on minimizing execution cost in the hybrid cloud. This could be due to the difficulty of calculating the total cost for these models owing to the heterogeneity of communication aspects and load balance challenges among resources (i.e. resource allocation, resource utilization and resource migration). Therefore, it is of paramount importance to focus on hybrid cloud for sharing the workflow management system's workload. In contrast, it might be possible in future to propose approaches that combine the private and public (hybrid) cloud models to offer sufficient power for processing to accomplish workflow in a specified execution timeframe.

Optimization method: Based on the frequency of optimization methods as reported in the literature, we found that heuristic (57%), market-oriented (10%), and meta-heuristic (11%) approaches have attained major attention of researchers compared to other optimization methods. Heuristic approaches have the highest potential to compute more accurate results. In contrast, market-oriented and meta-heuristic approaches are mainly used to achieve better performance compared to the fast heuristic methods, but with little compromise on execution time. Therefore, designing a hybrid approach (by integrating the features of existing heuristic and meta-heuristic search algorithms) certainly improves the scalability challenge due to concurrent processing.

Structural representation: From studying the frequency of structural representations, it can be clearly found that majority of the work (85%) have considered DAG or the modified DAG model as the structural representation. The DAG is able to handle very complex cost optimization workflow applications in grid and cloud computing systems. It shows the precedence constraints relationship between the workflow tasks. However, only few approaches (15%) that used different alternative structural representation methods have been adopted (i.e. grid broker (6%), just in time (3%), atomic task partitioning (3%), and multiplier level architecture (3%)). For future studies, there is a need to introduce a different kind of method that is applicable to large-scale data (data-intensive) for SWFS.

Scheduling technique: From analyzing the frequency of scalability aspects reported in the literature, it can be clearly found that majority of the work targeted dynamic approaches (77%). This is due to the fact that the dynamic method requires prior knowledge about the parameters. In contrast, some work (23%) has focused on the static methods for cost optimization SWFS approaches.

Workload type: The predicted (batch mode) type of workload remains a key focus (62%) of researchers in this cost aspect of SWFS. In contrast, some work (38%) has focused on considering the unpredicted (on-line mode) type of workload. This is due to the ability that batch mode offers to SWFS models by maximizing the throughput of the workload while minimizing the turnaround time (the time between task submission and task completion).

Optimization criteria: Most of the reviewed approaches (91%) have focused on multi-objective optimization, while only (9%) approaches targeted single-objective based optimization. This is mainly due to the fact that SWFS contains multiple objectives and constraints in cloud and grid computing environments.

5.2. Cost parameters

From analyzing the monetary cost parameters from the point of view of service consumers' profitability and service providers' profitability, surprisingly, few approaches have considered the estimation of execution cost parameter. Yet, estimated execution cost is one of the important challenges that requires the scheduler in handling the uncertainties of the input of SWFS algorithms. The majority of the proposed approaches emphasize cost in their approaches. This shows that there is a strong dependency between computation cost, communication cost and the total monetary cost.

During the scheduling stage, several models consider the cost of a service offered by service providers. This signifies that customers consider the cost of the service provider highly important, which must be calculated as an external cost to fulfil a service request to the service consumers. Only few models consider the cost of data storage in their approaches, which is potentially due to private resource usage instead of using a service offered by service providers. A large amount of work is to be expected to optimize data storage parameter due to recent focus on the big data challenge. Similarly, future researchers can utilize the resource elasticity (which plays a strong role in cloud) by providing an efficient resource allocation within the lowest cost.

From analyzing the temporal cost parameters according to the point of view of service consumers' and service providers' profitability, several approaches have focused on measuring pre-scheduling parameters due to the significance of determining the estimated execution time, which is required by the scheduler for handling the uncertainties of the input challenge for SWFS algorithms. At the same time, it is very important for the service providers to calculate the required execution time to schedule tasks to the available resources.

