IJCSEC- International Journal of Computer Science and Engineering Communications.
Vol.3, Issue 2, 2015, Page.793-, ISSN: 2347–8586

Deadline Based Execution of Scientific workflows on IaaS Clouds using Resource Provisioning and Scheduling Strategy

D.SILPA[1], A.SUDHA[2], P.M.GAJALAKSHMI [3], A.KALAIVANI [4]
[#]UG Scholar, Computer Science and Engineering,
R.M.K College of Engineering and Technology, Chennai, India.
d.silpa27@gmail.com[1], sudhaaju@gmail.com[2],
gajalakshmi040893@gmail.com[3],kalaiwind@gmail.com[4]

Abstract—Cloud computing is the latest distributed computing paradigm and it offers tremendous opportunities to solve large-scale scientific problems. However, it presents various challenges that need to be addressed in order to be efficiently utilized for workflow applications. Although the workflow scheduling problem has been widely studied, there are very few initiatives tailored for cloud environments. Furthermore, the existing works fail to either meet the user's quality of service (QoS) requirements or to incorporate some basic principles of cloud computing such as the elasticity and heterogeneity of the computing resources. This paper proposes a resource provisioning and scheduling strategy for scientific workflows on Infrastructure as a Service (IaaS) clouds. We present an algorithm based on the meta-heuristic optimization technique, particle swarm optimization (PSO), which aims to minimize the overall workflow execution cost while meeting deadline constraints. Our heuristic is evaluated using CloudSim and various well-known scientific workflows of different sizes. The results show that our approach performs better than the current state-of-the-art algorithms.

Keywords—Cloud computing, resource provisioning, scheduling, scientific workflow

I. INTRODUCTION

WORKFLOWS have been frequently used to model large scale scientific problems in areas such as bioinformatics, astronomy, and physics [1]. Such scientific workflows have ever-growing data and computing requirements and therefore demand a high-performance computing environment in order to be executed in a reasonable amount of time. These workflows are commonly modeled as a set of tasks interconnected via data or computing dependencies.

Over the years, distributed environments have evolved from shared community platforms to utility-based models; the latest of these being cloud computing. This technology enables the delivery of IT resources over the Internet [2], and follows a pay-as-you-go model where users are charged based on their consumption. There are various types of cloud providers [2], each of which has different product offerings. They are classified into a hierarchy of as-a-service terms: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). This paper focuses on IaaS clouds which offer the user a virtual pool of unlimited, heterogeneous resources that can be accessed on demand.

Moreover, they offer the flexibility of elastically acquiring or releasing resources with varying configurations to best suit the requirements of an application. Even though this empowers the users and gives them more control over the resources, it also dictates the development of innovative scheduling techniques so that the distributed resources are efficiently utilized. There are two main stages when planning the execution of a workflow in a cloud environment. The first one is the resource provisioning phase; during this stage, the computing resources that will be used to run the tasks are selected and provisioned. In the second stage, a schedule is generated and each task is mapped onto the best-suited resource. The selection of the resources and mapping of the tasks is done so that different user defined quality of service (QoS) requirements are met.

Virtual machine (VM) performance is an additional challenge presented by cloud platforms. VMs provided by current cloud infrastructures do not exhibit a stable performance in terms of execution times. This work is based on the meta-heuristic optimization technique, particle swarm optimization (PSO). PSO is then used to solve such problem and produce a schedule defining not only the task to resource mapping but also the number and type of VMs that need to be leased, the time when they need to be leased and the time when they need to be released. Our contribution is therefore, an algorithm with higher accuracy in terms of meeting deadlines at lower costs that considers heterogeneous resources that can be dynamically acquired and released and are charged on a pay-per-use basis.

II. PROBLEM FORMULATION

Application and Resource Models

A workflow application W={T,E} is modeled as a directed acyclic graph (DAG) where T={t1; t2; . . . ; tn} is the set of tasks and E is the set of directed edges. An edge eij of the form ðti; tjÞ exists if there is a data dependency between ti and tj, case in which ti is said to be the parent task of tj and tj is said to be the child task of tj. Based on this definition, a child task cannot be executed until all of its parent tasks are completed. A sample workflow is shown in Fig. 1. In addition, each workflow W has a deadline §W associated to it. A deadline is defined as a time limit for the execution of the workflow.

