International Journal of Recent Trends in Engineering & Research
ISSN (ONLINE) : 2455 – 1457
IMPACT FACTOR : 4.101

A Novel Resource Selection Method for Cost Optimized Workflow Scheduling with Deadline Constraint using Particle Swarm Optimization for IaaS Cloud

Kezia Rani.B[1], Dr.A.Vinaya Babu[2]
Dept. of Computer Science,JNTUHCEHHyderabad, INDIA

Abstract— Allocation and Scheduling of Cloud resources is an important area in Cloud Computing with minimum execution cost and time. As the cloud is a collection of software and hardware resources and the resources in the cloud are allocated by the cloud service providers, to several users simultaneously based on the user's requests, scheduling and optimization play an important role in better resource utilization, faster execution and minimizing the cost incurred in executing these applications. The algorithm used in this paper is based on the meta-heuristic optimization technique, Particle Swarm Optimization (PSO), which is a latest optimization technique. In this paper we present our work which aims to minimize the cost of execution of the workflow while maximizing the makespan to reach the deadline time with a deadline factor and less than or equal to one and hence the title fractional deadline time. The choice of cloud resources is an important factor which controls the cost and makespan of the workflow. The resources used are ordered based on cost and a varied sub-set of resources are considered which resulted in better cost minimization and the features of scalability and elasticity is taken advantage of, for minimizing the cost while meeting the deadlines. This work also compares two billing schemes, namely hourly billing and minute billing. A Java Application with Netbeans IDE was developed using CloudSim framework for implementation of the algorithm. Popular scientific workflows were considered as data for evaluation. The results obtained indicate improved cost minimization with deadline factors than the methods used in the current state-of-the-art methods.

IndexTerms— Cloud computing, resource provisioning, scheduling, scientific workflow, Particle Swarm Optimization

I. INTRODUCTION

Cloud computing is a new computing paradigm which aims to provide the consumer or end-user, computing environment with QOS (Quality of service) and based on the dynamic requirements. The definition of cloud computing according to National Institute of standards and Technology is "Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage application and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]". Scalability and Elasticity, Multitenancy, Reliability, Device and location independence, Pay-as-you-go-model are some of the important characteristics of Cloud Computing [2]. The three service models defined by NIST include Cloud Software as a Service (SaaS), Cloud Platform as a Service (PaaS), Cloud Infrastructure as a Sservice (IaaS).The different types of Clouds are Public clouds, Private Clouds, Hybrid clouds and community Clouds.

A. Scientific Workflows

A workflow is defined by a set of computational tasks with dependencies between them. Figure 1 shows a sample workflow with nine tasks. Workflow application can be represented using Directed

Acyclic Graph, W = (T, E) composed of a set of tasks T = {t_1, t_2, . . . , t_n }, and a set of directed edges E [18]. An edge e_{ij} of the form (t_i, t_j) exists if there is a data dependency between t_i and t_j, then t_i is said to be the parent task of t_j and t_j is said to be the child task of t_i. A child task cannot run until all of its parent tasks have completed their execution and its input data is available in the corresponding compute resource.

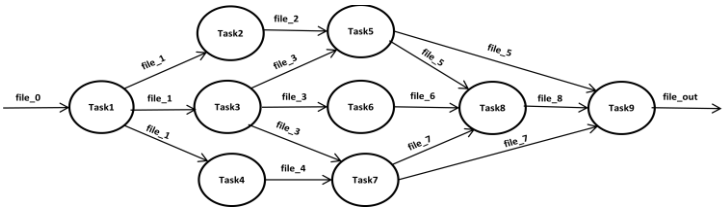


Fig.1 Sample workflow with nine tasks.

Some examples of scientific workflows are LIGO, SIPHT and Epigenomics, Montage and Cybershake. Figures 2 shows the structure of these scientific workflows respectively.

Montage workflow [3] is an astronomy application used to create custom mosaics of the sky, based on a set of input images. The structure of this workflow is shown in Figure 2.

Cybershake [4] is a earthquake hazard characterization application used by the Southern California Earth-quake Centre.

LIGO: The Laser Interferometer Gravitational Wave Observatory (LIGO) [5] application is used in the field of physics, to detect gravitational waves.

SIPHT: In bioinformatics, automation of the process of searching for sRNA encoding-genes for all bacterial replicas is performed by SIPHT in the National Centre for Biotechnology Information database [6].

Epigenomics: Epigenomics is used in the bioinformatics field [7], and the workflow automates the execution of various genome sequencing operations.
The full description of these workflows is presented by Juve et al. [8]

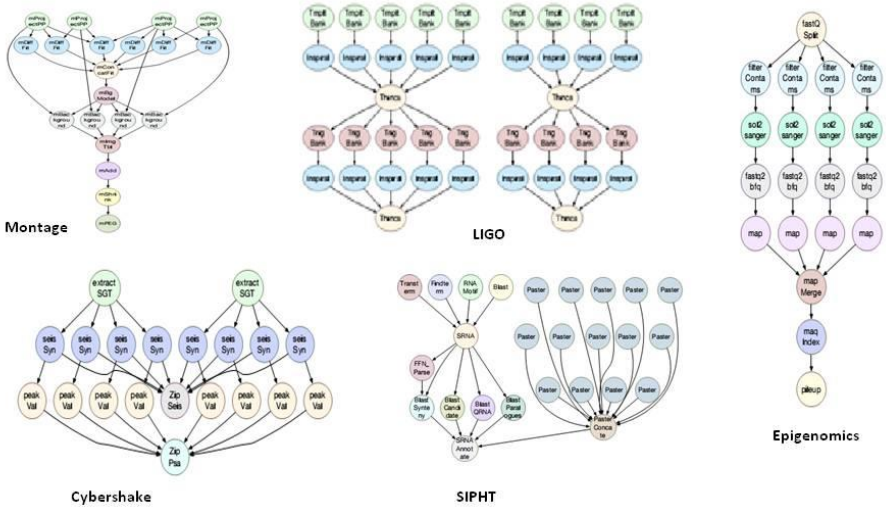


Fig.2 Scientific Workflow structures

B. Problem Definition: Workflow Scheduling in IaaS Clouds.

In general, the process of scheduling a workflow consists of assigning tasks to resources and coordinating their execution so that the dependencies between them are preserved. The mapping is also done so that different user-defined QoS requirements are met. These QoS parameters determine the scheduling objectives and are generally defined in terms of performance metrics such as execution time, execution cost, and also can be non-functional requirements such as security and energy consumption.

