Delay-Based Workflow Scheduling for Cost Optimization in Heterogeneous Cloud System

Madhu Sudan Kumar*, Indrajeet Gupta[†] and Prasanta K. Jana[‡]
Department of Computer Science and Engineering
Indian Institute of Technology (ISM), Dhanbad, India
Email: *mdhsdnkumar88@gmail.com [†]indrajeet7830@gmail.com, [‡]prasantajana@yahoo.com
http://www.iitism.ac.in

***Abstract*—Workflow scheduling has been widely adopted as a deep concern for heterogeneous cloud computing environment and recognized as a well-known NP-complete problem. The scheduling goal of any workflow scheduler is to map the user requests on the available virtual machines (VMs) which are deployed on various cloud servers such that overall processing time and incurred cost is minimized. This paper proposes a delay-based workflow scheduling algorithm for cost optimization in the heterogeneous cloud system. The proposed scheduling heuristic works in two phases. In the first phase, the proposed heuristic introduces a new priority scheme for the tasks. The prioritized tasks are then assigned to the appropriate VMs in the next phase. The algorithm is simulated on various synthetic workflows and two benchmark scientific workflows with different range of tasks. The experimental results of the proposed algorithm supercedes the existing scheduling algorithms in terms of makespan, schedule length ratio (SLR) and cost.**

Keywords*—Workflow Scheduling, Makespan, Cost, Schedule Length Ratio, Cloud Computing.

I. INTRODUCTION

Workflow scheduling is recognized as a burning research problem of highly connected and inter-dependent tasks and it belongs to NP-complete class. The Workflow applications liaises to various real life and scientific domains i.e., massively large web graphs and social networks. The contemporary growth of virtualization in the area of computing, communication and storage [1–3], a standalone or group of physical machines are able to create multiple virtual instances in form of virtual machines (VMs). These VMs can run simultaneously without interrupting to each other. The cloud users are charged by cloud service providers (CSPs) for renting the virtual resources (VMs) as per the pricing policies. However, the workflow applications always demand high computation power, network bandwidth and storage capacity due to the presence of precedence-constrains. Therefore, cloud computing is a well suited paradigm which offers multiple set of services to the cloud user with the help of virtualization. By focusing on the cloud user's budget, the cloud service provider (CSP) is to provide an efficient and cost conscious schedule plan for the user request. The pricing policy differs from VM to VM which means fast processing speed VM demands high charged (cost) while slower one offers lower VM rate [1, 2, 4]. For example Amazon EC2 offers 12 different types of VM instances for the cloud users [1, 5]. Therefore, a cost efficient resource provisioning and workflow scheduling is also accountable for suitable mapping of workflow tasks on the resources, so that the execution can be completed to satisfy criteria demanded by cloud users. Appropriate scheduling can provide a remarkable influence on the performance metrics (makespan, cost and energy) of the system [6–9].

In this paper, we address a cost conscious workflow scheduling algorithm by proposing a new priority scheme called delay based scheduling ($ADAP$) which is based on as-late-as-possible ($ALAP$) [10]. The new priority scheme $ADAP$ measures the possible delay of task start time without increasing the schedule length. In general the proposed algorithm keeps running in two stages. In first stage the static task characteristic $ADAP$ are calculated by visiting the directed acyclic graph (DAG) upwards from the exit task node. After that all the tasks are arranged in the decreasing order of $ADAP$ at each level (l_i) of DAG. In the second stage the prioritized tasks in the priority queue (Q_g) are scheduled according to earliest start time and earliest finish time. Our proposed algorithm uses heterogeneous nature of the tasks at each level that differs it from the original $ALAP$ which provides the task prioritization for the entire workflow rather than the heterogeneity at each level. The main advantage of exploring the leveled behavior of the tasks is to balance the differences in completion time of the tasks at a level. This leads to minimize the difference in the starting time of the tasks at the next level. The performance of the proposed scheme is evaluated by comparing it with other four popular existing dependent task scheduling heuristics namely ALAP [10], HEFT [11], PETS [12] and FCFS in terms of performance parameters.

