


Cloud Computing: A Multi-workflow Scheduling Algorithm with Dynamic Reusability

Mainak Adhikari¹ · Santanu Koley¹ 

Received: 17 January 2017 / Accepted: 13 July 2017 / Published online: 21 July 2017
© King Fahd University of Petroleum & Minerals 2017

Abstract Cloud computing provides a dynamic environment of well-organized deployment of hardware and software that are common in nature and the requirement for propping up heterogeneous workflow applications to realize high performance and improved throughput where the most demanding task is multiple workflow applications surrounded by their fixed deadline. These workflow applications consist of interconnected jobs and data. Nevertheless, hardly any initiations are tailored on multi-workflow scheduling exertion. These scheduling problems have been considered methodically in cloud atmosphere. Accessibility of the computing resources on the data center (DC) provides the exact time of execution of each process, whereas the execution time of every process within a workflow is pre-calculated in the majority of the existing multi-workflow scheduling problem. System overhead so far is an additional concern at the same time as dynamically generating virtual machines (VMs) with salvage them dipping the power eating. The aim of this paper is to reduce the execution time of every job and finalize the execution of all workflow within its deadline by producing VMs dynamically in DC and recycle them as necessary. We recommend a dynamic multi-workflow scheduling algorithm formally named as competent dynamic multi-workflow scheduling (CDMWS) algorithm. Simulation process describes one of the best algorithms so far in terms of performance among subsistent algorithm and moves toward a new era of multi-workflow relevance.

Keywords Heterogeneous workflow application · Cloud computing · Multi-workflow scheduling · Virtual machine · Deadline · Data center

1 Introduction

Cloud computing [1–3] is a new-fangled transformational paradigm which permits dynamic resource allocation using resource pool with an assortment of combination techniques from distributed computing, high-performance computing as well as platform virtualization techniques. Cloud service providers akin to Amazon, Google or Microsoft mingles numerous DCs [4] on poles apart geographical locations consisting of an enormous quantity of host servers which can generate manifold VMs as per request. It provides on demand services as software, hardware, platforms are included in it to serve user as pay-per-use concept. Cloud service provider (CSP) [4] ensures Quality of Service (QoS) [5,6] in Service Level Agreement (SLA) [7] to guarantee improved class on dissimilar bound of service. Dissimilar scientific applications are mold by workflows that have been generally used for a set of organized tasks via data. These workflows encompass considerable task that desires a number of computing resources. They must complete within certain pre-determined time edges known as the deadline. This is necessary for a cloud environment to complete a single workflow with minimum resources. This is really a challenging task to achieve such workflow with a certain deadline.

The wastage of computing resources is exceptionally ordinary in those DCs where simple multi-workflow scheduling problems are taken up. Here tasks have pre-computed execution time depending upon VMs. The diminution of the deployment of the server, shrinking flexibility, efficiency and elasticity of the resources are also the consequence of pre-

✉ Santanu Koley
santanukoley@yahoo.com

Mainak Adhikari
mainak.ism@gmail.com

¹ Department of Computer Science and Engineering,
Budge Budge Institute of Technology, Kolkata, India

configured VM. Now, this is clear enough to comprehend about the system that cannot be adequate to hold a wide number of user applications. Approximating the execution time of each task on the fly based on the SLA matrices of the workflow application and generating the right VMs dynamically can enhance the scalability and elasticity of the resources of cloud DC and facilitates to switch a huge number of user applications.

In this paper, we recommended a dynamic multi-workflow scheduling algorithm entitled as competent dynamic multi-workflow scheduling (CDMWS) in a cloud environment. Dynamic approximation of the period of accomplishment for every task is the focal point of the suggested algorithm. It also portrays the generation of appropriate VMs for carrying out those tasks with bare minimum resource requirement that reuse the VMs. Here, it is mandatory for all tasks of a workflow to complete their execution within the user-defined deadline. Construction of the VMs dynamically can augment the flexibility and elasticity of the cloud, and reusability of the VMs can trim down the overhead and the rate of power consumption. We become accustomed by means of a proficient mechanism to come across the execution time of each task within the application deadline even though it is not available. Now we are in a situation to articulate on the subject of our proposed algorithm that is better than existing algorithms with respect to various performance metrics by using through simulation.

We have organized our paper as in dissimilar sections. Section 2 describes the related work done so far in the area of workflow scheduling. Section 3 properly describes the problem along with problem formulation and the algorithm itself. The assessment parameter and performance-associated issues are highly structured in Sect. 4. As a final point, Sect. 5 concludes the paper and discusses about the opportunities for research direction.

2 Related Work

The cloud computing is the topical contraption in the vicinity of distributed approach and putting on the magnetism of the researchers throughout the world. The service-oriented approach [8] engages in recreation as win–win circumstances for the service providers as well as users who put together so popular today that entire services are moving in it. Now about workflow scheduling in a cloud environment is one of the well-liked neighborhood of research as many researchers are interested in it and did remarkable work so far for cloud [9–23] DCs. At this point, we analyze a number of associated scheduling algorithms for the data center based on cloud atmosphere.

Paton et al. [9] presented an optimized workflow scheduling algorithm based on a utility function. The algorithm

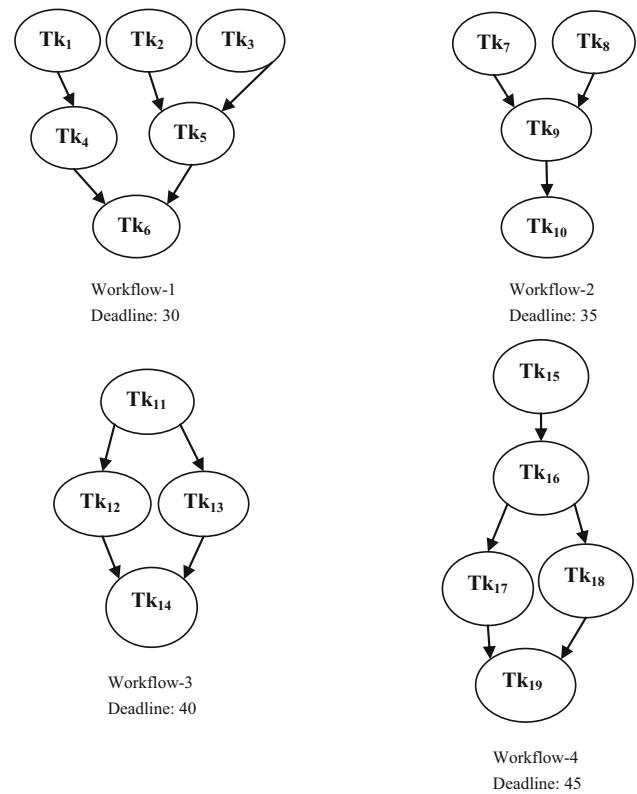


Fig. 1 A set of four workflows with its deadlines

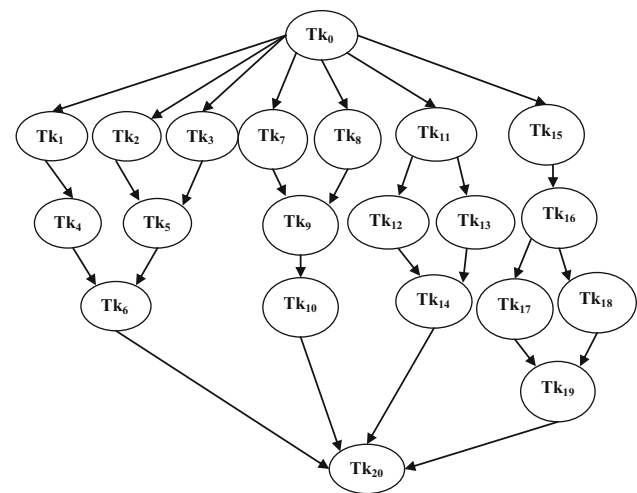


Fig. 2 A sample multi-workflow application

comes across the explicit desirability of dissimilar workloads with an evaluation stratagem.

Hu et al. [10] proposed an efficient workflow scheduling algorithm which distributes the task to the servers such a manner that all tasks of a workflow should complete their execution within time using a minimum number of servers.

