Cost Minimization Using Deadline Constrained

P PHANINDRA VARSHIT
Dept. Computer Science and Engineering
Anurag University
Hyderabad, Telangana, India.
phanivarshit@gmail.com

T ABHINAV REDDY
Dept. Computer Science and Engineering
Anurag University
Hyderabad, Telangana, India.
abhinavreddy038@gmail.com

P SIDDARTHA
Dept. Computer Science and Engineering
Anurag University
Hyderabad, Telangana, India.
siddarthayadav27@gmail.com

***Abstract*—** Over the past decade, there has been a significant surge in interest towards leveraging commercially available cloud computing resources for scientific computations. However, scheduling multiple workflows in cloud computing poses a formidable challenge due to various non-functional constraints and complex resource requirements. Existing scheduling algorithms, predominantly relying on search-based approaches, often incur substantial computational overhead and protracted execution times.

Addressing this issue, this paper introduces a novel approach termed Deadline-Constrained Cost Minimization (DCCM) algorithm for efficient resource scheduling in cloud computing environments. The key innovation lies in task grouping based on their scheduling deadline constraints and inter-task data dependencies. Unlike conventional methods, DCCM prioritizes meeting user-defined deadlines by subdividing tasks into different levels based on their urgency.

Simulation experiments demonstrate that DCCM outperforms state-of-the-art approaches, achieving notably higher success rates in meeting deadlines while minimizing costs. This underscores the efficacy of the proposed algorithm in enhancing the efficiency and reliability of cloud computing resource scheduling.

I. Introduction

In recent years, there has been a notable increase in research dedicated to addressing the scheduling complexities encountered in scientific workflows, which has been largely spurred by the growing adoption of cloud computing platforms for executing such workflows. While utilizing cloud resources offers numerous benefits, users often encounter challenges in selecting the most appropriate resource configurations for their applications. Traditional scheduling approaches have primarily focused on minimizing workflow execution time, employing techniques such as cluster, list, and duplicate scheduling.

Recent surveys have highlighted two prominent categories of workflow scheduling in cloud computing: budget-constrained and deadline-constrained, along with multi-objective scheduling. Budget-constrained strategies aim to reduce workflow execution time, while deadline-constrained strategies prioritize minimizing scheduling costs.

The scalability of cloud computing within distributed systems poses optimization challenges, particularly for inter-dependent tasks. Cost minimization becomes a complex problem when considering both cost and execution time simultaneously. Despite challenges such as acquisition delays for leased virtual machines, instance heterogeneity, and varying cloud system performance, simultaneous optimization holds the promise of significant cost reductions, particularly for deadline-sensitive applications such as disaster recovery and weather forecasting. As distributed intelligence becomes increasingly integral to managing complex systems like future networks, efficient cloud scheduling and management solutions are essential, particularly as network functions migrate towards the network edge.

In this context, we propose a dynamic resource scheduling strategy for cloud computing. Our approach involves grouping identical tasks based on their priorities and constraints to minimize queuing overhead. Subsequently, we distribute the workflow deadline across multiple task groups. We implemented a dynamic and scalable Virtual Machine (VM) scheduling strategy and conducted extensive simulations to evaluate its performance. Results indicate that our proposed algorithm outperforms existing task scheduling approaches in terms of meeting deadlines and mean load.

The subsequent sections of the paper are structured as follows: Section II provides a review of related work, while Section III outlines the system model. Our proposed algorithm is detailed in Section IV and compared with existing schemes using scientific workflows in Section V. Finally, conclusions are drawn in Section VI, where future research directions are also discussed.

II. Literature Review

The academic literature has extensively addressed the challenge of scheduling scientific workflows [16]. Various strategies have been proposed to enhance efficient resource

provisioning and scheduling, drawing from heuristics, meta-heuristics, and search-based techniques [10]. Typically, resource scheduling algorithms prioritize minimizing workflow execution time while considering two critical factors: monetary cost and deadline. This process typically involves two primary phases: resource provisioning, which determines the necessary resources for workflow execution, and resource scheduling, which handles task scheduling, placement, and execution. While many algorithms in cloud computing systems tackle both resource provisioning and scheduling, significant attention has been given to resource scheduling strategies [17, 18].