Residual part of the paper is orderly arranged as follows. Section II consists the related work as per the recognition of the research community. Problem statement and basic terminologies are described in Section

III followed by Section IV in which we present the proposed algorithm. Section V has been systematized by the simulation results and performance assessments after that Section VI concludes the paper.

II. Related Work

In recent years, several heuristics as well as meta-heuristics approaches have been proposed to deal with the problem of workflow scheduling for cost optimization. Tabu search and Simulated annealing were explained to solve the scheduling problems in Grid as well as cloud computing environment. In [13], multi-objective optimization based scheduling is explained by considering makespan and energy which uses two-phase genetic algorithm that involves multi-parent crossover parameter. In [14], Improved Genetic Algorithm (IGA) is explained which perform two times better than simple GA and it selects the VMs on the basis of dividend policy. In [15], Particle Swarm optimization based workflow scheduling is discussed with cost as the single objective. The cost model consider execution cost as well as communication cost, however it does not considers data transfer cost among the tasks. Also, the results were not compared with standard algorithm, and the algorithm were not simulated over scientific workflows.

Many research works have been proposed on workflow scheduling and resource provisioning algorithms of cloud computing by addressing various issues such as processing time [7, 8, 12, 16], cost [1, 2] and energy efficiency [3, 4]. In the list based workflow scheduling techniques various priority schemes have been proposed till now. Some of the schemes are b-level, t-level, HLF, CPOP, LPT, PETS and so on [10–12]. Based on above-mentioned priority schemes a number of workflow scheduling algorithms have been designed in cloud computing environment. However, task clustering based approach is also applicable for the cost efficient workflow scheduling but most of them are applicable in the homogeneous environment and they are very compatible for obtaining makespan but none of them have considered cost as the main objective [1]. In the clustered task category, there are more possibilities to reduce data transfer time but sometime cluster formation is not an easy job. Moreover, the intra-cluster data transfer time will be negligible but the data transfer time between inter cluster may be high due the multi-level dependencies among the task nodes in the workflow application.

III. Basic Terminologies and Problem Formulation

A Workflow application W_f = (T, E) can be modeled by a directed acyclic graph (DAG), where $T = \{t_1, t_2, \ldots, t_n\}$ is the task crew and E is the data transfer or precedence edges. Any two dependent task nodes t_i, $t_k \in T$ are scheduled in such manner that task t_k can not be scheduled until execution of t_i has been completed.

Before addressing the problem formulation, this section requires the detailed explanation of few important terminologies and formulas which are as follow.

Definition 1. The set of participated M cloud servers is represented as the $CS = \{CS_1, CS_2, CS_3, \ldots, CS_M\}$ which consists of m number of heterogeneous virtual machines (VMs). These VMs are the processing elements used for task execution and other operations on cloud servers. It is noteworthy that total number of created VMs may vary according to the deployment capacity of cloud servers. If any two VMs, VM_i and VM_j are deployed or created on same cloud server then the data transfer time between VM_i and VM_j will be considered zero otherwise the data transfer time between VM_i and VM_j will be considered.

Definition 2. (Estimated Computation Time): The estimated computation time ECT of a task t_i is the estimated execution time on a VM_j which is evaluated as dividing the task size (MI) by the processing speed of VM (MIPS). The mathematical representation of ECT can be organized in a matrix as follows.

