Journal of Network and Computer Applications 66 (2016) 64–82

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

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

Review

Towards workflow scheduling in cloud computing: A comprehensive analysis

Mohammad Masdari, Sima ValiKardan, Zahra Shahi, Sonay Imani Azar

Computer Engineering Department, Urmia Branch, Islamic Azad University, Urmia, Iran

ARTICLE INFO

Article history:
Received 12 August 2015
Received in revised form
17 December 2015
Accepted 30 January 2016
Available online 10 February 2016

Keywords:
Cloud computing
Workflow scheduling
QoS
Metaheuristic

ABSTRACT

Workflow scheduling is one of the prominent issues in cloud computing which is aimed at complete execution of workflows by considering their QoS requirements such as deadline and budget constraints. Numerous state of the art workflow scheduling schemes have been proposed in the literature for scheduling simple and scientific workflows in the cloud computing and this paper presents a comprehensive survey and analysis of these schemes. It illuminates the objectives of scheduling schemes in the cloud computing and provides a classification of the proposed schemes based on the type of scheduling algorithm applied in each scheme. Beside, each scheme is illustrated and a complete comparison of them is presented to highlight their objectives, properties and limitations. Finally, the concluding remarks and future research directions are provided.

Contents

E-mail addresses: M.Masdari@Iaurmia.ac.ir (M. Masdari), SimaValiKardan@yahoo.com (S. ValiKardan), Shahi_Za@yahoo.com (Z. Shahi), SonayImaniAzar@gmail.com (S.I. Azar).

http://dx.doi.org/10.1016/j.jnca.2016.01.018
1084-8045/

1. Introduction

Cloud computing is a technology that utilizes the internet and central remote servers to provide scalable services for its users (Kaur et al., 2011). It uses a great amount of heterogeneous distributed resources to deliver countless different services to its users with distinctive quality of service (QoS) requirements (Wu et al., 2013). Amazon EC2, GoGrid, Google App Engine, Microsoft Azure and Aneka are some of the prominent cloud computing platforms.

Generally, clouds are classified as public clouds, private clouds, community clouds, hybrid clouds and cloud federation (Huang, 2014). A public cloud can be accessed by any subscriber (Huth and Cebula, 2011), but private clouds and their infrastructure are owned and accessed by some organizations (Huang, 2014). Also, community clouds are shared between several organizations and can be maintained by them or other service providers (Huang, 2014). Hybrid clouds deal with resources from both public and private clouds (Marcon et al., 2013). Also, due to the availability issue of the single clouds, a movement towards multi-clouds has emerged (AlZain et al., 2012) which focuses on the federation of different clouds (Jensen et al., 2011; Buyya et al., 2010).

In addition, the services provided by cloud can be classified as software (SaaS), platform (PaaS), or infrastructure (IaaS) providers (Wang et al., 2014). SaaS provider leases enterprise software as a service to customers (Wu et al., 2011) and PaaS provider presents access to the required components over the internet to develop applications (Basishtha and Boruah). Also, IaaS clouds provide infrastructures resources such as processing, storage, networks, and soon (Dillon et al., 2010; Agarwal and Jain, 2014).

Virtualization is one of the key enabling technologies of cloud computing which allows multiple Virtual Machines (VMs) to reside on a single physical machine (Pandey et al., 2010). A Virtual Machine (VM) emulates a particular computer system and executes the user issued tasks (Wang et al., 2010). By using the instantiation of the VMs, users can deploy their applications on resources with various performance and cost levels. In each physical machine or server, the VMs are managed by a software layer called hypervisor or the VM monitor which facilitates the VMs creation and isolated execution.

Workflow scheduling is one of the prominent issues in the cloud computing which tries to map the workflow tasks to the VMs based on different functional and non-functional requirements (Jayadivya and Bhanu, 2012). A workflow consists of a series of interdependent tasks which are bounded together through data or functional dependencies and these dependencies should be considered in the scheduling (Kumar, 2014). However, workflow scheduling in the cloud computing is an NP-hard optimization problem and it is difficult to achieve an optimal schedule. Because there are numerous VMs in a cloud and many user tasks should be scheduled by considering various scheduling objectives and factors. The common objective of the workflow scheduling techniques is to minimize the makespan by the proper allocation of the tasks to the virtual resources (Rahman et al., 2013; Bala and Chana, 2011). For example, a scheduling scheme may try to support the promised SLAs, the user specified deadlines and cost constraints (Kapoor and Kakkar). Also, scheduling solutions may consider factors such as resource utilization, load balancing and availability of the cloud resources and services in the scheduling decisions (Bala and Chana, 2011; Maruthanayagam and Prakasam; Motahari-Nezhad et al., 2009; Barrett et al., 2011).

The workflow scheduling problem has been widely studied in the literature. This paper presents a complete survey of the workflow scheduling schemes proposed for cloud computing in the literature. For this purpose, it first illuminates the types and objectives of workflow scheduling and then provides a classification of the proposed schemes based on the algorithm which has been used in each workflow scheduling scheme. Also, the objectives, properties and the limitations of workflow scheduling schemes are assessed in detail and a complete comparison of them is presented. Although some schemes such as (Kaur; Nallakumar, 2014; Singh and Singh, 2013) discussed the workflow scheduling problem in cloud environment, none of them have provided an in-depth investigation and comparison of the proposed workflow scheduling schemes.

The rest of this article is organized as follows: Section 2 discusses about the workflow scheduling types and objectives, Section 3 investigates the proposed workflow scheduling and compares various aspects of them. Section 4 discusses about the cloud providers and simulators, Section 5 presents the comparison and discussion of the proposed schemes, and Section 6 exhibits the concluding remarks and future research directions.

2. Types and objectives of scheduling in cloud

Generally, there are two types of workflow which are simple and scientific workflows. Figure 1 indicates a simple workflow's DAG which contains 9 tasks.

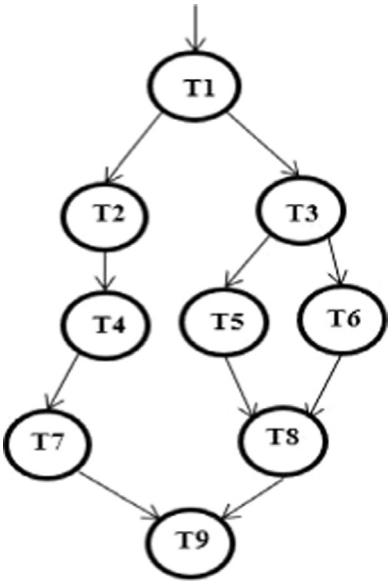

Fig. 1. A simple workflow.

Also, there are numerous scientific Workflows such as Montage, LIGO, Cyber Shake, SIPHT and Epigenomics applied in astronomy, earthquake researches and so on. Figure 2 indicates these scientific workflows which involve complex data of different sizes and need higher processing power. By providing unbounded on-demand virtual resources, the cloud computing paradigm can effectively handle the scientific workflows.

Generally, scheduling schemes can be categorized as follows (Chawla and Bhonsle, 2012):

- User level: the scheduler deals with problems raised by the service provision between the cloud providers and users.
- System level: deals with the resource management within the cloud data centers.
- Static scheduling: tasks arrive simultaneously and the available resource schedules are updated after each task is scheduled (Patel and Mer). This scheduling assumes a precise knowledge of the timing information about tasks which is difficult to obtain, but it incurs less runtime overhead. The Opportunistic Load Balancing or OLB is a static scheduling algorithm.

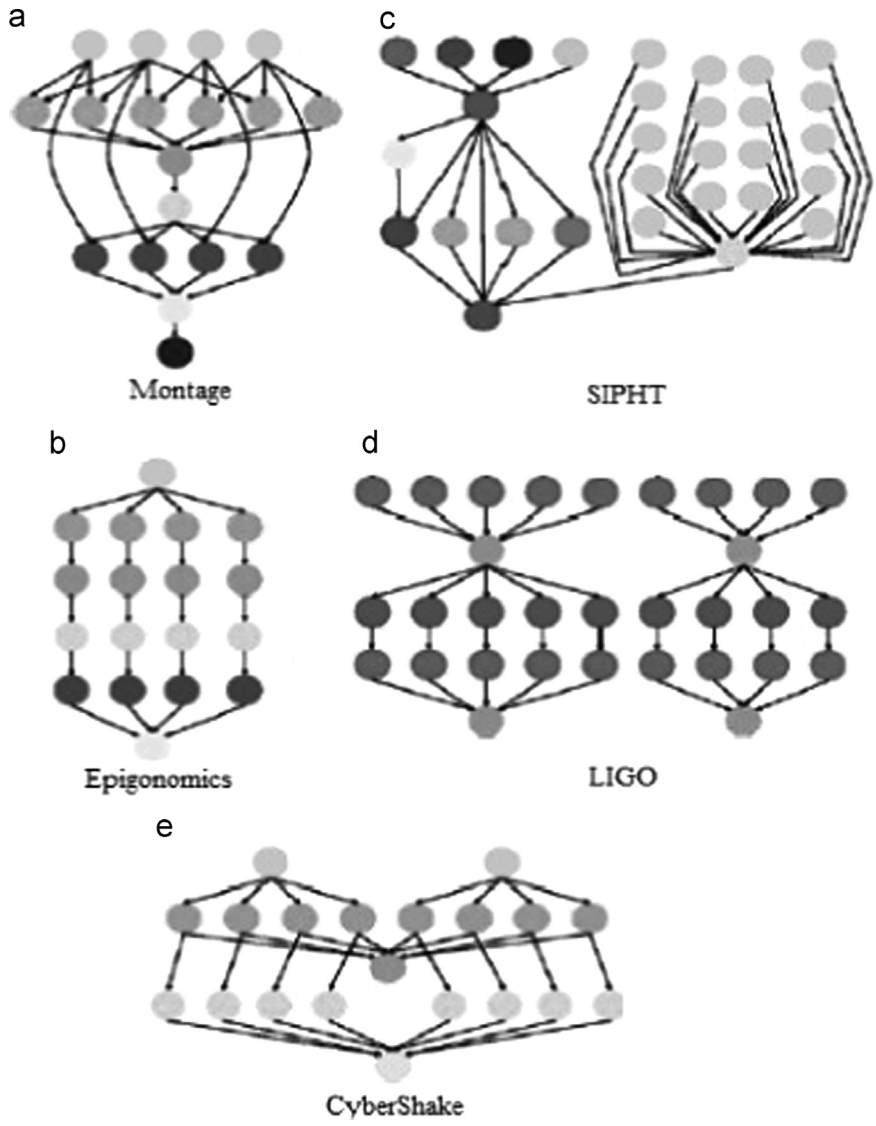

Fig. 2. Structure of five realistic scientific workflows.

- Dynamic scheduling: in which the information about the task arrivals is not known at runtime (Chawla and Bhonsle, 2012). Although it better adapts to the timing changes during the task execution, it incurs runtime overheads. Some dynamic scheduling algorithms are as follows: Earliest Deadline First (EDF) and Least Laxity First (LLF).
- Centralized scheduling: a master processor unit is used for the collection of tasks, which sends them to other processing units (Kaleeswaran et al., 1963).
- Distributed scheduling: it applies the local schedulers to manage the requests and maintain the job states. It has less efficiency in comparison to centralized scheduling (Tiwari).
- Preemptive scheduling: it allows each task to be interrupted during the execution and a task can be migrated to another resource (Patel and Bhoi, 2013).
- Non-preemptive scheduling: only when a task completes, its resource can be taken away (Xhafa and Abraham, 2010).
- Online scheduling: each task is scheduled only once and the scheduling result cannot be changed. It is suitable for the cases in which the arrival rate is low (Nagadevi et al., 2013) and is used in K-Percent Best (KPB) and Switching Algorithm.
- Offline scheduling: tasks are not mapped to the resources as they enter, but they are collected and are examined for mapping at the prescheduled times. Offline scheduling is called Batch mode heuristics (Vijayalakshmi and Vasudevan, 2015). Min-min and Max-min are offline algorithms.
- Task-level scheduling: deals with the optimization of the Task-to-VM assignment in the local cloud data centers where the overall running cost of the workflow should be minimized by considering QoS (Wu et al., 2013; Hong and Potkonjak, 1997).
- Service-level scheduling: deals with the Task-to-Service assignment where the workflow tasks are mapped based on their functional and non-functional QoS requirements.