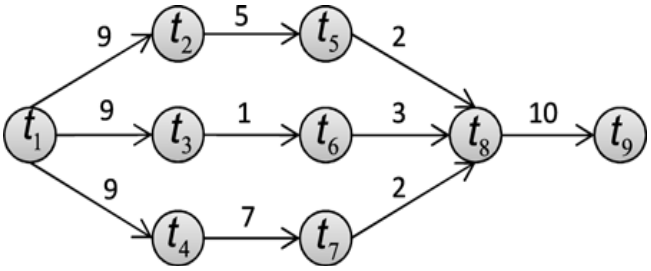

Fig. 1. Sample workflow. Each node represents a task and the arcs show the data transfer times between nodes.

The IaaS cloud provider offers a range of VM types. A VM type VMi is defined in terms of its processing capacity PVMi and cost per unit of time CVMi . In every VM type, the processing capacity in terms of floating point operations per second (FLOPS) is available either from the provider or can be estimated [17]. This information is used in our algorithm to calculate the execution time of a task on a given VM. Performance variation is modeled by adjusting the processing capacity of each leased VM and introducing a performance degradation percentage degVMi . The unit of time t in which the pay-per-use model is based is specified by the provider; any partial utilization of the leased VM is charged as if the full time period was consumed. For instance, for t ¼ 60 minutes, if a VM is used for 61 minutes, the user will pay for two periods of 60 minutes, that is, 120 minutes.

Also, we assume that there is no limitation on the number of VMs that can be leased from the provider. The execution time ETVMj ti of task ti in a VM of type VMj and is estimated using the size Iti of the task in terms of floating point operations (FLOP). This is depicted in Equation (1). Additionally, TTeij is defined as the time it takes to transfer data between a parent task ti and its child tj and is calculated as depicted in Equation (2). To calculate TTeij we assume that the size of the output data dout ti produced by task ti is known in advance and that the entire workflow runs on a single data center or region. This means that all the leased instances are located on the same region and that the bandwidth b between each VM is roughly the same. Notice that the transfer time between two tasks being executed on the same VM is 0. Finally, the total processing time PT VMj ti of a task in a VM is computed as shown in Equation (3), where k is the number of edges in which ti is a parent task and sk is 0 whenever ti and tj run on the same VMor 1.

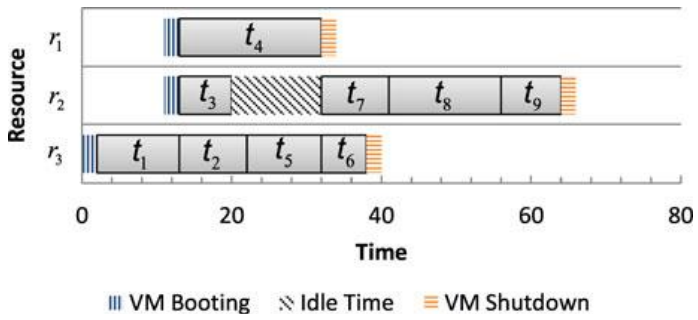

Fig. 2. Example of a schedule generated for the workflow shown in Fig. 1. Each task is mapped onto one of the three available resources.

Parent tasks are executed before their children ensuring the data dependencies are met.

Problem Definition

Resource provisioning and scheduling heuristics may have different objectives; 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. We define a schedule S ¼ ðR; M; TEC; TETÞ in terms of a set of resources, a task to resource mapping, the total execution cost and the total execution time. Fig. 2 shows a sample schedule generated for the workflow depicted in Fig. 1. R ¼ fr1; r2; . . . ; rng is the set of VMs that need to be leased;each resource ri has a VM type VMri associated to it as well as an estimated lease start time LSTri and lease end time LETri . M represents a mapping and is comprised of tuples of the form mr ti ¼ ðti; rj; STti;ETti Þ, one for each workflow task. A mapping tuple mrj ti is interpreted as follows: task ti is scheduled to run on resource rj and is expected to start executing a time STti and complete by time ETti . 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.

III. PARTICLE SWARM OPTIMIZATION

Particle swarm optimization is an evolutionary computational technique based on the behavior of animal flocks. It was developed by Eberhart and Kennedy [4] in 1995 and has been widely researched and utilized ever since. The algorithm is a stochastic optimization technique in which the most basic concept is that of particle. A particle represents an individual (i.e., fish or bird) that has the ability to move through the defined problem space and represents a candidate solution to the optimization problem.