$$ECT=$$

$$\begin{array}{c c}
 & \begin{array}{ccccccc} \multicolumn{3}{c}{\overbrace{VM_1 \quad \cdots \quad VM_{|CS_1|}}^{CS_1}} & \cdots & \multicolumn{3}{c}{\overbrace{VM_{\alpha+1} \quad \cdots \quad VM_{\alpha+|CS_M|}}^{CS_M}} \end{array} \\
\begin{array}{c} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_n \end{array} &
\begin{bmatrix}
ECT_{1,1} & \cdots & ECT_{1,|CS_1|} & \cdots & ECT_{1,\alpha+1} & \cdots & ECT_{1,\alpha+|CS_M|} \\
ECT_{2,1} & \cdots & ECT_{2,|CS_1|} & \cdots & ECT_{2,\alpha+1} & \cdots & ECT_{2,\alpha+|CS_M|} \\
ECT_{3,1} & \cdots & ECT_{3,|CS_1|} & \cdots & ECT_{3,\alpha+1} & \cdots & ECT_{3,\alpha+|CS_M|} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
ECT_{n,1} & \cdots & ECT_{n,|CS_1|} & \cdots & ECT_{n,\alpha+1} & \cdots & ECT_{n,\alpha+|CS_M|}
\end{bmatrix}
\end{array} \tag{1}$$

where, α is $\sum_{k=1}^{M-1} |CS_k|$.

In order to define problem definition, some other useful terminologies are explained as below.

1) The average estimated computation time (Avg_ECT): It is the average estimated computation time of any task t_i against all the m VMs which is described as follows.

$$Avg\_ECT(t_i) = \sum_{j=1}^{m} \frac{ECT_{i,j}}{m} \tag{2}$$

2) *EST*(t_i,VM_j): It is the earliest start time at which the task t_i is ready for execution on available VM pool. The EST of entry task will be zero because there are no predecessor task node of entry task t_i . $EST(t_i, VM_j)$ of any task t_i on any VM_j is defined as.

$$EST(t_i, VM_j) = max\left\{ \underset{m'}{Avl[j]},\ \max_{pt_i \in predr(t_i)} \left\{ AFT_{pt_i} + DTT_{p_i,t_i} \right\} \right\} \tag{3}$$

where *m’* is virtual machine type on any cloud server.

3) *EFT*(t_i,VM_j): Earliest finish time of task t_i on VM_j can be expressed by Eq. (4).

$$EFT(t_i, VM_j) = ECT_{i,j} + EST(t_i, VM_j) \quad (4)$$

where, $predr(t_i)$ is the set of just previous predecessor task nodes of task t_i and $Avl[j]$ is the minimum time at which VM_j is ready for task execution and $ECT_{i,j}$ is the estimated computation time of task t_i on VM_j.

A. Problem Definition

The mapping of the n task nodes of a given workflow on m active VMs in order to generate an optimal schedule plan with the following objectives.

Objective 1: The overall processing time of workflow application W_f i.e., *makespan* and schedule length ratio (SLR) are to be minimized.

Objective 2: The total cost (T_cost) incurred by the cloud user for the given workflow W_f also to be minimized.

The afore-mentioned objectives are vividly described in the upcoming statements.

1) **Makespan**: It is the overall workflow processing time by the participated cloud servers on m active VMs. Let $t_i \rightarrow VM_j$ denote mapping of task t_i on VM_j at cloud server C_k. It is calculated as follows:

$$Mkspn = min\{EFT(t_{exit})\} \quad (5)$$

2) **Schedule Length Ratio (*SLR*)**: It is the proportion between parallel execution time and the sum of computation time of those tasks lying on critical path running on fastest VM. The minimizing nature of SLR is attractive for any scheduling algorithm.

3) **Cost**: Now, to evaluate the cost we need the time for which the VMs are used for executing the workflow applications. Our cost model involves rate of all VMs to evaluate the total cost. We define single dimension cost matrix as shown in Table I. The total cost incurred for scheduling applications can be calculated using Eq. 6. In cost matrix unit can be in minutes, hour and days etc. It is depended on the requirement of users.