Workflow scheduling problem is NP-complete. To plan the execution of a workflow in a cloud environment, two issues need to be considered. The first one is known as resource provisioning where the issue is what resources need to be used from among the available resources. Let the different Vitrual Machines (VMs) provided by an IaaS vendor be represented by the set VM = { vm1, vm2, . . . , vmn }. Let R = { r1, r2, . . . , rk } be the set of resources used for the workflow execution where r_i= (vm_j, lease_start_time, VM_shutdown_time). VM is the set of available resources and R is the set of resources selected for use. The second issue is scheduling the timing of execution of tasks on the selected resources. It is the mapping of the tasks of workflow to the resources selected. Then the problem consists on finding the resources R needed to be leased and map each task $t_i \in T$ to a resource $r_i \in R$ and also satisfy scheduling objectives[12].

The remaining part of the paper is organized as follows. In Section II the related work is presented. Section III introduces the concept of Particle Swarm Optimization (PSO). Section IV presents the problem definition, Resource allocation Model and the task scheduling .Section V gives the experimental set up and section VI presents the results of the experiments. Finally, Section VII presents the conclusions.

II. RELATED WORK

Wu et al. [9] used PSO to generate a near-optimal schedule. They worked on minimizing either cost or time along with constraints like time deadlines and budget limits. This method uses heterogeneous resources and it assumes an initial set of VMs is available beforehand and hence this method lacks in utilizing the elasticity of IaaS clouds.

Byun et al. [10] developed an algorithm that estimated the optimal number of IaaS Cloud resources that need to be leased in order to minimize the execution cost of a workflow. Their algorithm is designed to produce a task to resource mapping and is capable to run online. The schedule and resources are updated for every billing period (i.e. every hour) depending on the current status of the running VMs and workflow tasks. Their approach takes total advantage of the elasticity and scalability of the cloud resources but fails to consider the heterogeneous nature of the Cloud computing resources by assuming that there is only one type of VM available.

Abrishami et al.[11] worked on partial critical paths to formulate their algorithm to schedule workflows on an IaaS Cloud. Their algorithm considered cloud features such as VM heterogeneity, pay-as-you-go model and time interval pricing. They try to optimize the execution cost using the heuristic of scheduling tasks in a partial critical path, executing on a single machine that can finish all the tasks before their latest finish time (which is calculated based on the application's deadline and the fastest available instance). However, they have not used any global optimization technique which is capable of producing a near-optimal solution; instead, they use a task-level optimization and hence have failed to utilize the complete workflow structure and characteristics for generation of a better solution.

Rodriguez and Buyya [12] used the meta-heuristic Particle Swarm optimization (PSO) for allocation of VMs and scheduling the workflow tasks to the VMs. Their algorithm considers a heterogeneous virtual pool of VMs and using PSO the actual VMs that need to be leased is found. Their method works with the aim of cost minimization while maintaining the makespan of the workflow to be within the deadline. Their algorithm generates schedules which specify the task to resource mapping along with the set of VMs to be leased and the start time and end time for the lease of the VM. The algorithm also considers boot time of the virtual machine while computing the makespan of the workflow. However, this method still has drawbacks because this model uses many types of resources which are numbered randomly. The entire set of VMs is considered by the algorithm and which makes the selection of the initial virtual machine pool important.

Hai-Hao Li & Zhi-Hui Zhan [13] proposed that the resources need to numbered in an order based on price or execution speed performance to make the learning among particles become more clear and reasonable. The results show that the resource renumber strategy is highly beneficial for PSO specially when using large scale cloud computing environments with many tasks.

III. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO)[14] is a latest evolutionary optimization technique build on the social conduct of living creatures. It was formulated by Eberhart and Kennedy and has been extensively used in research in various fields[15]. The particle is the central theme of this algorithm which is a stochastic technique The various inputs of a problem are represented using a particle and they have the ability to move through the specific limits and converge to the solution of the optimization problem.

The velocity of the particle defines the movement of particles and is a vector which has a magnitude and a direction. This velocity is calculated for each article using on the particle's best position and the global best position, which is the best position among all particles. A fitness function is used to measure the goodness of a particle's position is and the fitness function differs from one problem to another.

Every particle is characterized by its position (x) and velocity (v). The particle is initialized with random values within a specific range and the velocity of all particles is initialized to zero. The fitness function is evaluated for all particles and particle best (pbest) and global best (gbest) are determined and then the velocity and position of the particle is updated using equations 1 and 2.

$$\overrightarrow{x_i}(t+1) = \overrightarrow{x_i}(t) + \overrightarrow{v_i}(t) \quad \cdots \quad (1)$$

$$\overrightarrow{v_i}(t+1) = \omega.\overrightarrow{v_i}(t) + c_1 r_1 \left(\overrightarrow{x_i^*}(t) - \overrightarrow{x_i}(t)\right) + c_2 r_2 \left(\overrightarrow{x^*}(t) - \overrightarrow{x_i}(t)\right) \quad \cdots \quad (2)$$

ω = inertia
c_i = acceleration coefficient , i = 1, 2
r_i = random number, i = 1, 2 and $r_i \in [0, 1]$
x_i^* = best position of particle i
x^* = position of the best particle in the population
x_i = current position of particle i

This is repeated for a pre-set fixed number of iterations or till an accuracy condition is met. After each iteration the particles move towards the particle best and global best particle. The distance of particle movement depends on a random term generated with different random numbers used for acceleration towards particle best and global best locations [13]. Algorithm1 presents PSO pseudocode.

ALGORITHM 1: PSO

1. Initialize the population of particles with random positions and velocities.
2. Evaluate the fitness value of particles positions
3. Compare the particle's fitness value with the particle best pbest and store the value and position of the better one in pbest.
4. Compare all the pbest with the global best gbest and store the best value and position in gbest.
5. Update the velocity and position of each particle according to Eqs. (1) and (2).
6. Repeat step 2 to 5 until the stopping criterion is met.
7. The Result is the final gbest and the best mapping is given by the particle position

IV. SCHEDULING WORKFLOWS WITH PSO

A. Application and Resource Models

Each workflow W has a deadline δ_W associated to it which defines the time limit for the execution of the workflow. The IaaS cloud provider offers a range of VM types. VM_i is defined in terms of its processing capacity PVM_i and cost per unit of time CVM_i . It is assumed that the processing capacity, Floating Point Operations per Second (FLOPS) for all VMs, is known prior. The proposed algorithm uses this information to calculate the execution time of a task on a given VM.

