J Grid Computing (2015) 13:495–506
DOI 10.1007/s10723-015-9344-9

# Cost-Time Efficient Scheduling Plan for Executing Workflows in the Cloud

**Amandeep Verma · Sakshi Kaushal**

Received: 28 August 2014 / Accepted: 5 August 2015 / Published online: 20 August 2015

**Abstract** The emergence of Cloud Computing as a model of service provisioning in distributed systems instigated researchers to explore its pros and cons on executing different large scale scientific applications, i.e., Workflows. One of the most challenging problems in clouds is to execute workflows while minimizing the execution time as well as cost incurred by using a set of heterogeneous resources over the cloud simultaneously. In this paper, we present, *Budget and Deadline Constrained Heuristic* based upon *Heterogeneous Earliest Finish Time (HEFT)* to schedule workflow tasks over the available cloud resources. The proposed heuristic presents a beneficial trade-off between execution time and execution cost under given constraints. The proposed heuristic is evaluated for different synthetic workflow applications by a simulation process and comparison is done with state-of-art algorithm i.e. BHEFT. The simulation results show that our proposed scheduling heuristic can significantly decrease the execution cost while producing makespan as good as the best known scheduling heuristic under the same deadline and budget constraints.

A. Verma (✉) · S. Kaushal
University Institute of Engineering and Technology, Panjab University, Chandigarh, India
e-mail: amandeepverma@pu.ac.in

S. Kaushal
e-mail: sakshi@pu.ac.in

**Keywords** Workflow · Bi-criteria scheduling · HEFT · Cloud computing

## 1 Introduction

Cloud Computing is recently a booming area in distributing computing that delivers dynamically scalable services on demand over the Internet through virtualization of hardware and software [1]. It is based on a market-oriented business paradigm where users can consume these services based upon Service Level Agreement (SLA) and charged on a pay-as-you-go basis like conventional utilities [2]. The main advantages of clouds are its scalability and flexibility where user can lease and release resources/services as per the need [3]. Moreover, the cloud provider offers two resource provisioning plans, namely short-term on demand and long term reservation plans. The on-demand plan uses the dynamically provisioning of the resources at the moment when the resources are needed to fit the fluctuated and unpredictable demands. With the reservation plan, the user a priori reserves the resources in advance [4]. Amazon EC2 [5] and Go Grid [6], cloud providers offer services with both plans.

Scheduling of workflows requires massive computation and communication costs. These workflows, especially those related to scientific areas such as astronomy and biology, present a strong case for the usage of the cloud for their execution. Workflow scheduling is a process of mapping inter-dependent tasks on the available resources such that workflow application is able to complete its execution within the user's specified Quality of Service (QoS) constraints such as deadline and budget [7]. Initially, in case of grid workflows, the scheduling algorithms attempt to minimise the execution time without considering the cost of accessing resources. But, in case of cloud computing, service provider provides resources of different capabilities at different prices. Normally, faster resources are more expensive than the slower one. Thus, different scheduling plans of same workflow using different resources may result in different makespan and different monetary cost. Therefore, workflow scheduling problem in cloud, requires both time and cost constraints to be satisfied as specified by the user [8]. Time constraints ensure that the workflow is executed within the given deadline and the cost ensures that the budget specified by the user is not overshot. A good heuristic tries to balance both these values and still obtain a near optimal solution [9].

Minimizing makespan and minimizing execution cost are two conflicting objectives. In this paper, a novel, Budget and Deadline Constrained Heuristic is proposed to schedule workflow applications to cloud services that minimize the execution cost and the total execution time (i.e. *sum of processing time and data transmission time*) for running the workflow under the given budget and deadline constraint simultaneously. Our proposed heuristic includes a service level scheduling phase and a task level scheduling phase. In service level scheduling phase, the scheduler selects the suitable services from the available set of services based upon QoS constraints as specified in SLA. In the task level scheduling phase, the scheduler assigns the workflow tasks to the service from the suitable set such that the overall execution time and execution cost of running whole workflow are minimized simultaneously depending upon the spare budget and spare deadline of each workflow task. The proposed heuristic, namely, Budget *and Deadline Constraint Heterogeneous Earliest Finish Time (BDHEFT)*, is based upon *HEFT* algorithm, which minimizes the overall execution time of a workflow without considering the monetary cost and budget constraints while making the scheduling decisions. The proposed heuristic presents a beneficial trade-off between execution time and execution cost under given constraints.

The remaining paper is organized as follows: Section 2 presents the related work in the area of workflow scheduling. The problem description is presented in Section 3. The proposed heuristic, *BDHEFT*, is discussed with the help of an example in Section 4. The proposed *BDHEFT* algorithm is evaluated in Section 5 and Section 6 concludes the paper.

## 2 Related Work

Scheduling of workflows is an NP—complete problem [10]. Many heuristic algorithms such as Minimum Completion Time, Sufferage, Min-min, and Max-min are used as candidates for best-effort based scheduling strategies [11]. List scheduling has been very popular method for workflow task scheduling. In list scheduling, the priority is assigned to the workflow tasks and a task with higher priority is scheduled before a task with lower priority. There are different list based heuristic algorithms in literature like Dynamic Critical Path(DCP) [12], Dynamic Level Scheduling (DLS) [13], Critical Path on Processor (CPOP) [14], Heterogeneous Earliest Finish Time (HEFT) [14] etc. From all of these, HEFT outperforms in terms of makespan. But all of these heuristics just try to minimize the makespan without considering the monetary cost of executing the workflow tasks. So these methods are mainly suitable for grid environment.