TABLE I: VM rate for cost evaluation

	VM_1	VM_2	...	VM_m
cost/unit time	r_1	r_2	...	r_n

$$T\_cost = \sum_{i=1}^{m} Mkspn(VM_i) \times rate(VM_i) \quad (6)$$

The notations are used in the proposed algorithm are listed in Table II

TABLE II: Notations and Definitions

Notations	Definition
n	Total number of tasks in workflow W_f
Q_g	Ordered priority list
m	Total number of active VMs
M	Total number of cloud servers
CPL	Critical path length of a given workflow W_f
$ADAP(t_i)$	As delay as possible start time of task t_i
l_i	Workflow or DAG level
$predr(t_i)$	It is the predecessor set of task t_i
$succr(t_i)$	It is the successor set of task t_i
$EST_{i,j}$	Earliest Start Time of task t_i on VM_j
$EFT_{i,j}$	Earliest Finish Time of task t_i on VM_j
AFT_i	Actual Finish Time of task t_i
$DTT_{i,j}$	Data Transfer Time from task t_i to task t_j
$CSID$	Cloud Server ID
$VMID$	Virtual Machine ID

IV. Proposed Algorithm

In this paper, we propose a cost conscious workflow scheduling algorithm which is based on the $ADAP$ (As Delay As Possible) characteristic of the task's start time. The $ADAP$ start time of a task node can be measured as the delayed start time which does not affect on the overall execution time (makespan) for the given workflow W_f. The proposed algorithm runs in two phases. In first phase, it assigns the priority of each task t_i of the given workflow W_f by using $ADAP$ priority scheme. After-that, at each level (l_i) of the W_f the tasks are arranged as per their decreasing order of their respective priority. Then the second phase allocates the task on the suitable VM among the available VM pool by mapping the optimal possibilities of t_i. The mapping between task-VM is accommodated by calculating earliest start time (EST) and earliest finish time (EFT) for each task t_i. The above-mentioned steps are repeated until all the task nodes are scheduled of the workflow W_f.

The working process of proposed Algorithm is well explained in Algorithm 1 and Algorithm 2.

A. Illustrative Example

In order to understand the proposed algorithm more comprehensively, we have taken an example of DAG as shown in Fig. 1. In this example, there are 10 task nodes connected with 15 dependency edges which have to schedule on 3 heterogeneous VMs deployed 2 cloud servers. Moreover, the estimated computation time (*ECT*) matrix and corresponding VMs rate are given in Table III and Table IV.

First we find the average estimated computation time (Avg_ECT) for all the tasks as per the Eq. 1 then compute the $ADAP$ value of each task in the given DAG according to Algorithm 2. The Avg_ECT for all the 10 tasks will be as [13, 16.66, 14.33, 12.66, 11.66, 12.66, 11, 10, 16.66 and 14.66]. All the 10 tasks [t_1, t_2,

Algorithm 1 : Proposed Scheduling Algorithm

Input- W_f (T, E) workflow consisting of n tasks nodes

Output- Optimal Schedule Plan Generation S of W_f (T, E) by all m active VMs which are deployed on M cloud servers

```
1  Read the workflow $W_f$ and find the requirements of tasks
   against the all available VMs on different cloud server
2  Set the average estimated computation time
   $Avg\_ECT(t_i)$ of every tasks in $ECT$ matrix
3  Compute ADAP ($t_i$) for all tasks by traversing DAG in
   bottom up fashion from the exit task node
4  for each level $l_i$ of $W_f(T, E)$ do
5      Sort each tasks node in the Priority Queue ($Q_g$) as
       per the descending order of ADAP($t_i$) value
6      while the $Q_g$ consists of unscheduled task do
7          Choose the first task, $t_i$ from the $Q_g$
8          for each VM from the deployed set of m at the
           cloud server set M and $|VM_m| \geq |M_k|$ do
9              Calculate EST ($t_i$, $CS_k$) and EFT($t_i$, $CS_k$)
               by Eq. (3) and Eq. (4) by using insertion
               scheme
10             Assign task $t_i$ to the virtual machine $VM_j$
               at $CS_k$ that minimizes the EFT($t_i$, $CS_k$).
11         end for
12     end while
13 end for
```