The task execution time $ET_{t_i}^{VM_j}$ of task t_i on a Virtual Machine VM_j is calculated using the size I_{t_i} of the task and the Floating Point Operations (FLOPs) of the VM using Equation 1. Data Transfer time $TT_{e_{ij}}$ the time taken to transfer data from a parent task t_i and its child t_j and is estimated using Equation 2. To calculate $TT_{e_{ij}}$, it is assumed that all the virtual machines are available in the same data center and the size of the output data of the task t_i to be transferred, $d_{t_i}^{out}$ is known in advance. As all VMs are located in the same data center the bandwidth β between the VMs is approximately equal. When a parent task and the chid task run on the same VM the transfer tie will be considered as zero. A VM needs to remain active until all the output data of the running task is transferred to the VMs running the child tasks. Therefore the total processing time of a task $PT_{t_i}^{VM_j}$ in a VM instance VM_j is computed as shown in Equation 3, where k is the number of edges in which t_i is a parent task and s_k is zero whenever t_i and t_j run on the same VM or one otherwise.

$$ET_{t_i}^{VM_j} = I_{t_i} / P_{VM_j} \quad \cdots \quad (1)$$

$$TT_{e_{ij}} = d_{t_i}^{out} / \beta \quad \cdots \quad (2)$$

$$PT_{t_i}^{VM_j} = ET_{t_i}^{VM_j} + \left(\sum_{1}^{k} (TT_{e_{ij}} \times s_k)\right) \quad \cdots (3)$$

When a VM is leased, it requires an initial boot time and this time need to be considered as it impacts the makespan of the workflow. Resource provisioning and scheduling algorithms work with different objectives and this work focuses on finding a schedule to execute a workflow on IaaS computing resources such that the total execution cost is minimized and the deadline is met.

A schedule S = (R, M, TEC, TET) [12] is defined in terms of a set of resources, a task to resource mapping, the total execution cost and the total execution time. R = r1, r2, . . . , rn is the set of VMs that need to be leased; each resource ri has an estimated lease start time LST_{r_i} and lease end time LET_{r_i}. M represents a mapping and is comprised of tuples of the form $m_{t_i}^{r_j} = (t_i, r_j, ST_{t_i}, ET_{t_i})$, one

for each workflow task. A mapping tuple $m_{t_i}^{r_j}$ is interpreted as follows: task ti is scheduled to run on resource rj and is expected to start executing a time ST_{t_i} and complete by time ET_{t_i}. Equations 4 and 5 show how the total execution cost TEC and total execution time TET are calculated.

$$TEC = \sum_{i=1}^{|R|} C_{VM_{r_i}} \times \lceil (LET_{r_i} - LST_{r_i}) / \tau \rceil \quad \cdots (4)$$

$$TET = \max \{ ET_{t_i} : t_i \in T \} \quad \cdots \quad (5)$$

Based on the previous definitions, the problem can be formally defined as follows: find a schedule S with minimum TEC and for which the value of TET does not exceed the workflow's deadline.

Minimize TEC subject to $TET \leq \delta_W$ .... (6)

The time deadline is calculated by executing all the tasks of the workflow on the highest MFLOPs machine and then multiplied by the deadline factor to determine the deadline.

B. PSO Modeling for workflows.

A particle represents a workflow and its tasks; thus, the dimension of the particle is equal to the number of tasks in the workflow.

Co1	Co2	Co3	Co4	Co5	Co6	Co7	Co8	Co9
2.3	1.1	3.2	1.3	2.2	1.1	3.2	2.2	2.0

t_1,2	t_2,1	t_3, 3	t_4,1	t_5,2	t_6,1	t_7,3	t_8,2	t_9,2

Fig. 3. Particle's Position & Task to resource mapping

The dimension of a particle will determine the coordinate system used to define its position in space. For instance, the position of a 2-dimensional particle is specified by 2 coordinates (Co), the position of a 3-dimensional one is specified by 3 coordinates and so on. As an example, the particle depicted in Figure 1 represents a workflow with 9 tasks; the particle is a 9-dimensional one and its position is defined by 9 coordinates, coordinates 1 through 9. The range in which the particle is allowed to move is determined in this case by the number of resources chosen to run the tasks. As a result, the value of a coordinate can range from 0 to the number of VMs in the initial resource pool. Based on this, the integer part of the value of each coordinate in a particle's position corresponds to a resource index and represents the compute resource assigned to the task defined by that particular coordinate. In this way, the particle's position encodes a mapping of task to resources.

$$exeTime = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 \end{array} \\ \begin{pmatrix} 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 20 & 18 & 16 & 14 & 12 & 10 & 8 & 6 & 4 \\ 10 & 9 & 8 & 7 & 6 & 5 & 4 & 3 & 2 \\ 30 & 27 & 24 & 21 & 18 & 15 & 12 & 9 & 6 \\ 10 & 27 & 24 & 21 & 18 & 15 & 12 & 9 & 6 \\ 30 & 27 & 24 & 21 & 18 & 15 & 12 & 9 & 6 \\ 20 & 18 & 16 & 14 & 12 & 10 & 8 & 6 & 4 \\ 10 & 27 & 24 & 21 & 18 & 15 & 12 & 9 & 6 \\ 20 & 18 & 16 & 14 & 12 & 10 & 8 & 6 & 4 \end{pmatrix} \end{array}$$

$$transferTime = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 \end{array} \\ \begin{bmatrix} 0 & 5 & 5 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 4. Execution Time and Transfer Time for sample workflow.

C. Proposed Approach for selecting variable set of resources.

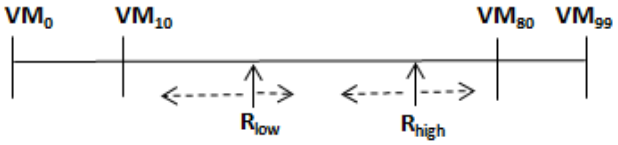


Fig. 5. Selecting $R_{initial}$ from R_{pool}.

