IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 1 February 2023, accepted 23 February 2023, date of publication 17 March 2023, date of current version 24 April 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3258682

RESEARCH ARTICLE

Deadline-Constrained Cost Minimisation for Cloud Computing Environments

SAMUEL MANAM[1], KLAUS MOESSNER[1,2], (Senior Member, IEEE), AND SERDAR VURAL[1]
[1]5G Innovation Centre, Institute for Communication Systems (ICS), University of Surrey, GU2 7XH Guildford, U.K.
[2]Faculty of Electronic Engineering and Information Technology, Chemnitz University of Technology, 09107 Chemnitz, Germany

Corresponding author: Samuel Manam (s.manam@surrey.ac.uk)

This work was supported in part by the European Union's Horizon 2020 Research and Innovation Programme through the Project DEDICAT 6G under Grant 101016499

ABSTRACT The interest in performing scientific computations using commercially available cloud computing resources has grown rapidly in the last decade. However, scheduling multiple workflows in cloud computing is challenging due to its non-functional constraints and multi-dimensional resource requirements. Scheduling algorithms proposed in literature use search-based approaches which often result in very high computational overhead and long execution time. In this paper, a Deadline-Constrained Cost Minimisation (DCCM) algorithm is proposed for resource scheduling in cloud computing. In the proposed scheme, tasks were grouped based on their scheduling deadline constraints and data dependencies. Compared to other approaches, DCCM focuses on meeting the user-defined deadline by sub-dividing tasks into different levels based on their priorities. Simulation results showed that DCCM achieved higher success rates when compared to the state-of-the-art approaches.

INDEX TERMS Cloud computing, deadline constraints, resource scheduling, scientific workflow, optimisation.

I. INTRODUCTION

Over last decade, there has been a large body of studies on scheduling problems in scientific workflows. This is due to the increase in the execution of scientific workflows in cloud computing platforms. Although this trend is advantageous, it is difficult for users to select resource configurations that are suitable for their applications [1], [2]. The focus on traditional workflow scheduling is at minimising the execution time of the workflow, using cluster, list and duplicate scheduling techniques as described in [3] and [4]. Recent surveys [5], [6] have shown that two main categories of workflow scheduling (budget-constrained and deadline-constrained) have been used frequently for resource scheduling in cloud computing in addition to multi-objective workflow scheduling. Budget-constrained scheduling strategies focus on the reduction in the time of execution of workflows while deadline-constrained strategies focused on the optimisation of the scheduling cost of the workflow [6], [7].

One of the main benefits of cloud computing is its scalability when applied to distributed systems. However, this often results in optimisation problems that are complex, in particular for tasks that are inter-dependent. One such problem is how to minimise cost. When cost and execution time are taken into consideration at the same time, the problem of optimising cost can be solved [8], [9], [10], [11], [12], [13]. This approach is very attractive as it can efficiently provide cost reductions for deadline-sensitive applications, as well as others such as disaster recovery, weather forecasting etc. However, a major drawback in this approach is the inconsistency in system requirements such as the acquisition delay which is the time taken for initialisation of a leased virtual machine [8], [9], instance heterogeneity [5], or the varying levels of performance in cloud systems [14], [15].

Complex systems like future networks rely increasingly on distributed intelligence for their management and as network

The associate editor coordinating the review of this manuscript and approving it for publication was Fan-Hsun Tseng.

functions are increasingly pushed towards the network edge, cloud scheduling and management solutions are necessary. We propose a dynamic resource scheduling strategy for cloud computing. To reduce processing cost, identical tasks were grouped together based on their priorities and constraints. This was done to reduce overheads from queuing, then the deadline of the entire workflow was sub-distributed across multiple groups of tasks. We implemented a resource scheduling strategy for Virtual Machines (VMs) that were dynamic and scalable. Extensive simulations were carried out to evaluate the performance of the proposed scheme. The simulation results showed that our proposed algorithm performs better than other existing task scheduling algorithms in terms of success rate at meeting the deadline and mean load.

The rest of the paper is organised as follows. We reviewed related work in Section II and presented the system model in Section III. The proposed algorithm is presented in Section IV and compared with existing schemes using scientific workflows in Section V. The paper was concluded in Section VI where future work was also discussed.

II. RELATED WORK

The problem of scheduling scientific workflows has been well studied in the literature. Several strategies have been proposed for effective resource provisioning and scheduling of scientific workflows [16]. These algorithms are based on heuristics, meta-heuristics and search-based techniques [10]. Generally, resource scheduling algorithms focus on the reduction in execution time of workflows with two main factors in consideration, monetary cost and deadline. Resource allocation in cloud computing comprises of two phases, resource provisioning and scheduling. Resource provisioning is responsible for the determination of resources required for the execution of workflows. Resource scheduling pays attention to the scheduling, placement and execution of tasks. Although most scheduling algorithms in cloud computing systems propose resource provisioning and scheduling algorithms for cloud computing systems. Prior research shows that much attention has been paid to resource scheduling strategies [17], [18].