At a given point in time, the movement of particles is defined by their velocity, which is represented as a vector and therefore has magnitude and direction. This velocity is determined by the best position in which the particle has been so far and the best position in which any of the particles has been so far. Based on this, it is imperative to be able to measure how good (or bad) a particle's position is; this is achieved by using a fitness function that measures the quality of the particle's position and varies from problem to problem, depending on the context and requirements.

Each particle is represented by its position and velocity. Particles keep track of their best position pbest and the global best position gbest; values that are determined based on the fitness function. The algorithm will then at each step, change the velocity of each particle towards the pbest and gbest locations. How much the particle moves towards these values is weighted by a random term, with different random numbers generated for acceleration towards pbest and gbest locations [18]. The algorithm will continue to iterate until a stopping criterion is met; this is generally a specified maximum number of iterations or a predefined fitness value considered to be good enough.

IV. PROPOSED APPROACH

PSO Modeling

There are two key steps when modeling a PSO problem. The first one is defining how the problem will be encoded, that is, defining how the solution will be represented. The second one is defining how the "goodness" of a particle will be measured, that is, defining the fitness function. To define the encoding of the problem, we need to establish the meaning and dimension of a particle. For the scheduling scenario presented here, a particle represents a workflow and its tasks; thus, the dimension of the particle is equal to the number of tasks in the workflow. The dimension of a particle will determine the coordinate system used to define its position in space. For instance, the position of a two-dimensional particle is specified by two coordinates, the position of a three-dimensional one is specified by three coordinates and so on.

The range in which the particle is allowed to move is determined in this case by the number of resources available 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.

Schedule Generation

Initially, the set of resources to lease R and the set of task to resource mappings M are empty and the total execution cost TECand time TET are set to zero. After this, the algorithm estimates the execution time of each workflow task on every resource ri 2 Rinitial. This is expressed as a matrix in which the rows represent the tasks, the columns represent the resources and the entry ExeTime½i; j_ represent the time it takes to run task ti on resource rj. This time is calculated using Equation (1). The next step is the calculation of the data transfer time matrix. Such matrix is represented as a weighted adjacency matrix of the workflow DAG where the entry TransferTime½i; j_ contains the time it takes to transfer the output data of task ti to task tj.

$$exeTime = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{array} \begin{array}{c} \begin{array}{ccc} r_1 & r_2 & r_3 \end{array} \\ \begin{bmatrix} 2 & 1 & 4 \\ 4 & 3 & 6 \\ 10 & 6 & 15 \\ 7 & 4 & 12 \\ 8 & 4 & 10 \\ 3 & 2 & 7 \\ 12 & 7 & 18 \\ 9 & 5 & 20 \\ 13 & 8 & 19 \end{bmatrix} \end{array}$$

(a)

$$transferTime = \begin{array}{c} \\ t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \\ t_9 \end{array} \begin{array}{c} \begin{array}{ccccccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 & t_7 & t_8 & t_9 \end{array} \\ \begin{bmatrix} 0 & 9 & 9 & 9 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

(b)

Fig. 3. Sample execution and data transfer time matrix. (a) Execution time matrix. (b) Data transfer time matrix.

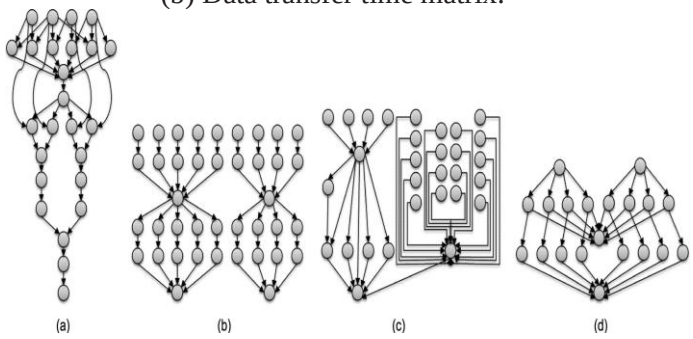

Fig. 4. Structure of the four different workflows used in the experiments. (a) Montage. (b) LIGO. (c) SIPHT. (d) Cyber Shake.