Algorithm 2 : ***COMPUTE ADAP(t_i)***

```
1  Build a tasks list in reverse topological order (Rev.
   Top List)
2  for each task node $t_i$ in Rev.Top List do
3      Min_Length = $CPL$
4      for each child $t_y$ of $t_i$ do
5          if ADAP($t_i$)$-DTT(t_i, t_y) < Min\_Length$
   then
6              $Min\_Length$ =
7                      $ADAP(t_i) - DTT(t_i, t_y)$
8          end if
9      end for
10     ADAP($t_i$)= $Min\_Length - Avg\_ECT(t_i)$
11 end for
```

t_3, t_4, t_5, t_6, t_7, t_8, t_9 and t_{10}] have the corresponding $ADAP$ values [1, 32, 29, 29, 40, 45.66, 66.33, 73.33, 64.66 and 94.33] respectively. Now at level 1 only t_1 is presented and it is entry task so it will be scheduled first and rest of the task will be arranged as per the decreasing order of the ADAP at each level. For example at level 2 there are five tasks t_2, t_3, t_4, t_5 and t_6. So at this level task t_6 have the highest ADAP value so among these five tasks then task t_6 will be scheduled by calculating EST and EFT as per the Eq. 3 and Eq. 4. We repeat this process at each level of DAG until whole workflow DAG is scheduled. Therefore, the complete task-VM mapping and scheduling are given in Fig. 2.

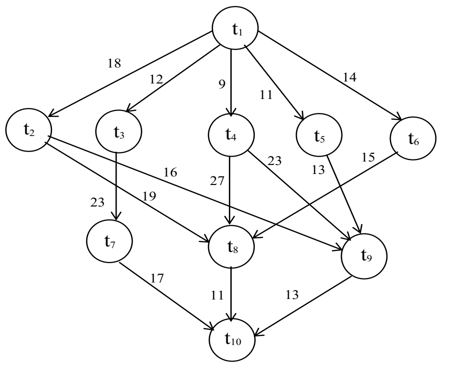

Fig. 1: An example of sample workflow with 10 task nodes

TABLE III: ECT matrix for given sample workflow

		t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
CS_1	VM_1	14	13	11	13	12	13	7	5	18	21
	VM_2	16	19	13	8	13	16	15	11	12	7
CS_2	VM_3	9	18	19	17	10	9	11	14	20	16

By applying Eq. 5, $Mkspn = min\{EFT(t_{exit})\}$ on the table of Fig. 2 the makespan of the given example is 61 time unit as per the proposed algorithm wheres the makespan of existing algorithms ALAP, PETS, HEFT and FCFS are 73, 70, 73 and 88 time unit respectively. As per the final schedule generation the makespan of VM_1, VM_2 and VM_3 are 25, 40 and 28 time unit respectively. Moreover, as per the Eq. 6 the incurred cost is 126.879 unit.

From the Table V, we can see clearly that proposed task delay based workflow scheduling performs much better than the existing algorithms for the given sample workflow example of Fig. 1.

	CS1				CS2		VMID	AFT	CSID
	VM1		VM2		VM3				
	EST	EFT	EST	EFT	EST	EFT			
t_1	0	14	0	16	0	9	3	9	2
t_6	23	36	23	39	9	18	3	18	2
t_5	20	32	20	33	18	28	3	28	2
t_2	27	40	27	46	28	46	1	40	1
t_3	40	51	21	34	28	47	2	34	1
t_4	40	53	34	42	28	45	2	42	1
t_8	42	47	42	53	69	83	1	47	1
t_7	47	54	42	57	57	68	1	54	1
t_9	54	72	42	54	65	85	2	54	1
t_{10}	54	75	54	61	71	87	2	61	1

Fig. 2: Overall task-VM mapping and scheduling of the given workflow

TABLE IV: VM rate in cost/unit time

VM_1	VM_2	VM_3
1.667	1.830	0.430

TABLE V: Comparison of performance metrics for the given DAG example

Scheduling Heuristic	makespan	SLR	Cost
Proposed	61	1.48	126.87
ALAP	73	1.78	129.70
PETS	70	1.70	126.97
HEFT	73	1.78	133.21
FCFS	88	2.14	152.79