A resource sub-pool $R_{initial}$ is selected from the larger pool by selecting the starting index R_{low} and the ending index R_{high} in the pool. If the indices of the larger pool range between 0 and 99 then the index range of $R_{initial}$ can be between 40 to 50. If the makespan need to be increased, then move the indices to the left and if the makespan needs to be decreased then move the indices to the right. Thus varying the indices we can reach the deadline time and get the appropriate cost minimization. The indices of the $R_{initial}$ can be varied for each experiment till the desired deadline is met, i.e the makespan becomes closer to the deadline. For example in Fig. 6 we find 9 resources in the larger pool but the resources considered by PSO during experimentation can vary between r_3 and r_5 or r_6 to r_8 and so on. This selection is possible as the resources are numbered in an order based on the cost and speed. Figure 6 shows the calculated execution times of all tasks on all resources and transfer times among all the tasks.

Since the fitness function is used to determine how good a potential solution is, it needs to reflect the objectives of the scheduling problem. Based on this, the fitness function will be minimized and its value will be the total execution cost TEC associated to the schedule S derived from the particle's position. The schedule is generated using algorithm 2.

D. Schedule Generation

Algorithm 2. Schedule Generation

Input: Set of workflow tasks T, Pre-Calculated ExeTime[$|T| * |R_{pool}|$], Pre-Calculated TransferTime[$|T| * |T|$]
Initial resource pool $R_{initial}$ (R_{low} to R_{high}) is a part of a larger pool of resources R_{pool}.
An array pos[|T|] represents a particle position

Procedure GENERATE SCHEDULE (T, $R_{initial}$, pos[$|T|$])
1. Initialize Schedule components
R = ∅, M = ∅, TEC = 0, TET = 0.
2. for $i = 0$ *to* $i = |T| - 1$
2.1. t_i=T[i], $r_{pos[i]}$= $R_{initial}$ [pos[i]]
2.2 if t_i has no parents then
$ST_{t_i} = LET_{r_{pos[i]}}$
else
ST_{t_i} = max(max{ET_{t_p}:t_p ϵ parents(t_i)}, $LET_{r_{pos[i]}}$)
end if
2.3 exe = exeTime[i][pos[i]]
2.4 for each child t_c of t_i do
2.5 if t_c is mapped to a resource different to $r_{post[i]}$ then
transfer+=TransferTime[i][c]
end if

end for

2.6 $PT_{t_i}^{r_{pos[i]}}$ = exe + transfer

2.7 $ET_{t_i} = PT_{t_i}^{r_{pos[i]}} + ST_{t_i}$

2.8 $m_{t_i}^{r_{pos[i]}} = (t_i, r_{post[i]}, ST_{t_i}, ET_{t_i})$

2.9 $M = M \cup m_{t_i}^{r_{pos[i]}}$

2.10: if $r_{pos[i]} \notin R$ then

$LST_{r_{pos[i]}} = \max(ST_{t_i}, bootTime)$

$R = R \cup \{ r_{post[i]} \}$

end if

2.11 $LET_{r_{pos[i]}} = PT_{t_i}^{r_{pos[i]}} + ST_{t_i}$

end for

3. calculate TEC according to equation 4.
4. calculate TET according to equation 5.
5. S = (R, M, TEC, TET)

end procedure

Algorithms 1 and 2 are combined to produce a near optimal schedule. In step 3 of Algorithm 1, instead of calculating the fitness value of the particle, we generate the schedule as outlined in Algorithm 2. Then we use TEC as a fitness value in steps 4 through 6 and introduce the constraint handling mechanism in step 4, ensuring that TET doesn't exceed the application's deadline.

V. EXPERIMENTAL SETUP

A Java Application with Netbeans IDE was developed using CloudSim framework for implementation of the algorithm. we assume a Cloud environment which consists of a service provider, offering 100 different computation services (similar to Amazon EC2 services)[16], with different processor speeds and different prices. The resources are ordered according to their cost and speed.The first among all the resources is the slowest resource with a cheapest cost and the last of the resources is the fastest and the costliest. The processor speeds are selected randomly such that the fastest service is roughly hundred times faster than the slowest one, and accordingly, it is approximately seventy times more expensive. The average bandwidth between the computation services is set to 20MBPS which is the approximate average bandwidth between the computation services (EC2) in Amazon.

For PSO implementation the number of particles considered is 100 and the number of iterations is set to 40.The values of c1 is set to 1.5 and c2 is set to 2.0 and ω is set to 0.5.

Another important parameter of the experiments is the billing time interval. Most of the current commercial Clouds, such as Amazon, charge users based on a long time interval equal to one hour. As a result, these Clouds do not exactly meet the goal of the pay-as-you-go pricing model, because the user has to pay for the whole last time interval, even if he has used a fraction of it. Today, a few service providers support short time intervals, for example CloudSigma [17] has a burst pricing model that its time interval is five minutes and every minute thereafter is considered as one unit. The cost of using a VM for 5 minutes is assumed as 1/5th of the hourly cost and the cost for every minute is 1/20th of the hourly cost. To evaluate the impact of short and long time intervals on our algorithms, we consider two different time intervals in the experiments: a long one equal to one hour, and a short one equal to five minutes.

Normalized Cost (NC) is considered as it can be used to evaluate the cost of execution of a workflow on any set of resources. The normalized cost when multiplied with the cost of execution of the workflow on the cheapest Virtual machine in the set of resources results in the original cost. Normalized cost of a workflow execution can be defined as follows:

NC = total schedule cost / C_C ----- (9)

where C_C is the cost of executing the same workflow with the Cheapest strategy [11].

VI. RESULTS

In this section we present results of the experiments conducted in order to evaluate the performance of the proposed approach.

A. Cost Evaluation:

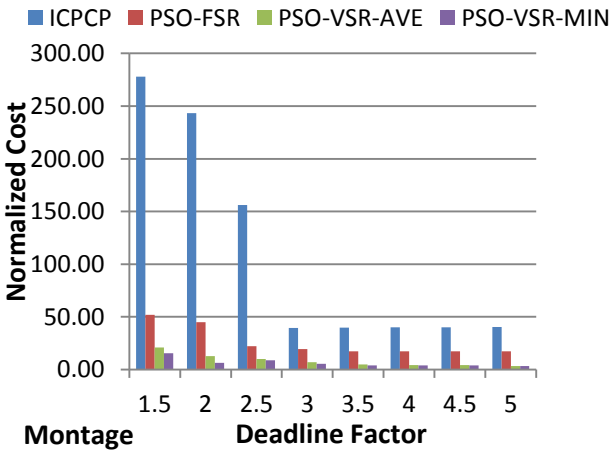
Fig. 6 shows the comparision of the Normalized costs obtained at different deadline factors varying from 1.5 to 5 with an increment of 0.5 for hourly billing for the methods ICPCP, PSO with fixed set of resources, PSO with variable set of resources average and PSO with variable set of resources minimum values.