V.PERFORMANCE EVALUATION

Experiments conducted in order to evaluate the performance of the proposed approach. We used the CloudSim framework [19] to simulate a cloud environment and chose four different workflows from different scientific areas: Montage, LIGO, SIPHT, and CyberShake. Each of these workflows has different structures as seen in Fig. 5 and different data and computational characteristics. The Montage workflow is an astronomy application used to generate custom mosaics of the sky based on a set of input images. The LIGO workflow is used in the physics field with the aim of detecting gravitational waves. This workflow is characterized by having CPU intensive tasks that consume large memory. SIPHT is used in bioinformatics to automate the process of searching for sRNA encoding-genes for all bacterial replicons in the National Center for Biotechnology Information1 database. Most of the tasks in this workflow have a high CPU and low I/O utilization. Finally, CyberShake is used to characterize earthquake hazards by generating synthetic seismograms and may be classified as a data intensive workflow with large memory and CPU requirements.

When leasing a VM from an IaaS provider, the user has the ability to choose different machines with varying configurations and prices. The first variation of our algorithm, referred to as PSO, considers this heterogeneous environment. However, in order to evaluate if this resource heterogeneity has an impact on the makepsan and cost of a workflow execution, we propose a second variation of our algorithm called PSO_HOM. PSO_HOM considers a homogeneous environment in which only a single type of VM is used.

The experiments were conducted using four different deadlines. These deadlines were calculated so that their values lie between the slowest and the fastest runtimes. To calculate these runtimes, two additional policies were implemented. The first one calculates the schedule with the slowest execution time; a single VM of the cheapest type is leased and all the workflow tasks are executed on it. The second one calculates the schedule with the fastest execution time; one VM of the fastest type is leased for each workflow task. Although these policies ignore data transfer times, they are still a good approximation to what the slowest and fastest runtimes would be. To estimate each of the four deadlines, the difference between the fastest and the slowest times is divided by five to get an interval size. To calculate the first deadline interval we add one interval size to the fastest deadline, to calculate the second one we add two interval sizes and so on. In this way we analyze the behavior of the algorithms as the deadlines increase from stricter values to more relaxed ones.

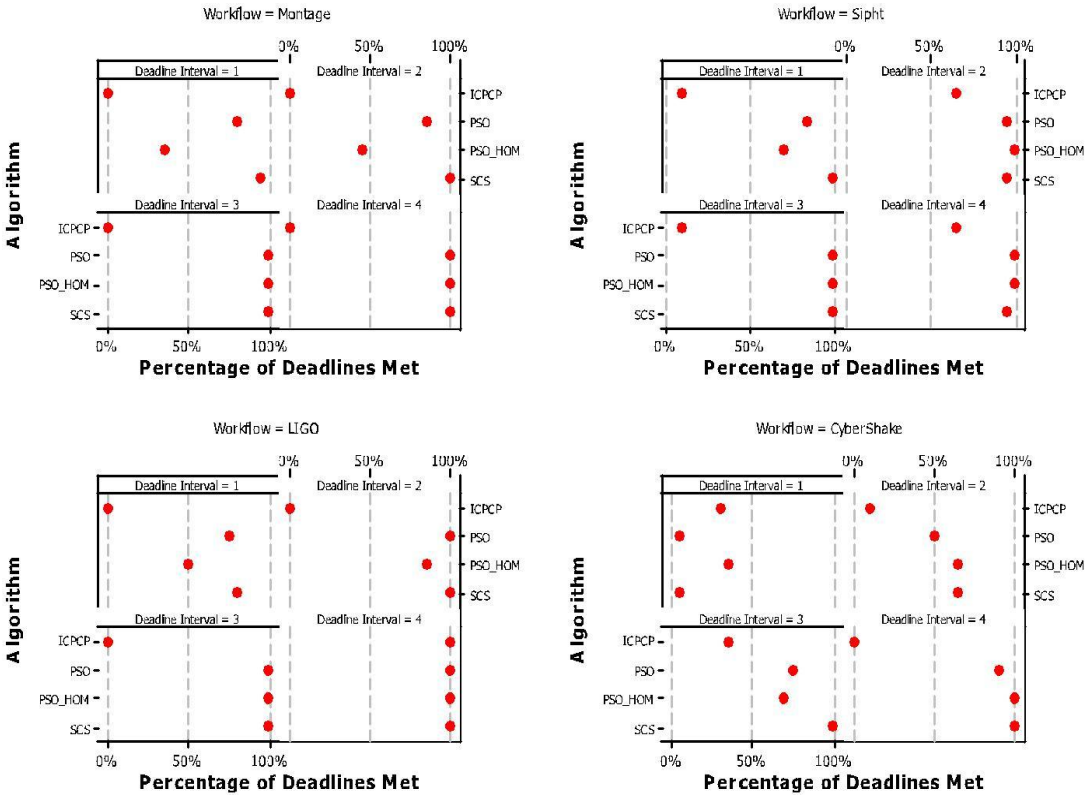

Fig. 5. Individual value plot of deadlines met for each workflow and deadline interval.