Fito et al. [11] developed an economic web hosting system called as CHP. The CHP endows with scalable web services

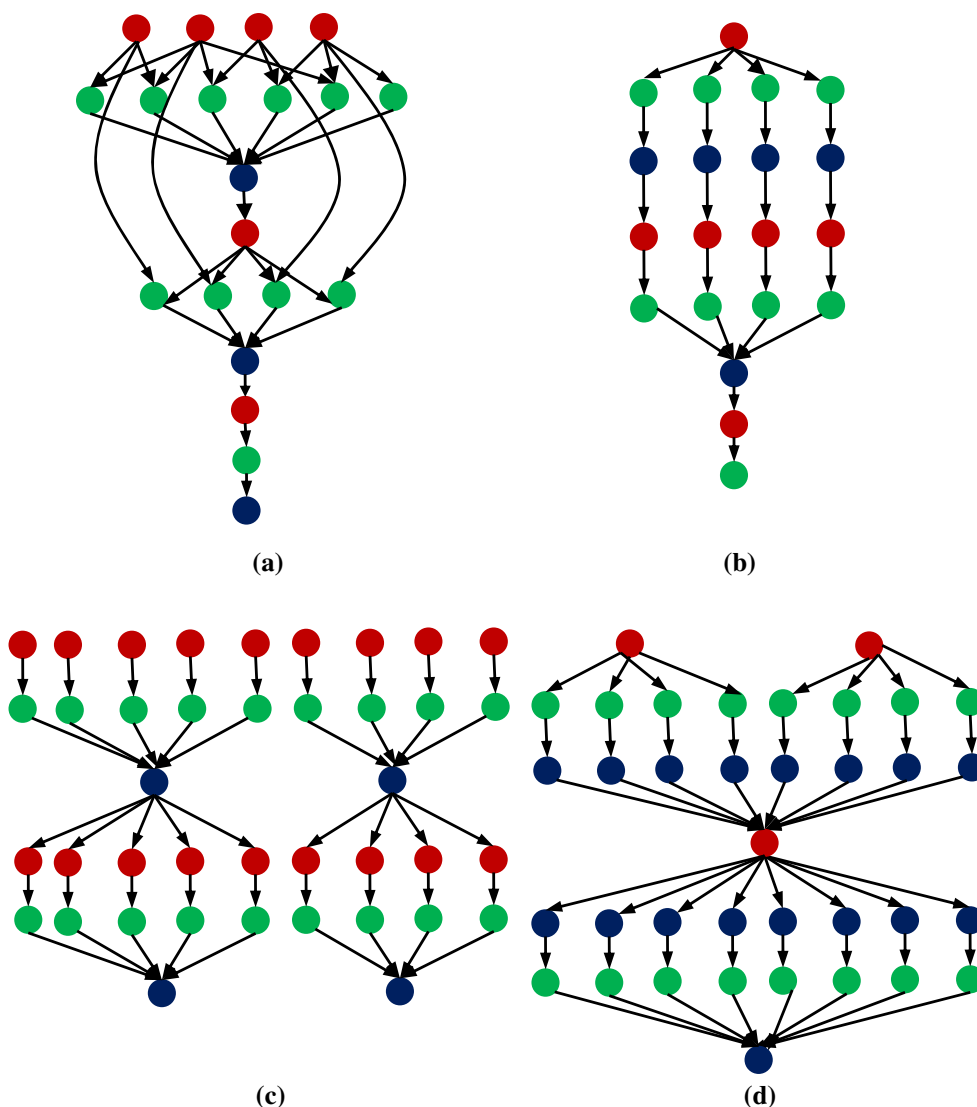


Fig. 3 Scientific workflows. **a** Epigenomics, **b** Montage, **c** LIGO, **d** Cybershake

by means of diverse cloud resources. The aim of the algorithm is to capitalize on the profit of the service provider.

Wu et al. [12] extended a PSO-based near-optimal workflow scheduling technique. The objective of the algorithm is to minimize the execution time of the tasks while meeting the scheduling constraints such as deadline and budget using some heuristic approaches. The algorithm uses a fixed set of VMs which shrinks elasticity of the cloud resources and is deficient in the deployment of resources.

Zhu et al. [13] expanded a multi-tire-based web applications using some virtualization technology in cloud data centers. The purpose of the algorithm is to satisfy users SLA matrices and maximize the overall profit of the providers.

Mao and Humphrey [14] proposed a dynamic workflow scheduling algorithm on cloud computing environment. Here cloud provider contains various types of VMs as a leased

manner based on the requirements of the user applications. The key idea of the algorithm is to reduce the execution time of the tasks using some heuristic approaches, but it fails to produce a near-optimal solution.

Byun et al. [15] suggested an online-based workflow scheduling algorithm to minimize the execution time of the tasks using an optimal number of resources. The provider contains same types of VMs as a leased manner based on the requirement of the application. The schedule and the resources are updated periodically every time interval based on the current status of the running VMs. This algorithm takes the advantage of elasticity of cloud but fails to consider the heterogeneity nature of computing resources.

Sharma et al. [16] advised a workload scheduling algorithm to select an appropriate resource for each task based on pricing and elasticity mechanism to optimize the cost of

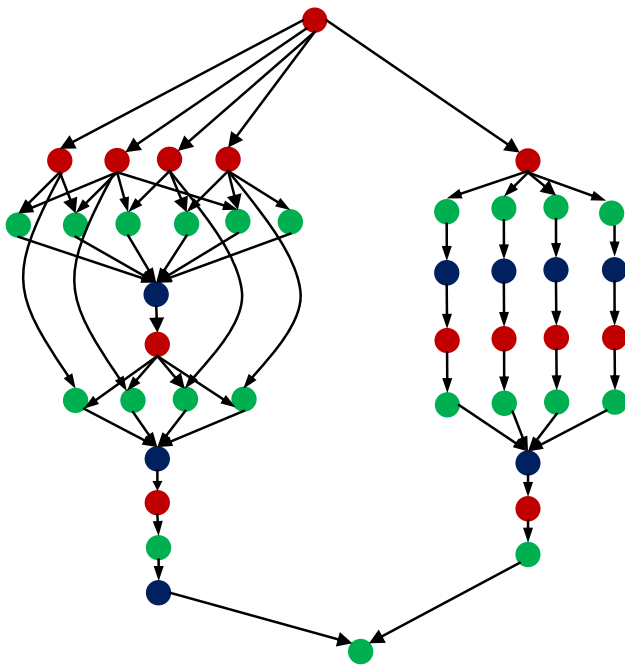


Fig. 4 Example of multi-workflow combination with two scientific workflows

the server. This algorithm is applicable to single workflow application without considering the SLA matrices but not to mixed workload application.

Bonvin et al. [17] widened a decentralized algorithm in public cloud environment for allocating cloud resources dynamically for different applications while meeting their SLA requirements. This algorithm is worked over web-based application only.

Abrishami et al. [18] offered a QoS-based workflow scheduling algorithm based on partial critical path approach to minimizing the cost of the tasks while meeting a deadline. This algorithm takes the advantage of the elasticity of the cloud but fails to consider heterogeneous nature of the computing resources by assuming only one type of VM present in the cloud.

Abrishami et al. [19] designed a static workflow scheduling algorithm in a cloud environment. The algorithm is based on workflow's partial critical paths, and it considers the basic cloud features such as VM heterogeneity, pay-as-use and time interval pricing model. The algorithm minimizes the execution cost based on heuristic approaches over all tasks present in a partial critical path on a single VM which can finish the tasks before their latest finish time. The algorithm produced a task-level optimization but fails to produce a near optimization technique and to utilize the whole workflow structure and characteristics to generate a better solution.

Malawski et al. [20] devised various types of dynamic and static algorithms to maximize the amount of work completed while meeting QoS constraints such as deadline and budget.

The algorithm is robust in nature that the estimated execution time of task may vary based on uniform distribution of tasks and cost safety margin to avoid generating a schedule that goes over budget. However, this algorithm considers only a single type of VM ignoring the heterogeneous nature of clouds.

Xiao et al. [21] make a blueprint of a dynamic resource provisioning algorithm which optimizes the resource allocation to the virtual machines based on the application demands.

Antonescu et al. [22] intended a prediction-based allocation mechanism for allocating the virtual machines to the data center in order to maximize the profits and meets all SLA metrics.

Rodriguez et al. [23] planed another PSO-based workflow scheduling algorithm to minimize the overall workflow execution cost while meeting with the scheduling constraint such as user-defined deadline. The algorithm produces task to resource mapping with higher accuracy in terms of meeting deadline. The solution considers the heterogeneity of resources present in the cloud that can be dynamically acquired and released resources and charged on a pay-per-use basis.

3 Proposed Work

This proposed work describes the main idea of dynamic reusability for virtual machines for different kind of tasks. Here, mathematical expressions and the algorithms are used to prove the above said. A single task is showing with its subtasks in parallel. Independent tasks can reuse the VMs which had previously been used by some other task.

3.1 Problem Statement

A multi-workflow (WF_m) application $WF_m = \{W_j, j = 1, 2, 3, \dots, n\}$ consists of a set of workflow applications which arrive at a time to the service provider. Let $WF(Tk, De)$ be a workflow, which is represented by a direct acyclic graph (DAG), where Tk is the set of tasks and De is the set of directed edges between the tasks. The directed edges of the DAG represent the dependency among the tasks which also represent the communication cost among them.

An edge $De_{ij}Tk_i, Tk_j$ exists if there is a dependency between task Tk_i and Tk_j . Task Tk_i is said to be the predecessor of Tk_j and Tk_j is said to be the successor of Tk_i . Based on this organization, a successor task cannot be executed until its entire predecessor tasks are completed their execution. Let DI be the deadline of workflow WF . Note that the deadline of each individual task is not available. Figure 1 shows the set of four sample workflows with a set of some independent tasks and a deadline for each of the workflows. The arriving time of all the four workflows must be same.