One of the widely used approaches in resource scheduling is Particle Swarm Optimisation (PSO) which was introduced by the authors in [19]. PSO is a self-adaptive optimisation technique that relies on a particles' social behaviour to solve task allocation and resource scheduling problems [2]. In [20], the authors proposed a resource scheduling strategy based on PSO. The proposed algorithm was aimed at reducing the cost of a single workflow execution and load balancing of task based on available resources. In [21], authors proposed Deadline Constrained Level Based (DCLB) which uses Level Load Balancing to refine deadline distribution as well as attaining lower communication cost. Wu et al. [22] also proposed a PSO algorithm to minimise cost and execution time, whilst considering budget and deadline constraints. The proposed algorithm focused on continuous optimisation problems that have no prior information. Rodriguez and Buyya [10] propose a static PSO algorithm for resource scheduling in complex scientific workflows. They introduced several characteristics of cloud services including heterogeneity of virtual machines, lease time interval, pricing model, and acquisition delay. Their algorithm effectively reduced the execution cost and met the deadline defined by the user using a global optimisation technique. One major disadvantage common to the highlighted PSO algorithm was their ability to manage heterogeneous resources. However, these algorithms are not suitable for deployment in Infrastructure as a Service (IaaS) as they do not consider the order of execution of task and the number and type of resources to be leased. The static PSO in [10] assigned tasks to instances randomly, thereby neglecting the characteristics and structure of the workflow.

In [8], the authors developed an IaaS Cloud Partial Critical Paths (IC-PCP) for the IaaS model. IC-PCP scheduled all task to the same VM instance which must be the cheapest instance that must meet the deadline given by the user. In IC-PCP, the critical path or set of tasks that is related to the exit node must first be found, then scheduled on the same VM. IC-PCP does not add additional overhead to the critical paths. However, the algorithm does not consider the boot time of VMs. Such shortcomings were identified by Calheiros and Buyya [23] who extended the algorithm Enhanced IC-PCP Algorithm with Replication (EIPR), by replicating task using idle instances and surplus in budgets. These enhancements show the possibility of meeting user defined deadlines by an increase in the replication of tasks. Such replicated computation resulted in a very high computational overhead. Although EIPR showed promising results in the mitigation of variable performance effects by the exploitation of the billing schemes and cloud elasticity. Is performance is low when the task execution time is near the billing period.

A. BUDGET-CONSTRAINED SCHEDULING ALGORITHM

In this section, we discussed the resource scheduling techniques proposed for scientific workflows in cloud computing. In [24], a budget constrained algorithm was proposed to deal with optimisation in workflow scheduling. In this approach, the total budget as defined by the user was proportionally distributed to each task based on the average time of execution. Arabnejad and Barbosa [9] introduced the concept of worthiness as an attribute in scientific workflow applications. In this approach, worthiness was used for resource selection and were determined based on cost and time of execution. Wu et al. [25] propose a budget-constrained algorithm PCB-B^2. This approach used a binary search method to execute its budget distribution. In [26], the authors proposed a task swapping strategy for scientific workflows. In this approach, tasks with smaller execution time were executed before others, which lowered the optimisation cost. In [21], the authors proposed a level based (DCLB) scheduling algorithm for cloud computing. DCLB is aimed at reducing the makespan of a workflow while meeting the user-defined deadline at the same time. The authors divided the workflow into logical levels using parallelism strategy while achieving load balancing at the same time. However, the context of

the logical level partitioning is not really clear and the task completion of the parent task is a prerequisite to the initiation of the child task. In [27], the authors proposed an algorithm for scheduling scientific workflows based on deadline distribution factor. The proposed algorithm (Just-in-time JIT-C) was developed to mitigate performance issues in VMs. This algorithm focused on three main objectives including; the variation in the performance of VMs, heterogeneity in cloud systems and resource acquisition delay. Although, the authors [27] claimed that this method was very effective for deadline, it was computationally expensive to the generate schedules when the deadline factor is low. Hadiri et al., [28], highlighted acquisition delay as a significant barrier in workflow scheduling. To optimise the total execution time required to complete a workflow task, they suggested a cost-effective technique (CEDA) for minimising task execution time while meeting deadlines. CEDA, on the other hand, is not ideal for large workflows.