There are also many scheduling heuristics in literature that are derived from list scheduling algorithms for workflow scheduling. Zheng and Sakellariou [15] proposed two scheduling heuristics LOSS and GAIN (based upon HEFT) for grid workflows that either tried to optimized time or cost, to meet the user's specified budget. So at a time, only one of the objectives, i.e., either time or cost is optimized. Cost and deadline constrained workflow scheduling in IaaS clouds was discussed in [16]. But, the resource model considered in the proposed algorithms consists of homogeneous resources. Saeid et al. [17] proposed two workflow scheduling algorithms for cloud environment: one-phase algorithm, IC-PCP and two-phase algorithm, IC-PCPD2. Both algorithms have a polynomial time complexity for scheduling large workflows and minimized the cost of workflow execution under deadline constrained. The authors considered different types of pricing models for simulation. Chopra and Singh [18] proposed HEFT based hybrid workflow scheduling algorithm for cost optimization within deadline in hybrid cloud. The authors used the concept of sub-deadline for rescheduling and allocation of resources in public cloud and optimized one of the objective, i.e., execution cost within assigned deadline. Similarly, the authors in [19] proposed a set of algorithms to schedule the deadline constrained bag of tasks applications on hybrid clouds to minimize the execution cost (again only one of the objectives is minimized). Also this work is only suitable for application consisting of number of independent tasks. In our previous work, we had proposed deadline and budget constrained heuristic based Genetic Algorithms to schedule workflow tasks over the cloud resources that minimize either computation cost or makesapn at a time [20–22]. The major key issue with all these heuristics and meta-heuristic techniques is that majority of them are mono-objective optimization techniques.

Only few works in the past considered bi-objective (time and cost mainly) criteria to schedule workflow tasks in distributed environment. In [23], the authors proposed Multi-Objective Evolutionary Algorithms (MOEAs) to solve multi-objective scheduling optimization problems in grid. An MOEA approach produces Pareto optimal set of solutions, which is the set consisted of all non-dominated solutions. In [24], the authors considered three conflicting objectives like execution time (makespan), total cost and reliability and proposed a multi-objective scheduling algorithm, using R-NSGA-II approach based on evolutionary computing paradigm that generates multiple scheduling solutions near the Pareto optimal front with small computation overhead. A multi-objective list scheduling (MOLS) algorithm [25] was proposed to find a dominant solution by using Pareto relation for heterogeneous environment. There are also bi-criteria scheduling heuristics derived from list scheduling algorithms for workflow scheduling. To achieve a trade-off between execution time and reliability, bi-objective dynamic level scheduling algorithm (BDLS) [26] was proposed that can be used for producing task assignment where the execution time is weighted against the failure reliability. In [27], the authors considered the time and cost and used the concept of Pareto dominance to execute large programs in the cloud to reduce the monetary cost without considering the budget and deadline constraints of user. Benyi et al. [28] proposed a fuzzy logic based Pliant system to migrate virtual machines and thus found a trade-off between energy consumption of IaaS datacenter and execution time. This work only considered the virtual machine scheduling. So, it cannot be directly applicable for workflow scheduling in clouds. From the review of literature, it has been found that majority of these multi-objective heuristics are only suitable for utility gird model.

Recently, Zheng and Sakellariou [29] proposed budget and deadline constrained, BHEFT, which is the extension of HEFT algorithm that gives *Budget and Deadline Constrained (BDC)* plan to check whether a workflow request should be accepted or not while considering the confirmed resource reservations from the other users. During creation of a BDC plan, the authors considered the spare budget for each task of workflow while selecting the resource. However, this heuristic is only applicable to the heterogeneous computing systems like utility grids, where the number of resources is fixed and if a user reserved a resource, then no one other user is able to execute its tasks on that resources for time. But, there are significant differences between clouds and utility grids. Firstly, the majority of current commercial cloud providers allow a user to launch a new instance of a resource even if that resource is reserved by some another user [5], while in case of utility grids, there are fixed resources with restricted available time slots. Secondly, current commercial clouds use the pay-as you-go pricing model where the users are charged based upon the number of time intervals that they have used. To address all these gaps, we introduced in this paper, a novel heuristic that generates a BDC schedule plan by considering the spare deadline along with spare budget while selecting the suitable resource for each workflow's task. To the best of our knowledge, this is the first multi-objective heuristic that extends a list based heuristic for workflow scheduling in cloud environment. The proposed work gives a BDC schedule plan which has significant reduced execution cost as compared to the BDC plan created by state-of-art scheduling heuristic under the same deadline and budget constraints.

## 3 System Model and Assumptions

### 3.1 Application Model and Cloud Model

A workflow application is modelled by a Directed Acyclic Graph (DAG), defined by a tuple $G(T, E)$, where $T$ is the set of $n$ tasks $\{t_1, t_2, ......, t_n\}$, and $E$ is a set of $e$ edges, representing the dependencies. Each $t_i \ \varepsilon \ T$, represents a task in the application and each edge $(t_i..........t_j) \ \varepsilon \ E$ represents a precedence constraint, such that the execution of $t_j \ \varepsilon \ T$ cannot be started before $t_i \ \varepsilon \ T$ finishes its execution [20]. If $(t_i, t_j) \ \varepsilon \ T$, then $t_i$ is the parent of $t_j$, and $t_j$ is the child of $t_i$. A task with no parent is known as an *entry* task and a task with no children is known as *exit* task. The task size $(z_i)$ is expressed in *Million of Instructions (MI)*.

Our cloud model consists of a service provider which offers, $m$, computational resources, $R = \{r_1, r_2, .., r_m\}$ at different processing power and different prices. It is assumed that any resource from the set, $R$ is able to execute all the tasks of a workflow. The processing power of a resource, $r_p \ \varepsilon \ R$, is expressed as *Million of Instruction per Second (MIPS) and is denoted by* $PP_{r_p}$. The pricing model is based upon pay-as-you-go basis similar to the current commercial clouds i.e. the user are charged based upon the number of time interval that they have used the resources, even if they have not completely used the last time internal.

Each task can be executed on different resources. The execution time, $ET_{(i,p)}$, of a task $t_i$ on a resource, $r_p$ is calculated by the following equation:

$$ET_{(i,p)} = \frac{z_i}{PP_{r_p}} \tag{1}$$

and the execution cost $EC_{(i,p)}$ is given by:

$$EC_{(i,p)} = \mu_p * ET_{(i,p)} \tag{2}$$

where $\mu_p$ is the price unit of using resource $r_p$ for each time interval. Moreover, all the computation resources of a service provider are assumed to be in same physical region, so data storage cost and data transmission costs are assumed to be zero and the average bandwidth between these resources is assumed to be roughly equal. Only, time to transmit data between two dependent tasks *(ct)*, which are mapped to different resources, is considered during experiment. Most of the commercial clouds (like Amazon) transfer the internal data at free of cost, so the data transfer cost is assumed to be zero in our model. Even, the data storage or storage cost have no effect on our scheduling heuristic, so we do not consider them in the model.