As during the scheduling process stage, the computation time and communication time parameters are extensively used. This shows that workflow application service consumers are more concerned about the time needed for the service to be accomplished by resources. This also highlights that there is a direct relationship between computation time and communication time parameters. So, in order to examine cost performance of the cost optimization SWFS algorithms, researchers should consider this relationship. From the pre-scheduling stage, there are not many approaches that consider the ready time and spare time. Thus, the scheduling approaches in the future study should adjust to the aforementioned cost optimization parameters for the execution time of workflow process model.

Regarding the response time challenge in cloud, traditionally, cloud systems aim to achieve better trade-off between performance (i.e. response time) and cost. The value of the response time (the duration of a service between calling and return) is specified in the service level agreement. In most cases, the service consumers of workflow application want fast response times regardless of cost, application owners want fast response times without spending too much money, and service providers seek to reduce the cost of running all applications within their service agreements, regardless of ownership [110]. Thus, the challenge faced by a service provider is how to minimize the cost while maintaining the resource utilization and low service response time [93, 143, 144]. SWFS approaches are required to consider achieving the application's submission time which is equal to the application execution start time. For the aforementioned reasons, cloud service providers usually use the auto-scaling mechanism to minimize the response time of customer requests to improve the user experience [52, 138].

5.3. Qualitative comparison of cost optimization SWFS approaches

In order to assess the quality of cost-optimization based SWFS approaches, we followed a qualitative methodology. The adopted methodology contains the following four main steps:

(i) formulating research questions,

(ii) identifying attributes and corresponding parameters from selected studies based on the devised research questions,

(iii) extracting the parameters' values for each defined attribute, and

(iv) constructing a comparative table.

In the first step of qualitative methodology, we formulated the following five research questions:

- What types of experimental tools have been used or adopted in the selected studies?
- How many computational resources have been utilized in the selectect studies ?
- What types of computational resources have been used to conduct the experiment?
- What types of data set have been executed?
- What is the average data set size considered in the selected studies?

After that, we identified several attributes and the corresponding parameters from selected studies based on the devised research questions. In total, six attributes were extracted: (a) name of the used tool, (b) type of tool, (c) number of computational resources, (d) type of resource (based on Amazon instance specifications as a standard [145]), (e) type of workflow application, and (f) size of workflow tasks. Note that researchers have considered a variety of computational resources, such as virtual machines, servers, and super computers, depending upon the selected tool and computational environment.

Next, we defined a varying set of parameters for each extracted attribute. For instance, regarding the type of tool attribute, two main parameters, including real-world experiment and simulator, have been used to map the obtained results of a particular study. Similarly, we defined three scales for task size attribute (i.e., small (<100 tasks), medium (100-1000 tasks), and large (>1000 tasks)). Subsequently, we developed a comparative table to map the attributes and corresponding parametrs (see Table 7). Note that the columns of the table represent the identified attributes, while rows represent a particular parameter for each selected approach.

Table 7: Qualitative comparison results of cost optimization SWFS approaches.