Topcuoglu et al. [3] propose some of the earliest resource scheduling algorithms for heterogeneous computing. Their two algorithms aimed at achieving fast scheduling time and high performance simultaneously. Heterogeneous Earliest Finish Time (HEFT) and Critical-Path-On-a-Processor (CPOP) are the two algorithms proposed by these authors. HEFT focuses on minimising the earliest finish time with a novel strategy called incentive-based task. On the other hand, CPOP pays attention to the prioritisation of tasks based on upward and downward ranks. One major disadvantage of HEFT is that load balancing is not considered and workflows are scheduled one at a time. In [29], the authors proposed a budget-aware algorithm (BDAS) for resource scheduling in heterogeneous e-scientific workflows. The study showed that high attention has been paid to optimisation in scientific workflows, with particular focus on time and cost constraints. BDAS approach tries to satisfy deadline and budget constraints by the introduction of heterogeneous instances on cost and time. Budget and Deadline HEFT (BDHEFT) was proposed by Verma and Kaushal [30]. BDHEFT is another extension of HEFT with emphasis on budget and deadline. The BDHEFT algorithm used a ranking mechanism for the identification of tasks, which is done in descending order. BDHEFT has shown promising results with reduction in makespan and execution cost of workflows. However, BDHEFT's dependence on prior information limits its static global point of view.

In this paper, we look at the grouping of tasks based on their related dependencies such as control data and user data to reduce computation overhead. We developed a deadline-constrained algorithm to minimise cost and meet the user defined deadlines. The system model is presented in the next section, followed by the algorithm description.

III. SYSTEM MODEL

In this section, we describe the system and cloud models used in this paper. As described in [31], workflow models are generally represented by directed acyclic graphs (DAG). We define the abstraction of the workflow as $W = (T, E)$

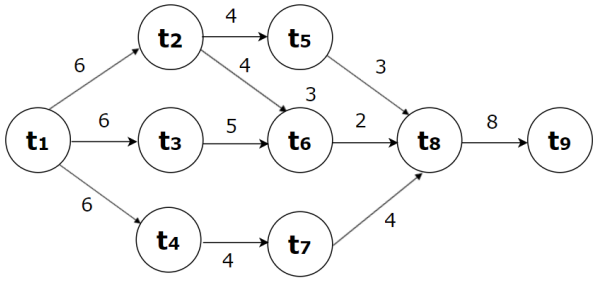


FIGURE 1. A Sample Workflow with 9 Tasks.

TABLE 1. Notation Description.

Notation	Description
W	Workflow
T	Set of Task
E	Set of directed edges
TC_{ms}	Makespan
DT	Data Transfer time for task
ETC	Estimated Time for Computation
β_{VM}	Available Bandwidth of VM
UR_t	Upward Rank of tasks
L_{t_i}	Degree of task dependency
AT_w	Available Time of workflow
EFT	Earliest Finish Time
FT_{t_i}	Finish time of last task
ESD	Estimated Sub-Deadline
MC_w	Monetary Cost of workflow
t_{entry}	Entry task
t_{exit}	Exit task
$\mathrm{ML}c_1 t_i$	Mean Load of task t_i on cloud provider c_1

where T represents a set of tasks $\{t_1, t_2, \ldots, t_n\}$ and E represents the task dependencies $e_{i,j} \in E$. As shown in Fig. 1, the directed arc between the two tasks t_i and t_j where $t_i, t_j \in T$. This implies that t_j starts when the execution of t_i is completed and t_i is the parent task of t_j. In Fig. 1, a sample workflow is presented where each node represents a task. The arcs denote the time (seconds) period during which data is transferred between tasks.

A. CLOUD MODEL

This paper considers cloud service providers with different capabilities. The instance type consist of compute resources including CPU, bandwidth, and storage. The cloud services are represented by $C = \{c_1, c_2, \ldots, c_n\}$ where c_n represents the number of instances. The total processing time of a workflow which is the task completion time is referred to as the makespan (MS). This is the time elapsed from execution of the entry task t_{entry} to the exit of the last task t_{exit}. These are tasks without predecessors and successors. For scientific workflows, a task is ready to be executed when the predecessor tasks have been successfully executed. The task completion time is represented in Eqn.6. Task completion time is a measure of the time it takes to complete or perform a task (from start to finish).

Definition 1: Data transfer time is the time taken to transfer data between endpoints. The data transfer time DT_{ij}

is represented in Eqn. 1:

$$DT_{ij} = \frac{M_{t_i}}{\beta_{VM}} \tag{1}$$

where M_{t_i} is the data size of t_i and β_{VM} is the available bandwidth of the *VM*.

Definition 2: The estimated time of computation is the overall time it takes to compute the end-to-end task.

The estimated time for computation (ETC) for task t_i is computed as shown in Eqn. 2:

$$ETC_{t_i} = \frac{S_{t_i}}{\rho(1 - P_{v_{VM}})} \tag{2}$$

where S_{t_1} represents the size of the task t_i, ρ represents the *VM*'s processing capacity and $P_{v_{VM}}$ represents the variation in performance of the *VM* such as CPU, memory and network degradation in real cloud environments.

Definition 3: Execution time of task is the time used by the system to execute the task, including run-time. Execution Time of task t_i is computed as shown in Eqn. 3

$$ET_{t_i} = ETC_{t_i} + \max_{t_i \in pred(t_i)} DT_{ij} + \delta \tag{3}$$

where δ represents the provisioning delay in the allocation of *VM* as a result of time zone differences, OS, VM migration delay, software setup, location of the data centre and many other factors.