Particle Swarm Optimization (PSO) has emerged as a widely used approach in resource scheduling, initially introduced in [19]. PSO is a self-adaptive optimization technique that utilizes particle social behavior to solve task allocation and resource scheduling problems [2]. In another study [20], a resource scheduling strategy based on PSO was proposed to reduce the cost of single workflow execution and balance task loads based on available resources. Additionally, a Deadline Constrained Level Based (DCLB) approach was presented in [21], utilizing Level Load Balancing to refine deadline distribution and minimize communication costs. Wu et al. [22] also introduced a PSO algorithm aimed at minimizing cost and execution time while considering budget and deadline constraints. This algorithm targeted continuous optimization problems lacking prior information.

Rodriguez and Buyya [10] introduced a static PSO algorithm for resource scheduling in complex scientific workflows, considering cloud service characteristics such as virtual machine heterogeneity, lease time intervals, pricing models, and acquisition delays. Their algorithm effectively reduced execution costs and met user-defined deadlines through global optimization techniques. However, a common limitation of these PSO algorithms is their inability to manage heterogeneous resources. Notably, they are unsuitable for Infrastructure as a Service (IaaS) deployment due to their neglect of task execution order and the number and type of resources to be leased. For instance, the static PSO approach [10] randomly assigned tasks to instances, disregarding workflow characteristics and structure.

In a separate study [8], authors developed an IaaS Cloud Partial Critical Paths (IC-PCP) model for IaaS deployment. IC-PCP scheduled all tasks to the same VM instance, which had to be the cheapest instance meeting the user-defined deadline. While IC-PCP did not add overhead to critical paths, it overlooked VM boot times. This limitation was addressed by Calheiros and Buyya [23], who extended the Enhanced IC-PCP Algorithm with Replication (EIPR). EIPR replicated tasks using idle instances and surplus budgets, showing potential for meeting user-defined deadlines through increased task replication. However, this resulted in high computational overhead, despite mitigating variable performance effects by exploiting billing schemes and cloud elasticity. Notably, performance decreased when task execution times neared billing periods.

This section explores various resource scheduling strategies proposed to optimize scientific workflows within cloud computing environments. Firstly, a budget-constrained algorithm [24] was introduced to address optimization challenges in workflow scheduling. This approach entailed proportionally distributing the total user-defined budget across tasks based on their average execution times. Moreover, Arabnejad and Barbosa [9] introduced the concept of worthiness as a decisive attribute in scientific workflow applications. Worthiness, derived from considerations of cost and execution time, played a pivotal role in task selection.

Wu et al. [25] put forward the PCB-B2 algorithm, employing a binary search method for budget distribution within a constrained framework. Additionally, a task swapping strategy [26] was proposed, prioritizing tasks with shorter execution times to reduce optimization costs. Furthermore, the DCLB algorithm [21] proposed a level-based scheduling approach aiming to minimize workflow completion time while adhering to user-defined deadlines. However, the specifics regarding the logical level partitioning and task dependencies remained somewhat unclear.

An algorithm based on deadline distribution factors, JITC [27], was designed to mitigate performance issues in virtual machines (VMs), focusing on variations in VM performance, cloud system heterogeneity, and resource acquisition delays. Although effective in meeting deadlines, JITC incurred high computational costs, particularly when deadline factors were low. Hadiri et al. [28] highlighted acquisition delay as a significant bottleneck in workflow scheduling and proposed the CEDA technique to minimize task execution time while meeting deadlines. However, CEDA's applicability to large workflows was questioned.

Topcuoglu et al. [3] introduced early resource scheduling algorithms for heterogeneous computing, including HEFT and CPOP, aimed at achieving fast scheduling times and high performance simultaneously. While HEFT focused on minimizing earliest finish times through incentive-based

task strategies, CPOP prioritized tasks based on upward and downward ranks. However, HEFT lacked load balancing considerations and scheduled workflows sequentially.

Another notable contribution was the BDAS algorithm [29], a budget-aware approach for resource scheduling in heterogeneous e-scientific workflows. This approach attempted to satisfy both deadline and budget constraints by introducing heterogeneous instances based on cost and time considerations. Furthermore, Verma and Kaushal [30] proposed BDHEFT, an extension of HEFT emphasizing budget and deadline considerations through task identification based on a descending ranking mechanism. Despite promising results in reducing workflow makespan and execution costs, BDHEFT's reliance on prior information limited its adaptability.

In this paper, we explore task grouping based on related dependencies such as control data and user data to mitigate computational overhead. We devised a deadline-constrained algorithm to minimize costs and meet user-defined deadlines. The subsequent section presents our system model, followed by a detailed description of the algorithm.