Ref.	Name of tool	Type of tool	No. of resources	Type of resource	Type of SWFA	Tasks Size
[3]	SwinDeW-C based on Hadoop	Real-world experiment (empirical study)	30	t2.medium, t2.large	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium
[57]	Invmod, Austrian Grid, and Water Balance Simulation Model ETH (WaSiM-ETH)	Simulator (open source) and Real-world experiment (Case study)	16	M3.large, M3.extra-large	parallel, random, and CSTEM	Medium
[68]	Developed tool in Java	Simulator	9	t2.small, t2.medium, t2.large, S3	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium-Large
[5]	TeraGrid	Batch experimentation (empirical study)	10	t2.small, t2.medium, t2.large	Lead (weather), Motif (storm-surge), Scoop (flood-plain mapping), Ncfs (time-sensitive)	Large
[69]	GridSim and DAS-3	Simulator (open source)	272	t2.small, t2.medium, t2.large	Montage, CyberShake, Epigenomics, LIGO and SIPHT.	Small-Medium-Large
[70]	CloudSim	Simulator (open source)	9	t2. micro, t2.small, t2.medium, t2.large	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small
[71]	Project scheduling problem library	Real-world experiment (Empirical study)	10	t2.small, t2.medium, M4.large	10 types of scientific WFs	Medium-Large
[72]	Fair-share scheduling policy in C++	Simulator, and real-world experiment (Case study)	6	t2.medium	Weather research and forecasting (WRF) Climate modeling workflow structure	Small-Medium-Large
[73]	ASKALON, GroundSim, and Eucalyptus middleware in Java	Simulator (open source)	5	M1.large, C1.xlarge	Wien2k, Invmod, and meteoRG	Medium
[74]	Developed tool	Real-world experiment	3	N/A	serial and parallel	Small
[75]	Developed tool in Java	Simulator	10	t2.medium	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium-Large
[7]	GridSim and GridSim Index Service (GIS)	Simulator (open source)	15	N/A	neuro-science, EMAN, protein annotation Montage	Small
[76]	Rescheduling policy	Simulator (open source)	8	N/A	Fork-Join, Laplace equation solver, FFT	Small-Medium
[19]	GridSim and GridSim Index Service (GIS)	Simulator (open source)	4	t2.small, t2.medium and service type (*Align_wap* and *Reslice)*	neuro-science workflow, Hybrid structure, protein annotation	Small
[77]	ICENI, GridSim toolkit, Mixed-Integer Non-linear programming (MINLP), NEOS Server, and SBB MINLP optimiser	Simulator	24	M3.large, M3.extra-large	N/A	Large
[78]	SwinDeW-C based on Hadoop	Simulator	8	N/A	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium-Large
[79]	Phosphorus network and CloudSigma	Simulator	3	C4.8 extra-large	N/A	Large
[80]	Developed tool	Real-world experiment	3	N/A	FFT, Fork-Join, Laplace, Random DAGs	Medium
[81]	Java and IBM ILOG CPLEX Optimizer	Real-world experiment	4	M3.2 extra-large	Montage fork-join DAG	Small
[21]	IEEE 118-node, UDDI, WSDL, GonQoS, and CFM1	Real-world experiment (Case study)	8	t2.micro	N/A	Small-Medium
[6]	Developed tool and Amazon EC2	Simulator (open source)	10	t2.small, t2.medium, t2.large	Montage, cyberShake, Inspiral, Sipht, Epigenomics	Small-Medium-Large
[83]	GridSim, and GridSim Index Service (GIS)	Simulator (open source)	15	N/A	pipeline, parallel and hybrid-*Align_wap* and *reslice*	Small

Continued on next page

Table 7 – continued from previous page

Ref.	Name of tool	Type of tool	No. of resources	Type of resource	Type of SWFA	Tasks Size
[17]	Developed tool	Real-world experiment	10	N/A	Gauss–Jordan Algorithm, LU decomposition, Laplace Transform	Small-Medium
[18]	JSwarm package, and Amazon CloudFront	Real-world experiment (Empirical study)	3	t2.small, t2.medium, t2.large	N/A	Large
[85]	Pwrake workflow system, InTrigger Kore, Gfarm distributed file system in Ruby language	Real-world experiment (Empirical study)	8	M4.extra-large, M4.10 extra-large	Montage 2MASS	Large
[4]	Amazon Elastic Compute Cloud	Real-world experiment (Empirical study)	3	t2.small, t2.large, M4. extra-large	Montage, AIRSN, CSTEM, LIGO-1 and LIGO-2, Chimera-1, Chimera-2, median filter image processing	Small-Medium
[44]	Pegasus 4.0, and Condor 7.8	Simulator (open source) and Real-world experiment (Empirical study)	20	C1.extra-large, S3	Montage, cyber-Shake, Inspiral, Sipht, Epigenomics	Small-Medium-Large
[146]	SwinDeW-C, and Hadoop	Real-world experiment (Empirical study)	8	N/A	Montage, cyber-Shake, Inspiral, Sipht, Epigenomics	Small-Medium-Large
[86]	AMS	Real-world experiment (Empirical study)	5	t2.small, t2.medium, t2.large	Stable, On-and-Off, Crowing, Bursting	Small-Medium-Large
[13]	SciCumulus	Developed tool in Java	8	M4.extra-large	DNA sequences	Small-Medium
[31]	CloudSim	Simulator (open source)	9	t2.small, t2.medium, t2.large	Montage, CyberShake, Epigenomics, LIGO and SIPHT	Small-Medium-Large
[87]	Developed tool in C++	Real-world experiment (Test cases)	15	t2.medium	N/A	small-Large
[54]	CloudSim	Simulator (open source)	9	t2.small	VM-Amazon EC2	Small-Medium-Large