Definition 4: Earliest start time is the earliest time in the schedule at which a task can begin. Earliest Start Time (EST) is the time for task t_i to be executed on the VM. This is computed as shown in Eqn. 4

$$EST_{t_i} = Avail_{VM} DT_{ij} + ET_{t_i} \tag{4}$$

Definition 5: Earliest finish time represents the earliest time a task can possibly finish, if all predecessors and successors also finish on their respective early finish time. The earliest finish time (EFT) is computed as shown in Eqn. 5

$$EFT_{t_i} = ETC_{t_i} + \max_{t_i \in pred t_j} (DT_{ij} + ESD_p) \tag{5}$$

where ESD_p is the estimated sub-deadline for predecessor t_j.

Definition 6: Finish time of the last task t_i represents the completion time of the last task. The finish time (FT_{t_i}) is computed as shown in Eqn. 6

$$FT_{t_i} = ET_{t_i} + EST_{t_i} \tag{6}$$

Definition 7: Task completion time is a measure of the time it takes to complete or perform a task (from start to finish) as shown in Eqn. 7.

$$TC_{ms} = TF_{t_{com}} \tag{7}$$

where $TF_{t_{com}}$ is the completion time of the last task.

Definition 8: We assume that the each cloud provider c_1 has compute resources with X dimension (memory, CPU, or network) and the Mean Load is computed as shown in Eqn. 8

$$ML_{c_1 t_i} = (\frac{c^m_{1(t_i)}}{c^m_1} + \frac{c^B_{1(t_i)}}{c^B_1})/X \tag{8}$$

where $ML_{c_1 t_i}$ represents the mean load of t_i on c_1, $c^m_{1(t_i)}$ and $c^B_{1(t_i)}$ represent the used memory and bandwidth by t_i on c_1.

IV. PROPOSED ALGORITHM

The main objective of the proposed algorithm is to minimise cost while meeting budget and deadline constraints. The proposed scheme is made up of three main phases which include, partitioning of the workflow, deadline distribution, selection of task and resource scheduling algorithm.

A. PARTITIONING OF WORKFLOW

In workflow partitioning, the main aim is to prioritise and group tasks on the same level with similar functionalities and dependencies. First, tasks are prioritised based on dependencies. For task t_i, we assign a priority based on the [3] upward rank *UR* as proposed in in Eqn. 9

$$UR_{t_i} = ETC_{t_i} + \max_{t_i \in pred t_j} DT_{ij} + UR_{t_j} \tag{9}$$

we consider this approach very effective because it computes the length of the critical path from the entry task to the exit task by using a ranking process where the ranks are computed recursively by traversing the DAG to the entry node. We compute the degree of dependency in Eqn. 10

$$L_{t_i} = \sum_{t_i \in pred t_j} E^i_{t_i} + \sum_{t_i \in pred t_j} E^o_{t_i} \tag{10}$$

where $\sum_{t_i \in pred t_j} E^i_{t_i}$ and $\sum_{t_i \in pred t_j} E^i_{t_i}$ represent the internal and external edges as shown in fig. 1. These tasks are grouped based on similar functionalities. Each group of tasks consist of single or multiple tasks.

B. TASK SELECTION

In the proposed scheme, tasks are held in the execution queue until their predecessors are executed. For tasks grouped together, their execution can be carried out simultaneously. Before this was done, we considered the deadline distribution of tasks which ensured that global deadline of tasks was achieved by distributing deadline among different levels. As described in Eqn 1, the data transfer time DT_{ij} is computed using available VM instances. To guarantee that the overall deadline was met, sub-level deadline which is the deadline for each sub task was implemented. We first find the available time *AT* which is the elapsed time between the workflow's deadline and the estimated makespan. The available time *AT* is computed as shown in Eqn. 11.

$$AT_W = D_W - TC_{ms} \tag{11}$$

where D_W is the deadline of the workflow and TC_{ms} represents the makespan. To meet with the overall deadline, we also compute the available time for group level task AT_{GL} in Eqn. 12.

$$AT_{GL} = \frac{ETC^{GL}_{t_n}}{ETC^{W}_{t_n}} \times AT_W \tag{12}$$

where $ETC^{GL}_{t_n}$ represents all the task at group level and $ETC^{W}_{t_n}$ represents all the task in the workflow. The sub-deadline of task t_i is represented as shown in eqn. 13;

$$D_{t_i sub} = \max_{t_i \in pred t_n} (DT_{ij} + ETC + AT_{GL}) \tag{13}$$

C. GROUPING OF TASKS

Tasks are integrated into a group of tasks using the horizontal grouping strategy adopted by the authors in [6]. This means that tasks in the same group have same capabilities and same depth to enable parallel processing. Rodriguez and Buyya et. al. [6] described how tasks were grouped horizontally using the same colour. A group of tasks can be made up of single or multiple tasks. A single immediate processor is shared by tasks that are homogeneous which form part of a group of task with similar attributes in terms of data size, input/output size, and computing cost, have the same data distribution pattern and are of the same type.