The common objectives for workflow scheduling schemes are as follows:

- Budget: is the cost which the consumers pay for the usage of the cloud resources (Mary, 2014).
- Deadline: deadline is the time limit for the execution of the workflow (Rodriguez and Buyya, 2014) and its supporting is an important QoS requirements.
- Reliability: it is the probability that the task can be completed successfully. For this purpose, techniques such as active replications and backup/restart schemes corresponding to the resource and time redundancy may be applied in the scheduling (Zhao et al., 2013).
- Availability: by the proper workflow scheduling, tasks are executed faster and the execution is terminated quickly. This improves the availability of the cloud resource (Sajid and Raza, 2013).
- Minimizing the makespan: the makespan of a workflow is the time at which the last workflow task finishes its execution (Lopez et al., 2006; Liu et al., 2010).
- Supporting Service Level Agreement (SLA): SLA is a document that has various considerations of the service consumers and providers. These include the explanations of QoS delivery performance guarantees (Alkhanak et al., 2015).
- Security: attackers may misuse some cloud features and components to launch cloud specific attacks (Tao et al., 2009). A secure scheduler produces a safe scheduling to mitigate the effects of the security attacks.
- Load Balancing: a scheduler should optimize the resource usage to avoid the overload of any cloud resources(Anju Baby, 2013).

3. Analysis of the workflow scheduling schemes

Figure 3 first classifies the workflow scheduling schemes into heuristic and metaheuristic solutions, and then further classifies them based on type of the scheduling algorithms. Table 1 presents the acronyms which will be applied in the next sections of this article.

Often in these schemes, a workflow is specified by a DAG (Directed Acyclic Graph) or weighed DAG, in which each computational task T_i is indicated by a vertex. Also, each data or control dependency between the tasks is indicated by a weighed directed edge E_{ij} which may be computed using different factors in each scheduling scheme (Wu et al., 2013). Each directed edge E_{ij} states that T_i is the parent task of T_j and T_j and can only be executed once all its parent tasks have been completed.

3.1. Metaheuristic-based scheduling

To generate optimal schedules, metaheuristic scheduling schemes may apply algorithms such as PSO, ACO, SA (Yaseen and Al-Slamy, 2008).

3.1.1. PSO-based workflow scheduling

Particle Swarm Optimization or PSO is a population based stochastic optimization algorithm developed by Eberhart and Kennedy in 1995 (Nallakumar, 2014; Rahman et al., 2013). In the PSO-based workflow scheduling, the dimension of the particles is the number of tasks, and each position of a particle indicates a mapping between the VMs and tasks. Also, each particle corresponds to a candidate solution and has its own position in the space, with a fitness value corresponding to the position. Moreover, each particle has a velocity to determine the speed and direction which it flies. The PSO-based workflow scheduling schemes often initially create a random swarm of particles, but some schemes try to produce a better initial swarm. Moreover, some scheduling schemes only use the basic PSO algorithm, but others improve the PSO (Verma and Kaushal, 2015).

Pandey et al. (2010) present a PSO-based solution which takes both computation cost and data transmission cost into account. First, this algorithm computes the mapping of all the overall costs of computing the workflow application. To validate the dependencies between the tasks, the algorithm assigns the ready tasks to resources according to the mapping given by PSO. After dispatching the tasks to the resources for execution, the scheduler waits for the polling time. Depending on the number of tasks completed, the ready list is updated. Then, they update the average values of the communication between the resources based on the current network load. When the communication costs change they recompute the PSO mappings. Also, when the remote resource management systems are not able to allocate a task to the resources due to the resource unavailability, the recomputation of the PSO makes the heuristic dynamically balance other tasks' mappings. With the recomputed PSO mappings, they assign the ready tasks to the resources and these steps are continued until all the workflow tasks are scheduled.

Huang et al. (2013) in models the mapping between the tasks and resources and achieves a tunable fitness function on the basis of which a workflow schedule may be selected for the minimal cost or minimal makespan. In addition, a heuristics is proposed to address a bottleneck problem and attain a smaller makespan. In this scheme, first the position and velocity of all the particles are randomly initialized. If the iteration stopping criterion is not met, the algorithm repeatedly performs the following operations: for each particle, it first calculates its fitness value using one of the fitness functions, and then updates its local best positions. Then, it calculates the global best position among all the particles and

Fig. 3. Classification of the workflow scheduling schemes in the cloud environment.

updates the velocity and position of all the particles. Finally, when the stopping condition is met, the global best is the optimal mapping.

Chen and Zhang (2012) propose a set-based PSO (S-PSO) workflow scheduling scheme which minimizes both the cost and makespan and increases the reliability by addressing the QoS

constraints. The S-PSO extends the PSO in the discrete space and is suitable for the cloud workflow scheduling because the service instances available in the cloud can be considered as a resource set. In this scheme, the allocation of the service instances is considered as the selection problem from a set of service instances, the set-based representation scheme in the S-PSO is appropriate for the cloud workflow scheduling problem. They define penalty-based fitness functions to consider several QoS constraints and integrate the S-PSO with seven heuristics.

A Bi-Criteria Priority based PSO (BPSO) (Verma and Kaushal, 2014) is proposed by Verma et al. to reduce the execution cost and time under the deadline and budget constraints. Each workflow task is the assigned priority using bottom level, and priorities are used to initialize the PSO. All particles have fitness values indicating their performances and velocities which direct the flight of the particles. Also, the experience of the neighboring particles is taken into account during the optimization. A particle status on the search space is specified by its velocity and position, which are updated in every generation. In the BPSO, the workflow tasks are executed based on their priority computed using the bottom level which is the same as the one defined in HEFT.

Jianfang et al. (2014) uses discrete PSO and consider the security issues and try to reduce the completion time and cost. They use the On-Demand method to charge the computing power by the hour and has three stages: initialization, execution and the end. At the initialization phase the initial population is randomly generated according to the cloud workflow coding scheme. At the execution phase, local adjustment of the particle position can be made to ensure that it is a feasible solution, and three kinds of QoS attribute values of the particles can be calculated; a decision matrix is constructed according to the QoS attribute values, combined with the users' preferences, the objective function of multi QoS perception is aggregated into a single objective function, making possible that the particles are selected by a numerical comparison among each other. At the end phase the optimal scheduling solution is achieved.

The Local Minima Jump PSO is proposed by Chitra et al. (2014) that attempts to optimize the execution time, cost, load balancing and makespan. This algorithm first initializes random swarm particles and each particle's fitness function is packet delivery ratio. If fitness function is less than pbest, then the value of fitness function is assigned to pbest and compares each particle's fitness function with the global best particle. If fitness function is less than gbest, then gbest takes the value of fitness function. When the value of gbest does not change much with each iteration, it has been found that convergence is poorer as gbest gets struck with the local minima problem. Then, revises the velocity for each particle and moves each particle to a new position.

Table 2 compares the properties of the PSO-based workflow scheduling algorithms. Table 3 presents the objectives of the PSO-based schemes.

3.1.2. GA-based workflow scheduling

Some of GA-based workflow scheduling schemes apply the basic GA algorithm, while others modify it to achieve better results. Also, most of them generate better initial population to achieve better results (Madić et al., 2013; Nair and Sooda, 2010).

A bi-objective scheduling scheme called DWSGA is proposed in Aryan and Delavar which considers the completion time and distribution of the workload on resources. At first, by using bidirectional tasks prioritization, it makes a good initial population. Then, it gets the most appropriate possible solution by optimizing the makespan and provide good distribution of the workload on resources. It uses some conversant methods such as making initial population by ordering the tasks, based on a bi-directional priority method and other goal-oriented operations that lead the algorithm to fulfill the objectives by speeding up the good solution finding process. Moreover, it controls the search using a mutation method that reassigns resources based on the workload to consider the most effective task. In GA-based scheduling schemes, a task to the VM mapping is indicated by a single gene in the chromosome and a valid chromosome contains a sequence of genes. The order of the genes exhibits the schedule execution order on the chosen resources. A feasible solution to the scheduling problem must maintain the precedence constraints between the tasks specified in the workflow (Barrett et al., 2011).

Verma and Kaushal (2013) present BCHGA to optimize the execution cost and data transfer cost with a budget constraint. Each workflow's task is the assigned priority using bottom level (b-level) and top level (t-level). BCHGA, at first, calculates the b-level and t-level of all workflow tasks and creates the initial population which for all individuals the priority of each task is set equal to the total of its b-level. Then, all the tasks are assigned to the available VMs with their priority. While termination criteria are not met, BCHGA evaluates the fitness of the individual in the

Table 1
Acronyms and abbreviations.

Abbreviation	Definition
VM	Virtual Machine
VMM	VM Monitor
PSO	Particle Swarm Optimization
ACO	Ant Colony Optimization
SA	Simulated Annealing
GA	Genetic Algorithm
CSO	Cat Swarm Optimization
HEFT	Heterogeneous Earliest Finish Time
SLA	Service Level Agreement
QoS	Quality of Service
DAG	Directed Acyclic Graph

Table 2
Comparison of the PSO-based workflow scheduling schemes.

Scheme	Type of workflow	Type of cloud	Simulator/environment	Type of PSO algorithm	Discrete/continuous
(Pandey et al., 2010)	Scientific workflow	–	JSwarm	Standard PSO	Discrete
(Rodriguez and Buyya, 2014)	Scientific workflow	–	CloudSim	Standard PSO	Continuous
(Huang et al., 2013)	Simple workflow	Hybrid Cloud	CloudSim	Standard PSO	Continuous
(Verma and Kaushal, 2014)	Scientific workflow	–	JAVA	Bi-Criteria PSO	Continuous
(Jianfang et al., 2014)	Simple workflow	–	CloudSim	Discrete PSO	Discrete
(Chitra et al., 2014)	Simple workflow	Hybrid Cloud	–	Modified PSO	Continuous
(Chen and Zhang, 2012)	Simple workflow	–	Amazon EC2	Set-based PSO	Discrete
(Pragaladan and Maheswari)	Scientific workflow	Multi Cloud	–	Standard PSO	Continuous
(Wu et al., 2010)	Simple workflow	–	Amazon EC2	Discrete PSO	Discrete
(Verma and Kaushal, 2015)	Simple workflow	–	CloudSim	Bi-Criteria PSO	Continuous
(Sridhar and Babu, 2015)	Simple workflow	–	CloudSim	Hybrid PSO	Continuous

Table 3
Objectives of PSO-based workflow scheduling schemes.