### 3.2 Workflow Scheduling Model

The different entities in our workflow scheduling model are: ***User, Scheduler and Resource Provider (RP)***. **RP** has a set of computational resources with different capabilities. The information related to set of available resources like their processing power and prices is publicly published. The RP responds to the queries from the scheduler about the availability of requested resources. The **user** submits a workflow application along with budget, $B$ and deadline, $D$ to the scheduler. The **scheduler** decides how to execute workflow tasks over available resources. The whole steps are summarized in Fig. 1.

In the following section, the proposed heuristic, namely, *BDHEFT*, which used this system model, is presented.

## 4 Proposed Heuristic

The proposed heuristic, *BDHEFT*, is based upon *HEFT* [11], which is a well known DAG scheduling heuristic. It is an extension of HEFT and considers budget and deadline constraints while scheduling tasks over available resources. In *BDHEFT*, each task is assigned a priority using upward rank as defined in HEFT and is given by (3).

$$rank(t_i) = w_i + \max_{t_j \varepsilon \ succ(t_i)} \{d_{ij} + rank(t_j)\} \tag{3}$$

where $w_i$ is the average execution time of the task on the different computing resources; $succ(t_i)$ includes all the children tasks of $t_i$; $d_{ij}$ is the average data transmission time from a task $t_i$ to $t_j$.

Let $EST(t_i, r_p)$ and $EFT(t_i, r_p)$ denote, the earliest execution start time and the earliest finish time of

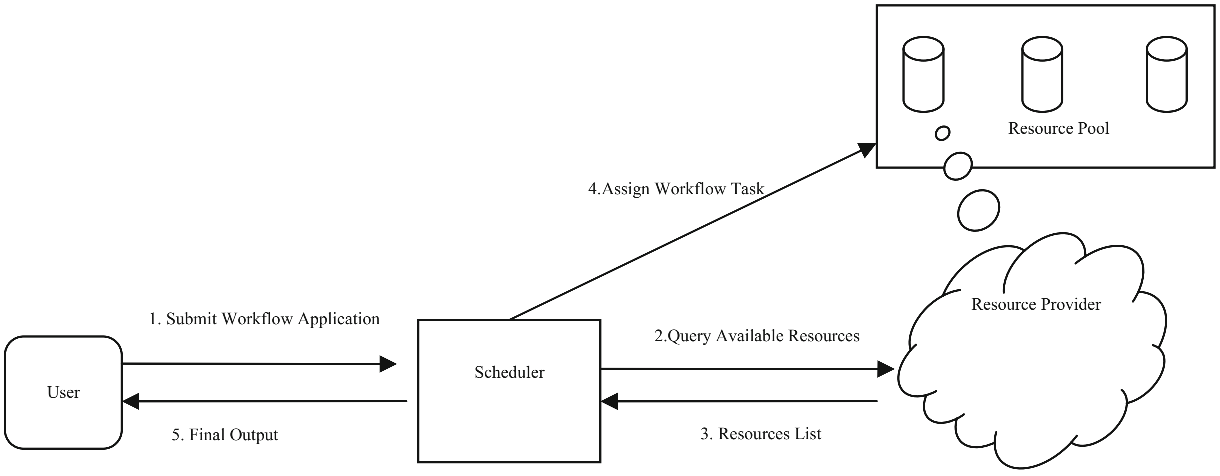


**Fig. 1** Workflow scheduling model

a task $t_i$ on a resource $r_p$, respectively. For the entry task, we have:

$$EST(t_{entry}, r_p) = avail(r_p) \tag{4}$$

For the other tasks in DAG, we compute EST and EFT recursively as follows:

$$EST(t_i, r_p) = max \begin{cases} avail(r_p) \\ \max\limits_{t_j \varepsilon\ pred(t_i)} \{AFT(t_j) + ct_{ij}\} \end{cases} \tag{5}$$

$$EST(t_i, r_p) = ET_{(i,p)} + EST(t_i, r_p) \tag{6}$$

where $pred(t_i)$ is the set of immediate predecessor tasks of task $t_i$, and $avail(r_p)$ is the time when the resource $r_p$ is ready for task execution. The inner ***max*** block in the *EST* equation returns the ready time. The actual start and finish time of task $t_i$ on service $r_p$, is denoted by $(ASTt_i, r_p)$ and $AFT(t_i, r_p)$, respectively, may be different from its earliest start time $(ESTt_i, r_p)$ and finish time $(EFTt_i, r_p)$. The makespan is equal to the maximum of actual finish time of the exit tasks $t_{exit}$.

$$M = max\{AFT(t_{exit})\} \tag{7}$$

The makespan is also referred to as the running time for the entire DAG.

The *BDHEFT* heuristic includes two phases: a service level scheduling phase and a task level scheduling phase.In service level scheduling phase, the tasks are selected in descending order according to their rank using (3). For each selected task, the set of best possible resources is constructed using the six variables: *Spare Workflow Budget (SWB)*, *Spare Workflow Deadline (SWD)*, *Current Task Budget (CTB)*, *Current Task Deadline (CTD)*, *Budget Adjustment Factor (BAF) and Deadline Adjustment Factor (DAF)*. From these variables, *SWB*, *CTB* and *BAF* are same as defined in [29] and *SWD*, *CTD*, and *DAF* are proposed by us. For a task $t_i$, the value of these variables is given by (8) to (13), as follows:

$$SWB_i = B - \sum\nolimits_{k=0}^{i-1} c_i - \sum\nolimits_{j=1}^{n-1} \bar{c}_j \tag{8}$$

$$SWD_i = D - \sum_{k=0}^{i-1} e_i - \sum_{j=i}^{n-1} \bar{e}_j \tag{9}$$

$$BAF_i = \begin{cases} c_i / \sum_{k=1}^{n-1} \bar{c}_k, & SWB_i \geq 0 \\ 0, & SWB_i < 0 \end{cases} \tag{10}$$

$$DAF_i = \begin{cases} e_i / \sum_{k=i}^{n-1} \bar{e}_k, & SWD_i \geq 0 \\ 0, & SWD_i < 0 \end{cases} \tag{11}$$

$$CTB_i = \bar{c}_i + SWB_i * BAF_i \tag{12}$$

$$CTD_i = \bar{e}_i + SWD_i * DAF_i \tag{13}$$

where $B$ and $D$ are given budget and deadline respectively, $c_k$ and $e_k$ are the execution cost and execution time of allocated task $k$, $\bar{c}_j$ and $\bar{e}_j$ are the average execution cost and average execution time of un-allocated task $j$ over different resources, $n$ is the total number of tasks in the workflow. It is clear from the above equations that $SWB_i$ and $SWD_i$ are the values that are used to predict the expected spared budget and spared deadline when planning task $t_i$. $BAF_i$ and $DAF_i$ are used to adjust the amount of spare budget and spare deadline for the whole workflow given to the current task.