D. DEADLINE DISTRIBUTION

The deadline distribution strategy is presented in Algorithm 1, the cheapest VM is selected from the VM instances available. This ensures that the estimated makespan for every task is less than the user-defined deadline. The next non-expensive VM is chosen if the deadline is by the makespan estimated until the completion time of the workflow is less than the user-defined deadline. AT_W is then distributed proportionally across all levels of task based on the task duration using AT_{GL}. The estimated sub-deadline is computed for each task using $D_{t_i sub}$. This implies that tasks that form a group of tasks are given the same sub-deadline which will be equivalent to start time of a successor task. Algorithm 2, describes the dynamic scheduling of task grouping using sub-deadline distribution which is computed used upward rank and degree of dependency. The task are sorted based on descending order of priorities for each level. Once the group of task are identified, they are placed in priority queue and sorted based on relevance and placed in the execution queue.

Algorithm 1 Deadline Distribution

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

V. PERFORMANCE 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 of VMs. The configurations of the VM types which were based on the Amazon EC2 cloud offering was presented in Table 2.

Algorithm 2 Sub-Task Distribution

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

TABLE 2. VM Instance Specifications.

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

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 evaluate the performance of the proposed scheme and the compared approaches, we considered the following metrics:

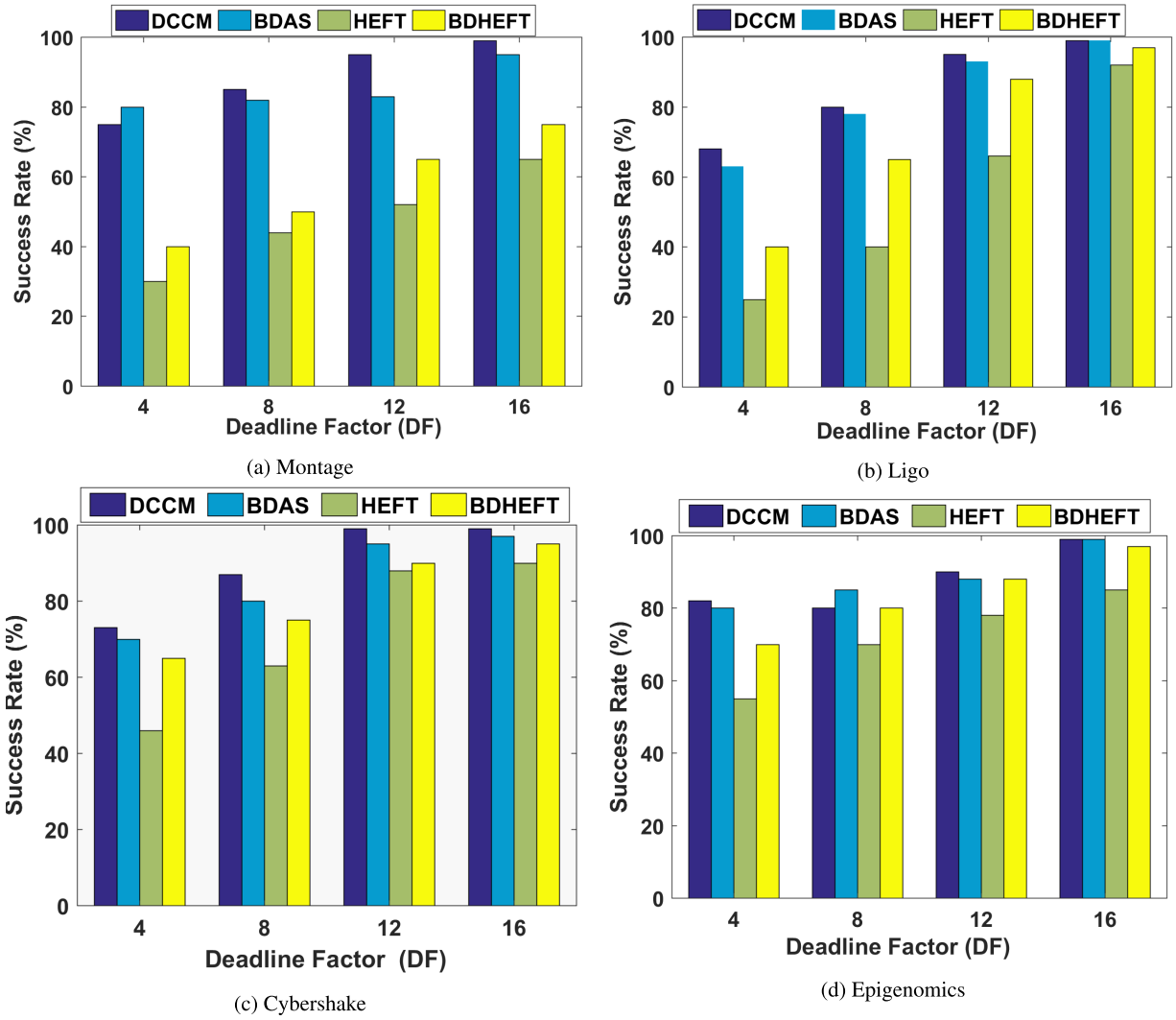
- *Success Rate (SR)*: The success rate of a resource scheduling algorithm is computed as the percentage of valid schedules that are obtained in an experiment. This is calculated as in Eqn. 14