III. Proposed Method

In this segment, we present the proposed algorithm crafted to minimize costs while ensuring compliance with budget and deadline constraints in the scheduling of scientific workflows within cloud computing environments. The algorithm unfolds in three core phases: workflow partitioning, deadline distribution, and task selection and resource scheduling.

A. Workflow Partitioning:

In the workflow partitioning phase, tasks undergo a meticulous process of prioritization and grouping, hinged upon similarities in functionality and dependencies. This stage aims to streamline the workflow by categorizing tasks into cohesive groups. To determine task priorities, we employ the upward rank (UR) method. This method efficiently assigns priorities by evaluating the critical path's length from the initial task to the final one. Furthermore, it gauges the degree of dependency between tasks, facilitating the clustering of tasks sharing similar functionalities.

$$UR_{t_i} = ETC_{t_i} + \max_{t_i \in predt_j} DT_{ij} + UR_{t_j} \tag{1}$$

We compute the degree of dependency

$$L_{t_i} = \sum_{t_i \in predt_j} E^{i}_{t_i} + \sum_{t_i \in predt_j} E^{o}_{t_i} \tag{2}$$

B. Task Selection:

In this phase, tasks are queued for execution, with grouped tasks being executed simultaneously once their predecessors are completed. Prior to execution, deadlines are distributed among tasks to ensure overall deadline adherence. The available time (AT) for each task level is computed to meet the workflow deadline. Subsequently, sub-deadlines for individual tasks are determined to regulate task execution.

$$AT_W = D_W - TC_{ms} \tag{3}$$

where DW is the deadline of the workflow and TCms represents the makespan. To meet with the overall deadline, we also compute the available time for group level task ATGL in Eqn. 4.

$$AT_{GL} = \frac{ETC^{GL}_{t_n}}{ETC^{W}_{t_n}} \times AT_W \tag{4}$$

C. Grouping of Tasks:

Tasks are grouped using the horizontal grouping method detailed by Rodriguez and Buyya et al. [6], aimed at ensuring tasks with similar characteristics and complexities are processed in parallel. This grouping approach entails assigning tasks to groups based on various attributes, including data size, input/output size, computing cost, data distribution pattern, and type. Each group may contain either a single task or multiple tasks, with homogeneous tasks within a group sharing a single immediate processor to enhance efficiency.

D. Deadline Distribution:

The deadline distribution process, illustrated in Algorithm 1, involves selecting the most cost-effective virtual machine (VM) among the available instances. This selection ensures that the estimated completion time for each task aligns with the user-defined deadline. In cases where the estimated completion time exceeds the deadline, the next least expensive VM is chosen. Following this, the available time (ATW) is distributed proportionally across all task levels based on their durations using the ATGL metric. Subsequently, sub-deadlines (Dtisub) are computed for each task to regulate task execution, ensuring that tasks within a group share the same sub-deadline, equivalent to the start time of a successor task. Algorithm 2 delineates the dynamic scheduling of task groupings using sub-deadline distribution, wherein tasks are prioritized based on descending order of priorities for each level. Identified task

groups are then arranged in a priority queue and sorted based on relevance before being added to the execution queue.

Algorithm 1 Deadline Distribution

Workflow W, Deadline D, Estimated time for Computation of tasks ETC
$VM_{cheap} \leftarrow$, obtain cheap VM where $TC_{ms} > D_w$
if VM_{cheap} = null **then**
  while $VM_{cheap} \neq$ null **do**
    $VM_{cheap} \leftarrow$ next VM_{cheap} type $TC_{ms} > D_w$
  end while
end if
Compute Available Time AT_w according to Equation (11)
Proportionally allocate AT_w on all levels
Proportionally allocate AT_w on all task on each level
For each task, compute estimated sub-deadline
For each group level task, update sub-deadline as finish time

IV. PERFOMANCE EVALUATION

In this section, the performance of the proposed algorithm was compared with [30], [32], and [3]. For this comparison, we used CloudSim [33] to conduct these experiments. The simulation environment was configured as an IaaS provider cloud with a single data centre and six different types

Algorithm 2 Sub-Task Distribution

Require: A connected graph $W = (T, E)$
Add t_{entry} and t_{exit} with interrelated edges
for all $t_i \in T$ **do**
  Compute UR_{t_i} according to Equation (9)
  Compute L_{t_i} according to Equation (10)
end for
Classify task based on relevance
Identify all group level task
Compute group level task based on dependencies
for all Group level task **do**
  Prioritise based on relevance
  Place group level task in priority queue
  Place group level task ready in execution queue
end for