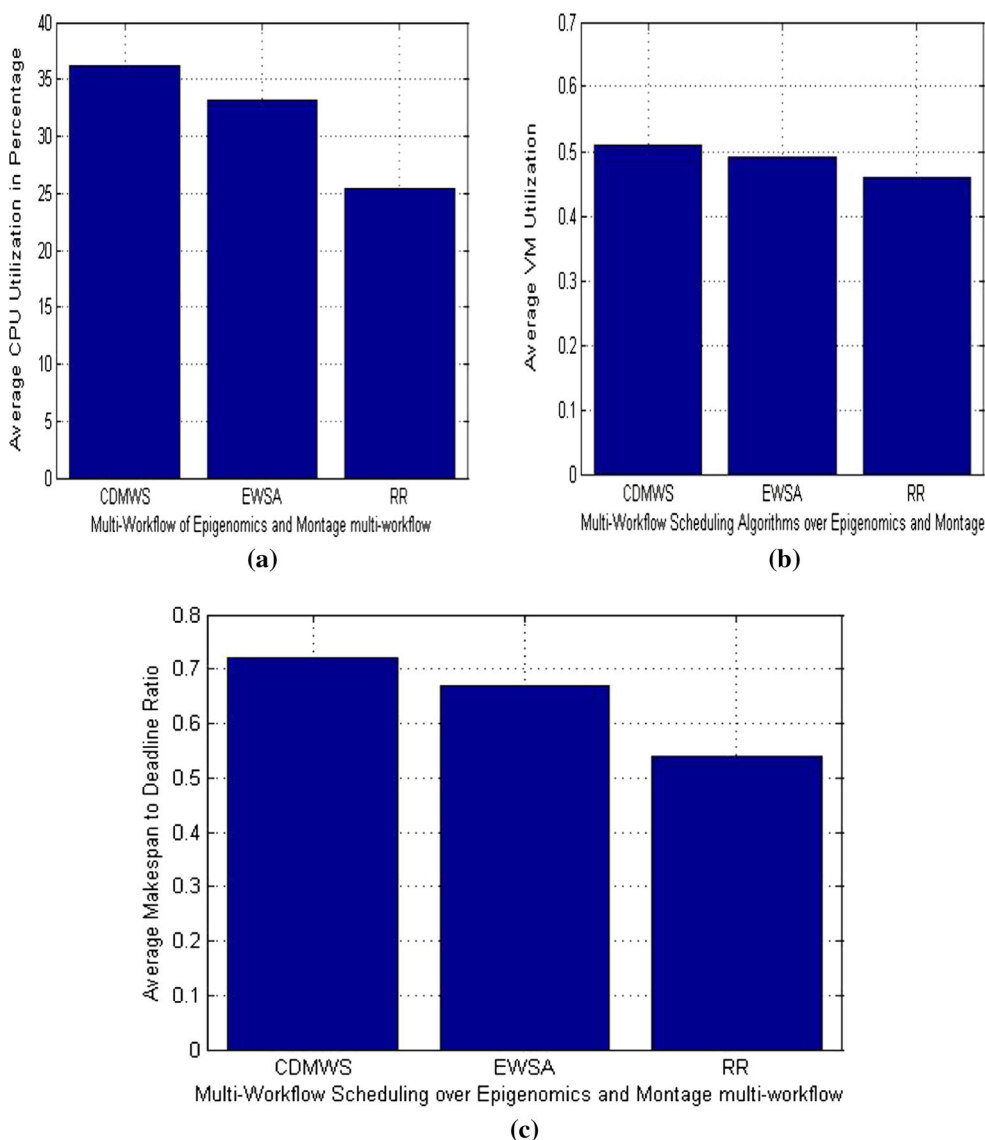


Fig. 5 Multi-workflow scheduling over Epigenomics and Montage workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio

From the set of simple workflows, we create a simple workflow with the help of another two dummy tasks—one is called pseudo entry task Tk_{EN} and another one is pseudo exit task Tk_{EX} . The execution time of these two tasks and communication cost between those tasks with other tasks must be zero. Those tasks only help to create a multi-workflow DAG using the help of a set of some simple workflow DAGs. Figure 2 shows the multi-workflow DAG which combines the four workflows as shown in Fig. 1. In Fig. 2, task Tk_0 is pseudo entry task and Tk_{20} is pseudo exit task. Tk_0 helps to combine all root nodes of the DAGs, and Tk_{20} helps to combine all the leaf nodes of the DAG. From Fig. 2, we observe that there exist nine possible paths, each path having set of tasks, e.g., (path labeled: 0-1-4-6-20). Note that the deadline of each workflow is $De_i = De_k, k = 1, 2, 3 \dots N$. Tasks

are also categorized into levels; tasks belonging to each level can be executed in parallel, e.g., Level 1: (labeled 1, 2, 3, 7, 8, 11 and 15). The size of each task is represented in the form of α_{MI} (million instructions).

3.2 Proposed Algorithm

The proposed algorithm is divided into two phases, update and task-VM mapping. The objective of the update phase is to set execution time for each task with the help of the deadline of the workflow and the communication cost between the tasks and to find a required capacity VM to execute each task. In task-VM mapping phase, the tasks are scheduled to the VMs dynamically and reuse the VMs for those tasks which required same capacity VM to execute. The VM reusability

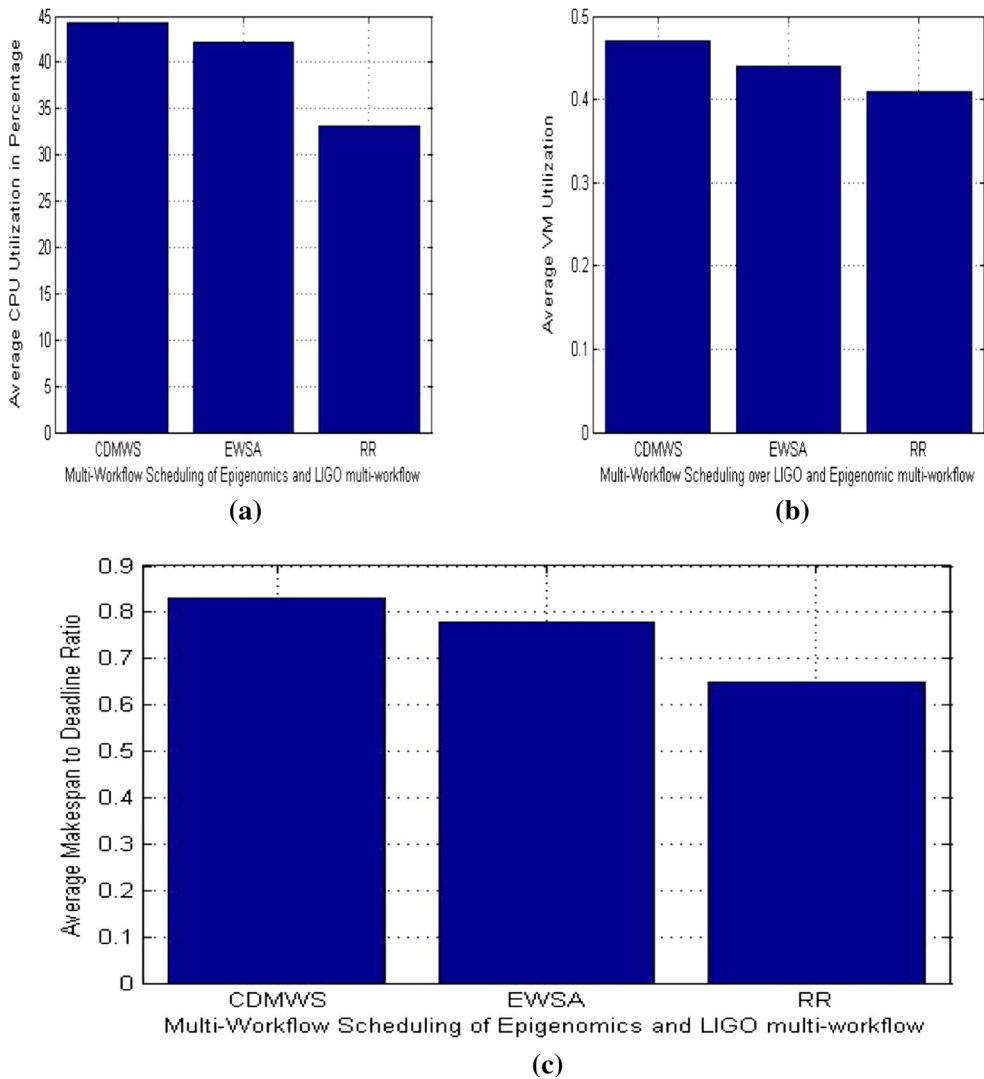


Fig. 6 Multi-workflow scheduling over LIGO and Epigenomics workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio

technique of task-VM phase reduces the system overhead and power consumption. They are discussed as follows.

Update Phase $Pt_1, Pt_2, Pt_3, \dots, Pt_q$ are the q paths that exist in WF_m and $Tk_1^i, Tk_2^i, Tk_3^i, \dots, Tk_t^i$ are the set of tasks, and $De_{12}^i, De_{22}^i, De_{32}^i, \dots, De_{t(t-1)}^i$ are the edges belonging to a path Pt_i , and the deadline of each path is De_k which depends on the workflow of the multi-workflow application. The size of a task Tk_j^i is represented by αMI^i_j and $C^i(Tk_x^i; Tk_y^i)$ is the communication cost between the tasks Tk_x^i and Tk_y^i where $Tk_x^i, Tk_y^i \in Pt_i$. Let us consider that all the tasks belong to the path Pt_i as a single virtualized task, i.e., Tk^i , whose size is αMI^i which is computed as follows:

$$\alpha MI^i = \sum_{j=1}^t \alpha MI^i_j \tag{1}$$

The execution time $Ex(T^i)$ of task T^i is calculated as follows:

$$Ex(Tk^i) = \left(De_k - \sum_{j=1}^{t-1} C^i(Tk_j^i, Tk_{j+1}^i) + (Tk_{oh} \times t) \right) \tag{2}$$

where Tk_{oh} is the overhead time during task-VM mapping. Let CPt^i be the capability (in terms of MIPS) of a VM that requires to execute each task belonging to the path Pt_i in order to execute all the tasks within De_k depending on the workflow of the multi-workflow application. The capability CPt^i is calculated as follows:

$$CPt^i = \frac{\alpha MI^i}{De_k(Tk^i)} \tag{3}$$

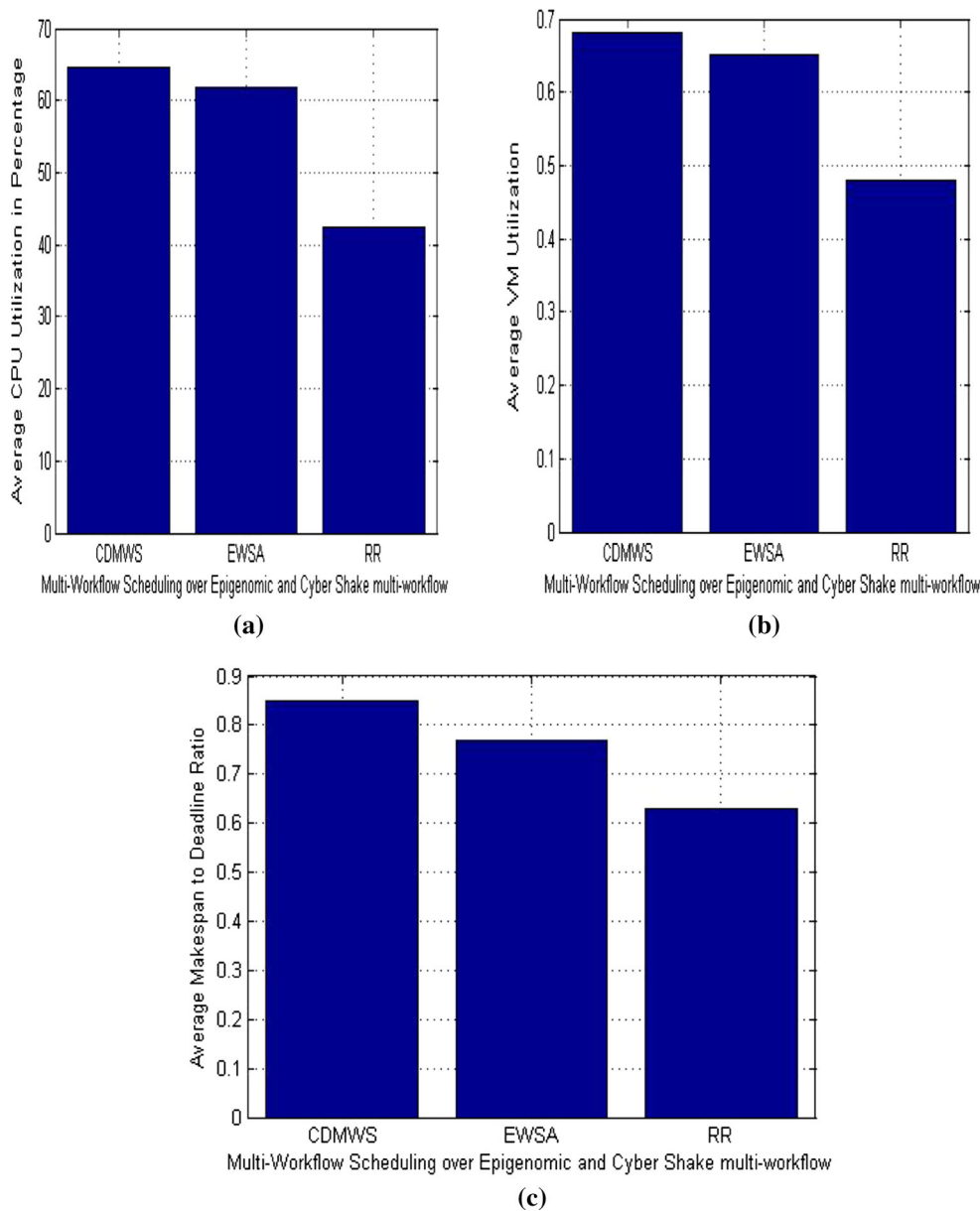


Fig. 7 Multi-workflow scheduling over Epigenomics and Cybershake workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio

Initially, the execution time of each task is initialized to -1 . In other words, the execution time of tasks is not available. If the execution time α_{ET}^i of a task Tk_j^i is -1 , the α_{ET}^i is estimated as per the following equation:

$$ET_j^i = \frac{\alpha MI_j^i}{CPI^i} \tag{4}$$

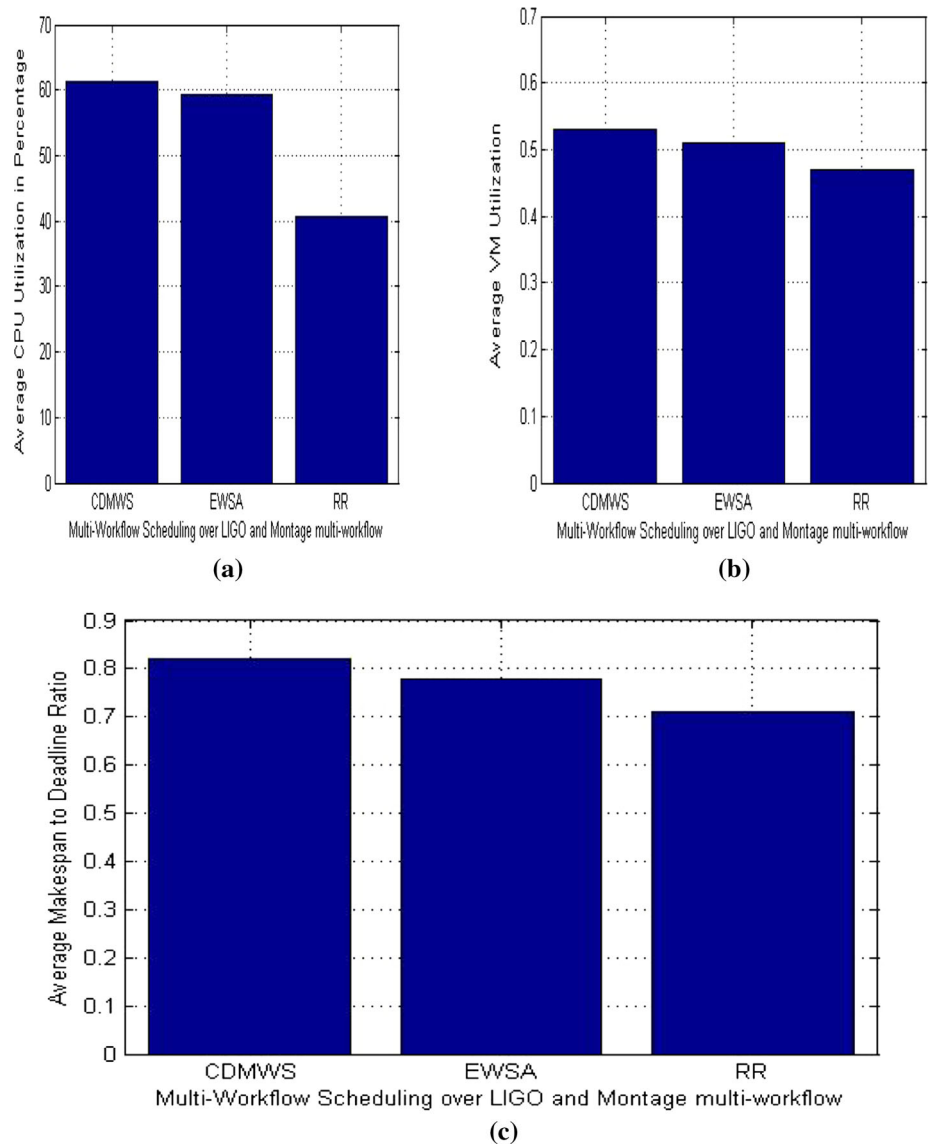
Consider a situation, $Tk_i^a \in Pt_a$ and $Tk_i^b \in Pt_b$ where $Tk_i^a = Tk_i^b$. The capability of VMs requires to execute Tk_i^a and Tk_i^b which are CPI^a and CPI^b , respectively, such that

Tk_i^a and Tk_i^b have execution having ET_i^a and ET_i^b . Then, the execution time of task Tk_i^a ($Tk_i^a = Tk_i^b$) is updated as follows:

$$ET_j^a = ET_j^b = \begin{cases} \frac{\alpha MI_j^a}{CPI^a} & \text{iff } CPI^a > CPI^b \\ \frac{\alpha MI_j^b}{CPI^b} & \text{iff } CPI^b > CPI^a \end{cases} \tag{5}$$

Task-VM Mapping Here, tasks are mapped onto the VMs level wise. Let multi-workflow WF_m organized into levels l levels, $L_0, L_1, L_2, \dots, L_{l-1}$. The number of tasks belonging to a level L_i is labeled as $|L_i|$. Therefore, the number of

Fig. 8 Multi-workflow scheduling over LIGO and Montage workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio



VMs required to execute all the tasks (in parallel) belonging to the level L_i is $|L_i|$. Let us consider level L_0 first and $Tk_0^1, Tk_0^2, Tk_0^3 \dots Tk_0^{|L_i|}$ are the tasks belonging to the level. Using Eqs. (3), (4) and (5), the VMM creates and assigns a corresponding VM to all the tasks. But if the task required Tk_i^b a CPt^b capacity VM which is already created for the task Tk_i^a with capacity CPt^a where $CPt^a = CPt^b$ and then the VMM assigned the task Tk_i^b to the earlier VM without creating a new VM. This reduces the overhead of the VMM as well as the DC. Once the execution all the tasks are completed, the VMM starts creating and scheduling VMs for the execution tasks belonging to the next level. The VMM halts only after completion of all task in all the levels. Let $Pt(Tk)$ and $St(Tk)$ be the set of predecessor and successor of task Tk . Using the following equations, starting time (ST) and completion time (CT) of a task are calculated. The starting time ST_j of task Tk_j is computed as

$$ST_j = \begin{cases} 0, & \text{iff } Tk_j \in L_0 \\ CT_i + T_{koh}, & \text{iff } \forall k \neq i, CT_i > CT_k, Tk_k, \\ & Tk_i \in P(Tk_j) \end{cases} \tag{6}$$

Note that the overhead time T_{oh} is added only when a new VM is created to execute the task T_j ; otherwise, its value is zero. The completion time CT_j of task T_j is derived as follows

$$CT_j = \begin{cases} ET_j, & \text{iff } T_j \in L_0 \\ CT_i + ET_j, & \text{iff } \forall k \neq i, CT_i > CT_k, T_k, \\ & T_i \in P(T_j) \end{cases} \tag{7}$$

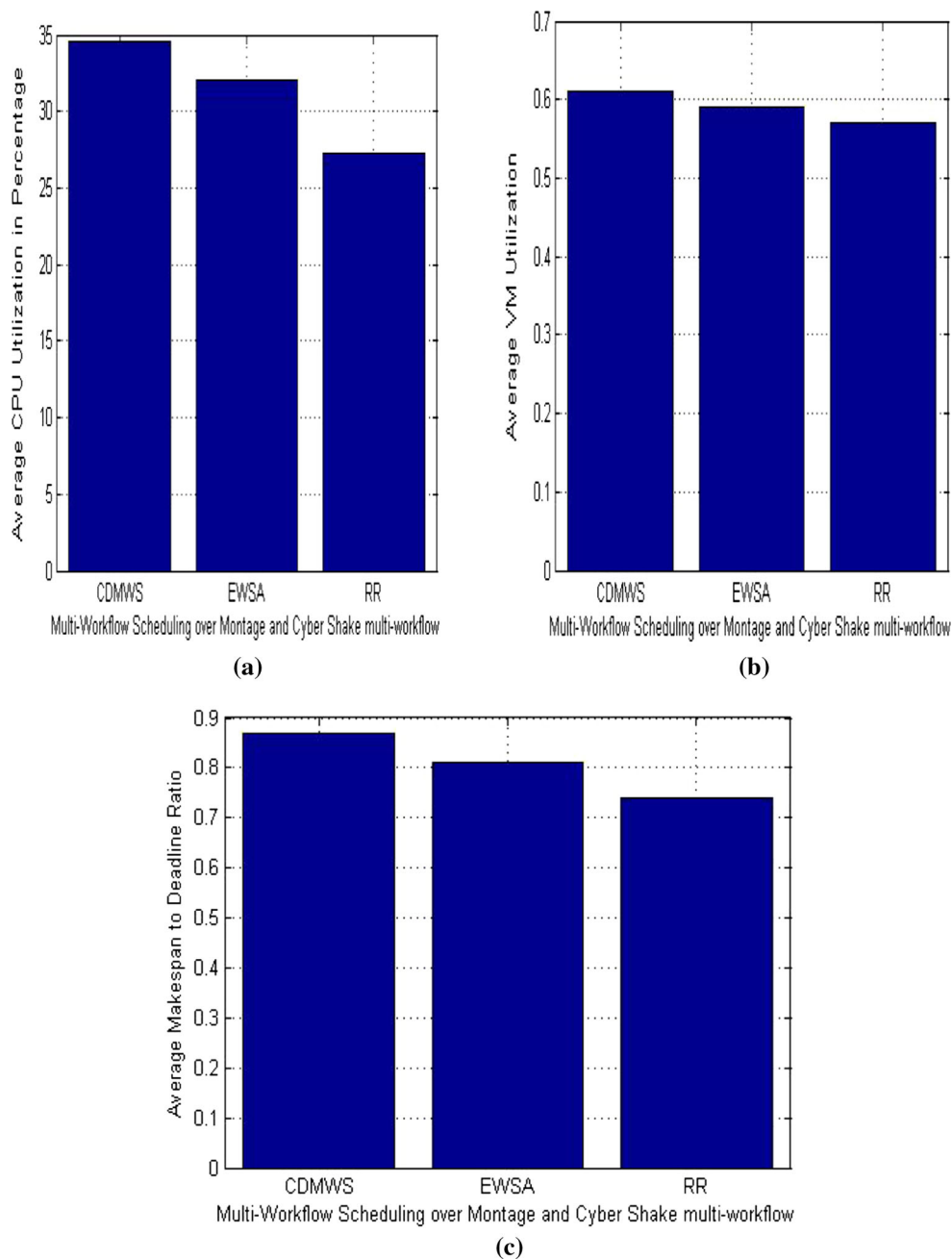


Fig. 9 Multi-workflow scheduling over Montage and Cyber Shake workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio

4 Performance Evaluation

In this section, we present the simulation results of proposed CDMWS algorithm.

We assess the proposed algorithm using a combination of some popular scientific workflows, such as Montage, Epigenomics, LIGO and Cybershake [19]. Deadline of each workflow is set as 100. Figure 3 provides detailed structure of scientific workflows, and Fig. 4 shows one exam-

ple of multi-workflow which combines with two scientific workflows—Montage and Epigenomics. To the best of our knowledge, we are first to estimate the execution time of the tasks dynamically and create VMs dynamically inside DC according to the requirement of the tasks. But if the task required a same capacity VM which already exists inside DC, then it reuses the VM without creating a new one which saves the system overhead and time. There is no such existing algorithm solving the problem in a similar way.

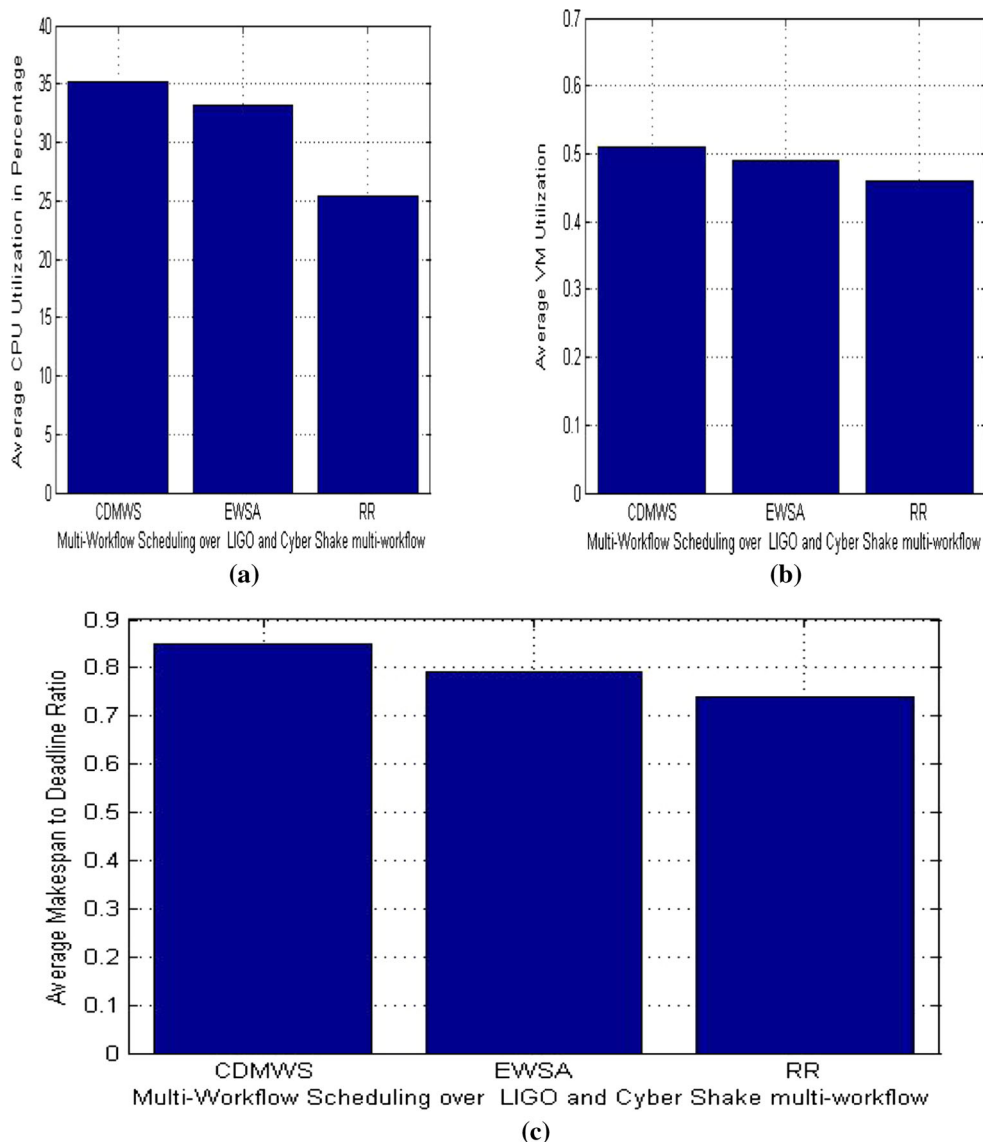


Fig. 10 Multi-workflow scheduling over LIGO and Cybershake workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio

Here, we adapted a static multi-workflow scheduling algorithm and a dynamic multi-workflow scheduling algorithm without VM reusability concept for the sake of comparison. In the static algorithm, VMs are pre-configured and execution time of the task is estimated based on these VMs, and the algorithm has adopted a best-fit strategy to schedule VMs. In the case of dynamic scheduling, the VMs are created dynamically for each task according to their requirement without reusing them.