$$SR = \frac{W_s}{W_t} \times 100 \tag{14}$$

where W_s represents the number of successful workflows and W_t 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. 15

$$MC = \frac{MC_W}{VM_{cheap}} \tag{15}$$



(a) Montage

(b) Ligo

(c) Cybershake

(d) Epigenomics

FIGURE 2. Performance comparison of success rate for Montage, Ligo, Cybershake and Epigenomics.

B. EXPERIMENTAL RESULTS

1) PERFORMANCE COMPARISON FOR SUCCESS RATE

The success of DCCM, BDAS, HEFT and BDHEFT obtained for the four scientific workflows are presented in Fig. 2(a), (b), (c), (d). The results show that the proposed scheme performs better than the compared approaches. We consider the deadline factor for the scientific workflows used in the experiments as discussed in the Budget-Constrained Scheduling Algorithm section. Figure 2(a) shows the results obtained from simulations for montage workflow. DCCM and BDAS perform better than HEFT and BDHEFT even with lower deadline factors. Although all the algorithms show an increase in trend, DCCM performs better than BDAS as the deadline factor increases and outperforms HEFT and BDHEFT with more than 20% when the $DF = 16$.

In Fig. 2(b), the results obtained from the experiment with LIGO are presented. It was observed that DCCM and BDAS achieve above 65% in terms of their planning success rate with the lowest deadline factor when compared with HEFT and BDHEFT which have about 25% and 40% success rate respectively. However, with an increase in the factor, there is a significant improvement with all the algorithms as DCCM, BDAS, HEFT and BDHEFT achieve 99%, 99%, 90%, and 95%, respectively. In Fig. 2(c) and (d), the results obtained from Cybershake and Epigenomics workflows are presented. For the compared approaches, the algorithms achieved more than 80% success rate at $DF = 16$. DCCM performed better than the compared approaches. HEFT had the lowest success rate at $DF = 4$ when compared with other algorithms. It was observed that DCCM and BDAS achieve similar results but DCCM performs better than BDAS by 5-15% success rate.

2) PERFORMANCE COMPARISON FOR COST RATIO

In the previous subsection, the success rate of the proposed algorithm was compared with other approaches. This subsection highlighted the performance comparison of monetary cost ratio. Fig. 3 (a), (b), (c), (d) shows the cost ratio of DCCM, BDAS, HEFT and BDHEFT under the same