Scheme	Makespan	Load balancing	Deadline constrained	Budget constrained	Storage	Bandwidth	Memory requirement	QoS Support (SLA)
(Pandey et al., 2010)	–	✓	–	–	–	–	–	–
(Rodriguez and Buyya, 2014)	_	_	✓	–	–	–	–	–
(Huang et al., 2013)	✓	✓	–	–	–	–	–	✓
(Verma and Kaushal, 2014)	–	–	✓	✓	–	–	–	✓
(Jianfang et al., 2014)	–	✓	–	–	–	–	–	✓
(Chitra et al., 2014)	✓	✓	–	–	–	–	–	✓
(Chen and Zhang, 2012)	✓	–	✓	✓	–	–	–	✓
(Pragaladan and Maheswari)	–	–	✓	✓	–	–	–	–
(Wu et al., 2010)	✓	✓	✓	–	–	–	–	✓
(Verma and Kaushal, 2015)	–	–	✓	–	–	–	–	–
(Sridhar and Babu, 2015)	_	_	✓	_	_	_	_	_

Table 4
Comparison of the metaheuristic-based workflow scheduling schemes

Scheme	Type of workflow	Type of cloud	Simulator/environment	Type of scheduling	Type Of algorithm
(Huang, 2014)	Scientific workflow	–	CloudSim	–	Genetic Algorithm
(Barrett et al., 2011)	Scientific workflow	–	CloudSim	–	Genetic Algorithm
(Zhou and Huang, 2013)	Simple workflow	–	CloudSim	–	Ant Colony Optimization
(Singh and Singh)	Simple workflow	Private cloud	CloudSim	–	Ant Colony Optimization
(Verma and Kaushal, 2013)	Scientific workflow	–	JAVA	–	Genetic Algorithm
(Jian et al., 2013)	Simple workflow	–	–	–	Simulated Annealing
(Aryan and Delavar)	–	–	–	–	Genetic Algorithm
(Bilgaiyan et al., 2014)	Simple workflow	–	–	Dynamic	Cat Swarm Optimization
(Ponniselvi and Seetha)	Scientific workflow	Multi Cloud	–	–	ESEMO
(Gogulan et al., 2012)	Simple workflow	Private cloud	ETC Matrix	–	Ant Colony Optimization

population, afterwards it applies the selection operator to select the parent. Then, it applies the crossover operator on the selected parent using crossover probability to create the children and applies the mutation operator on the newly created children. Then, it validates each child according to the fitness function. Finally, it adds the valid child to create the new population.

Barrett et al. (2011) propose a scheduler which supports the user defined QoS like the cost and makespan. It utilizes a genetic algorithm to generate optimal schedules and applies a Markov Decision Process (MDP) which chooses from them. By using an MDP this scheme is capable of observing temporal dynamics and tuning schedule selection accordingly. At first, users submit their workflows with non-functional QoS constraints. Then, the average execution time that a given task takes to run on a resource is calculated. A number of solvers with ranging configurations are instantiated to produce schedules of varying cost and makespan. From these schedules an agent utilizing an MDP computes the optimal schedule based on the current state of the cloud environment. The scheduling plan is performed on the cloud by the Executor module which monitors the tasks and returns the results to the user interface. Once a processing schedule has been completed on the cloud, the QoS monitor returns the actual cost and makespan incurred by the schedule.

Table 4 represents the comparison of the metaheuristic-based workflow scheduling schemes in the cloud environment.

3.1.3. SA-based workflow scheduling

Simulated annealing or SA is a random search method for the global optimization problems that is able to handle local optima problems. It imitates the metals recrystallization in the process of annealing (Madić et al., 2013).

Normally, cloud customers focus on the time and cost factors when choosing the cloud resources providers. For this purpose, in Jian et al. (2013) apply the SA to minimize the workflow scheduling cost under time-constraints and load balancing the resources. They consider both the time-cost of the tasks execution and time-cost of the data transmission between different tasks. Scheduling by the SA costs less time because of the better quality of the optional solution selected by Metropolis criterion. This scheme first initializes the solution space and the objective function. The value of the objective function is equal to the total time-cost. Then, it initializes the temperature and its parameters. The number of the iterations on each given temperature and the temperature decay function f are given at the time. When the temperature reaches T_k they can calculate the objective function. In each iterative process, it can be decided whether that current solution is global optimal solution or not. If it cannot meet the condition, then it steps into the next operation: it calculates the probability of the selection and generates a random number and compares it with P. If $k < P$, they set the current solution as the optimal solution. Otherwise the optimal solution remains unchanged. The temperature is cooled down by the temperature decay function and these steps are repeated. If the algorithm reaches the maximum number of iterations, then it stops. Also, the result of this scheme is compared with the PSO.

3.1.4. CSO -based workflow scheduling

Cat Swarm Optimization (CSO) has been introduced by Chu and Tsai in 2007 and operates in seeking mode and tracing mode. In

seeking mode, cats do not move and stay in a certain position and sense for the next best move, but in tracing mode, cats move to their next position with some velocity which indicates how cats chase their target (Bilgaiyan et al., 2014).

A CSO based solution is presented by Bilgaiyan et al. (2014), which considers both the data transmission cost between the dependent resources and execution cost. It reduces iterations to obtain scheduling scheme and provides fair load balancing on the available resources. By using two operation modes, this scheme reduces the energy wastage in random movement, and updates the positions of cats to attain the optimal solutions. It uses an initial population of N cats among which some of them are in seeking mode and the others are in tracing mode. In seeking mode the majority of cats search the global space while being in a resting state by the intelligent position updating. This scheme applies two factors called SMP or Seeking Memory Pool and CDC or Count of Dimension to Change. The SMP exhibits the number of copies to be made for each cat and the CDC specifies the number of allocations to be altered in a single copy. Tracing mode indicates the cats that are in a fast moving mode and search the local space by moving towards the next best position. In this scheme, each cat indicates a task resource mapping updated as per mode that the cat is in. Assessing the fitness value of cats leads to finding the mapping which has the minimum cost. Moreover, in each iteration, a new set of cats is chosen to be in tracing mode. The final solution, provides the best mapping which has the minimum cost.

3.1.5. ACO-based workflow scheduling

Ant Colony Optimization (ACO) is a paradigm for designing Meta heuristic algorithms for the combinatorial optimization problems (Yaseen and Al-Slamy, 2008).

The ACO algorithm is used by Singh and Singh to optimize the execution cost of those workflow tasks whose sub deadlines are missed at the private cloud and shifted to the public cloud to complete their execution within the sub deadlines. It finds a schedule at the private cloud which satisfies the user defined deadlines. However, when deadlines are missed at the private cloud then the sub deadlines are assigned to the workflow tasks and they migrate to the public cloud. At the public clouds, the ACO calculates the pheromone value of each VM based on the execution cost and selects those VMs from the public clouds which are cost effective and can execute the workflow application within the sub deadlines.

Zhou and Huang (2013) propose an ACO-based solution to reduce the time to find the computing resource in the cloud. In this scheme, the ants are divided into the Forward-Ants and the Back-Ants, where Forward-Ants are used to find the available VMs in the cloud and Back-Ants are produced when the Forward-Ant finds the resources. The Back-Ant returns in the original way and leaves the pheromone of the resource to the node. In this scheme first each node's pheromone is initiated.

Then, the batch of jobs is submitted to the Master node, which selects the first job. The Master node starts a timer and sends the forward-ants, and each of them randomly selects the next node. When the forward-ant comes into a node, it will be sent into the N_s of Forward-Ant. The pre- execution time will be calculated if the pre-execution time is less thanET_0, the node i is the available node, otherwise. it is not. If the node i is the available node, a Back-Ant is produced which will get the N_s of the Forward-Ant, the pheromone of the node, and the pre-execution time. The pheromone of the node in the back path is updated. If a node is not the available node and the value of its pheromone does not reach the threshold τ_0, the Forward-Ant selects the next node randomly. If a node is not the available node, but its pheromone reaches the threshold τ_0, the forward-ant selects the next node. Before the timer of the Master node reaches zero, if the Master node receives the Back-Ant, it will assign the tasks to the k available nodes which have the least pre-execution time; otherwise, the Master node does not assign tasks. When the tasks are completed or failed, the pheromone of the available node will be updated. The tasks that are not completed will be assigned to another node by the Master node. These processes are done for the next job.

Multiple Pheromone Algorithm (MPA) is proposed by Gogulan et al. (2012) which is based on the ACO algorithm to mitigate the makespan, the task completion time and resource utilization. In this scheme, ants use different pheromone value for each task and apply them to select tasks for different resources. In this solution, first all pheromone values and parameters are initialized. Then, the M ants are initialized to select the N tasks and each ant builds a solution to the M resources. In each iteration, ants are randomly selected to build a constructive direction. After the M ants map a solution to the M resources, pheromone value is updated by a local pheromone updating rule. If an ant selects a new task, the pheromone is increased to improve the efficient allocation. At the end of the iteration, the pheromone value is updated by using a global pheromone value to find the best-so-far solution. The MPA algorithm operates based on the reliability, cost and makespan constraints.

Table 5 indicates the objectives of metaheuristic-based workflow scheduling schemes in the cloud computing environment.

3.1.6. Enhanced superior element multitude optimization (ESEMO) algorithm

Ponniselvi and Seetha present a solution which minimizes the workflow execution cost and considers the deadline constraints. This scheme calculates the data transfer cost between the data centers to minimize the cost of the execution in multicloud environments. Also, it considers basic IaaS cloud issues such as heterogeneity, multi-cloud, and cloud provider of the resources. At first, this scheme initializes resources for each resource, and then calculates the fitness value. When the fitness value is better than the best fitness value in the list, then the current value is set as the new optimal best. Then, the resource capacity is calculated using the capacity update equation and resources position is updated using the position update equation.

3.2. Heuristic workflow scheduling

3.2.1. Deadline constraint scheduling algorithm

Singh and Singh (2013) propose a score based deadline constrained scheme that reduces the execution time under manageable cost by the user specified deadline. It also reduces the failure rate of the workflow and applies the concept of score to represent the capabilities of the resources. It **s**ubmits a list of the workflow tasks and gets the available virtual resources from the data center. Afterwards, it imposes the user deadline on the entire workflow by sub deadlines of the tasks. Then, it obtains the final scores of the VMs from the component minimum sub-scores and picks the minimum score VM from that satisfies the threshold of the task which is based on its length. If the selected VM can run the task with a deadline, then the task is assigned to it; otherwise, the next minimum score VM is selected. This process is repeated until the mapping of all the tasks to the VMs is created.

3.2.2. HEFT-based scheduling

The HEFT algorithm operates based on the earliest time in the execution from the available resources and ignores other factors affecting the execution time of the tasks.

Bala proposes an improved HEFT algorithm which is able to reduce the turnaround time by selecting resources based on the multiple factors such as inter-node bandwidth between the VM nodes and the RAM, bandwidth, and the storage factors. This

Table 5
Objectives of the metaheuristic-based workflow scheduling schemes.

Scheme	Makespan	Load balancing	Deadline constrained	Budget constrained	Storage	Bandwidth	Memory requirement	QoS Support (SLA)
(Huang, 2014)	–	–	✓	✓	–	–	–	✓
(Barrett et al., 2011)	✓	–	✓	✓	✓	–	✓	✓
(Zhou and Huang, 2013)	–	✓	–	–	–	–	–	–
(Singh and Singh)	–	–	✓	–	–	–	–	–
(Verma and Kaushal, 2013)	✓	–	✓	✓	–	–	–	✓
(Jian et al., 2013)	–	✓	–	–	–	–	–	–
(Aryan and Delavar)	–	–	–	–	–	–	–	–
(Bilgaiyan et al., 2014)	–	–	–	–	✓	–	–	✓
(Ponniselvi and Seetha)	–	–	✓	–	–	–	–	–
(Gogulan et al., 2012)	✓	–	–	–	–	–	–	✓

algorithm first computes the average execution time for each task, and then calculates the average data transfer time between the tasks and their successors. If the score of the request is less than the score of the available, the compound rank value for each task is computed. Then tasks in a scheduling list are sorted by decreasing the order of the task rank value.

Lin and Lu (2011) propose the SHEFT algorithm which is an extension of the HEFT algorithm and it is aimed at optimizing the workflow execution time and enables the resources to scale elastically at runtime. This scheme models a cloud environment by partitioning all resources into a number of clusters which have the same computing capability, same network communication, and the same inter-cluster and intra cluster data transfer rate. Then, they formalize a scientific workflow as a weight DAG in which the communication cost is determined by the weighed of the edge in the graph, and the task computation cost is determined by the weight of the vertex in the graph.

3.2.3. Priority Impact Scheduling Algorithm

Priority Impact Scheduling Algorithm or PISA is proposed by Wu et al. (2012) who considered fairness and scheduled multiple workflows by considering priority and the task weights. This algorithm is able to increase the scheduling success rate significantly and it takes into account the user priority, and extends the existing FIFO algorithm. The simplest scheduling strategy is to increase expenditure on workflows that are more important and to prevent expenditure on less important workflows. Importance may be implied by proximity to deadline, current demand of the anticipated output or whether the application is in a test or production phase. This scheme defines the importance as workflow priority and task weight.