Based on the allocated deadline and budget to a task $t_i$, a set $BS_i$ is constructed by considering possible resources for a task $t_i$, by:

$$BS_i = \{S_{i,p} | \ni S_{i,p}, EC_{i,p} \le CTB_i \ and \ ET_{i,p} \le CTD_i\} \quad (14)$$

where $S_{i,p}$ is a set $p$ of resources for a task $i$, from the given $m$ resources such that $EC_{i,p}$ of task $i$ over resource $p$ must be less than or equal to current task budget and $ET_{i,p}$ of task $i$ over resource $p$ must be less than or equal to current task deadline.

In the task level scheduling phase, the best possible resource for a task is selected using the following rules:

1. if $BS_i \neq \phi$, then the best resource is chosen from this set that minimizes the following expression:

   $$\alpha * EFT_{i,j} + (1-\alpha) * EC_{(i,j)} \ for \ all \ j \varepsilon BS_i \quad (15)$$

   where $EFT_{i,j}$ is the earliest finish time and $EC_{i,j}$ is the execution cost of a task $i$ over all possible $j$ resources in $BS_i$ respectively and $\alpha$ is the cost-time balance factor in a range of [0,1] which represents the user preference for execution time and execution cost.
2. if $BS_i = \phi \ and \ SWB >= 0$, then the resource from all the available resources that minimize the above equation is chosen.
3. if $BS_i = \phi \ and \ SWB < 0$ and $SWD < 0$, the cheapest resource is selected from all the available ones.

The algorithm terminates when all tasks according to their rank are considered. The Fig. 2 outlines the proposed heuristic, BDHEFT and its working with an illustrative example is examined in Section 4.1.

```
Input:   DAG G with Budget B and Deadline D
Output : Workflow Schedule Plan
1.  Compute rank using equation (3) for all the tasks.
2.  Sort all the tasks in a list in descending order of rank.
3.  for i=0 to n do
      (i)   Select i^th task from the list.
      (ii)  Compute SWB and SWD for a task using equations (8) and (9).
      (iii) Compute CTB and CTD for a task $t_i$ using equations (10) and (11).
      (iv)  Construct the possible resources set $BS_i$ using equation (14).
      (v)   for each resource in $BS_i$, for task $t_i$, do
                Compute EFT of mapping task $t_i$ .
            end for
      (vi)  Select a resource for a task $t_i$ using the defined selection rules.
    end for
```

**Fig. 2** The BDHEFT Heuristic

## 4.1 An Example

An example workflow with 9 tasks as shown in Fig. 3a is considered to illustrate the working of *BDHEFT*. Each edge is representing the amount of data to be transferred (in MB) between the dependent tasks. The Fig. 3b shows the execution time of these tasks on three different available resources along with their ranks computed using (3). The price for running a task on different resources is shown in Fig. 3c. It is assumed the bandwidth between all the resources is 20 Mbps and all resources are able to execute all types of workflow tasks.

Assume a deadline of 200 time units and budget of 100 price units. Table 1 summarizes the values of different variables and the steps executed using *BDHEFT*. The workflow tasks are sorted in order of their rank. The value of $\alpha$ is chosen as 0.5 i.e. there is 50:50 preference of user for both time and cost. The values for different variables computed for each task clearly shows how *BDHEFT* selects the resource that must be within deadline and budget constraint. Like, for task 1, the possible set of resources consists of two resources: $r_0$, and $r_2$. So, *BDHEFT* will choose a resource that minimizes the expression (15). For task 2, 5, and 3, their corresponding *BS* is null, so for task 2, rule 3 is applied and for task 3 and 5, rule 2 is applied.

### 4.2 Time Complexity

To find out the time complexity of *BDHEFT* algorithm, suppose that the scheduler receives a Workflow, $G(T, E)$ as an input with $n$ tasks and $e$ dependencies. As $G$ is directed graph, so maximum number of dependencies is $((n-1)(n-2))/2 \approx O(n^2)$. The first step of the algorithm is to find the rank of all the tasks which requires processing of all workflow tasks and edges, so its time complexity equals $O(n+e) \approx O(n^2)$. Similarly, the step 3, consists of two nested loops. The outer loop is for $n$ tasks and inner loop is for all the possible $m$ resources. Therefore time complexity of scheduling all tasks is $O(n.m)$. If we assume that the number of possible resources is at most equal to $n$, then the time complexity of step 3 becomes $O(n^2)$ which gives the overall time complexity of *BDHEFT* algorithm equals $O(n^2)$.

## 5 Performance Evaluation

In this section, the simulation of the proposed heuristic, BDHEFT is presented. To evaluate the proposed workflow scheduling algorithm, we used five synthetic workflows based on realistic workflows from diverse scientific applications, which are:

- Montage: Astronomy
- Genome: Biology
- CyberShake: Earthquake
- LIGO: Gravitational physics
- SIPHT: Biology

The detailed characterization for each workflow including their structure, data and computational requirements can be found in [30]. The Directed Acyclic Graph in XML (DAX) format for all these workflows are available at website (http://confluenece.peagasus.isi.edu/display.peagasus/WorkflowGenerator). Figure 4 shows the approximate structure of each workflow.

### 5.1 Experiment Setup

For simulation, we assume a cloud environment consisting of a service provider, which offers 20 different computation resources with different processing speed and hence with different prices. For this study, we have used the CloudSim [31] simulator. The existing CloudSim simulator allows modelling and simulating cloud environment by dealing only with single workload. It is not suitable for workflow scheduling as multiple tasks need to be scheduled together. So, the core framework of CloudSim simulator is extended to handle workflow scheduling. One of the crucial changes is to read DAX files (http://confluenece.peagasus.isi.edu/display.peagasus/WorkflowGenerator) of the workflow structures and to extract the required parameters like run time, input file size, output file size and task dependencies. The processor speeds of different resources are selected randomly in the range of 1000–10000 MIPS such that fastest resource is roughly ten times faster than the slowest one as well as ten times more expensive. The average bandwidth between these resources is set equal to 20 Mbps. We are using the pricing model similar to Amazon. The reasonable values for deadline $D$, and Budget $B$ are generated as:

***Deadline D*** $= LB_D + k_1 * (UB_D\text{-}LB_D)$, where $LB_D = M_{HEFT}$ *(makespan of HEFT)*, $UB_D = 5 * M_{HEFT}$ and $k_1$ is a deadline ratio in range from 0 to 1.