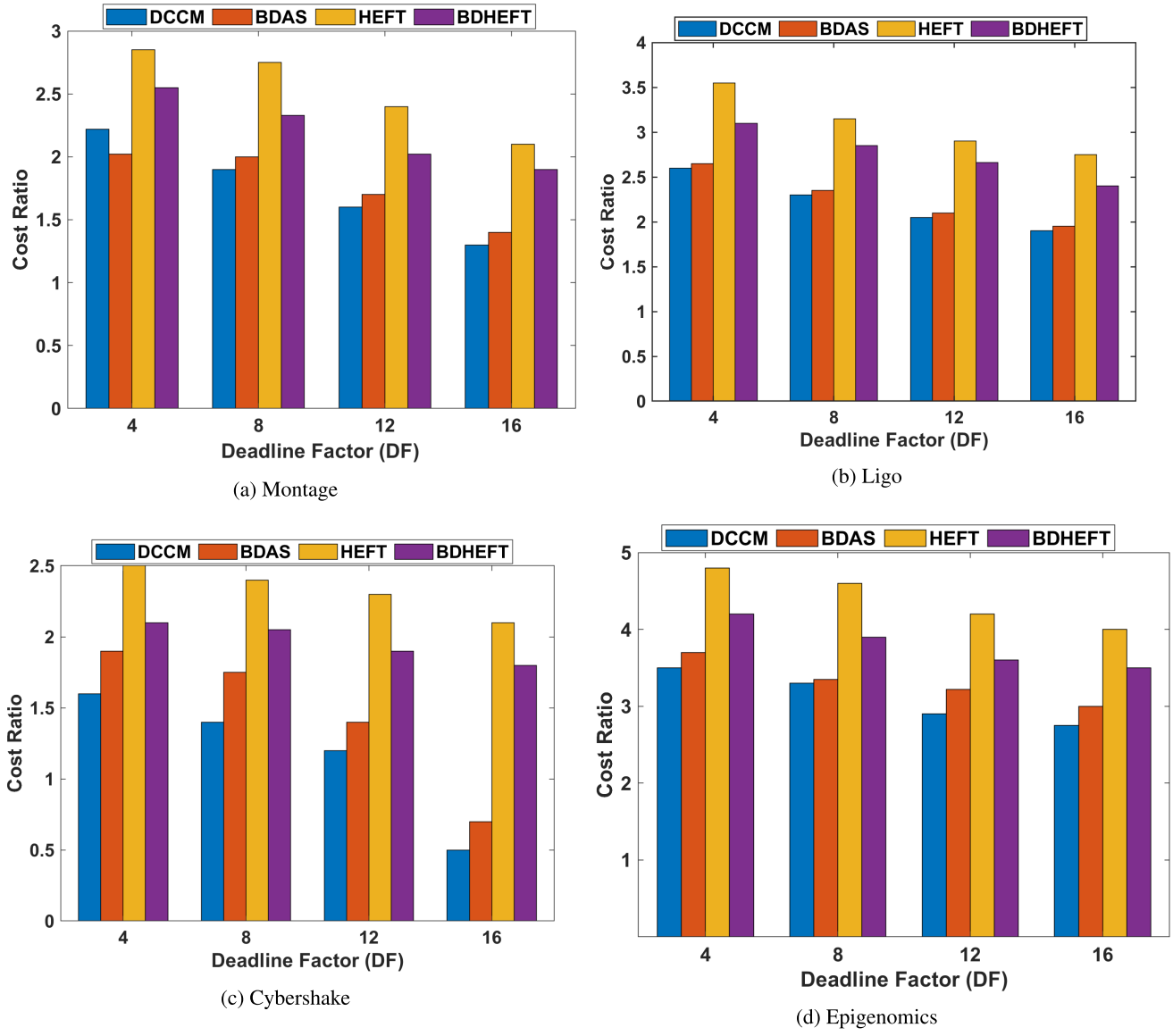


FIGURE 3. Performance comparison of cost ratio for Montage, Ligo, Cybershake and Epigenomics.

deadline actor used for the comparison of their success rate. It was observed that the cost ratios of all the workflows decrease as the deadline factor increases. In Fig. 3 (a) and (b), the results obtained from Montage and LIGO scientific workflows are presented. The proposed scheme and BDAS outperform HEFT and BDHEFT algorithms. Although BDAS performs better than DCCM when $DF \leq 4$, DCCM performs better than BDAS as the deadline increases. DCCM performance can be attributed to the sub-deadline division which reduces the number of instances. Similar results are also observed in Fig. 3(c) and (d) for Cybershake and Epigenomics workflows. DCCM performs better than the compared algorithms while HEFT and BDHEFT algorithms have very high cost ratio which affect the schedule length of the workflows. From the results, it can be seen that DCCM and BDAS have very close results in terms of performance. This is because both schemes utilise minimum budget. However, DCCM categorises task into different sub-levels by the distribution of deadline using a deadline top level approach.

C. PERFORMANCE COMPARISON FOR MEAN LOAD

In Fig. 4 (a), (b), (c) and (d), the performance comparison for the mean load of DCCM and BDAS are presented. From the results earlier presented in Fig. 2. and 3, a close observation shows that BDAS and DCCM are very close results in terms of the cost ratio and success rate. As shown in Fig. 4 (a) Montage, (b) Ligo, (c) Cybershake and (d) Epigenomics, DCCM performs better than BDAS in terms of the mean load in all the scientific workflows compared. It is observed that the mean load increases as the number of task increases. DCCM performs better than BDAS dues to its pre-selection phase whose goal is to always to choose the provider with the least execution time.