3.2.4. Hybrid Cloud Optimized Cost scheduling algorithm

The HCOC scheduling algorithm is proposed by Bittencourt and Madeira (2011) to minimize the costs and makespan and reduce the number of schedules with the makespan higher than the deadline. Moreover, it decides to execute a workflow task in the private or public cloud's resource. In this algorithm, the tasks are selected to reschedule, and then resources are picked from the public cloud. While the former decides the tasks that can have their execution time reduced by using more powerful resources from the public cloud, the latter determines the performance and costs of the new schedule. By taking the prices and performance of the resources into account, this algorithm decides how many resources to request from the public cloud. Also, it considers the number of the clusters of the tasks being rescheduled to select these resources and their cost, number of cores, and performance. By considering the number of resource cores, the task dependency costs are suppressed, since the communication time between the tasks in the same processor is set to null.

3.2.5. CTC scheduling algorithm

Liu et al. (2010) present a scheme called Compromised-Time-Cost or CTC which supports instance-intensive and cost-constrained workflows by compromising the execution time and cost with the user input. The algorithm presents a graph of the time-cost relationship for the users to choose an acceptable compromise before the next round of scheduling begins. If no user input is provided, then default scheduling is applied. Afterwards, the algorithm focuses on sharing, conflicting and competition of the services caused by the multiple concurrent instances running on the cloud platform. It focuses on the background load to estimate the execution time more accurately, which is considered when calculating the task execution time on each specific server. In addition, to adapt to a load change, the server may reschedule the tasks. Moreover, to handle the execution failures, the uncompleted tasks are rescheduled with a higher priority in the next scheduling round.

3.2.6. Deadline and budget distribution-based cost-time optimization (DBD-CTO)

DBD-CTO (Verma and Kaushal, 2012) tries to reduce the execution cost and time and addresses the user defined deadline and budget constraints. This scheme categorizes the workflow tasks as synchronization tasks and simple tasks; synchronization tasks have a task which has more than one parent or child task. Moreover, the workflow is partitioned in such a way that a set of simple tasks are executed sequentially between two synchronization tasks. Then, it estimates the minimum execution time and cost for each task from the available set of services. Afterwards, it calculates the total expected completion time/cost by adding the data processing time/cost and minimum execution time/cost. The workflow runs when the calculated values are less than the deadline and budget provided by the user. Then, it distributes the user's overall deadline and budget into every task partition proportion to their minimum processing time and processing cost. Finally, all the service lists are sorted in decreasing the order of their cost, and a service should be chosen to execute some task so that processing the cost and execution time can be less than the partition's deadline and budget value.

3.2.7. Time and cost optimization for the hybrid clouds (TCHC) algorithm

Kumar and Ravichandran (2012) present a scheme called TCHC which decreases the execution time and cost of the multiple workflows scheduling. In the dependent workflow scheduling, the switching between the private and public clouds leads to the increased execution time and cost. To reduce this problem, TCHC buffers the resource in the local resource pool which may help when there is a change in the on demand resource price and reduces the request cost. It dynamically schedules multiple

Table 6
Comparison of the Heuristic-based workflow scheduling algorithms' properties.

Scheme	Type of workflow	Type of cloud	Simulator/environment	Type of algorithm
(Marcon et al., 2013)	Scientific workflow	Hybrid cloud	Amazon EC2	Workflow scheduling with security constraint
(Jayadivya and Bhanu, 2012)	Multiple Workflow	–	–	QoS based scheduling
(Liu et al., 2010)	Simple workflow	–	–	CTC
(Zhang)	Scientific workflow	Hybrid cloud	IBM EC2	IOO
(Singh and Singh, 2013)	Scientific workflow	–	CloudSim	Deadline constraint scheduling
(Bala)	Simple workflow	–	CloudSim	HEFT
(Verma and Kaushal, 2012)	Simple workflow	–	JAVA	DBD-CTO
(Kumar and Ravichandran, 2012)	Simple workflow	Hybrid Cloud	–	TCHC
(Lin and Lu, 2011)	Scientific workflow	–	–	SHEFT
(Wu et al., 2012)	Simple workflow	–	–	PISA
(Xu et al., 2009)	–	–	–	MQMW
(Bittencourt and Madeira, 2011)	Scientific workflow	Hybrid Cloud	Amazon EC2	HCOC
(Lu et al., 2014)	Simple workflow	–	–	Concurrent level based workflow scheduling
(Zhu et al., 2014)	Scientific workflow	–	–	Scientific workflow scheduling under deadline constraint
(Hoenisch et al., 2013)	Simple workflow	–	–	Resource-efficient workflow scheduling
(Guo et al., 2015)	Simple workflow	Private Cloud	CloudSim	FCBWTS
(Poola et al., 2014)	Scientific workflow	-	CloudSim	Workflow scheduling using spot instances
(Bessai et al., 2012)	Scientific Workflow	–	–	Bi-criteria workflow tasks allocation and scheduling
(Fard et al., 2013)	Scientific workflow	Multi Cloud	–	Dynamic workflow scheduling Mechanism for commercial multicloud environment
(Abrishami et al., 2013)	Scientific workflow	_	–	Deadline-constrained workflow scheduling
(Calheiros and Buyya, 2014)	Scientific workflow	Public Cloud	CloudSim	Meeting deadlines of scientific workflows
(Lee et al., 2015)	Simple workflow	–	_	Resource-efficient workflow scheduling
(Zeng et al., 2015)	Simple workflow	–	_	SABA
(Malawski et al., 2015)	Scientific workflow	Multi Cloud	Amazon EC2	Scheduling multilevel deadline-constrained scientific workflows

workflows, verifies the dependency range, and evaluates a minimized execution time and cost. This algorithm makes an initial schedule that considers only the private resources and checks if they meet the desired deadline. If the deadline is not met, it requests resource from the public cloud based on the performance, cost, and number of tasks to be scheduled in the public cloud.

Table 6 provides the comparison of the Heuristic-based algorithm properties.

3.2.8. *Multiple QoS constrained scheduling strategy of multi-workflows (MQMW)*

Xu et al. (2009) present MQMW in which tries to satisfy the users' QoS requirements and minimizes the makespan and cost of the workflows. In this scheme, users should submit their workflow with the QoS requirements. Then, the system allocates appropriate services for processing the workflow tasks and schedules them on the services by the QoS requirements. The system consists of Preprocessor, Scheduler and Executor components. The Preprocessor computes the available service number, cost and other information. Then, it submits the ready tasks to the Scheduler queue, a sorted set containing tasks from different users waiting to be scheduled. Then, the Scheduler re-computes the attributes of the queued tasks and then resorts all of them. The Executor selects the best service to sequentially execute the tasks in the queue. When a task finishes, the Executor notifies the Preprocessor which the task belongs to of the completion status. First, a task with a minimum available service number and the tasks which belong to the workflow with the minimum time surplus and cost surplus and the task with minimum covariance should be scheduled. So the task should be scheduled at first. Otherwise, users should pay more or the time would increase more.

3.2.9. *QoS based workflow scheduling*

The authors in Jayadivya and Bhanu (2012) propose a QoS based Scheduling (QWS) to schedule the workflows based on the user defined QoS parameters like Deadline, Reliability and Cost. This scheme considers two types of servers which are storage server and computational server. QWS process is designed by using three modules which are Preprocessor module (PM), Scheduler module (SM) and Executor module (EM) with a rescheduling if required. In this scheme, users submit their workflows with QoS. Then, the PM discovers the services required for those tasks and generates their DAG and divides the tasks based on the computation services and storage services. The SM allocates services to these tasks based on the QoS parameters and the attributes of the services. After mapping, the EM sends the tasks to the mapped servers and checks for the result, of these tasks. If the EM gets the successful result, then it activates all the tasks which are dependent on these tasks. If it fails, then the SM reschedule them.

Table 7 presents the objectives of the heuristic-based workflow scheduling schemes designed for the cloud environment. Also, Figure 4 indicates the percentage of metaheuristic workflow scheduling schemes which are presented in the literature.

3.2.10. *Scientific workflow scheduling under the deadline constraint*

The authors in Zhu et al. (2014) propose a two-step workflow scheduling algorithm called HiWAE to reduce the cloud overhead under a user-specified execution time limit. This scheme considers the resource availability of both computer nodes and network links and increases the system throughput. In the HiWAE, the completion time of the workflow can be specified as a QoS requirement. In this scheme, first the modules are topologically sorted into different layers to determine the module mapping order starting from the first layer. Each module is assigned with a certain priority value based on its computational complexity and mapped to the node that yields the lowest partial end-to-end delay as the execution time from the starting module to the current one. This mapping process is repeated until a convergence point is reached. The main goal of the second step is to improve the resource utilization rate by minimizing the overhead of the VM's startup and shutdown time and its idle time.

3.2.11. *IaaS cloud partial critical paths*

Abrishami et al. (2013) propose two workflow scheduling algorithms called IC-PCP and IC-PCPD2. These algorithms are based on the PCP scheduling algorithm which consists of Deadline

Table 7
Objectives of the Heuristic-based workflow scheduling schemes.

Scheme	Makespan	Load balancing	Deadline constrained	Budget constrained	Storage	Bandwidth	Memory requirement	QoS Support (SLA)
(Marcon et al., 2013)	–	✓	✓	–	–	✓	–	✓
(Jayadivya and Bhanu, 2012)	✓	–	✓	–	–	–	–	✓
(Liu et al., 2010)	✓	✓	✓	–	–	–	–	–
(Zhang)	✓	–	–	–	–	–	–	–
(Singh and Singh, 2013)	✓	–	✓	–	–	–	–	✓
(Bala)	✓	–	–	–	✓	✓	✓	✓
(Verma and Kaushal, 2012)	–	–	✓	✓	–	–	–	✓
(Kumar and Ravichandran, 2012)	✓	–	✓	–	✓	–	–	–
(Lin and Lu, 2011)	✓	–	–	–	–	–	–	–
(Wu et al., 2012)	–	–	–	–	–	–	–	–
(Xu et al., 2009)	✓	–	–	–	–	–	–	✓
(Bittencourt and Madeira, 2011)	✓	–	✓	✓	–	✓	–	–
(Lu et al., 2014)	–	–	✓	–	–	–	–	✓
(Zhu et al., 2014)	–	–	–	–	–	✓	–	–
(Hoenisch et al., 2013)	–	–	✓	–	–	–	–	–
(Guo et al., 2015)	–	✓	✓	–	–	–	–	✓
(Poola et al., 2014)	✓	–	✓	–	–	_	–	✓
(Bessai et al., 2012)	✓	–	–	–	–	–	–	_
(Fard et al., 2013)	✓	–	–	–	–	–	–	–
(Abrishami et al., 2013)	✓	–	✓	✓	–	–	–	–
(Calheiros and Buyya, 2014)	_	–	✓	✓	–	–	–	–
(Lee et al., 2015)	–	–	–	–	–	–	–	_
(Zeng et al., 2015)	–	–	–	–	✓	✓	✓	_
(Malawski et al., 2015)	–	–	–	–	–	–	–	_

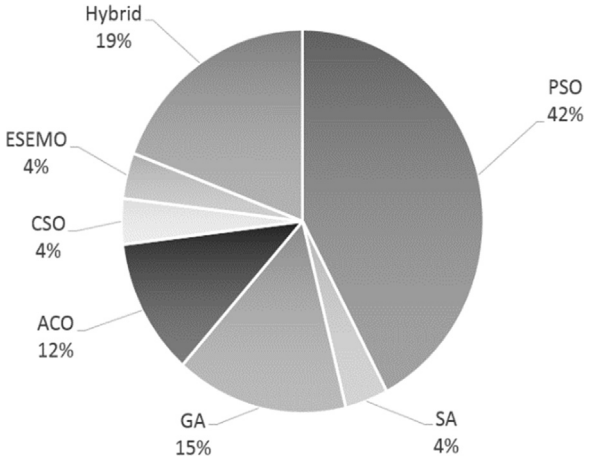

Fig. 4. Percentage of the metaheuristic schemes.