4.1 Performance Metrics

The simulation programs are written in Dev C++ and Matlab. To evaluate the performance of the proposed algorithm, we use few performance metrics elaborated as follows:

- (a) Completion time:** The completion time of a workflow W is defined as the complete execution time of tasks. In other words, when all tasks of the workflow are completed their execution, it is called the completion time of W . It is also called makespan, denoted by M_W .
- (b) VM utilization:** This parameter indicates that how many times VMs are rescheduled to execute the different tasks after their creations. In other words, rescheduling VMs may lead to minimizing the number of VMs. If VM utilization is higher, it also results in saving computing resources which can be utilized for executing few more application.
- (c) CPU utilization:** This parameter indicates the maximum CPU utilization at any instance of time while execut-

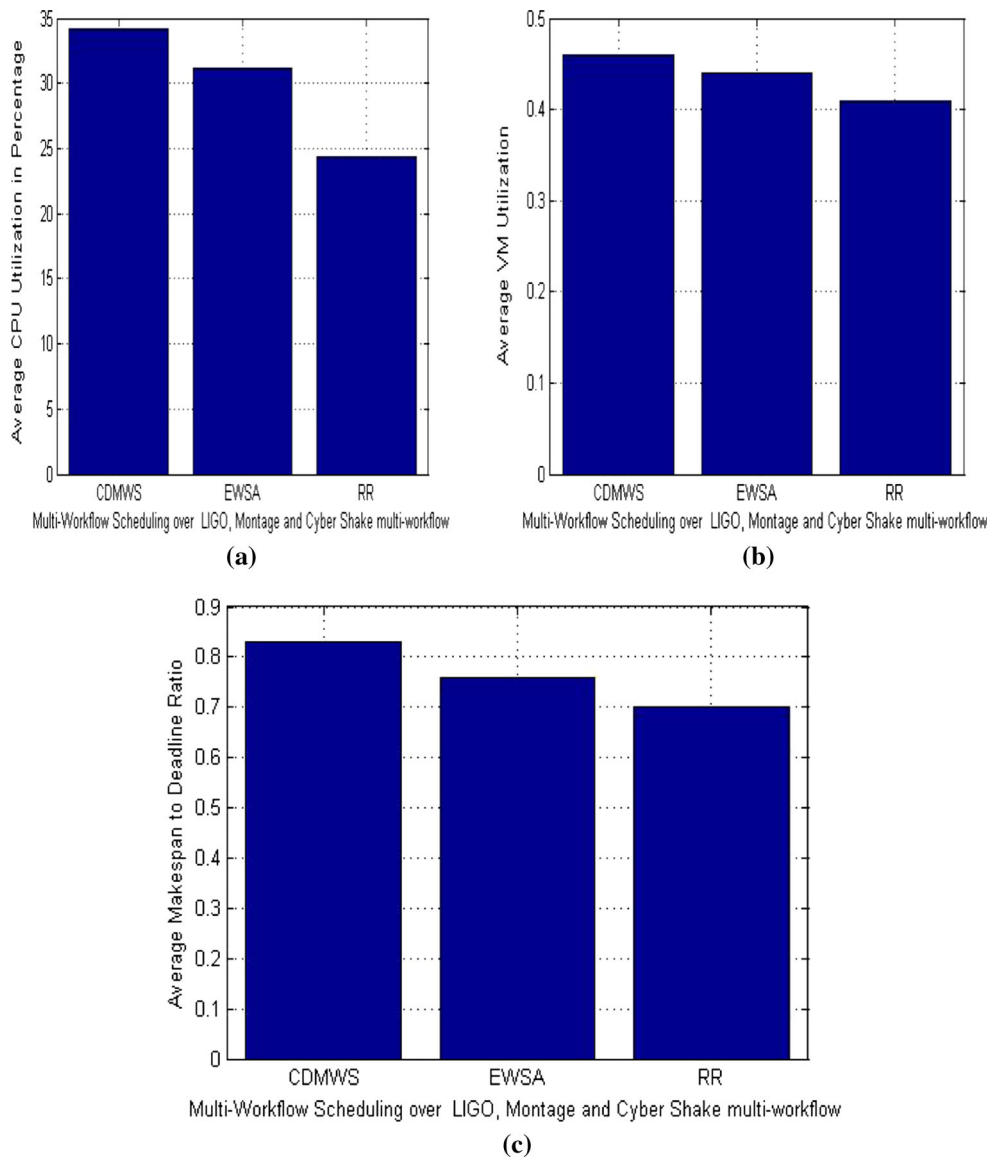


Fig. 11 Multi-workflow scheduling over LIGO, Montage and Cybershake workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio

ing the task. This basically depends on the maximum number of active VMs.

- (d) Makespan–deadline ratio: This parameter indicates how close the multi-workflow application completes its work to its deadline.

Therefore, the ultimate objective of the proposed algorithm is to maximize the makespan–deadline ratio, VMs utilization and overall CPU utilization.

4.2 Performance Evaluation

Assessment method contains the comparison results of the proposed algorithm as well as existing algorithms like

enriched workflow scheduling algorithm (EWSA) [24] and round robin (RR), by means of multi-workflow which mingle with different scientific workflows. Figures 5, 6, 7, 8, 9, 10, 11, 12, 13 and 14 illustrate the performance of the proposed algorithm (CDMWS) with static scheduling and dynamic scheduling in terms of VM utilization, CPU utilization and makespan–deadline ratio using ten different multi-workflows blends with some scientific workflows.

Commencing through multi-workflow scheduling like Epigenomics and Montage workflow (Fig. 5a), we examine that the CPU utilization is higher for CDMWS as compared to EWSA and RR. In Fig. 5b, we observe that the VM utilization is advanced than the used algorithms. From Fig. 5c, it is scrutinized that the average

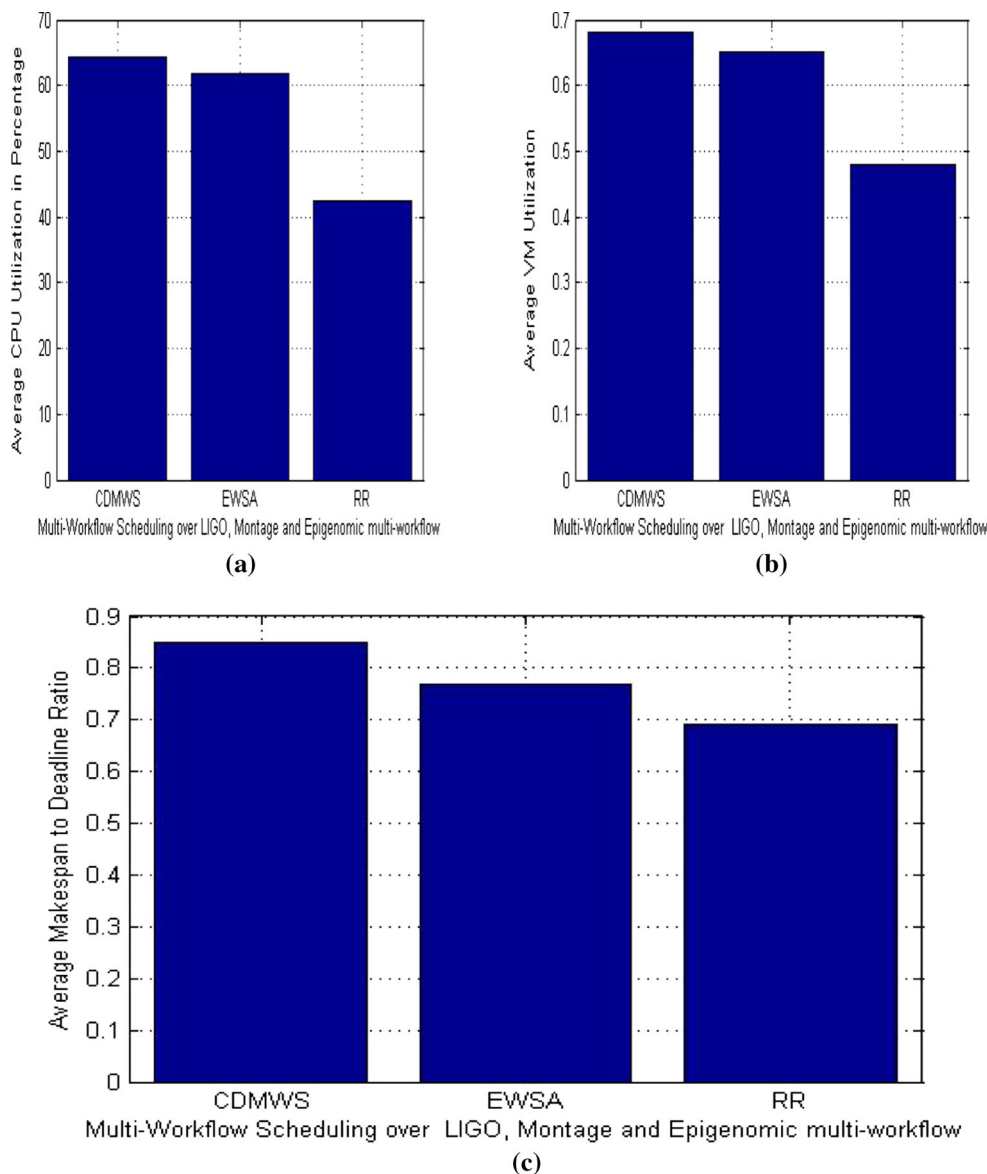


Fig. 12 Multi-workflow scheduling over LIGO, Montage and Epigenomics workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio

makespan-to-deadline ratio is privileged than other algorithms.