***Budget B*** $= LC_B + k_2 * (UC_B - LC_B)$, where $LC_B$ is the lowest cost obtained by mapping each task to the cheapest service and $UC_B$ is the highest cost obtained conversely and $k_2$ is a budget ratio in range from 0 to 1.

### 5.2 Performance Metrics

The performance metrics chosen for the comparison are *Normalized Schedule Cost (NSC)* and *Normalized Schedule Length (NSL).*

The *NSC* of a schedule is calculated as:

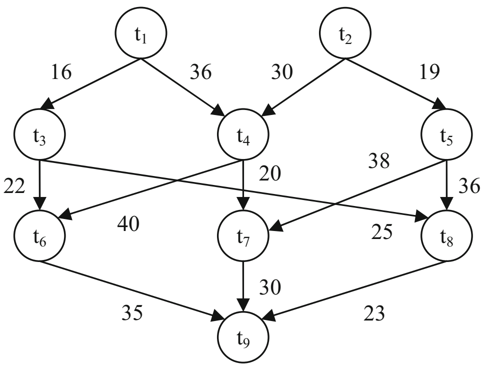
$$NSC = \frac{Total\ Cost}{C_c} \tag{16}$$

where $C_c$ is the execution cost of the same workflow by executing all the tasks on the cheapest resource, according their precedence constraints.

The *NSL* of a schedule is calculated as:

$$NSL = \frac{Total\ Execution\ Time}{M_c} \tag{17}$$

where $M_c$ is the execution time of the same workflow by executing all the tasks on the fastest resource, according their precedence constraints.



(a) A Sample Workflow

| Tasks | $R_0$ | $R_1$ | $R_2$ | Rank |
|---|---|---|---|---|
| $T_1$ | 22 | 42 | 10 | 140.07 |
| $T_2$ | 26 | 34 | 12 | 137 |
| $T_3$ | 28 | 40 | 14 | 104.53 |
| $T_4$ | 24 | 34 | 11 | 101 |
| $T_5$ | 30 | 40 | 14 | 104.6 |
| $T_6$ | 22 | 38 | 10 | 62 |
| $T_7$ | 26 | 44 | 12 | 64 |
| $T_8$ | 30 | 50 | 20 | 67.2 |
| $T_9$ | 24 | 36 | 14 | 24.67 |

(b) Estimated execution time of tasks over available resources, along with their rank

| Service | Price |
|---|---|
| $R_0$ | 0.40 |
| $R_1$ | 0.29 |
| $R_2$ | 0.92 |

(c) Price unit of different services

**Fig. 3** An example of BDHEFT Heuristic

## 5.3 Experiment Results

For comparison purpose, we adapted a state-of-art workflow scheduling algorithm for fixed heterogeneous computing resources with reserved time slots to the cloud environment. This algorithm is called *BHEFT* [29]. We have compared *BDHEFT* algorithm and *BHEFT* algorithm (without considering the resource reservations) on the basis of makespan and monetary cost. For the purpose of performance comparison of two algorithms, an average value of *NSC* and *NSL* is captured through 50 runs of simulations. The comparison is further enhance by choosing three different values of cost-time balance factor ($\alpha$), i.e., $\alpha = 0.3$, 0.5, and 0.7. The different variants of *BDHEFT* corresponding to the values of $\alpha$ are represented by *BDHEFT (0.3), BDHEFT (0.5)* and *BDHEFT (0.7)* as shown in Figs. 4 and 5. Figures 4 and 5 shows the average *NSC* and average *NSL*, respectively, of scheduling different workflows with *BDHEFT and BHEFT* for three different values of deadline ration ($k_1$), i.e., $k_1 = 0.2$, 06, *and* 1.0

**Table 1** An example to illustrate the working of BDHEFT using the workflow in Fig. 3a

| $Task_k$ | $SWB_k$ | $CTB_k$ | $SWD_k$ | $CTD_k$ | $BS_k$ | Selected resource | Start time | Finish time | Cost |
|---|---|---|---|---|---|---|---|---|---|
| 1 | −1.41 | 10.06 | −35.66 | 24.67 | $r_0,r_2$ | $r_0$ | 0 | 22 | 8.8 |
| 2 | −0.15 | 10.43 | −32.99 | 24 | – | $r_1$ | 0 | 34 | 9.86 |
| 5 | 0.42 | 12.22 | −42.99 | 28 | – | $r_2$ | 41.6 | 55.6 | 12.88 |
| 3 | 0.98 | 12.05 | −44.99 | 27.33 | – | $r_0$ | 22 | 50 | 11.2 |
| 4 | 1.67 | 10.15 | −15.66 | 23 | $r_2$ | $r_2$ | 55.6 | 66.6 | 10.12 |
| 8 | 1.41 | 15.42 | −3.66 | 33.33 | $r_0$ | $r_0$ | 96.6 | 126.6 | 12 |
| 7 | 4.38 | 12.96 | −0.33 | 27.33 | $r_0,r_2$ | $r_2$ | 66.6 | 78.6 | 11.04 |
| 6 | 4.74 | 11.89 | 15 | 30.62 | $r_0,r_2$ | $r_2$ | 78.6 | 88.6 | 9.2 |
| 9 | 5.21 | 16.18 | 28.33 | 53 | $r_0,r_1,r_2$ | $r_0$ | 126.6 | 150.6 | 9.6 |

Total Time Taken = 150.6 and total cost = 95.7, given deadline, D = 200 units and budget, B= 100 units