Distribution and Planning phases. The first phase of this scheme finds the critical path of the workflow and calls the path assigning algorithm to distribute the deadline among the critical nodes. After this distribution, each critical task has a sub deadline which can be used to compute a sub deadline for all of its predecessors. The PCP carries out the same procedure for all the tasks. Finally, the Planning algorithm schedules the workflow by assigning each task to the cheapest service which meets its sub deadline. The ICPCP is a one-phase algorithm which instead of assigning sub deadlines to the tasks of a partial critical path, it tries to actually schedule them by finding an instance of a computation service which can execute the entire path before its latest finish time. IC-PCPD2 is a two-phase algorithm which replaces three path assigning policies with a single new policy to adapt to the new pricing model. Also, in the planning phase, it utilizes the remaining time of the existing instances of the computation services for task scheduling, and if it fails, then it start to launch a new instance to execute the task before its sub deadline.

3.2.12. *Just-in-time and adaptive scheduling heuristic Algorithm*

Poola et al. (2014) schedules tasks on the cloud resources using spot and on-demand instances pricing models to reduce the execution cost and meet the workflow deadline. It tolerates the premature termination of the spot instances and performance variations of the cloud resources. This scheme evaluates the critical path for every ready task and computes the slack time, which is the time difference between the deadline and the critical path time. When the slack time reduces because of the failures or performance variations in the system, this algorithm switches to the on-demand instances and applies a bidding strategy and check pointing to reduce the cost to meet the deadline.

3.2.13. *Bi-criteria workflow scheduling*

Bessai et al. (2012) propose three bi-criteria workflow scheduling schemes on the distributed Cloud resources, which considers the overall execution time and the cost incurred by using a set of resources. The first algorithm aims to minimize the execution and communication costs using several allocation strategies. For each obtained solution the completion time is computed. The second algorithm attempts to minimize the execution and communication times, and for each obtained solution the overall computation cost is computed. The third algorithm is called cost time-based approach, based on the obtained Pareto solutions by the two first algorithms. Thus, by using the solutions produced by the cost and time-based approaches only the non-dominated solutions are selected.

3.3. *Hybrid metaheuristic and heuristic scheduling*

This section discusses about the schemes which combine the metaheuristic algorithms with the heuristic scheduling algorithms such as Best Fit, Round Robin and Max-Min, to solve the workflow scheduling problem.

Table 8
Comparison of the hybrid workflow scheduling algorithms' properties.

Scheme	Type of workflow	Type of cloud	Simulator/environment	Continuous/discrete	Combined algorithm
(Wu et al., 2013)	Scientific workflow	–	Amazon EC2	–	Market-oriented hierarchical scheduling
(Yassa et al., 2013)	Scientific workflow	–	Amazon EC2	Discrete	Particle Swarm Optimization + HEFT
(Kaur and Ghumman)	Simple workflow	–	CloudSim	–	Ant Colony Optimization + Max-min
(Delavar and Aryan, 2012)	Simple workflow	–	MATLAB	–	Genetic Algorithm + Best Fit + Round Robin
(Delavar and Aryan, 2014)	Scientific workflow	–	–	–	Genetic Algorithm + Best Fit + Round Robin

3.3.1. Hybrid improved max min ant algorithm

Yassa et al. (2013) combine the Ant colony and Max-min algorithm and focuses on total processing time and cost. They try to balance the total system load and minimize the total makespan. In Max-min algorithm, large tasks have higher priority than smaller ones. This scheme reduces the waiting time of the short jobs by assigning large tasks to the slower resources. Thus, small tasks are executed concurrently on the fastest resource to finish large number of tasks during finalizing at least one large task on the slower resource. In max min if it cannot execute the tasks concurrently, makespan becomes larger. To overcome such limitations, a new modification is applied for the Max-min scheduling algorithm. This approach improves the total cost and time factor and is only concerned with the number of the resources and tasks. Improved max min provides an optimal solution during the preliminary stage. Furthermore, during the starting stage of ant algorithm, the searching speed is very slow for the lacking of pheromones, but after the pheromones reach a certain degree, the speed of the optimal solution improves quickly. The main aim of the dynamic combination between the max min algorithm and ant colony algorithm is that the max min algorithm can be utilized to get advantages in the initial stage and obtains the optimal solution by the ant algorithm in last stages.

3.3.2. A goal-oriented workflow scheduling

Delavar and Aryan (2012) propose a hybrid metaheuristic method based on Genetic Algorithm named GMSW to find a suitable solution to map the requests on the resources which have different communication costs. The GMSW reduces the number of the GA operation iterations by making an optimized initial population and considers the reliability and suitable distribution of the workload on resources. This scheme obtains the solutions by using two evaluation functions, one of which measures the priority of each task in the DAG for bi-directional form based on their influence on the tasks, and the other one which evaluates the value of the produced solutions. This scheme tries to get an improved method rapidly considering the makespan, reliability and distribution of the workload on resources. It sorts the tasks by a bi-orientation priority method and addresses their horizontal and vertical influences. Then, it creates a good initial population by using Best-Fit and Round Robin algorithms and a bi-directional task prioritization in an unbalanced-structured workflow. This scheme speeds up a good solution finding process and leads the search by a special mutation method that reassigns the resources based on the workload, and failure frequency and considers the most effective task.

3.3.3. A hybrid heuristic algorithm for workflow scheduling

Delavar and Aryan (2014) propose a solution to optimize the workflow makespan, load balancing and speed-up ratio. It combines the Genetic Algorithm with the Best Fit and Round Robin algorithms. It tries to decrease the number of the GA operation iteration with starting the algorithm by an optimized initial population. At first, this algorithm makes tasks prioritization in the complex graph and considers their impact on others. Then, it merges the Best-Fit and Round Robin algorithms to make an optimal initial population. In this scheme, to make a chromosome, each task is mapped to a selected resource from a list of available virtual resources and a set of multiple possible solutions or chromosomes are referred to as a population. For this purpose, to build the initial population, after the tasks are sorted by priority, they are placed in the first row of the genes in the chromosome. Then, for each task, a resource is selected with minimum running time from the virtual resource list and this is repeated for all genes as Best-Fit. Thus, in first chromosome for each gene, the fittest resource is selected from the first place in the virtual list, but for the second chromosome, the best resource is found from the second place in the virtual list. The fittest candidate-resource will be searched from the next point, after the last chromosome is started from the last place in the virtual list and so on, like the Round-Robin method, but for the resource selection. Moreover, the priority of the tasks in the graph topology are computed and a list of task orderings is provided by descending to make the first row of the chromosomes. Two parents and some of their genes are selected randomly and two other solutions are created. A chromosome and one of its genes are selected randomly, and at the amount of a resource is selected randomly from the virtual list of the resources. The selected resource is replaced with the selected gene if its failure rate is better than the last candidate resource.

Table 8 indicates a Comparison of the properties of the hybrid workflow scheduling algorithms.

3.3.4. Multi-objective workflow scheduling

Yassa et al. (2013), propose a multi-objective scheme which uses hybrid PSO and considers the heterogeneity and the marketization of the resources to optimize the makespan, cost and energy. It applies the Dynamic Voltage and Frequency Scaling technique and operates processors in different voltage supply levels. But a trade-off between the quality of the schedules and energy is made. It produces a set of non-dominated solutions and the users can select a schedule that meets their QoS requirements. This algorithm begins by initializing the positions and velocities of the particles. To obtain the position of a particle, the voltage and frequency of each resource is randomly initialized. Then the HEFT algorithm is applied several times to generate an efficient solution to minimize the makespan. Initially, the velocity and the best position for each particle pBest are attributed as the particle itself. After all these initializations, the new velocity and position of each particle are calculated after selecting the best overall position in the external archive and eventually performing a mutation, then the particle is evaluated and its corresponding pBest is updated. The external archive is updated after each iteration and once the termination condition is reached, the external archive containing the Pareto front is returned.

Table 9
Objectives of hybrid workflow scheduling schemes.

Scheme	Makespan	Load balancing	Deadline constrained	Budget constrained	Storage	Bandwidth	Memory requirement	QoS Support (SLA)
(Wu et al., 2013)	✓	–	–	–	–	–	✓	✓
(Yassa et al., 2013)	✓	–	–	–	–	–	–	✓
(Kaur and Ghumman)	✓	✓	–	–	–	–	–	–
(Delavar and Aryan, 2012)	✓	–	–	–	–	–	–	✓
(Delavar and Aryan, 2014)	✓	✓	–	–	–	–	–	✓

3.4. Task and workflow scheduling

Some scheduling schemes, such as Partly Dependent Task (PDT) (Shengjun et al., 2012) are able to perform both the task and workflow scheduling.

Shengjun et al. (2012) define a new scheduling model for Partly Dependent Tasks (PDTs) which merges workflow and independent tasks together. It supports the QoS constrains and minimizes the total cost. Also, it uses the basic idea of the ACO to assign tasks, and treats the PDTs as a whole colony. To get a better allocation method and evaluate the finish time of the PDTs, they divide the PDTs into several colony-sets. Before the division, they arrange the independent tasks by sorting the workflow tasks by the execution order. In this scheme, the workflow in the PDTs only has one input and one output. After division, it initializes the number of the ants and iterations to start the ACO-based scheduling, then presents the solution for each ant. When the iteration is not over, it calculates the probability distribution of the solution and updates the solutions of each ant. After the iterations, it selects the best solution among the all solutions provided by the ants and binds the PDTs to the VMs with the best solution. Table 9 indicates the objectives of the hybrid workflow scheduling algorithms designed for the cloud. As indicated in this table, each scheduling scheme only tries to achieve a subset of the objectives outlined in the previous sections.

4. Discussion

This section provides a complete discussion and analysis of the workflow scheduling schemes, about the following items:

- The cloud simulators and providers.
- Applied Algorithms.
- Scheduling objectives.

4.1. Simulation of the workflow scheduling

Various cloud simulators and providers which have been utilized in the workflow scheduling schemes.

4.1.1. CloudSim

CloudSim is a popular cloud simulator that models and simulates the infrastructures containing the Data Centers, Datacenter Broker class, Cloudlet class, users, user workloads, and pricing models, VMs and resource provisioning policies (Calheiros et al., 2011). Some of the features of this Simulator are as follows:

- Availability of a virtualization engine that aids the creation and management of multiple, Independent, Co-hosted virtualized services on a data center node.
- Flexibility in switching between the space-shared.
- Time-shared allocation of processing cores to the virtualized services.

Developing new application provisioning algorithms for the Cloud computing would be sped up due to these features of the CloudSim (Calheiros et al., 2011). Layers of the CloudSim are as follows (Kaur and Ghumman):

- CIS (Cloud Information Service): supplies the database level match-making services and maps the user requests to the best suitable cloud providers.
- Data center Broker: helps for intermediating between the users and service providers and depending on the QoS requirements, the broker deploys the service tasks.
- VM Scheduler: illustrates the policies such as space-shared and time-shared, requested to allocate processing power to all VMs.
- VM Allocation Policy: selects an available host in a datacenter for a VM deployment.

4.1.2. Eucalyptus

Eucalyptus is an open source software framework for the cloud computing that performs what is generally referred to as Infrastructure as a Service (IaaS) (Nurmi et al., 2009; Amiry et al., 2012). Eucalyptus allows the users to run and control the overall VM instances deployed over a variety of physical resources like the cloud. Eucalyptus has been practically used, and it allows the users that are aware of the existing Grid systems to explore new cloud functionality while maintaining access to the existing, familiar application development software and Grid middleware. Eucalyptus presents several profits by addressing a variety of subjects with the cloud such as: the VM instance scheduling, the VM and user data storage, the cloud computing administrative interfaces, construction of the virtual networks, defining and execution of several level agreements and the cloud computing user interfaces (Igugu and Biltoria, 2011).