As of multi-workflow scheduling resembling LIGO and Epigenomics workflow (Fig. 6a), we inspect the CPU utilization is higher for CDMWS as compared to EWSA and RR. Figure 6b portrays the VM utilization is superior to the used algorithms. Figure 6c depicts average makespan-to-deadline ratio is advantaged than other algorithms.

The multi-workflow scheduling as Epigenomics and Cybershake described earlier (individually) in Fig. 7a expresses the CPU utilization is higher for CDMWS as measured up to EWSA and RR. Figure 7b represents the VM utilization is greater to the used algorithms. Figure 7c shows a picture

of average makespan-to-deadline ratio is providential than other algorithms.

Figure 8a–c proves the comparison outcome of the proposed and existing algorithms in terms of CPU utilization, VM utilization and average makespan-to-deadline ratio for LIGO and Montage workflow.

Here it can be noted two multi-workflow scheduling algorithms combination of Montage and Cybershake describes the proposed algorithm is at its best in terms of average CPU utilization, average VM utilization and average makespan-to-deadline ratio from Fig. 9a–c.

Figure 10a–c shows a picture of higher CPU, VM utilization and makespan-to-deadline ratio on an average basis for

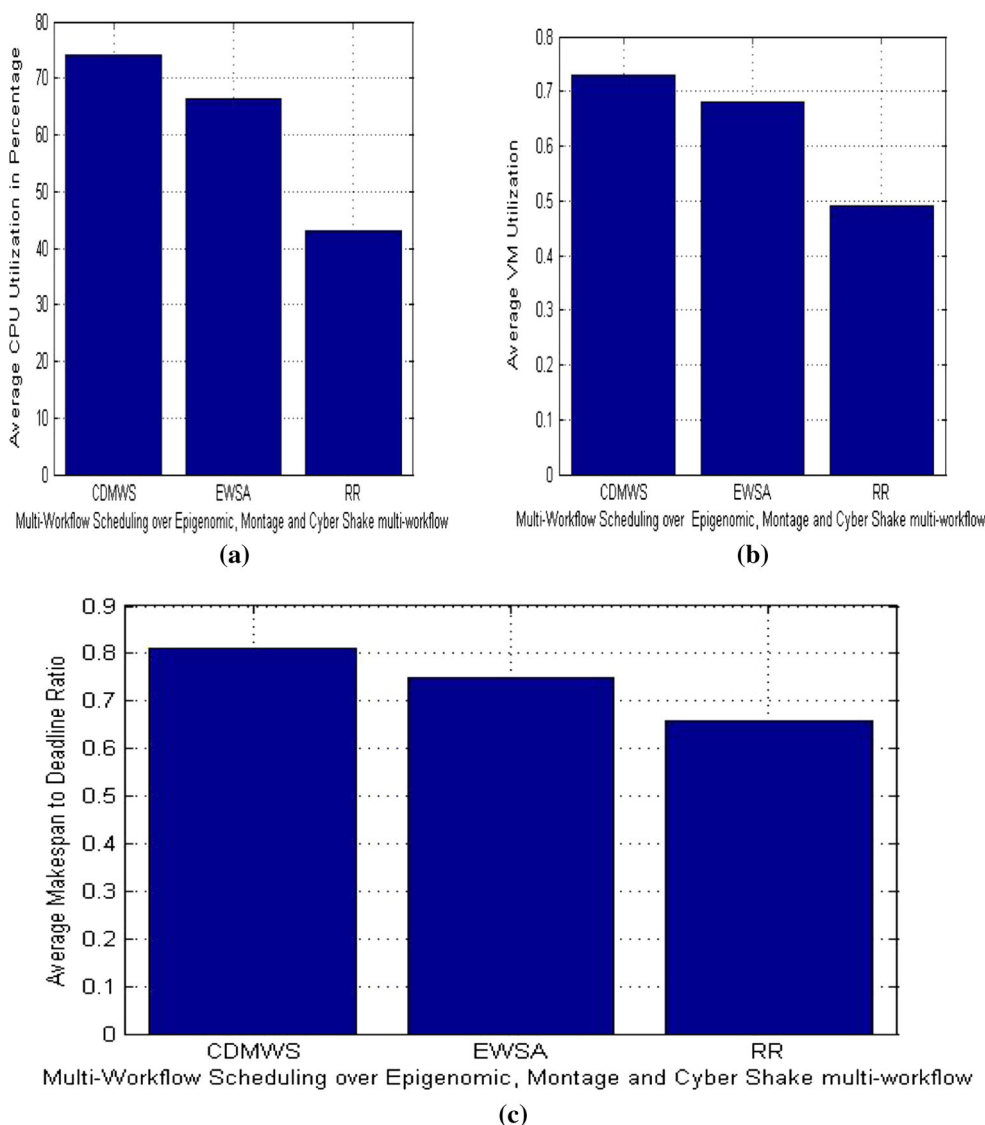


Fig. 13 Multi-workflow scheduling over Epigenomics, Montage and Cybershake workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio

different multi-workflow scheduling algorithms like LIGO and Cybershake.

At this time origination all the way through multi-workflow scheduling like LIGO, Montage and Cybershake workflow (Fig. 11a) is being examined that the CPU utilization is higher for CDMWS as compared to EWSA and RR. From Fig. 11b, we detect that the VM utilization is highly developed than the used algorithms. From Fig. 11c, it is analyzed that the average makespan-to-deadline ratio is advantaged than additional algorithms. Here, we have used three different algorithms.

Now the multi-workflow scheduling resembling like LIGO, Montage and Epigenomics workflow (Fig. 12a) is

being examined that the CPU utilization is higher for CDMWS as compared to EWSA and RR. From Fig. 12b, we become aware of that the VM utilization is extremely developed than the used algorithms. From Fig. 12c, it is analyzed that the average makespan-to-deadline ratio is fortunate than supplementary algorithms. At this instant also, we have drawn on three different algorithms.

At this moment of multi-workflow scheduling similar to Epigenomics, Montage and Cybershake workflow (Fig. 13a), we give something the once-over the CPU utilization is privileged for CDMWS as judge against to EWSA and RR. Figure 13b renders the VM utilization is bigger to the used algorithms. Figure 13c represents average makespan-

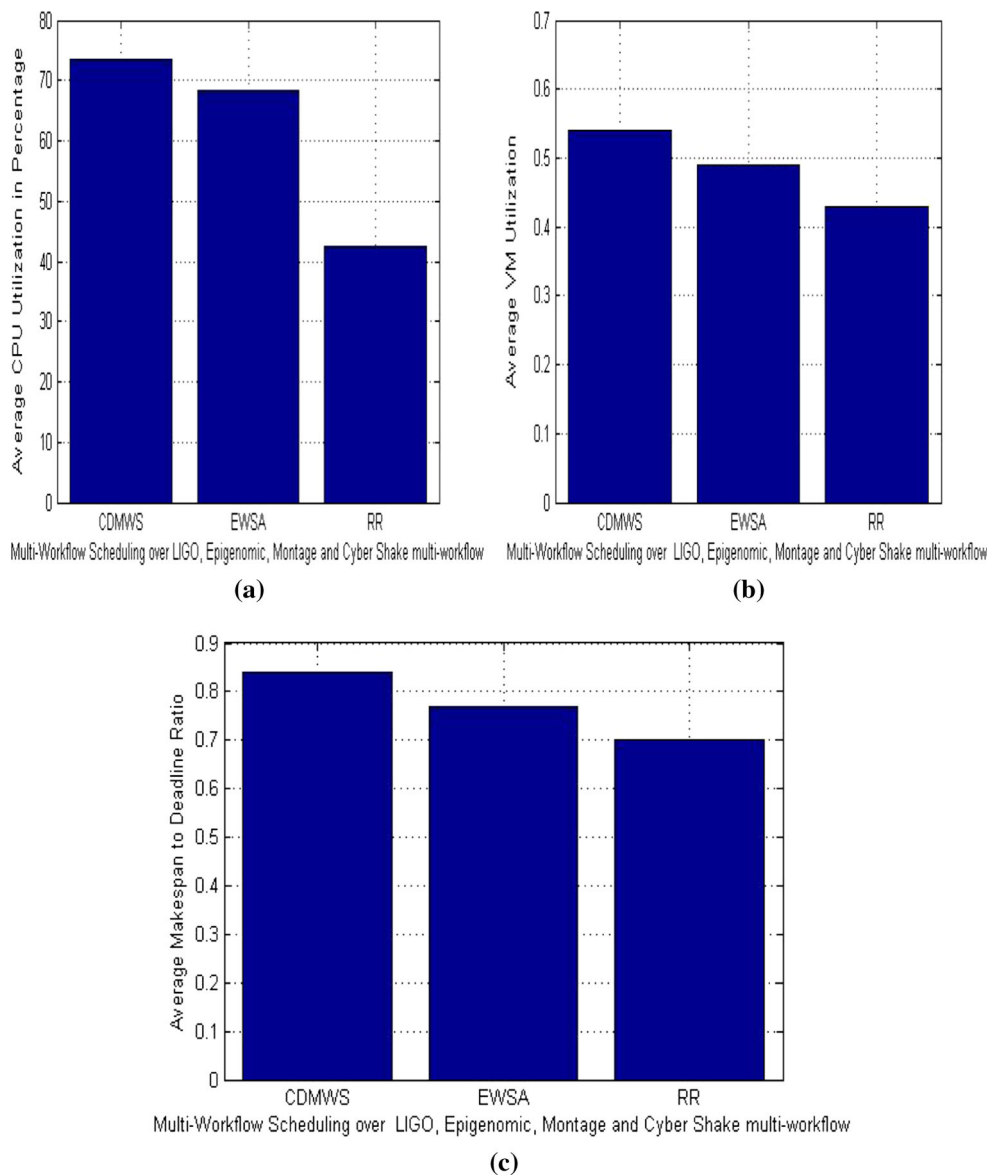


Fig. 14 Multi-workflow scheduling over LIGO, Epigenomics, Montage and Cybershake workflow. **a** Average CPU utilization, **b** average VM utilization, **c** average makespan-to-deadline ratio