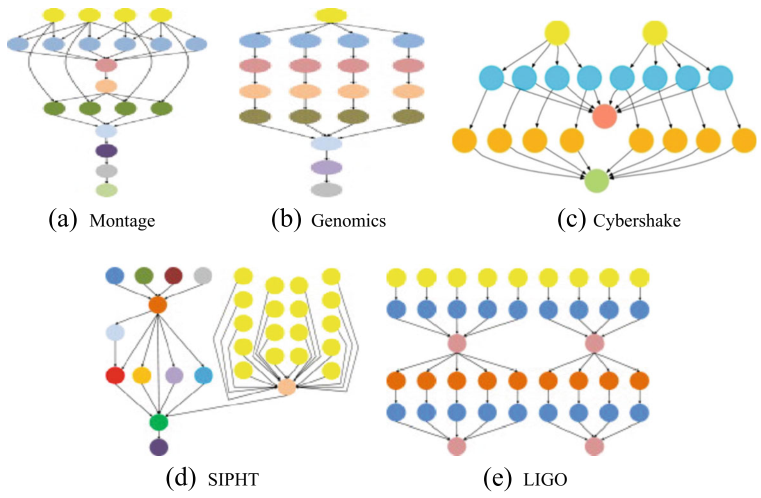
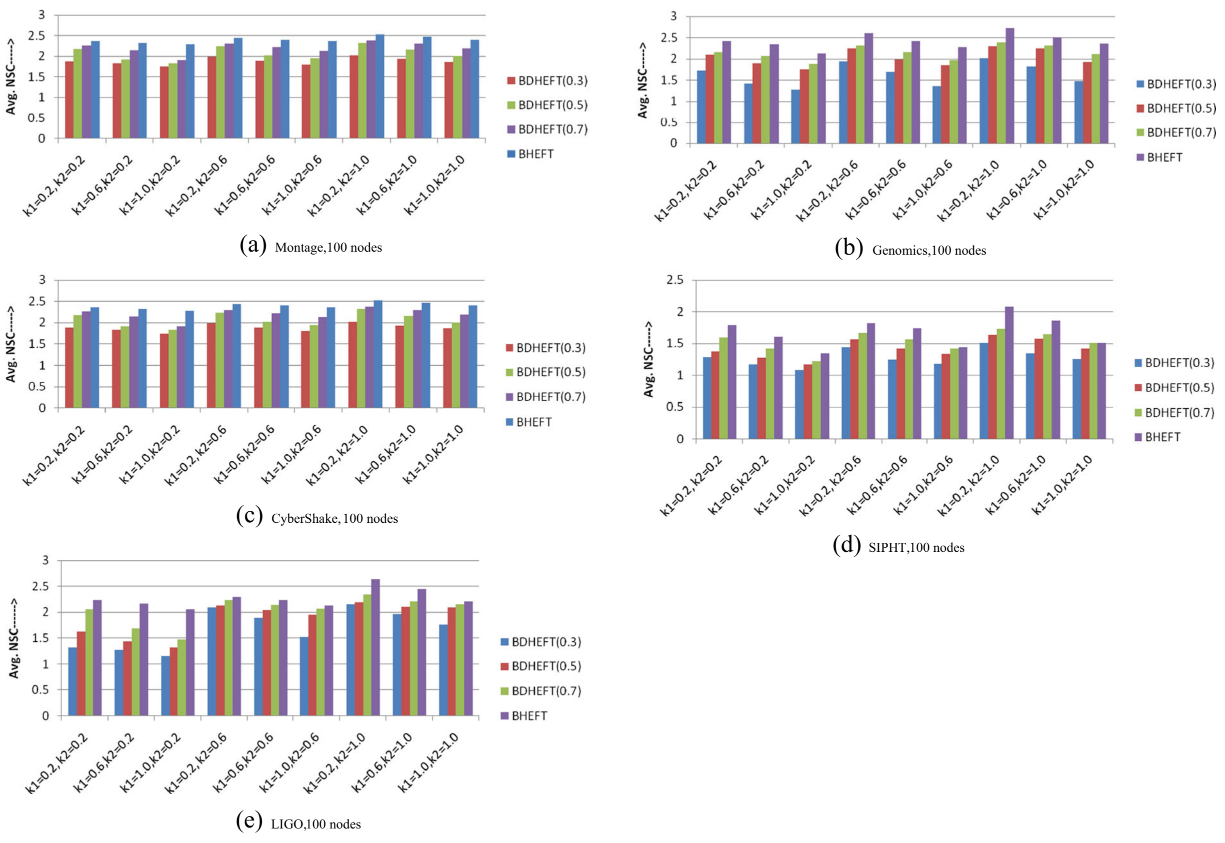


**Fig. 4** Structure of various workflows [30]

(a) Montage (b) Genomics (c) Cybershake (d) SIPHT (e) LIGO

and three different values of budget ration ($k_2$), i.e., $k_2 = 0.2, 0.6,$ *and* 1.0, in total 9 combinations.

The variation in average *NSC* and average *NSL* for three different variants of *BDHEFT*, i.e., *BDHEFT (0.3), BDHEFT (0.5)* and *BDHEFT (0.7)* is due to the fact that when the user sets $\alpha = 0.3$, then the user gives more preference to minimize the total execution cost as compared to minimize the total execution time (according to expression (15)), for complete workflow. As the value of $\alpha$ increase, the user preference



(a) Montage,100 nodes (b) Genomics,100 nodes (c) CyberShake, 100 nodes (d) SIPHT,100 nodes (e) LIGO,100 nodes