TABLE 1. VM Instance Specification

VM Types	Memory(GB)	ECU	Price ($/h)
m3.medium	3.75	3	0.067
m4.large	8	6.5	0.126
m3.xlarge	15	13	0.266
m4.2xlarge	32	26	0.504
m4.4xlarge	64	53.5	1.008
m4.10xlarge	160	124.5	2.520

of VMs. The configurations of the VM types which were based on the Amazon EC2 cloud offering was presented in Table 1. The Pegasus workflow generator was used in the generation of different sizes of synthetic workflow application. The evaluation of the proposed scheme with other algorithms is evaluated using four scientific workflows (Cybershake, Epigenomics, Montage and LIGO) which have been widely used in literature. These workflows are well explained in [34] and [35].

A. Performance Metrics:

To assess the effectiveness of both the proposed scheme and the compared approaches, we utilized a set of performance metrics. These metrics include:

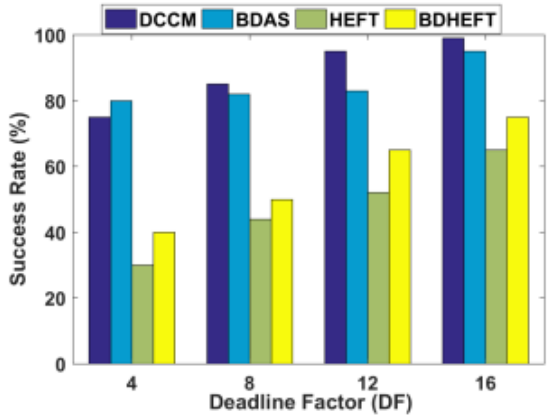
• Success Rate (SR): The success rate of a resource scheduling algorithm measures the percentage of valid schedules achieved in an experiment. This metric is calculated according to Equation 5.

$$SR = \frac{W_s}{W_t} \times 100 \tag{5}$$

where Ws represents the number of successful workflows and Wt represents the total workflows in the experiment.

• Cost Ratio: Cost Ratio is defined as the ratio of total monetary cost and the cost of the cheapest workflow schedule. The monetary cost is computed as shown in Eqn. 6.

$$MC = \frac{MC_W}{VM_{cheap}} \tag{6}$$

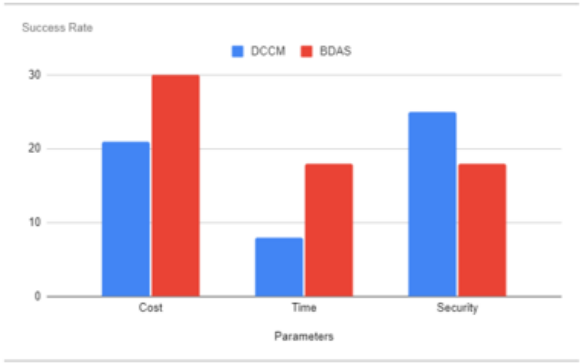


V. RESULTS

Based on the information depicted in the bar graph, it is evident that the DCCM (Dynamic Cluster-based Collaborative Monitoring) algorithm showcases superior performance compared to the BDAS (Bayesian Deep Anomaly Segmentation) algorithm across three crucial parameters: cost, time, and security. The bar representing the cost parameter for the DCCM algorithm is notably shorter compared to that of the BDAS algorithm. This indicates that the DCCM algorithm incurs significantly lower costs in terms of resource utilization or implementation expenses. This cost-effectiveness suggests that the DCCM algorithm may offer financial advantages over the BDAS algorithm. Similarly, the DCCM algorithm exhibits a substantially shorter bar for the time parameter, signifying that it requires notably less time for execution or processing compared to the BDAS algorithm. This efficiency in terms of processing time implies that the DCCM algorithm can accomplish tasks more swiftly, potentially leading to improved productivity and responsiveness in practical applications. In terms of security, the DCCM algorithm again demonstrates superiority over the BDAS algorithm, with a lower security rating. While both algorithms likely prioritize security, the lower security rating associated with the DCCM algorithm suggests that it may offer comparable or even better security measures with potentially fewer resource requirements or computational overhead.

The comprehensive comparative analysis underscores the overall superiority of the DCCM algorithm across the evaluated parameters. Its efficiency, cost-effectiveness, and security credentials position it as a compelling choice for the project. This information can serve as a compelling rationale for selecting the DCCM algorithm over the BDAS algorithm, as it notonly outperforms its counterpart but also maintains a commendable level of security.