to-deadline ratio is advantaged than other algorithms. Once more three different multi-workflow scheduling algorithms are utilized.

At this juncture, we use four different multi-workflow scheduling algorithms like LIGO, Epigenomics, Montage and Cybershake. The best part is once again we find average CPU utilization, VM utilization and average makespan-to-deadline ratio much higher than others. Figure 14a–c describes the whole theory as proven one.

Finally, we observe that the performance of the proposed algorithm is better than the existing algorithms, in terms of VMs utilization, CPU utilization and makespan-deadline

ratio, using scientific multi-workflows as well as sample multi-workflows. The superior performance of the proposed algorithm is summarized as follows:

1. Estimating execution time of the tasks on the fly which helps in creating suitable VM without wasting computing resources.
2. Creating suitable VMs helps in executing the tasks and reuses them when required so that the entire application completes within its deadline.
3. Scheduling same VM to execute the different tasks leads to minimization of a number of active VMs.

Table 1 The pseudo code of the proposed CDMWS algorithm

Algorithm: CDMWS	
/*Update phase*/	
1:	For each path P_i
2:	Find αMT^i , $ET(Tk^i)$, and CP^i using Eq. (1), Eq. (2), and Eq. (3) respectively
3:	For each task in the path
4:	Calculate and set ET using Eq. (4) and Eq. (5) respectively.
5:	End for
6:	End for
/*Task-VM mapping*/	
7:	For each level
8:	For each task Tk_i in the level
9:	If existing VM is available
	/* Required capacity VM already present in DC*/
10:	Reuse the VM and schedule the VM so that the execution of the task Tk_i is ETk_i
11:	Else
12:	VMM creates a new VM and schedule it so that the execution of the task Tk_i is ETk_i
13:	End if
14:	End for
15:	End for
16:	For each level
17:	For each task in the level
18:	Calculate ST and CT , using Eq. (6) and Eq. (7) respectively.
19:	End for
20:	End for

The execution of workflow with a minimum number of active VMs results in efficient computing utilization of data centers.

5 Conclusion

The most important endeavor of the proposed algorithm called CDMWS is to take full advantage of the resource utilization of the DC using VM reusability technique and complete the execution of the workflows of the multi-workflow application within their deadline correspondingly. To assess the proposed algorithm, it is executed over multi-workflows which combine with various scientific workflows and compare with other two different workflow scheduling algorithms. We have considered various performance metrics such as CPU utilization, makespan–deadline ratio and VM utilization to judge the performance of the proposed algorithm. From the simulation result, it is clear that the proposed algorithm successfully maximizes the VM utilization and CPU utilization as well as improves the makespan–deadline ratio while meeting the user-defined deadline of the corresponding multi-workflow applications.

Our future research plan is to extend the proposed algorithm for rescheduling tasks in response to changes in VMs and network loads.

References

- Xiong, K.; Perros, H.: Service Performance and Analysis in Cloud Computing, 978-0-7695-3708-5/09 \$25.00 © 2009 IEEE, pp. 693–700 (2009)
- Sotomayor, B.; Montero, R.S.; Llorente, I.M.; Foster, I.: Virtual Infrastructure Management in Private and Hybrid Clouds, 1089-7801/09/\$26.00 © 2009 IEEE (2009)
- Chatterjee, T.; Ojha, V.K.; Banerjee, S.; Biswas, U.; Snasel, V.: Design and implementation of a new datacenter broker policy to improve the QoS of a Cloud. In: Springer International Publishing Switzerland 2014, Proceedings of ICBA 2014, Advances in Intelligent Systems and Computing, vol. 303, pp 281–290 (2014)
- Banerjee, S.; Kar, S.; Biswas, U.: Development and analysis of a new cloudlet allocation strategy for QoS improvement in cloud. Arab. J. Sci. Eng. **40**(5), 1409–1425 (2014). (Springer, ISSN: 1319-8025)
- Yeo, C.; Buyya, R.: Service level agreement based allocation of cluster resources: handling penalty to enhance utility. In: Proceedings of the 7th IEEE international conference on cluster computing, Boston, USA (2005)
- Sousa, T.; Silva, A.; Neves, A.: Particle swarm based data mining algorithms for classification tasks. Parallel Comput. **30**(5), 767–783 (2004)
- Garg, S.K.; Toosi, A.N.; Gopalaiyengar, S.K.; Buyya, R.: SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter. J. Netw. Comput. Appl. **45**, 108–120 (2014)
- Koley, S.; Singh, N.: Cdroid: used in Fujitsu server for mobile cloud. GE Int. J. Eng. Res. **2**(7), 1–14 (2014). (ISSN: 2321-1717)
- Paton, N.W.; Aragão, M.A.T.; Lee, K.; Fernandes, A.A.A.; Sakellariou, R.: Optimizing utility in cloud computing through autonomic workload execution. IEEE Data Eng. Bull. **32**(1), 51–58 (2009)
- Hu, Y.; Wong, J.; Iszlai, G.; Litoiu, M.: Resource provisioning for cloud computing. In: CASCON'09: Proceedings of the 2009 conference of the Center for Advanced Studies on Collaborative Research, Ontario, Canada (2009)
- Fito, J.O.; Goiri, I.; Guitart, J.: SLA-driven elastic cloud hosting provider. In: Proceedings of the 18th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), Pisa, Italy (2010)
- Wu, Z.; Ni, Z.; Gu, L.; Liu, X.: A revised discrete particle swarm optimization for cloud workflow scheduling. In: Proceedings of the IEEE International Conference on Computational Intelligence and Security (CIS), pp. 184–188 (2010)
- Zhu, Z.; Bi, J.; Yuan, H.; Chen, Y.: SLA based dynamic virtualized resources provisioning for shared cloud data centers. In: Proceedings of 2011 IEEE International Conference on Cloud Computing (CLOUD), Washington DC, USA (2011)
- Mao, M.; Humphrey, M.: Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In: Proceedings of the International Conference on High-Performance Computing, Networking, Storage and Analysis (SC), pp. 1–12 (2011)
- Byun, E.K.; Kee, Y.S.; Kim, J.S.; Maeng, S.: Cost optimized provisioning of elastic resources for application workflows. Future Gen. Comput. Syst. **27**(8), 1011–1026 (2011)
- Sharma, U.; Shenoy, P.; Sahu, S.; Shaikh, A.: A cost-aware elasticity provisioning system for the cloud. In: Proceedings of the



- 31st International Conference on Distributed Computing Systems (ICDCS), Minneapolis, Minnesota, USA (2011)
17. Bonvin, N.; Papaioannou, T.G.; Aberer, K.: Autonomic SLA-driven provisioning for cloud applications. In: Proceedings of the 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Newport Beach, CA, USA (2011)
 18. Abrishami, S.; Naghibzadeh, M.: Deadline-constrained workflow scheduling in software as a service Cloud. *Sci. Iran. Trans. D Comput. Sci. Eng. Electr. Eng.* **19**(3), 680–689 (2011)
 19. Abrishami, S.; Naghibzadeh, M.; Epema, D.: Deadline-constrained workflow scheduling algorithms for IaaS Clouds. *Future Gen. Comput. Syst.* **23**(8), 1400–1414 (2012)
 20. Malawski, M.; Juve, G.; Deelman, E.; Nabrzyski, J.: Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. In: Proceedings of the International Conference on High-Performance Computing, Networking, Storage and Anal, (SC), 22 (2012)
 21. Xiao, Z.; Song, W.; Chen, Q.: Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Trans. Parallel Distrib. Syst.* **24**(6), 1107–1117 (2013)
 22. Antonescu, A.-F.; Robinson, P.; Braun, T.: Dynamic SLA management with forecasting using multi-objective optimization. In: Proceeding of 2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), Ghent, Belgium (2013)
 23. Rodriguez, M.A.; Buyya, R.: Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. *IEEE Trans. Cloud Comput.* **2**(2), 222–235 (2014)
 24. Saxena, S.; Saxena, D.: EWSA: an enriched workflow scheduling algorithm in cloud computing (2015). DOI:[10.1109/CCCS.2015.7374202](https://doi.org/10.1109/CCCS.2015.7374202)