TABLE 1: Type of VMs Used in the Experiments

	EC2 Units	Processing Capacity (MFLOPS)	Cost per Hour
m1.small	1	4400	$0.06
m1.medium	2	8800	$0.12
m1.large	4	17600	$0.24
m1.xLarge	8	35200	$0.48
m3.xLarge	13	57200	$0.50
m3.doubleXLarge	26	114400	$1.00

Results and Analysis

Deadline Constraint Evaluation

To analyze the algorithms in terms of meeting the user defined deadline, we plotted the percentage of deadlines met for each workflow and deadline interval. The results are displayed in Fig. 6. For the Montage workflow, ICPCP fails to meet all of the deadlines. PSO_HOM meets fewer than 50 percent of the deadlines on interval 1 but improves its performance on interval 2 and achieves a 100 percent hit rate for both intervals 3 and 4. PSO and SCS are the best performing algorithms in terms of deadlines met with PSO meeting slightly less deadlines on intervals 1 and 2 but achieving 100 percent on the last two intervals. Even though our approach is offline, as is IC-PCP, we succeed in considering the dynamic nature of the cloud and the variability of CPU performance. Overall, PSO, PSO_HOM and SCS meet the most number of deadlines for all of the workflows, making them the most appealing algorithms for the scheduling problem stated in this paper. There is a considerable difference in the percentage of deadlines met by IC-PCP and these three algorithms.

Makespan Evaluation

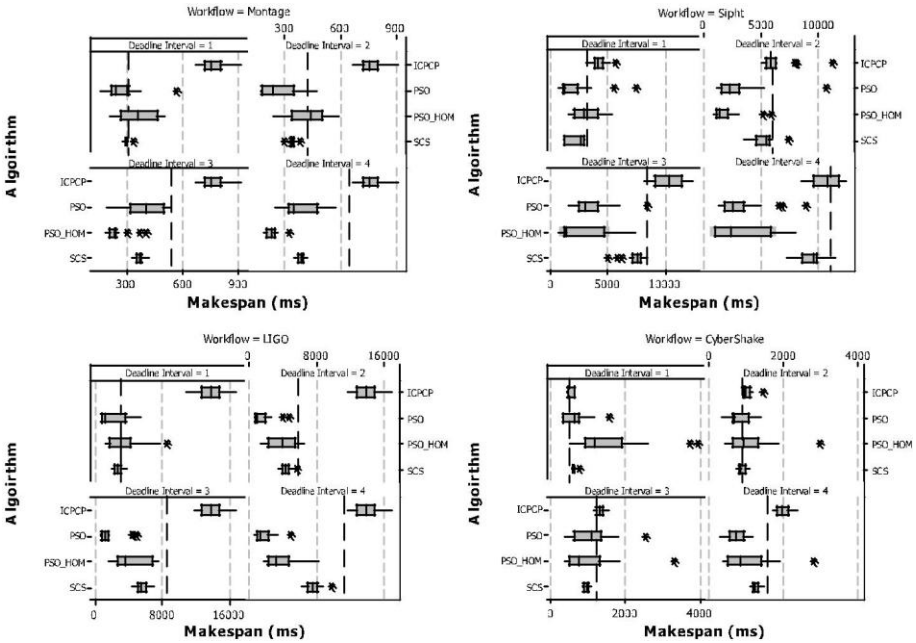

Fig. 6. Boxplot of makespan by algorithm for each workflow and deadline interval. The reference line on each panel indicates the deadline value of the corresponding deadline interval

The dotted line on each panel of each graph corresponds to the deadline value for the given interval. Evaluating the makespan with regards to this value is essential as all of the algorithms were designed to meet the given deadline. For the LIGO and Montage workflows, IC-PCP fails to meet this goal by producing schedules which take longer time to execute on average than the workflow's deadline.