The data presented in the table shows that the proposed method significantly outperforms the existing method across the key metrics of cost and time. The proposed method reduces the cost by nearly half, from $0.98 to $0.45, while also drastically cutting the processing time from 12 seconds to just 2 seconds, a 5-fold improvement. These substantial improvements in efficiency and cost-effectiveness indicate that the proposed method represents a superior approach compared to the existing method, and its implementation could lead to notable benefits for the project or organization.



The cost of storing files of identical sizes can fluctuate depending on the specified deadlines due to the Deadline Constraint Cost Minimization (DCCM) approach's methodology. This strategy focuses on selecting the most optimal virtual machine (VM) for each file, tailored to meet the specified deadlines while minimizing costs. Consequently, varying deadlines may prompt different VM selections by DCCM, resulting in differing costs for file storage. Essentially, the choice of deadline influences the selection criteria for VMs, impacting the overall cost optimization strategy employed by DCCM.

VI. CONCLUSION

This study introduces the DCCM algorithm designed for resource scheduling within cloud computing environments. DCCM specifically targets cost reduction within user-defined deadlines. Through evaluation and comparison with four established scientific workflows, our experimental findings reveal that the proposed scheme consistently outperforms existing approaches by 16-25% in terms of planning success rate and makespan cost ratio. As part of our future endeavors, we aim to develop a dual-objective scheduling model that simultaneously minimizes both makespan and overall monetary service cost.

VII. REFERENCES

[1] Y.-K. Kwok, ‘‘Parallel program execution on a heterogeneous PC cluster using task duplication,’’ in Proc.

9th Heterogeneous Comput. Workshop (HCW), 2000, pp. 364–374.

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, ‘‘Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility,’’ Future Generat. Comput. Syst., vol. 25, no. 6, pp. 599–616, 2009.

[3] H. Topcuoglu, S. Hariri, and M.-Y. Wu, ‘‘Performance-effective and lowcomplexity task scheduling for heterogeneous computing,’’ IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3, pp. 260–274, Mar. 2002.

[4] J. J. Durillo and R. Prodan, ‘‘Multi-objective workflow scheduling in Amazon EC2,’’ Cluster Comput., vol. 17, no. 2, pp. 169–189, Jun. 2014.

[5] M. Malawski, K. Figiela, M. Bubak, E. Deelman, and J. Nabrzyski, ‘‘Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization,’’ Sci. Program., vol. 2015, pp. 1–13, Jan. 2015.

[6] M. A. Rodriguez and R. Buyya, ‘‘Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods,’’ ACM Trans. Auto. Adapt. Syst., vol. 12, no. 2, pp. 1–22, May 2017.

[7] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, ‘‘Cloud computing resource scheduling and a survey of its evolutionary approaches,’’ ACM Comput. Surv., vol. 47, pp. 63:1–63:33, Jul. 2015.

[8] S. Abrishami, M. Naghibzadeh, and D. H. J. Epema, ‘‘Deadlineconstrained workflow scheduling algorithms for infrastructure as a service clouds,’’ Future Generat. Comput. Syst., vol. 29, no. 1, pp. 158–169, 2013.

[9] H. Arabnejad and J. G. Barbosa, ‘‘A budget constrained scheduling algorithm for workflow applications,’’ J. Comput., vol. 12, no. 4, pp. 665–679, Dec. 2014.

[10] M. A. Rodriguez and R. Buyya, ‘‘Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds,’’ IEEE Trans. Cloud Comput., vol. 2, no. 2, pp. 222–235, Apr. 2014.

[11] W. Chen, G. Xie, R. Li, Y. Bai, C. Fan, and K. Li, ‘‘Efficient task scheduling for budget constrained parallel applications on heterogeneous cloud computing systems,’’ Future Gener. Comput. Syst., vol. 74, pp. 1–11, Sep. 2017.

[12] Z. Cai, X. Li, R. Ruiz, and Q. Li, ‘‘A delay-based dynamic scheduling algorithm for bag-of-task workflows with stochastic task execution times in clouds,’’ Future Gener. Comput. Syst., vol. 71, pp. 57–72, Jan. 2017.

[13] J. Chen, C. Du, F. Xie, and B. Lin, ‘‘Scheduling non-preemptive tasks with strict periods in multi-core real-time systems,’’ J. Syst. Archit., vol. 90, pp. 72–84, Oct. 2018.

[14] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, and J. Wen, ‘‘Deadline-constrained cost optimization approaches for workflow scheduling in clouds,’’ IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 12, pp. 3401–3412, Aug. 2017.