4.1.3. EC2

EC2 is relied on the Linux based and Xen (3), and various O/S images, Amazon Machine Images (AMIs), can be supported (Evangelinos and Hill, 2008; Gai et al., 2015). The Amazon Elastic Compute Cloud (Amazon EC2) service is the largest provider of the public cloud services and is particularly helpful for running the large scale simulation as parallel processes on a computing cluster (Diallo). It is a web service that supplies resizable compute capacity in the cloud. It makes web-scale computing easier for the developers. Amazon EC2's simple web service interface is used to take and configure the capacity with minimal friction. It provides a complete control of the computing resources. Also, Amazon EC2 reduces the required time to obtain and boot new server instances to minutes, allowing to quickly scale capacity, both up and down, as the requirements change. It enables the developers with tools to

build failure resilient applications and isolate them from the common failure (Instances, 2008; Zhao et al., 2015).

4.1.4. IBM RC2

IBM Research has produced a cloud computing platform called Research Compute Cloud (RC2) to be used by the worldwide IBM Research community. The Research Compute Cloud (RC2) is an infrastructure-as-a-service cloud built by leveraging the existing IT infrastructure of the IBM Research division. Its goals are two categories: make a shared infrastructure for the daily use by the research population, and a living lab to experiment with the new cloud technologies (Ryu et al., 2010).

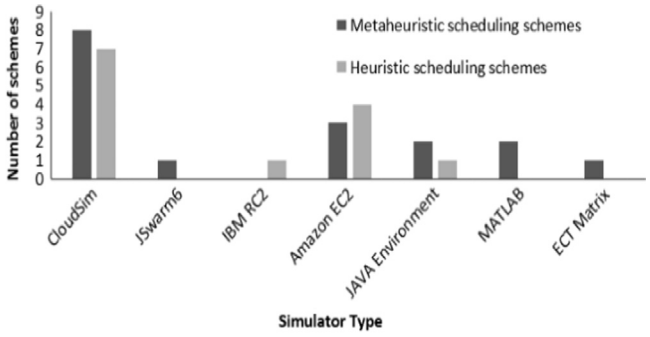

Fig. 5. Number of simulators used by workflow scheduling schemes.

4.1.5. Simulation analysis

Figure 5 exhibits the number of schemes which have applied each simulator or cloud provider in both heuristic and metaheuristic workflow scheduling schemes. Also, as it is clear from this figure, CloudSim simulator is the most popular simulator in the scheduling schemes. Also, Amazon EC2 is the most applied IaaS provider in both heuristic and metaheuristic workflow scheduling solutions. By using these simulator software and cloud providers, each scheduling scheme has evaluated various factors affecting its operations and objectives. Table 10 indicates the factors which have been evaluated in the simulation of the studied workflow scheduling schemes.

Often workflow scheduling schemes consider the real execution time for n tasks and m VMs in a matrix where each cell represents the exact amount of the time that takes a VM to execute some task. The input of a workflow scheduling algorithm includes the workflow DAG and information about the target cloud, including the VMs processing capacity, the VMs network bandwidth and other factors applied in the scheduling decisions. However, based on the simulation parameters such as the number of workflow tasks, the number of VMs and physical machines, these matrixes may become very large and often most scheme does not specify their content and values.

4.2. Applied Algorithms

In the previous sections, the workflow scheduling schemes are classified based on the algorithm type. Figure 6 indicates the

Table 10
Factors evaluated in the simulation scenarios of workflow scheduling schemes.

Scheme	
(Pandey et al., 2010)	Computation cost/total data size, computation cost /range of computation cost of compute resources, total cost of computation /number of iteration
(Huang et al., 2013)	Cost/cost weight value
(Jianfang et al., 2014)	Security utility values/number of iteration, completion/number of iteration, cost/number of iteration load balancing deviation/number of tasks
(Sridhar and Babu, 2015)	Ratio of successful execution/number of tasks
(Huang, 2014)	Implementation cost/algebra operation, implementation time/algebra operation
(Barrett et al., 2011)	Total cost/data size, total cost/time
(Zhou and Huang, 2013)	Time/number of tasks
(Verma and Kaushal, 2013)	NSL(Normalized Schedule length)/Budget
(Jian et al., 2013)	Time-cost/number of tasks, time-cost/number of iteration
(Bilgaiyan et al., 2014)	Total cost/iteration of obtaining the cost, total cost/size of data
(Ponniselvi and Seetha)	Total execution cost/number of resources, total execution time/number of resources
(Marcon et al., 2013)	Security/number of virtual infrastructures, acceptance ratio/cloud load
(Liu et al., 2010)	Execution cost/deadline, execution time/execution cost
(Zhang et al.,)	Workload/experiment time
(Singh and Singh, 2013)	Execution cost/number of cloudlets, execution time/number of cloudlets, failure rate/number of iterations
(Verma and Kaushal, 2012)	Execution time/budget
(Wu et al., 2012)	Success rate/services
(Xu et al., 2009)	Success rate/number of concurrent workflows, mean execution time/number of concurrent workflows, mean execution cost/number of concurrent workflows
(Lu et al., 2014)	Different with DBL or deadline Min-Cost, CLWS stratifies/number of tasks
(Zhu et al., 2014)	Utilization rate/ different scheduling algorithms, EED(the execution time from the starting module to the current one)/different scheduling algorithms
(Hoenisch et al., 2013)	Amount of virtual machines/time/ amount of parallel workflow requests
(Guo et al., 2015)	Performance for each algorithm/number of tasks, performance for each algorithm/ number of processing units, performance for each algorithm/ccr(communication to computation ratio)
(Poola et al., 2014)	Number of task failures/deadline, execution cost/ volatility of spot market
(Bessai et al., 2012)	Ratio/number of tasks
(Fard et al., 2013)	Coverage/problem size/balance situation, execution time/problem size/balance situation
(Abrishami et al., 2013)	Normalized cost/deadline
(Calheiros and Buyya, 2014)	Normalized average execution time/different application sizes
(Lee et al., 2015)	Resource usage reduction/makespan increase, effective reduction/workflow application, makespan/makespan delay limit, number of resources/makespan delay limit, normalized makespan/makespan delay limit
(Malawski et al., 2015)	Runtime/time limit ratio/time, optimization time/time, execution cost/deadline
(Wu et al., 2013)	Optimization ratio of makespan/number of tasks, makespan/different scheduling algorithms, optimization ratio of cost/number of tasks, CPU Time/Number of Tasks, CPU Time/different scheduling algorithms
(Yassa et al., 2013)	Energy/makespan/cost
(Kaur, Ghumman)	Total processing time/number of cloudlets, total processing cost/number of tasks, total processing cost/number of cloudlets

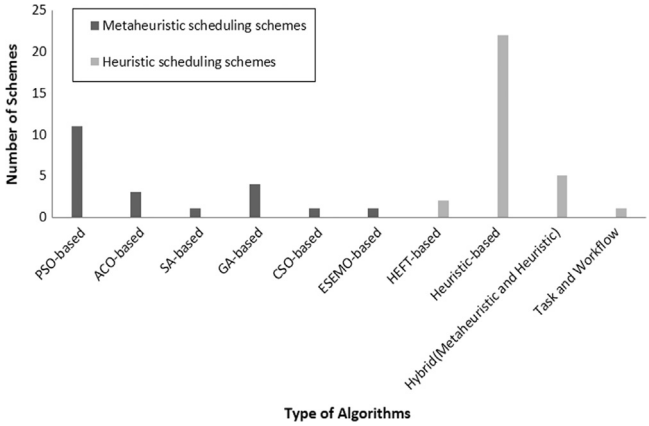

Fig. 6. Algorithms applied in the cloud workflow scheduling schemes.

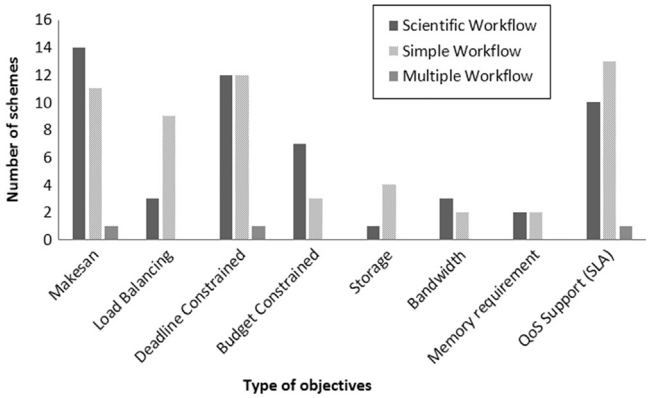

Fig. 8. Objectives and scheduling factors considered in workflow scheduling schemes.

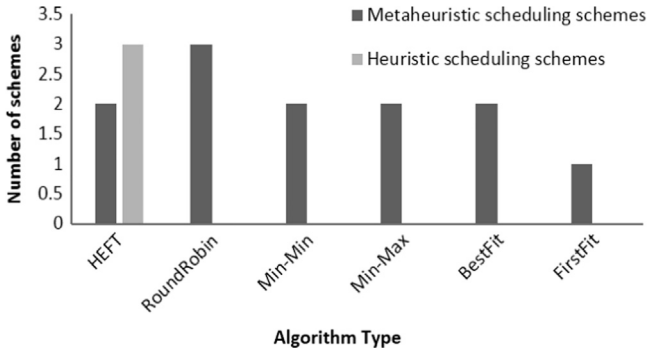

Fig. 7. Heuristic algorithms applied in the cloud workflow scheduling schemes.

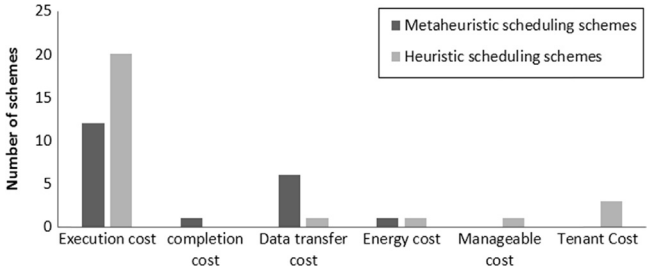

Fig. 9. Cost factors considered in workflow scheduling schemes.

number of the heuristic and metaheuristic workflow scheduling schemes which have designed based on the each algorithm. As indicated in this figure, in metaheuristic workflow scheduling, the PSO algorithm, because of its fast convergence time is mostly used. Also, in the heuristic scheduling algorithms HEFT algorithm is the mostly used for workflow scheduling. Sometimes, this algorithm is applied in combination with other scheduling algorithms and even some schemes try to improve and enhance its capabilities.

To achieve better scheduling results, the metaheuristic workflow scheduling schemes may perform one of the following operations:

- Combining multiple metaheuristic algorithms with each other.
- Combining metaheuristic algorithms with some heuristic algorithm such as HEFT.
- Using improved versions of the metaheuristic algorithms or proposing new concepts or changes in these algorithms.

Often, these are performed to generate better initial population or to solve some of the inherent problems of existing metaheuristic algorithms such as preventing local optima problem. Figure 7 indicates the number of schemes which have utilized the heuristic workflow scheduling algorithms solely or in combination with other scheduling methods.

4.3. Scheduling objectives

The main objectives of the workflow scheduling schemes are previously listed and described briefly in Section 2.1. Figure 8 indicates the number of heuristic and metaheuristic workflow scheduling schemes which have considered each objective. It indicates that the makespan, supporting the SLAs and deadline constraints, are the factors which have been focused by numerous scheduling schemes.

In metaheuristic workflow scheduling schemes, the pursued objectives are often considered in the fitness function of the metaheuristic algorithm. For this purpose, when multiple objective is applied for scheduling, different coefficients are assigned for each objective that can tune the effect of each objective on the overall workflow scheduling. Also, for example in PSO-based schemes, the factors which affect the scheduling objectives are integrated in the various parameters that adjust the movement and velocity of each particle.