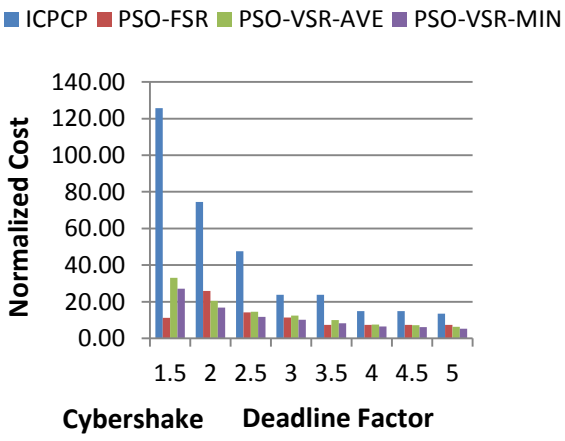
In Cybershake, the performance of PSO-VSR is better for all deadline factors when compared to ICPCP in hourly billing and minute billing. PSO-VSR when compared with PSO-FSR, is better at higher deadline factors. In minute billing PSO-FSR is better at lower deadline factors and PSO_VSR is better at higher deadlines.

In montage hourly billing the normalized cost of PSO-VSR is lesser compared to PSO_FSR and far lesser when compared to ICPCP. The normalized cost PSO-VSR is reduced by almost 10 times when compared to ICPCP in hourly billing and reduced by 5 times in minute billing.

In SIPHT, in hourly billing, the normalized cost of PSO-VSR-MIN is always lesser than ICPCP and PSO-VSR-AVE is similar to ICPCP. PSO-FSR resulted in higher normalized costs when compared to ICPCP and PSO-VSR. In minute billing the normalized cost of PSO-VSR can be observed that it is always lesser than ICPCP and PSO-FSR.

In LIGO, in hourly billing and minute billing , the normalized cost of PSO-VSR is lesser than ICPCP and PSO-FSR. It can be observed that the normalized cost of PSO-FSR is comparatively high for higher deadline factors when compared to ICPCP and PSO-VSR in hourly billing.





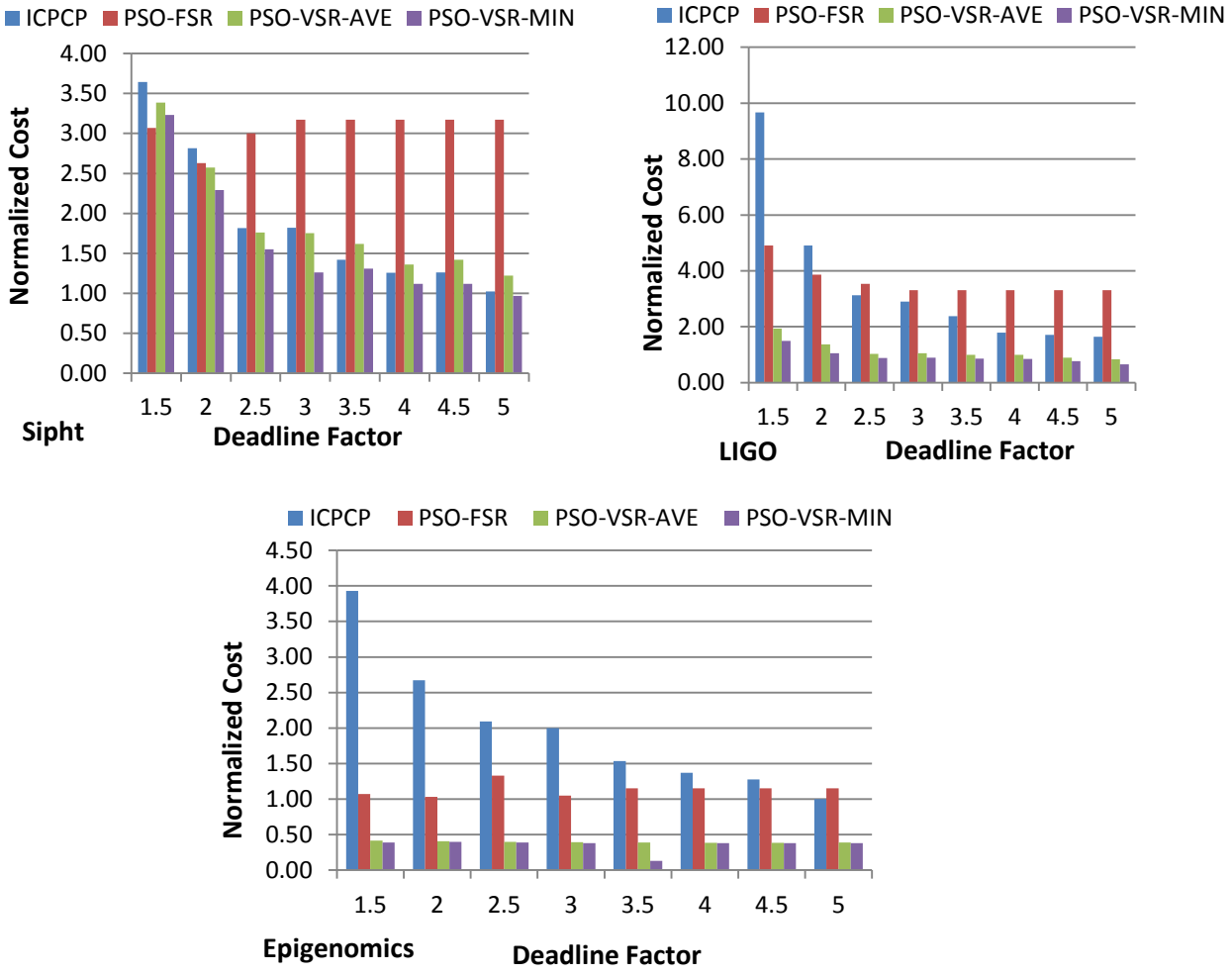
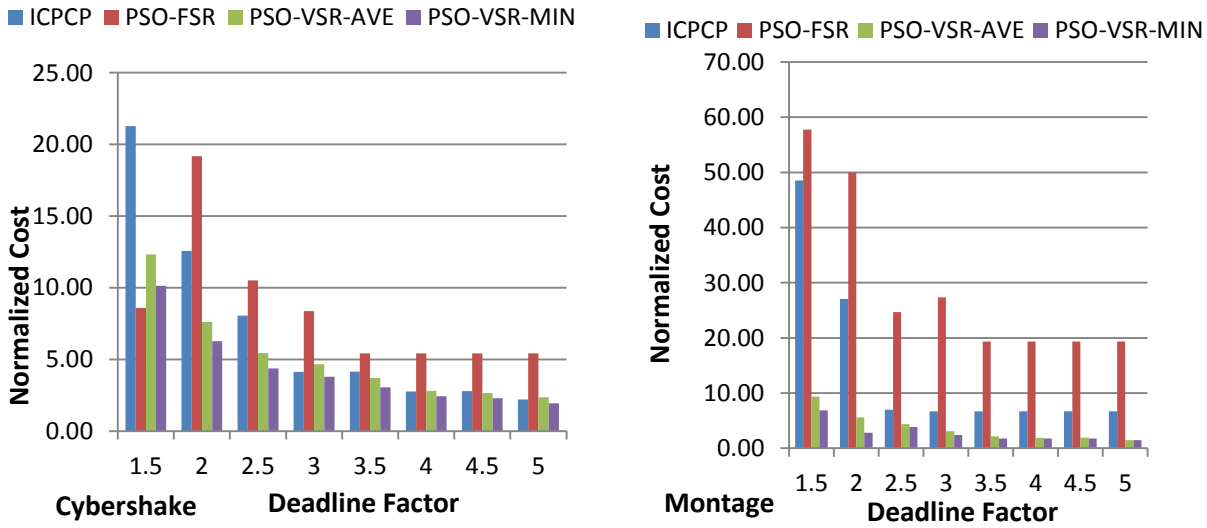