**Fig. 5** Average NSC of different workflows

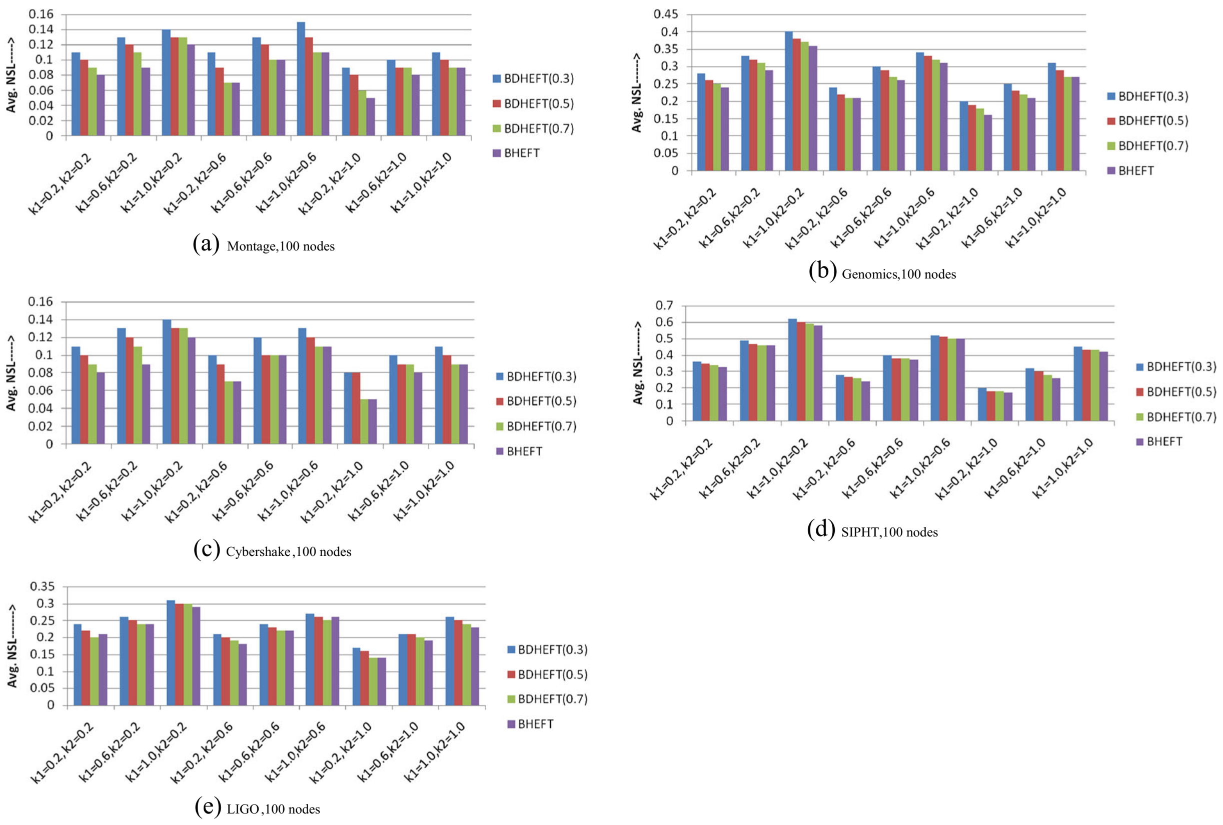


**Fig. 6** Average NSL of different workflows

for minimizing the execution cost is decreasing and the value of *NSC* increases and *NSL* decreases respectively. At a fixed budget ratio, i.e., $k_2$, the deadline is relaxed by increasing the value of deadline ratio, i.e., $k_1$ from 0.2 to 1.0. As a result, the scheduler is able to choose the cheaper resources or slowest resources for assigning workflow tasks. Thus, the *NSC* of created schedule plan is reduced and the *NSL* of same plan is increased under the same budget ratio as shown in Figs. 5 and 6, respectively. Similarly, by fixing the deadline ratio, i.e., $k_1$, and by varying the value of budget ratio, i.e., $k_2 = 0.2$, 0.6, and 1.0, the budget is relaxed for each of these values. Now, the scheduler is able to choose the expensive resources or

**Table 2** Comparative results of BDHEFT vs. BHEFT

| Algorithm / Workflow structure | BDHEFT(0.3) over BHEFT | | BDHEFT(0.5) over BHEFT | | BDHEFT(0.7) over BHEFT | | Average of BDHEFT(0.3), BDHEFT(0.5), and BDHEFT(0.7) over BHEFT | |
|---|---|---|---|---|---|---|---|---|
| | Cost | Makespan | Cost | Makespan | Cost | Makespan | Cost | Makespan |
| Montage (100 nodes) | −20 % | +21 % | −16 % | +12 % | −7 % | +3.4 % | −14.33 % | +12.1 % |
| Genome (100 nodes) | −26 % | +11.5 % | −16 % | + 7 % | −12 % | 0 % | −18 % | +6.3 % |
| CyberShake (100 nodes) | −20 % | +22 % | −13 % | +11 % | −10 % | 0 % | −14.3 % | +11 % |
| SIPHT (100 nodes) | −32 % | +8.1 % | −24 % | +3.8 % | −11 % | +2.7 % | −22.33 % | +4.9 % |
| LIGO (100 nodes) | −34 % | +9 % | −32 % | +4 % | −27 % | 0% | −31 % | +4.3 % |

fastest resources for assigning workflow tasks. Thus tend to increase the *NSC* and decrease the *NSL* of created schedule plan under the same deadline ratio as shown in Figs. 5 and 6, respectively, for different types of workflows under study. Table 2 shows the overall comparison of execution cost and makespan of schedule plan created by *BDHEFT (0.3), BDHEFT (0.5)* and *BDHEFT (0.7)* over schedule plan created by *BHEFT*. The last column of Table 2 shows the average comparison of execution cost and makespan of schedule plan created by *BDHEFT (0.3), BDHEFT (0.5)* and *BDHEFT (0.7)* over the schedule plan created by *BHEFT*. We can see the average cost of all the variants of *BDHEFT* is reduced by 14.33 % for Montage with about 12.1 % more makespan, by 18 % for Genome with about 6.3 % more makespan, by 14.3 % for CyberShake with about 11 % more makespan, by 22.33 % for SIPHT with 4.9 % more makespan and by 31 % for LIGO with 4.3 % more makespan as in comparison with the cost and makespan of *BHEFT*. The overall results shows that all variants of *BDHEFT* algorithm outperform *BHEFT* algorithm significantly by reducing the execution cost of schedule while making the makespan as good as given by *BHEFT* under the same deadline and budget constraint and using same pricing model in all cases.

This work can be applicable to handle real scientific workflow applications in real IaaS cloud environment like creating instances over Amazon EC2 of different capacities at different costs.

## 6 Conclusion and Future Work

In this paper, we devise a new workflow scheduling algorithm, namely, *BDHEFT*, for cloud environment. The *BDHEFT* algorithm is an extension of *HEFT* algorithm. The proposed algorithm is evaluated with synthetic workflows that are based on realistic workflows with different structures and different sizes. The comparison of proposed algorithm is done with *BHEFT* heuristic, under same deadline and budget constraint and pricing. The simulation results show that our proposed algorithm outperforms *BHEFT* algorithm in terms of monetary cost while producing the makespan as good as produced by *BHEFT* algorithm. In future, we intend to improve our work by embedding the results of *BDHEFT* into meta-heuristic multi-objective optimization techniques and try to find the near optimal schedule plan for real cloud environment.