V. Simulation and Performance Evaluation

We performed meticulous experiments of the proposed workflow scheduling algorithm using MATLAB R2012a version 8.3.0.532 on an Intel core i3 and 4 GB RAM running on Window 7 Ultimate platform. The experiments were carried out over two scientific workflows (Cybershake and Epigenomic) which is considered by [17] and various random DAGs up to 1000 task nodes. The results were analyzed over different performance metrics like makespan, SLR and cost. To show the effectiveness of our proposed approach we compare our results with four other existing scheduling algorithms, i.e., ALAP, PETS, HEFT and FCFS and simulated results show the efficacy of our proposed approach. Moreover, we have used uniform distribution to set the values of ECT matrix and data transfer time in the workflows. Moreover, in the random DAGs we have taken care of shape parameter and connecting edge probability between the task nodes.

The makespan, cost and SLR for random workflows are shown graphically in Figs 3, 4 and 5 respectively. The graphical comparison of cost and SLR are presented in Figs. 6, 7 for scientific workflows cybershake and epigenomic. The bar and line graphs clearly show the considerable lead in the performance parameters of the proposed workflow scheduling algorithm over the existing algorithms.

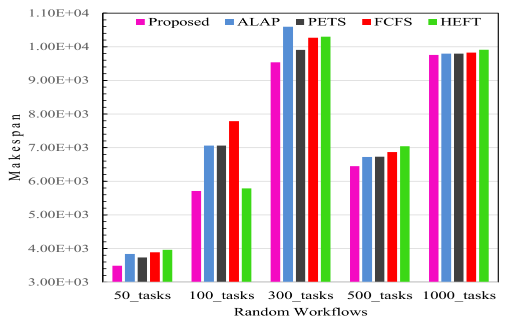

Fig. 3: Makespan comparison for random workflows

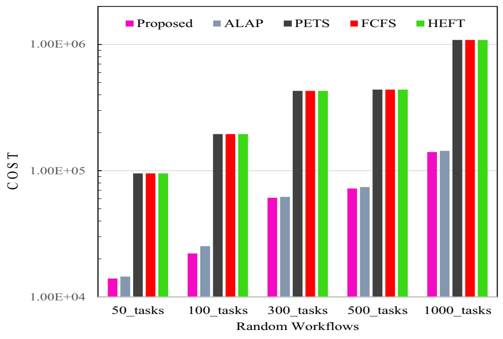

Fig. 4: Cost comparison for random workflows

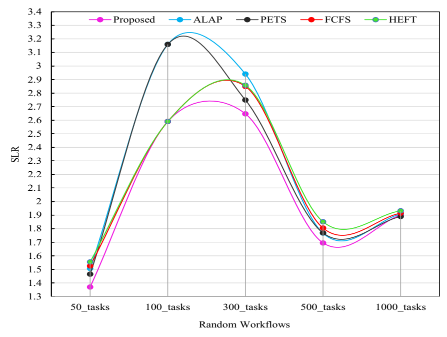

Fig. 5: SLR comparison for random workflow

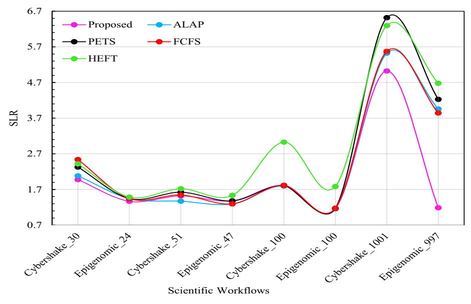

Fig. 7: SLR comparison for cybershake and epigenomics workflows

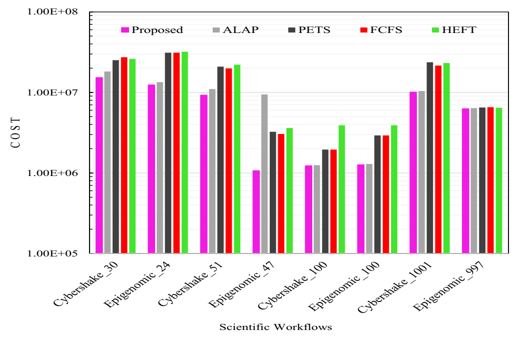

Fig. 6: Cost comparison for cybershake and epigenomics workflows

VI. Conclusion

In this paper, we have presented a cost conscious workflow scheduling heuristic using the delay characteristic of the task node in a given workflow application. We have also introduced a new priority scheme for task ordering which is being accommodated on each level of DAG. The proposed heuristic kept an effective balance between time and cost incurred for VM renting when compared to existing state of art of algorithms. We have tested the scheduling algorithm on two scientific benchmark workflows (cybershake and epigenomic) and different sizes of random DAGs upto 1000 task nodes using arbitrary distribution of ETC matrix and data transfer time. The paper also evaluates the performance metrics (*makespan*, SLR and *Cost*) and results were compared with ALAP, HEFT, PETS and FCFS based cost effective scheduling technique considering same parameters. The simulation results pacify the effectiveness of the proposed approach over existing scheduling algorithms. However, the analysis was carried out considering the static behavior of workflows instead of dynamic condition.