[15] S. Abrishami and M. Naghibzadeh, ‘‘Deadline-constrained workflow scheduling in software as a service cloud,’’ Scientia Iranica, vol. 19, no. 3, pp. 680–689, Jun. 2012.

[16] S. Singh and I. Chana, ‘‘A survey on resource scheduling in cloud computing: Issues and challenges,’’ J. Grid Comput., vol. 14, no. 2, pp. 217–264, Jun. 2016.

[17] M. A. Rodriguez and R. Buyya, ‘‘A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments,’’ Concurrency Comput., Pract. Exper., vol. 29, no. 8, p. e4041, Apr. 2017.

[18] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. Netto, and A. N. Toosi, ‘‘A manifesto for future generation cloud computing: Research directions for the next decade,’’ ACM Comput. Surv., vol. 51, no. 5, p. 105, 2018.

[19] J. Kennedy and R. Eberhart, ‘‘Particle swarm optimization,’’ in Proc. Int. Conf. Neural Netw. (ICNN), vol. 4, Aug. 1995, pp. 1942–1948.

[20] S. Pandey, L. Wu, S. M. Guru, and R. Buyya, ‘‘A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments,’’ in Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl., 2010, pp. 400–407.

[21] S. Omranian-Khorasani and M. Naghibzadeh, ‘‘Deadline constrained load balancing level based workflow scheduling for cost optimization,’’ in Proc. 2nd IEEE Int. Conf. Comput. Intell. Appl. (ICCIA), Sep. 2017, pp. 113–118.

[22] Z. Wu, Z. Ni, L. Gu, and X. Liu, ‘‘A revised discrete particle swarm optimization for cloud workflow scheduling,’’ in Proc. Int. Conf. Comput. Intell. Secur., Dec. 2010, pp. 184–188.

[23] R. N. Calheiros and R. Buyya, ‘‘Meeting deadlines of scientific workflows in public clouds with tasks

replication,'' IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 7, pp. 1787–1796, Jul. 2014.

[24] W. Zheng and R. Sakellariou, ''Budget-deadline constrained workflow planning for admission control,'' J. Grid Comput., vol. 11, no. 4, pp. 633–651, Dec. 2013.

[25] F. Wu, Q. Wu, Y. Tan, R. Li, and W. Wang, ''PCP-B2 : Partial critical path budget balanced scheduling algorithms for scientific workflow applications,'' Future Gener. Comput. Syst., vol. 60, pp. 22–34, Jul. 2016.

[26] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. D. Dikaiakos, Scheduling Workflows With Budget Constraints. Boston, MA, USA: Springer, 2007, pp. 189–202.

[27] J. Sahni and D. P. Vidyarthi, ''A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment,'' IEEE Trans. Cloud Comput., vol. 6, no. 1, pp. 2–18, Mar. 2015.

[28] R. A. Haidri, C. P. Katti, and P. C. Saxena, ''Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing,'' J. King Saud Univ.-Comput. Inf. Sci., vol. 32, no. 6, pp. 666–683, Jul. 2020.

[29] V. Arabnejad, K. Bubendorfer, and B. Ng, ''Budget and deadline aware e-Science workflow scheduling in clouds,'' IEEE Trans. Parallel Distrib. Syst., vol. 30, no. 1, pp. 29–44, Jan. 2019.

[30] A. Verma and S. Kaushal, ''Cost-time efficient scheduling plan for executing workflows in the cloud,'' J. Grid Comput., vol. 13, no. 4, pp. 495–506, Dec. 2015.

[31] I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, Workflows for E-Science: ScientificWorkflows for Grids. Cham, Switzerland: Springer 2014.

[32] D. Poola, S. K. Garg, R. Buyya, Y. Yang, and K. Ramamohanarao, ''Robust scheduling of scientific workflows with deadline and budget constraints in clouds,'' in Proc. IEEE 28th Int. Conf. Adv. Inf. Netw. Appl., May 2014, pp. 858–865.

[33] R. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, ''CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,'' Softw., Pract. Exper., vol. 41, no. 1, pp. 23–50, Aug. 2011.

[34] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, ''Characterizing and profiling scientific workflows,'' Future Gener. Comput. Syst., vol. 29, no. 3, pp. 682–692, Mar. 2013.

[35] J.-S. Vöckler, G. Juve, E. Deelman, M. Rynge, and B. Berriman, ''Experiences using cloud computing for a scientific workflow application,'' in Proc. 2nd Int. Workshop Sci. Cloud Comput., New York, NY, USA, Jun. 2011, pp. 15–24.