Table 7 presents the comparison results of selected approaches. It can be clearly seen that researchers have mainly considered simulator based tools compared to the real-world experimentation. Availability of tools justifies this trend, since standard data set in terms of open source is easily available than real-world experimentation. Regarding the number of resources attribute, several types and specifications were utilized due to the need for heterogeneous computational resources. It is evident that majority of the cost-optimization SWFS approaches used a large size of resources due to the nature of some Scientific Workflow Application (SWFA). As for the type of SWFA, several types have been used such as Montage, CyberShake, Epigenomics, LIGO, and SIPHT [147]. Moreover, the considered applications supporting different types of tasks dependencies (e.g., process, pipeline, data distribution, data aggregation, or data redistribution). Notice that not all the approaches have considered the three scales of tasks (i.e. small, medium, large). Therefore, to effectively measure the performance of cost optimization SWFS approaches, it is important to consider these three scales of workflow tasks.

6. Conclusion

The cost optimization of Scientific Workflow Scheduling (SWFS) especially in cloud and grid computing remains an important challenge for both service consumers and service providers. The current work analyzes the cost optimization problem for SWFS in cloud and grid computing. After careful selection of the relevant papers in this field of study, we identified the cost optimization aspects of SWFS that should be considered while scheduling workflows in cloud and grid computing. We introduced two classifications (i.e. aspects and parameters) of cost optimization SWFS, which is one of the major contributions of this work. The proposed classifications aimed at providing a solid foundation for developing an economical SWFS approach that meets the demands of future workflow applications. We classified the related works according to the devised classifications and identified a correlation between cost optimization parameters and profitability of SWFS. Besides, we presented the relevent cost and time formulas, which are used to determine the overall SWFS cost considering a number of relevant cost parameters. Moreover, we provided several recommendations for developing a cost optimization scheduling approach.

From the detailed analyses, we found that majority of cost optimization SWFS approaches considered various aspects of SWFS: heuristic methods (57%), grid computing (34%), directed acyclic graph (85%), dynamic technique (77%), and predicted workload (62%). Regarding the QoS constraints, there is a variety of research focus such as privacy, response time, reliability, and security. From the parameters aspect, most of the proposed approaches have applied temporal and monetary cost parameters during various stages of SWFS. In contrast, researchers have paid more attention to profitability from the point of view of service providers than the service consumers.

It is evident from the observed trends related to SWFS that researchers are mainly focusing on large-scale data intensive applications for evaluation purpose. The other research direction could be on proposing a generic multi-objective scheduling approach with lower complexity. However, considering trade-off between many constraints can significantly increase the complexity of scheduling process. Therefore, future research should focus on proposing hybrid approaches by considering the strengths of heuristics and meta-heuristics optimization methods. Similarly, high accuracy is required in estimating the execution cost and time to reduce the level of uncertainty of required inputs to the scheduler, by proposing more optimal scheduling for the real-time applications. Furthermore, this work could lead researchers to develop more generic cost optimization scheduling approaches in cloud and grid computing environments.

Acknowledgment

This work is developed within the framework of the research project with reference RG114-12ICT, supported by University of Malaya. We would like to thank all the following researchers: Saif Ur Rehman Khan, Saeid Abolfazli, and Zohre Sanaei from University of Malaya, Malaysia, for their valuable feedback.

References

[1] W.-J. Wang, Y.-S. Chang, W.-T. Lo, Y.-K. Lee, Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments, The Journal of Supercomputing 66 (2) (2013) 1–29, ISSN 0920-8542.