4.3.1. Cost factors

Figure 9 provides various cost-related factors considered in both heuristic and metaheuristic schemes.

As indicated in this figure, only a limited number of schemes have considered the energy factor. Regarding the importance of the green cloud computing and cost of energy, more researches in the context of energy-aware workflow scheduling schemes should be conducted (Chopra and Singh, 2013; Rius et al., 2013).

Table 11 presents more elaborate information about the cost factors considered in the scheduling schemes. In this table, application of the following costs factors are analyzed for each studied scheme:

- Execution cost.
- Completion cost.
- Communication cost.
- Data transfer cost.

Table 11
Cost factors applied in the cloud workflow scheduling schemes.

Scheme	Execution cost	Completion cost	Communication cost	Data transfer cost	Energy cost	Manageable cost	Tenant Cost
(Wu et al., 2013)	✓	–	–	–	–	–	–
(Huang, 2014)	✓	–	–	–	–	–	–
(Marcon et al., 2013)	–	–	–	–	–	–	✓
(Pandey et al., 2010)	✓	–	–	✓	–	–	–
(Jayadivya and Bhanu, 2012)	✓	–	–	–	–	–	–
(Barrett et al., 2011)	✓	–	✓	✓	–	–	–
(Rodriguez and Buyya, 2014)	✓	–	–	–	–	–	–
(Huang et al., 2013)	✓	–	–	–	–	–	–
(Bittencourt and Madeira, 2011)	✓	–	✓	–	–	–	–
(Verma and Kaushal, 2014)	✓	–	–	–	–	–	–
(Chitra et al., 2014)	✓	–	–	–	–	–	–
(Pragaladan and Maheswari)	✓	–	–	✓	–	–	–
(Wu et al., 2010)	✓	–	✓	✓	–	–	–
(Verma and Kaushal, 2015)	✓	–	–	–	–	–	–
(Yassa et al., 2013)	✓	–	–	–	✓	–	–
(Verma and Kaushal, 2013)	✓	–	–	✓	–	–	–
(Jian et al., 2013)	✓	–	–	✓	–	–	–
(Singh and Singh)	✓	–	–	–	–	–	–
(Bilgaiyan et al., 2014)	✓	–	–	✓	–	–	–
(Gogulan et al., 2012)	–	✓	–	–	–	–	–
(Ponniselvi and Seetha)	✓	–	–	–	–	–	–
(Singh and Singh, 2013)	✓	–	–	–	–	✓	–
(Bala)	–	–	✓	–	–	–	–
(Liu et al., 2010)	✓	–	–	–	–	–	–
(Xu et al., 2009)	✓	–	–	–	–	–	–
(Verma and Kaushal, 2012)	✓	–	–	–	–	–	–
(Kumar and Ravichandran, 2012)	✓	–	–	–	–	–	–
(Lu et al., 2014)	✓	–	–	–	–	–	–
(Hoenisch et al., 2013)	✓	–	–	–	–	–	–
(Shengjun et al., 2012)	✓	–	–	–	–	–	–
(Poola et al., 2014)	✓	–	–	–	–	–	–
(Bessai et al., 2012)	✓	–	✓	–	–	–	–
(Fard et al., 2013)	–	–	–	–	–	–	✓
(Abrishami et al., 2013)	✓	–	–	–	–	–	–
(Calheiros and Buyya, 2014)	✓	–	–	–	–	–	–
(Lee et al., 2015)	–	–	–	–	✓	–	–
(Zeng et al., 2015)	–	–	–	–	–	–	✓
(Malawski et al., 2015)	✓	–	–	–	–	–	–

- Energy cost.
- Manageable cost.
- Tenant Cost.
- Type of scheduling.

From these items, communication cost indicates the cost incurred for the data transfer between the VMs which execute the workflow tasks, and the data transfer cost indicates the cost of the data transfer between the VM and the storage components in the data intensive applications (Parsa and Entezari-Maleki, 2009).

4.3.2. *Time factors*

Considering the importance of the time factors which are used in the scheduling process, Table 12 presents more elaborate information of these factors considered in the workflow scheduling schemes presented for the cloud computing. In this table, application of the following time factors are analyzed in in each scheduling scheme:

- Execution time.
- Communication time.
- Response time.
- Completion time.
- Data Transfer time.

5. Conclusion

A workflow is a logical sequence of the dependent tasks. Numerous studies have been conducted about the workflow scheduling in the cloud computing which utilize heuristics and metaheuristics algorithms to obtain the approximated solutions.

This paper provided a comprehensive analysis of the workflow scheduling schemes proposed for the cloud computing environment. These schemes are classified based on the type of the algorithm utilized in the workflow scheduling and their various objectives and properties are analyzed and compared. Also, various factors which are considered in each scheduling scheme are highlighted, and the limitations and advantages of the scheduling schemes are specified.

One of the issues which have less been addressed in most workflow scheduling schemes is the concept of the secure scheduling. Considering the ever increasing security attacks against the cloud environment, various attacks have been presented in the literature which target the cloud schedulers and other cloud components such as the VMs. As a result, in the future research and studies, the security and trust-related concept should be integrated in the cloud scheduling to differentiate the workflow requests issued by the attackers and the trusted cloud users. Thus, regarding the security vulnerabilities of the cloud environment, cloud scheduler should be

Table 12
Time factors considered in the cloud workflow scheduling schemes.

Scheme	Execution time	Communication time	Response time	Completion time	Data Transfer time
(Wu et al., 2013)	✓	–	–	–	–
(Huang, 2014)	✓	–	–	–	–
(Pandey et al., 2010)	–	–	–	–	–
(Barrett et al., 2011)	–	–	–	✓	✓
(Rodriguez and Buyya, 2014)	✓	–	–	–	–
(Huang et al., 2013)	–	–	–	✓	–
(Liu et al., 2010)	✓	✓	–	✓	–
(Bittencourt and Madeira, 2011)	✓	–	–	–	–
(Chen and Zhang, 2012)	–	–	–	–	–
(Verma and Kaushal, 2014)	✓	–	–	–	–
(Jianfang et al., 2014)	–	–	–	✓	–
(Chitra et al., 2014)	✓	–	–	–	–
(Pragaladan and Maheswari)	–	–	–	–	–
(Wu et al., 2010)	✓	–	–	–	–
(Verma and Kaushal, 2015)	✓	–	–	–	–
(Sridhar and Babu, 2015)	✓	–	–	–	–
(Yassa et al., 2013)	✓	✓		–	–
(Verma and Kaushal, 2013)	✓	–	–	–	–
(Jian et al., 2013)	✓	–	–	–	–
(Zhou and Huang, 2013)	✓	–	–	–	–
(Singh and Singh)	–	–	–	–	–
(Bilgaiyan et al., 2014)	–	–	–	–	–
(Gogulan et al., 2012)	✓	–	–	–	–
(Kaur and Ghumman)	✓	–	✓	–	–
(Delavar and Aryan, 2012)	–	–	✓	–	–
(Delavar and Aryan, 2014)	–	–	–	✓	–
(Ponniselvi and Seetha)	✓	–	–	–	–
(Zhang)	–	–	–	–	–
(Singh and Singh, 2013)	✓	–	–	–	–
(Bala)	✓	✓	✓	–	–
(Lin and Lu, 2011)	✓	–	–	–	–
(Wu et al., 2012)	–	–	–	–	–
(Verma and Kaushal, 2012)	✓	–	–	–	–
(Kumar and Ravichandran, 2012)	✓	–	–	–	–
(Lu et al., 2014)	✓	–	–	–	–
(Zhu et al., 2014)	✓	–	–	✓	–
(Hoenisch et al., 2013)	✓	–	–	–	–
(Shengjun et al., 2012)	✓	–	–	–	–
(Bessai et al., 2012)	✓	✓	–	–	–
(Fard et al., 2013)	✓	–	–	✓	✓
(Abrishami et al., 2013)	✓	–	–	✓	✓
(Calheiros and Buyya, 2014)	✓	–	–	–	–
(Zeng et al., 2015)	✓	–	–	–	–
(Malawski et al., 2015)	–	–	–	✓	–
(Guo et al., 2015)	✓	✓	–	✓	–

empowered with the secure algorithms to prevent or thwart various attacks such as the cloud internal denial of the service attacks, the VM sprawl attacks, the VM neighbor attacks, the VM migrate and escape attacks and so on.

References

AlZain M, et al. Cloud computing security: from single to multi-clouds. In: Proceedings of the 2012 45th Hawaii international conference on system science (HICSS); 2012, IEEE.

Agarwal D, Jain S. Efficient optimal algorithm of task scheduling in cloud computing environment. arXiv preprint arXiv:1404, 2076, 2014.

Alkhanak EN, Lee SP, Khan SUR. Cost-aware challenges for workflow scheduling approaches in cloud computing environments: taxonomy and opportunities. Future Gener Comput Syst 2015.

Anju Baby J. A survey on honey bee inspired load balancing of tasks in cloud computing. In: Proceedings of the international journal of engineering research and technology; 2013. ESRSA Publications.

Aryan Y, Delavar AG. Abi-objective workflow application scheduling in cloud computing systems.

Abrishami S, Naghibzadeh M, Epema DH. Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds. Future Gener Comput Syst 2013;29(1):158–69.

Amiry V, et al. Implementing hadoop platform on eucalyptus cloud infrastructure. In: Proceedings of the P2P, 2012 seventh international conference on parallel, grid, cloud and internet computing (3PGCIC); 2012, IEEE.

Buyya R, Ranjan R, Calheiros RN. Intercloud: utility-oriented federation of cloud computing environments for scaling of application services, In: Algorithms and architectures for parallel processing; 2010, Springer. p. 13–31.

Basishtha S, Boruah S. Cloud computing and its security aspects.

Bala A, Chana I. A survey of various workflow scheduling algorithms in cloud environment. In: Proceedings of the 2nd national conference on information and communication technology (NCICT); 2011.

Barrett E, Howley E, Duggan J. A learning architecture for scheduling workflow applications in the cloud. In: Proceedings of the 2011 ninth IEEE European conference on web services (ECOWS); 2011. IEEE.

Bilgaiyan S, Sagnika S, Das M. Workflow scheduling in cloud computing environment using cat swarm optimization. In: Proceedings of the 2014 IEEE International advance computing conference (IACC); 2014. IEEE.

Bala, RaGS. An improved heft algorithm using multi-criterian resource factors.

Bittencourt LF, Madeira ERM. HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. J Internet Serv Appl 2011;2(3):207–27.

Bessai K, et al. Bi-criteria workflow tasks allocation and scheduling in Cloud computing environments. In: Proceedings of the 2012 IEEE 5th international conference on cloud computing (CLOUD); 2012. IEEE.

Chawla Y, Bhonsle M. A study on scheduling methods in cloud computing. Int J Emerg Trends Technol Comput Sci 2012;**1**(3):12–7.

Chen W-N, Zhang J. A set-based discrete PSO for cloud workflow scheduling with user-defined QoS constraints. In: Proceedings of the 2012 IEEE international conference on systems, man, and cybernetics (SMC).

Chitra S, et al., Local minima jump PSO for workflow scheduling in cloud computing environments, In: Advances in computer science and its applications; 2014, Springer. p. 1225–1234.

Calheiros RN, Buyya R. Meeting deadlines of scientific workflows in public clouds with tasks replication. Parallel Distrib Syst IEEE Trans 2014;25(7):1787–96.