References

[1] E. N. Alkhanak, S. P. Lee, R. Rezaei, and R. M. Parizi, "Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues," *Journal of Systems and Software*, vol. 113, pp. 1–26, 2016.

[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 Generation computer systems*, vol. 25, no. 6, pp. 599–616, 2009.

[3] M. Malawski, G. Juve, E. Deelman, and J. Nabrzyski, "Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds," *Future Generation Computer Systems*, vol. 48, pp. 1–18, 2015.

[4] L. Zhang, K. Li, C. Li, and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," *Information Sciences*, vol. 379, pp. 241–256, 2017.

[5] M. A. Rodriguez and R. Buyya, "Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 222–235, 2014.

[6] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, 2013.

[7] M.-A. Vasile, F. Pop, R.-I. Tutueanu, V. Cristea, and J. Kołodziej, "Resource-aware hybrid scheduling algorithm in heterogeneous distributed computing," *Future Generation Computer Systems*, vol. 51, pp. 61–71, 2015.

[8] S. K. Panda, I. Gupta, and P. K. Jana, "Task scheduling algorithms for multi-cloud systems: allocation-aware approach," *Information Systems Frontiers*, pp. 1–19, 2017.

[9] I. Gupta, M. S. Kumar, and P. K. Jana, "Task duplication-based workflow scheduling for heterogeneous cloud environment," in *Contemporary Computing (IC3), 2016 Ninth International Conference on*, pp. 1–7, IEEE, 2016.

[10] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys (CSUR)*, vol. 31, no. 4, pp. 406–471, 1999.

[11] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, no. 3, pp. 260–274, 2002.

[12] E. Ilavarasan and P. Thambidurai, "Low complexity performance effective task scheduling algorithm for heterogeneous computing environments," *Journal of Computer sciences*, vol. 3, no. 2, pp. 94–103, 2007.

[13] F. Tao, Y. Feng, L. Zhang, and T. Liao, "Clps-ga: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling," *Applied Soft Computing*, vol. 19, pp. 264–279, 2014.

[14] H. Zhong, K. Tao, and X. Zhang, "An approach to optimized resource scheduling algorithm for open-source cloud systems," in *ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*, pp. 124–129, IEEE, 2010.

[15] 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 *Advanced information networking and applications (AINA), 2010 24th IEEE international conference on*, pp. 400–407, IEEE, 2010.

[16] I. Gupta, M. S. Kumar, and P. K. Jana, "Compute-intensive workflow scheduling in multi-cloud environment," in *Advances in Computing, Communications and Informatics (ICACCI), 2016 International Conference on*, pp. 315–321, IEEE, 2016.

[17] S. Bharathi, A. Chervenak, E. Deelman, G. Mehta, M.-H. Su, and K. Vahi, "Characterization of scientific workflows," in *Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on*, pp. 1–10, IEEE, 2008.