## References

1. Foster, I., Zhao, Y., Raicu, L., Lu, S.: Cloud computing and grid computing 360-degree compared. In: Proceeding of Grid Computing Environment Workshop, Austin, pp. 1–10 (2008)
2. Gabriel, M., Wolfgang, G., Calvin, J.R.: Hybrid computing—where hpc meets grid and cloud computing. J. Futur. Gener. Comput. Syst. **27**(5), 440–453 (2011)
3. Verma, A., Kaushal, S.: Cloud computing security issues and challenges: a survey. In: Proceeding of International Conference on Advances in Computing and Communications, Part-IV, Kochi, India. Series Title: Communications in Computer and Information Sciences, vol. 193, pp. 445–454. Springer (2011)
4. Sinadon, C., Bu-Sung, L., Dusit, N.: Optimization of resource provisioning cost in cloud computing. IEEE Trans. Serv. Comput. **5**(2), 166–177 (2012)
5. Amazon EC2, http://aws.amazon.com/ec2 (2014)
6. Go Grid, http://www.gogrid.com (2014)
7. Taylor, I., Deelman, E., Gannon, D., Shields, M.: Workflows for e-science: scientific workflows for grid, 1st edn. Springer (2007)
8. Pandey, S.: Scheduling and management of data intensive application workflows in grid and cloud computing environment. PhD Thesis, University of Melbourne, Australia (2010)
9. Ke, L., Hai, J., Jinjun, C.X.L., Dong, Y.: A compromised time-cost scheduling algorithm in SwinDeW-C for instance-intensive cost-constrained workflows on cloud computing platform. Int. J. High Perform. Comput. Appl. 1–16 (2010)
10. Yu, J., Buyya, R.: Workflow scheduling algorithms for grid computing. In: Xhafa, F., Abraham. A. (eds.) Metaheuristics for scheduling in distributed computing environment, Springer, Berlin. ISBN: 978-3-540-69260-7 (2008)
11. Yu, J., Buyya, R.: Taxonomy of workflow management systems for grid computing. J. Grid Comput. **3**(1–2), 171–200 (2008)
12. Kwok, Y.K., Ahmad, I.: Dynamic critical path scheduling: effective techniques for allocating task graphs onto multiprocessors. IEEE Trans. Parallel Distrib. Syst. **7**(5), 506–521 (1996)
13. Sih, G.C., Lee, E.: A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architecture. IEEE Trans. Parallel Distrib. Syst. **4**(2), 175–187 (1993)
14. Haluk, T., Salim, H., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogenous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
15. Sakellariou, R., Zhao, H., Tsiakkouri, E., Dikaiakos, M.D.: Scheduling workflows with budget constraint. In: Gorlatch, S., Danelutto, M. (eds.) Integrated Research in GRID Computing, pp. 189–202. Springer (2007)

16. Malawki, M., Juve, G., Deelman, E., Nabrzyski, J.: Cost and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds. In: Proceeding of IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, UT, pp. 1–11 (2012)
17. Saeid, A., Mahmoud, N., Dick, H.J.E.: Deadline constrained workflow scheduling algorithms for Infrastructure as a service clouds. J. Futur. Gener. Comput. Syst. **29**(1), 158–169 (2013)
18. Chopra, N., Singh, S.: HEFT based workflow scheduling algorithm for cost optimization within deadline in hybrid clouds. In: Proceeding of Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT), India, pp. 1–6 (2013)
19. Bossche, R.V.D., Vanmechelen, K., Broeckhove, J.A.: Online cost-efficient scheduling of deadline-constrained workloads on hybrid clouds. J. Futur. Gener. Comput. Syst. **29**(4), 973–985 (2013)
20. Verma, A., Kaushal, S.: Deadline and budget distribution based cost-time optimization workflow scheduling algorithm for cloud. In: IJCA Proceeding of International Conference on Recent Advances and Future Trends in IT, Patiala, India, pp. 1–4 (2012)
21. Verma, A., Kaushal, S.: Deadline constraint heuristic based genetic algorithm for workflow scheduling in cloud. J. Grid Util. Comput. **5**(2), 96–106 (2014)
22. Verma, A., Kaushal, S.: Budget constraint priority based genetic algorithm for workflow scheduling in cloud. In: Proceeding of IET International Conference on Recent Trends in Information, Telecommunication and Computing, India, pp. 8–14 (2013)
23. Aimin, Z., Bo-Yang, Q., Hui, L.C., Shi Zheng, Z., Ponnuthurai, N.S., Qingfu, Z.: Multiobjective evolutionary algorithms: A survey of the state of the art. J. Swarm Evol. Comput. **1**(1), 32–49 (2011)
24. Garg, R., Singh, A.K.: Multi-objective optimization to workflow grid scheduling using reference point based evolutionary algorithm. Int. J. Comput. Appl. **22**(6), 44–49 (2011)
25. Fard, H.M., Prodon, R., Barrionuevo, J.J.D., Fahringer, T.: A multi-objective approach for workflow scheduling in the heterogeneous environment. In: Proceeding of IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, Ottawa, Canada, pp. 300–309 (2012)
26. Dogan, A., Ozguner, R.: Bi-objective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems. Comput. J. **48**(3), 300–314 (2005)
27. Su, S., Li, J., Huang, Q., Huang, X., Shuang, K., Wang, J.: Cost-efficient task scheduling for executing large program in the cloud. J. Parallel Comput. **39**(4–5), 177–188 (2013)
28. Benyi, A., Dombi, J.D., Kertesz, A.: Energy-aware VM Scheduling in IaaS Clouds using Pliant logic. In: Proceeding of the 4th International Conference on Cloud Computing and Services Science (CLOSER'14), Barcelona, Spain, pp. 519–526 (2014)
29. Zheng, W., Sakellariou, R.: Budget-deadline constrained workflow planning for admission control. J. Grid Comput. **11**(4), 633–651 (2013)
30. Bharathi, S., Lanitchi, A., Deelman, E., Mehta, G., Su, M.H., Vahi, K.: Characterization of scientific workflows. In: Workshop on Workflows in Support of Large Scale Science, CA, USA, pp. 1–10 (2008)
31. Rodrigo, N.C., Ranjan, R., Anton, B., Cesar, A.F.D.R., Buyya, R.: Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. J. Softw. Pract. Exp. (SPE) **41**(1), 23–50 (2011)