Fig. 6. Normalized cost for hourly billing with Deadlines for ICPCP, PSO with fixed set of resources, PSO with variable set of resources (average and minimum)



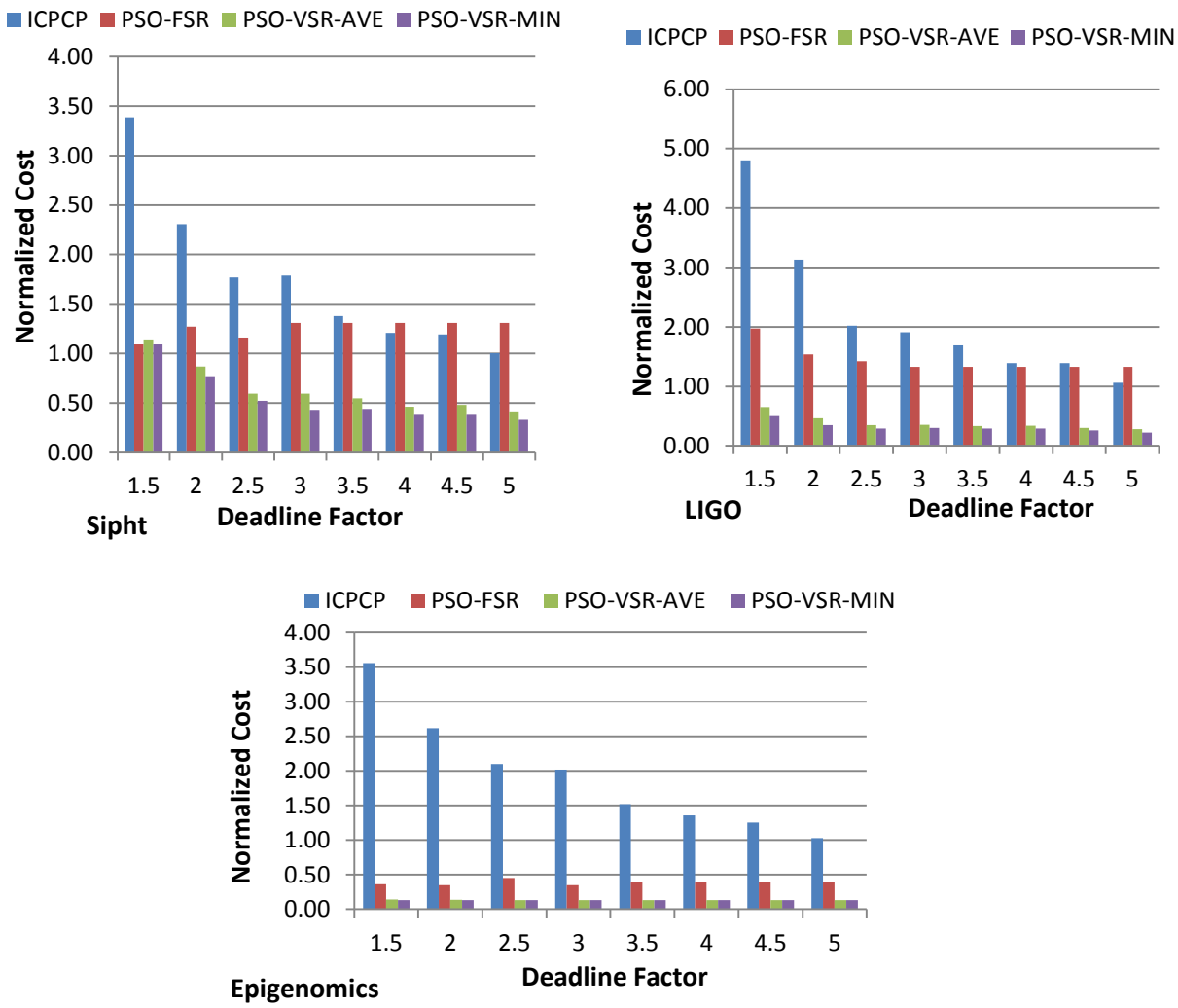


Fig. 7 Normalized cost for minute billing with Deadlines for ICPCP, PSO with fixed set of resources, PSO with variable set of resources (average and minimum)