Calheiros RN, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw: Pract Exp 2011;41(1):23–50.
Chopra N, Singh S. HEFT based workflow scheduling algorithm for cost optimization within deadline in hybrid clouds. In: Proceedings of the 2013 fourth international conference on computing, communications and networking technologies (ICCCNT); 2013. IEEE.
Dillon T, Wu C, Chang E. Cloud computing: issues and challenges. In: Proceedings of the 2010 24th IEEE international conference on advanced information networking and applications (AINA); 2010. IEEE.
Delavar AG, Aryan Y. A goal-oriented workflow scheduling in heterogeneous distributed systems. Int J Comput Appl 2012;52(8):27–33.
Delavar AG, Aryan Y. HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems. Cluster Comput 2014;17(1):129–37.
Diallo S, et al. Simulator of amazon EC2 spot market.
Evangelinos C, Hill C. Cloud computing for parallel scientific HPC applications: Feasibility of running coupled atmosphere-ocean climate models on Amazon's EC2. Ratio 2008;2(2.40):2–34.
Fard HM, Prodan R, Fahringer T. A truthful dynamic workflow scheduling mechanism for commercial multicloud environments. Parallel Distrib Syst IEEE Trans 2013;24(6):1203–12.
Gogulan R, Kavitha A, Karthick Kumar U. An multiple pheromone algorithm for cloud scheduling with various QOS requirements. Int J Comput Sci 2012;9:3.
Guo F, et al. A workflow task scheduling algorithm based on the resources' fuzzy clustering in cloud computing environment. Int J Commun Syst 2015;28(6):1053–67.
Gai K, et al. Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing. J Netw Comput Appl 2015.
Huang J. The workflow task scheduling algorithm based on the GA model in the cloud computing environment. J Softw 2014;9(4):873–80.
A.HuthJ.CebulaThe basics of cloud computing. United States Computer; 2011.
Hong I, Potkonjak M. Power optimization in disk-based real-time application specific systems. In: Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design; 1997. IEEE Computer Society.
Huang J, et al. A tunable workflow scheduling algorithm based on particle swarm optimization for cloud computing. Criterion 2013;12:14.
Hoenisch P, Schulte S, Dustdar S. Workflow scheduling and resource allocation for cloud-based execution of elastic processes. In: Proceedings of the 2013 IEEE 6th international conference on service-oriented computing and applications (SOCA); 2013. IEEE.
Igugu JO, Biltoria P. STAF-on-eucalyptus: a cloud-based software testing environment for distributed systems; 2011.
Instances E. Getting started with cloud computing: amazon EC2 on red hat enterprise linux; 2008.
Jensen M, et al. Security prospects through cloud computing by adopting multiple clouds. In: Proceedings of the 2011 IEEE international conference on cloud computing (CLOUD); 2011. IEEE.
Jayadivya S, Bhanu SMS. Qos based scheduling of workflows in cloud computing. Int J Comput Sci Electr Eng 2012:2315–4209 ISSN.
Jianfang C, Junjie C, Qingshan Z. An optimized scheduling algorithm on a cloud workflow using a discrete particle swarm. Cybern Inform Technol 2014;14(1):25–39.
Jian C, et al. Time-constrained workflow scheduling in cloud environment using simulation annealing algorithm. J Eng Sci Technol Rev 2013;6(5):33–7.
Kaur N, Aulakh TS, Cheema RS. Comparison of workflow scheduling algorithms in cloud computing. Int J Adv Comput Sci Appl 2011;2:10.
Kapoor N, Kakkar P., Workflow scheduling in mobile cloud computing environment.
Kumar PaSA. Priority Based Workflow Task Scheduling In Cloud Computing Environments. Aust J Basic Appl Sci 2014;8:17.
Kaur A. A review of workflow scheduling in cloud computing environment.
Kaleeswaran A, Ramasamy V, Vivekanandan P. Dynamic scheduling of data using genetic algorithm in cloud computing. Park Coll Eng Technol 1963.
Kumar BA, Ravichandran T. Time and cost optimization algorithm for scheduling multiple workflows in hybrid clouds. Eur J Sci Res 2012;89(2):265–75.
Kaur R, Ghumman N. Hybrid improved max min ant algorithm for load balancing in cloud.
Lopez MM, Heymann E, Senar M. Analysis of dynamic heuristics for workflow scheduling on grid systems. In: Proceedings of the fifth international symposium on parallel and distributed computing, 2006. ISPDC'06; 2006. IEEE.
Liu K, et al. A compromised-time-cost scheduling algorithm in SwinDeW-C for instance-intensive cost-constrained workflows on cloud computing platform. Int J High Perform Comput Appl 2010.
Lin C, Lu S. Scheduling scientific workflows elastically for cloud computing. In: Proceedings of the 2011 IEEE international conference on cloud computing (CLOUD) 2011. IEEE.
Lu G, et al. QoS constraint based workflow scheduling for cloud computing services. J Softw 2014;9(4):926–30.
Lee YC, et al. Resource-efficient workflow scheduling in clouds. Knowl.-based Syst 2015;80:153–62.
Marcon DS, et al. Workflow specification and scheduling with security constraints in hybrid clouds. In: Proceedings of the 2nd IEEE Latin American conference on cloud computing and communications (LatinCloud); 2013. IEEE.
Maruthanayagam D, Prakasam TA. Job scheduling in cloud computing using ant colony optimization. Int J Adv Res Comput Eng Technol, 3, p. 540–47.
Motahari-Nezhad HR, Stephenson B, Singhal S. Outsourcing business to cloud computing services: opportunities and challenges. IEEE Internet Comput 2009;10.
Mary NABaKJ. An extensive survey on QoS in cloud computing. Int J Comput Sci Inform Technol 2014;5:1.
Madić M, Marković D, Radovanović M. Comparison of metaheuristic algorithms for solving machining optimization problems. Facta Univ Series: Mech Eng 2013;11(1):29–44.
Malawski M, et al. Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization. Sci Progr 2015;2015.
Nallakumar R. A survey on deadline constrained workflow scheduling algorithms in cloud environment. arXiv preprint arXiv:1409, 7916; 2014.
Nagadevi S, Satyapriya K, Malathy D. A survey on economic cloud schedulers for optimized task scheduling. Intemational J Adv Eng Technol 2013;4(1):58–62.
Nair T, Sooda K. Comparison of genetic algorithm and simulated annealing technique for optimal path selection in network routing. arXiv preprint arXiv:1001, 3920; 2010.
Nurmi D, et al. The eucalyptus open-source cloud-computing system. In: Proceedings of the 9th IEEE/ACM international symposium on cluster computing and the grid, 2009. CCGRID'09; 2009. IEEE.
Pandey S, et al. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: Proceedings of the 2010 24th IEEE International Conference on advanced information networking and applications (AINA); 2010. IEEE.
Patel R, Mer H. A survey of various Qos-based task scheduling algorithm in cloud computing environment.
Patel S, Bhoi U. Priority based job scheduling techniques in cloud computing: a systematic review. Int J Sci Techno Res 2013;2:147–52.
Pragaladan R, Maheswari R. Improve workflow scheduling technique for novel particle swarm optimization in cloud environment.
Ponniselvi MD, Seetha E. Analysis of workflow scheduling process using enhanced superior element multitude optimization in cloud.
Poola D, Ramamohanarao K, Buyya R. Fault-tolerant workflow scheduling using spot instances on clouds. Proc Comput Sci 2014;29:523–33.
Parsa S, Entezari-Maleki R. RASA: a new task scheduling algorithm in grid environment. World Appl Sci J 2009;7:152–60.
Rahman M, et al. Adaptive workflow scheduling for dynamic grid and cloud computing environment. Concurr Comput: Pract Exp 2013;25(13):1816–42.
Rodriguez MA, Buyya R. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. Cloud Comput IEEE Trans 2014;2(2):222–35.
Ryu KD, et al. RC2-A living lab for cloud computing. In: Proceedings of the LISA; 2010.
Rius J, Cores F, Solsona F. Cooperative scheduling mechanism for large-scale peer-to-peer computing systems. J Netw Comput Appl 2013;36(6):1620–31.
Singh L, Singh S. A survey of workflow scheduling algorithms and research issues. 2013.
Sajid M, Raza Z. Cloud computing: issues and challenges. In: Proceedings of the international conference on cloud, big data and trust; 2013.
Sridhar M, Babu G. Hybrid particle swarm optimization scheduling for cloud computing. In: Proceedings of the 2015 IEEE International Advance Computing Conference (IACC); 2015. IEEE.
Singh L, Singh S. Deadline and cost based ant colony optimization algorithm for scheduling workflow applications in hybrid cloud.
Singh R, Singh S. Score based deadline constrained workflow scheduling algorithm for Cloud systems. Int J Cloud Comput: Serv Archit 2013;3(6):31–41.
Shengjun X, Jie Z, Xiaolong X. An improved algorithm based on ACO for Cloud service PDTs Scheduling. Adv Inf Sci Serv Sci 2012;4:18.
Tiwari SM., Cloud computing using cloud-level scheduling: a survey.
Tao Q, et al. QoS constrained grid workflow scheduling optimization based on a novel PSO algorithm. In: Proceedings of the eighth international conference on grid and cooperative computing, 2009. GCC'09; 2009. IEEE.
Vijayalakshmi R, Vasudevan V. Static batch mode heuristic algorithm for mapping independent tasks in computational grid. J Comput Sci 2015;11(1):224.
Verma A, Kaushal S. Cost minimized PSO based workflow scheduling plan for cloud computing; 2015.
Verma A, Kaushal S. Bi-criteria priority based particle swarm optimization workflow scheduling algorithm for cloud. In: Proceedings of the 2014 Recent Advances in Engineering and Computational Sciences (RAECS); 2014. IEEE.
Verma A, Kaushal S. Budget constrained priority based genetic algorithm for workflow scheduling in cloud. In: Proceedings of the Fifth International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013); 2013. IET.
Verma A, Kaushal S. Deadline and budget distribution based cost-time optimization workflow scheduling algorithm for cloud. In: Proceedings of the IJCA on International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT'12). 2012.
Wu Z, et al. A market-oriented hierarchical scheduling strategy in cloud workflow systems. J Supercomput 2013;63(1):256–93.
Wang J, et al. Workflow as a service in the cloud: architecture and scheduling algorithms. Proc Comput Sci 2014;29:546–56.
Wu L, Garg SK, Buyya R. SLA-based resource allocation for software as a service provider (SaaS) in cloud computing environments. In: Proceedings of the 2011 11th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid); 2011. IEEE.

Wang L, et al. Schedule distributed virtual machines in a service oriented environment. In: Proceedings of the 2010 24th IEEE international conference onadvanced information networking and applications (AINA); 2010. IEEE.
Wu Z, et al. A revised discrete particle swarm optimization for cloud workflow scheduling. In: Proceedings of the 2010 international conference on computational intelligence and security (CIS); 2010. IEEE.
Wu H, Tang Z, Li R. A priority constrained scheduling strategy of multiple workflows for cloud computing. In: Proceedings of the 2012 14th international conference on advanced communication technology (ICACT); 2012. IEEE.
Xhafa F, Abraham A. Computational models and heuristic methods for Grid scheduling problems. Future Gener Comput Syst 2010;26(4):608–21.
Xu M, et al. A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing. In: Proceedings of the 2009 IEEE international symposium on parallel and distributed processing with applications; 2009. IEEE.
Yaseen SG, Al-Slamy NM. Ant colony optimization. IJCSNS 2008;8(6):351.
Yassa S. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. Sci World J 2013;2013.
Zhao L, Ren Y, Sakurai K. Reliable workflow scheduling with less resource redundancy. Parallel Comput 2013;39(10):567–85.
Zhou Y, Huang X.. Scheduling workflow in cloud computing based on ant colony optimization algorithm. In: 2013 sixth international conference on proceedings of the business intelligence and financial engineering (BIFE); 2013. IEEE.
Zhang F, et al. Adaptive workflow scheduling on cloud computing platforms with iterative ordinal optimization.
Zhu MM, Cao F, Wu CQ. High-throughput scientific workflow scheduling under deadline constraint in clouds; 2014.
Zeng L, Veeravalli B, Li X. SABA: a security-aware and budget-aware workflow scheduling strategy in clouds. J Parallel Distrib Comput 2015;75:141–51.
Zhao Q, et al. A new energy-aware task scheduling method for data-intensive Applications in the cloud. J Netw Comput Appl 2015.