Cost Evaluation

Makespan is the algorithms should be able to generate a cost-efficient schedule but not at the expense of a long execution time. The reference line on each panel displaying the mean makespan is the deadline corresponding to the given deadline interval. We present this as there is no use in an algorithm generating very cheap schedules but not meeting the deadlines; the cost comparison is made therefore, amongst those heuristics which managed to meet the particular deadline in a give case.

Further Analysis

The results found for the small and extra large workflows were similar to the ones presented here, although, our approach seems to perform better with smaller sized workflows which is reasonable as bigger workflows mean larger search spaces. We also found that in some cases our solution tends to generate schedules with lower makespans and higher costs as long as the deadline is met. Some users might prefer this as a solution whereas others might prefer the makespan to be closer to the deadline and pay a lower price. As future work, experiments will be held to analyze the performance of the scheduling algorithm in these scenarios using different optimization techniques such as genetic algorithms. Another finding is the significant impact that the selection of the initial pool of resources has on the performance of the algorithm. The same set of experiments was conducted with an initial VM pool composed of one VM of each type for each task. The results show that the algorithm takes longer to converge and find an optimal solution producing schedules with higher execution times and costs. We noticed this strategy leads the algorithm into leasing more VMs with lower utilization rate and hence incurring in higher costs and more delays such as startup and data transfer times.

VI.CONCLUSIONS AND FUTUREWORK

This is a combined resource provisioning and scheduling strategy for executing scientific workflows on IaaS clouds. The scenario was modeled as an optimization problem which aims to minimize the overall execution cost while meeting a user defined deadline and was solved using the meta-heuristic optimization algorithm, PSO. The proposed approach incorporates basic IaaS cloud principles such as a pay-as-you-go model, heterogeneity, elasticity, and dynamicity of the resources. Furthermore, our solution considers other characteristics typical of IaaS platforms such as performance variation and VM boot time. The simulation experiments conducted with four wellknown workflows show that our solution has an overall better performance than the state-of-the-art algorithms, SCS and IC-PCP. In every case in which IC-PCP fails to meet the application's deadline, our approach succeeds. Furthermore, our heuristic is as successful in meeting deadlines as SCS, which is a dynamic algorithm. Also, in the best scenarios, when our heuristic, SCS and IC-PCP meet the deadlines, we are able to produce schedules with lower execution costs. As future work, we would like to explore different options for the selection of the initial resource pool as it has a significant impact on the performance of the algorithm. We would also like to experiment with different optimization strategies such as genetic algorithms and compare their performance with PSO.

Another future work is extending the resource model to consider the data transfer cost between data centers so that VMs can be deployed on different regions. Extending the algorithm to include heuristics that ensure a task is assigned to a VM with sufficient memory to execute it will be included in the algorithm. Finally, we aim to implement our approach in a workflow engine so that it can be utilized for deploying applications in real life environments.

REFERENCES

[1] 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.

[2] 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.

[3] R. Buyya, J. Broberg, and A. M. Goscinski, Eds., Cloud Computing: Principles and Paradigms, vol. 87, Hoboken, NJ, USA: Wiley, 2010.

[4] J. Kennedy and R. Eberhart, "Particle swarm optimization," in Proc. 6th IEEE Int. Conf. Neural Netw., 1995, pp. 1942–1948.

[5] Y. Fukuyama and Y. Nakanishi, "A particle swarm optimization for reactive power and voltage control considering voltage stability," in Proc. 11th IEEE Int. Conf. Intell. Syst. Appl. Power Syst., 1999, pp. 117–121.

[6] C. O. Ourique, E. C. Biscaia Jr., and J. C. Pinto, "The use of particle swarm optimization for dynamical analysis in chemical processes," Comput. Chem. Eng., vol. 26, no. 12, pp. 1783–1793, 2002.

[7] T. Sousa, A. Silva, and A. Neves, "Particle swarm based data mining algorithms for classification tasks," Parallel Comput., vol. 30, no. 5, pp. 767–783, 2004.

[8] M. R. Garey and D. S. Johnson, Computer and Intractability: A Guide to the NP-Completeness, vol. 238, New York, NY, USA: Freeman, 1979.

[9] M. Rahman, S. Venugopal, and R. Buyya, "A dynamic critical path algorithm for scheduling scientific workflow applications on global grids," in Proc. 3rd IEEE Int. Conf. e-Sci. Grid Comput., 2007, pp. 35–42.