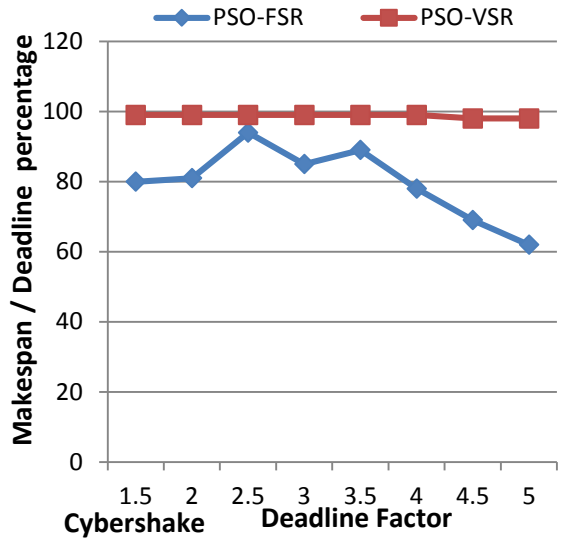
In Epigenomics, the normalized cost of PSO-VSR is lesser than the normalized cost observed in both ICPCP and PSO-FSR . It can be observed that the normalized cost of ICPCP is almost the same in hourly billing and minute billing where as a considerable reduction in normalized cost is observed in PSO-FSR and PSO-VSR.

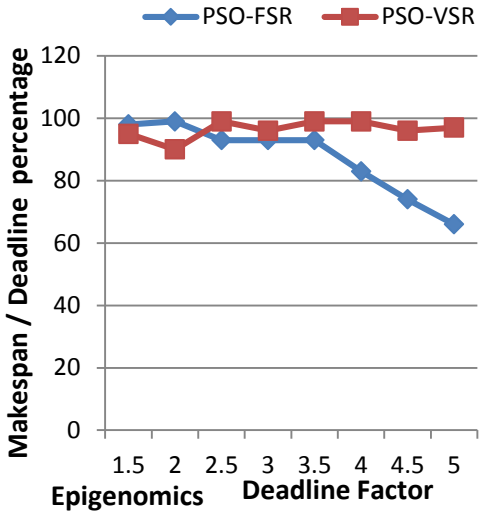
Fig. 7 shows the comparision of the Normalized costs obtained at different deadline factors varying from 1.5 to 5 with an increment of 0.5 for minute billing for the methods ICPCP, PSO with fixed set of resources, PSO with variable set of resources average and PSO with variable set of resources minimum values. It can be observed from the results that the hourly normalized costs are of higher value when compared to the minute normalized costs.

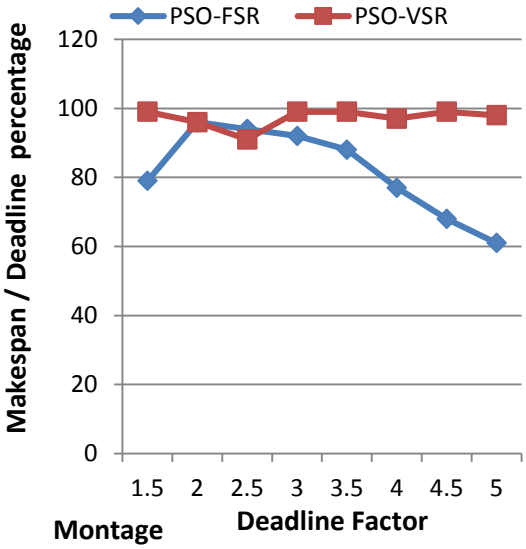
B. Deadline Constraint Evaluation.

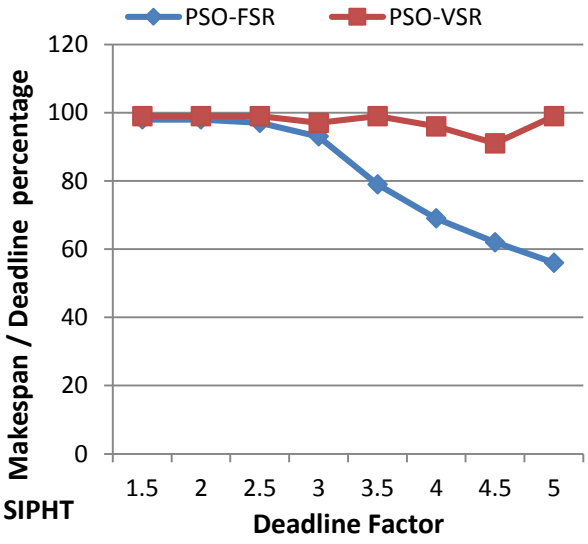
Using PSO with fixed set of resources fail to reach the deadline time and hence limits the minimization of cost. whereas PSO with variable set of resources achieved a lot of flexibility in the makespan of the workflow. The makespan achieved in PSO-FSR does not extend to extensive ranges as the set of resources are fixed and hence the makespan does not always match the deadline but is far below the deadline. In Cybershake the makespan is nearer to the deadline for the deadline factors 1.5, 2, and 2.5 whereas in Epigenomics, LIGO, SIPHT and Montage applications, the makespan could reach the deadline only for the deadline factor 1.5. As the deadline factor increases the

makespan could not match the deadline. It can be observed that the makespan achieved in PSO-VSR is above 90% for all deadline factors and in all the applications considered. This could be achieved because of the variable initial resource pool considered by PSO algorithm for allocation of resources.









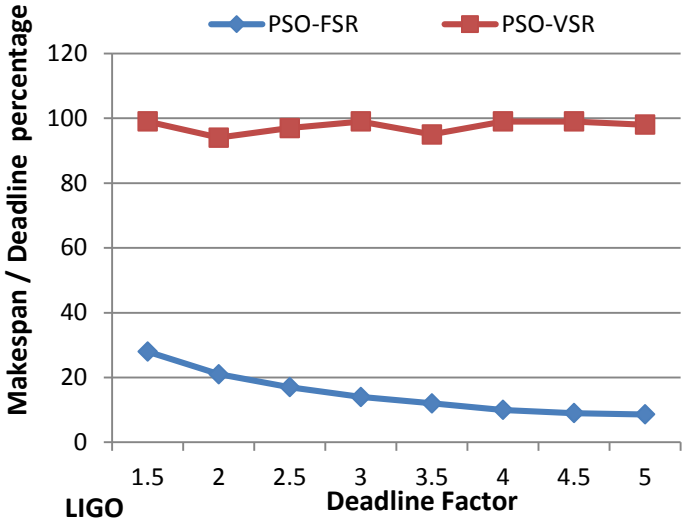


Fig. 8. Percentage of deadline met

C. Comparision of Hourly Billing and Minute billing:

In hourly billing every 60 minutes is considered as 1 unit and if the Virtual machine is used for 61minutes, it will be billed for 2 units. In the minute billing system the first 5 minutes is considered as 1 unit and then for every minute thereafter it is billed as a unit with a minute price. The comparision shown in the Fig. indicates that using PSO-VSR the hourly billing to minute billing comparision varied between 2.6 times to 3.07 times for the applications considered in the experiment. It can be observed that hourly billing is on an average 3 time the cost of minute billing. Using ICPCP the hourly billing to minute billing comparision varied between 1.03 times to 7 times and in PSO-FSR the hourly billing to minute billing ratio varies between 0.96 and 2.97. From the results we observe that the hourly billing to minute billing ratio is stable and best in PSO-VSR algorithm.