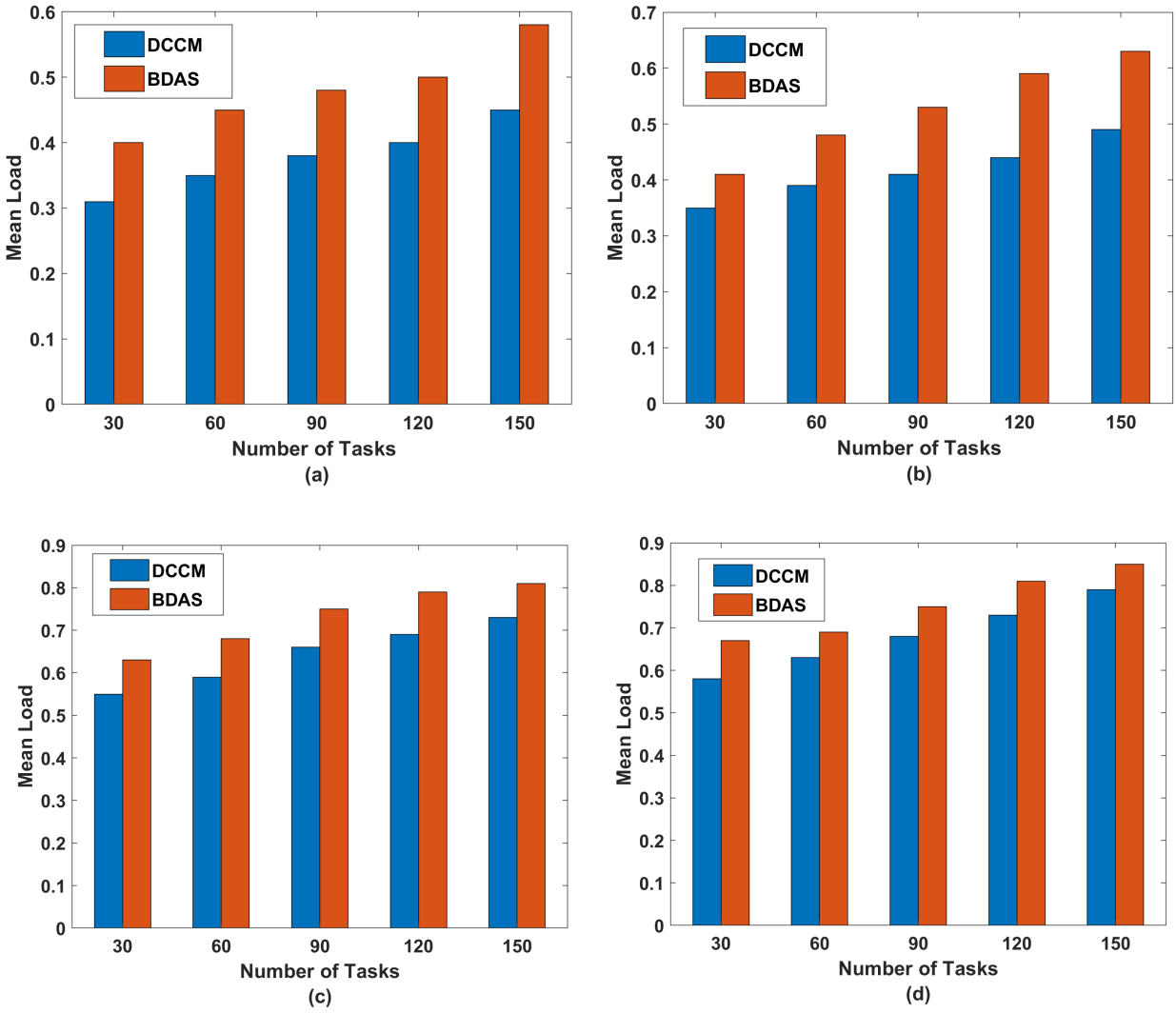


FIGURE 4. Performance Comparison of Mean Load.

VI. CONCLUSION

This paper proposes an algorithm (DCCM) for scheduling of resources in cloud computing environments. DCCM focuses on the reduction of cost in a deadline defined by the user. DCCM is evaluated and compared with four scientific workflows well-known in the literature. Experimental results show that the proposed scheme performs $16 - 25\%$ higher than the compared approaches in terms of the planning success rate and makespan cost ratio. In our future work, we plan to build a dual-objective scheduling model that minimises both the makespan and the overall monetary service cost at the same time.

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 low-complexity 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, "Deadline-constrained 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-B^2: 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.

SAMUEL MANAM received the B.Sc. degree in computer science from the University of Calabar, in 2001, the M.Sc. degree in wireless communications systems engineering from the University of Greenwich, U.K., in 2010, and the Ph.D. degree from the Department of Electrical and Electronic Engineering, Institute of Communication Systems, University of Surrey, U.K., in 2022. His research interests include network function virtualization, cloud networks, resource scheduling and management in cloud, Machine Learning (ML) and Artificial Intelligence (AI), network security, and Software-Defined Networks (SDNs).

KLAUS MOESSNER (Senior Member, IEEE) is currently a Professor of communications engineering with the Chemnitz University of Technology and also a Professor in cognitive networks with the Institute for Communication Systems, University of Surrey. He was involved in a large number of projects in cognitive communications, service provision, and the IoT areas. He was responsible for the work on cognitive decision-making mechanisms in the CR Project ORACLE. He led the work on radio awareness in the ICT FP7 Project QoSMOS and the H2020 Speed5G Project. He was the Founding Chair of the IEEE DYSPAN Working Group (WG6) on sensing interfaces for future and cognitive communication systems. He has also led several EU-funded ICT projects. His research interests include cognitive networks, the IoT deployments, sensor data based on knowledge generation, and reconfiguration and resource management for reliable wireless communication.

SERDAR VURAL received the B.S. degree in electrical and electronics engineering from Boğaziçi University, Istanbul, Turkey, in 2003, and the M.Sc. and Ph.D. degrees in electrical and computer engineering from The Ohio State University, Columbus, OH, USA, in 2005 and 2007, respectively. He is currently a Principal Network Architect with the 5G Innovation Centre, University of Surrey. He is actively involved in research and development activities on various topics, specifically Software-Defined Networks (SDNs).