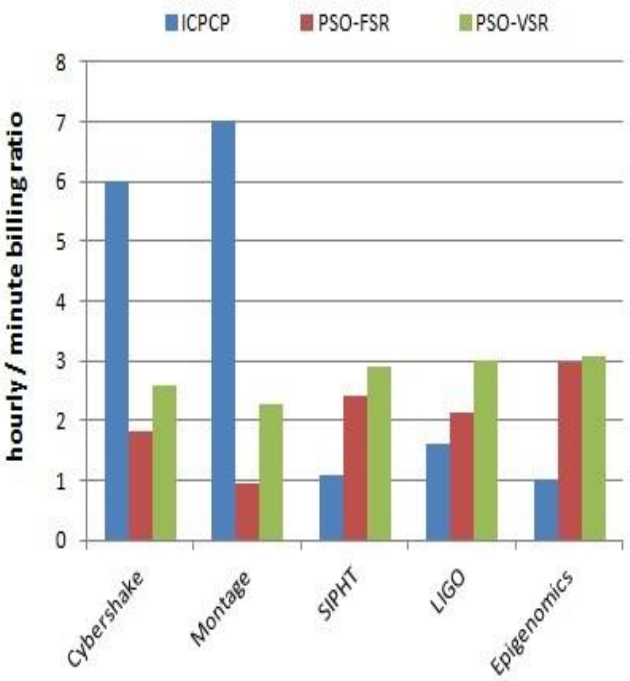


Fig.9. Comparision of Hourly Billing and Minute billing:

VII. CONCLUSION

Scheduling and optimization play an important role in resource utilization, faster execution and minimizing the cost incurred in executing the scientific applications. In this paper we proposed a resource provisioning and scheduling strategy for scientific workflow execution on Infrastructure as a Service (IaaS) clouds. The algorithm presented is based on the meta-heuristic optimization technique, particle swarm optimization (PSO). The results show that the cost of executing scientific workflows is minimized when the resources are selected using Particle swarm optimization with variable set of resources when compared to the state of the art work. This was possible because of the flexibility and availability of Cloud resources. The algorithm proposed gave better makespan in terms of meeting the deadlines and minimizing the cost. Comparision of the hourly billing and minute billing showed that there can be a cost saving upto 1/3rd of the hourly cost, if minute billing is used.

REFERENCES

I. P. Mell, T. Grance, "The NIST definition of cloud computing recommendations of the National Institute of Standards and Technology" Special Publication 800-145, NIST, Gaithersburg, 2011.
II. R. Buyya, J. Broberg, and A. M. Goscinski, Cloud Computing: Principles and Paradigms. Wiley. com, 2010, vol. 87.
III. G. Berriman, A. Laity, J. Good, J. Jacob, D. Katz, E. Deelman, G. Singh, M. Su, and T. Prince, "Montage: The architecture and scientific applications of a national virtual observatory service for computing astronomical image mosaics," in Proceedings of Earth Sciences Technology Conference, 2006.
IV. R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner et al., "Cybershake: A physics-based seismic hazard model for southern california," Pure and Applied Geophysics, vol. 168, no.3-4, pp. 367–381, 2011.
V. A. Abramovici, W. E. Althouse, R. W. Drever, Y. G ¨ ursel, S. Kawamura, F. J. Raab, D. Shoemaker, L. Sievers, R. E. Spero, K. S. Thorne et al., "Ligo: The laser inter-ferometer gravitational-wave observatory," Science, vol. 256, no. 5055, pp. 325–333, 1992.
VI. J. Livny, H. Teonadi, M. Livny, and M. K. Waldor, "High-throughput, kingdom-wide prediction and annotation of bacterial non-coding rnas," PloS one, vol. 3, no. 9,p. e3197, 2008.
VII. "USC Epigenome Center," http://epigenome.usc.edu, accessed: October 2015.
VIII. G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K.Vahi, "Characterizing and profiling scientific workflows," Future Generation Comput. Syst., vol. 29, no. 3, pp. 682–692, 2012.
IX. Z. Wu, Z. Ni, L. Gu, andX. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," in Proceedings of the International Conference on Computational Intelligence and Security (CIS). IEEE, 2010, pp. 184–188.
X. E.-K. Byun, Y.-S. Kee, J.-S. Kim, and S. Maeng, "Cost optimized provisioning of elastic resources for application workflows," Future Generation Computer Systems, vol. 27, no. 8, pp. 1011–1026, 2011
XI. S. Abrishami, M. Naghibzadeh, and D. Epema, "Deadline-constrained workflow scheduling algorithms for IaaS clouds," Future Generation Comput. Syst., vol. 23, no. 8, pp. 1400–1414, 2012
XII. M. A. Rodriguez and R. Buyya, "Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds," IEEE Transactions on Cloud Computing, vol. 2, no. 2, April-June 2014.
XIII. H. H. Li, Y. W. Fu, Z. H. Zhan, J. J. Li, "Renumber strategy enhanced particle swarm optimization for cloud computing resource scheduling", IEEE Congress on Evolutionary Computation (CEC) Sendai, pp. 870-876, 2015.
XIV. A. Lazinica, Particle swarm optimization. InTech Kirchengasse, 2009.
XV. J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. IEEE Int. Conf. Neural Networks, 1995, pp. 1942-1948.
XVI. "Amazon EC2 Spot Instances," https://aws.amazon.com/ec2/purchasing-options/spot-instances/, accessed: October 2015.
XVII. "Cloudsigma," https://www.cloudsigma.com, accessed: October2015
XVIII. A. Barker and J. Van Hemert, "Scientific workflow: a survey and research directions," in Parallel Processing and Applied Mathematics. Springer, 2008, pp. 746–753.