An Effective Multi-Objective Workflow Scheduling in Cloud Computing: A PSO based Approach

Shubham[*], Rishabh Gupta[†], Vatsal Gajera[‡], Prasanta K. Jana[§], *IEEE Senior Member*
Department of Computer Science and Engineering
Indian School of Mines, Dhanbad, India
[*]shubhambce2010@gmail.com, [†]rg.aimt@gmail.com, [‡]vgajera7@gmail.com, [§]prasantajana@yahoo.com

***Abstract*—Cloud computing has emerged as prominent paradigm in distributed computing which provides on-demand services to users. It involves challenging areas like workflow scheduling to decide the sequence in which the applications are to be scheduled on several computing resources. Due to NP-complete nature of workflow scheduling, finding an optimal solution is very challenging task. Thus, a meta-heuristic approach such as Particle Swarm Optimization (PSO) can be a promising technique to obtain a near-optimal solution of this problem. Several workflow scheduling algorithms have been developed in recent years but quite a few of them focuses on two or more parameters of scheduling at a time like usage cost, makespan, utilization of resource, load balancing etc. In this paper, we present a PSO based workflow scheduling which consider two such conflicting parameters i.e., makespan and resource utilization. With meticulous experiments on standard workflows we find that our proposed approach outperforms genetic algorithm based workflow scheduling in all cases achieving 100% results.**

Keywords : Cloud computing; Workflow scheduling; Particle Swarm Optimization

I. Introduction

IaaS cloud consists of numerous data-centers which contain several computing resources to execute the tasks submitted by users [1]. However, due to limitation of availability of computational resources, provisioning of resources is a challenging task. It involves two steps to provide services: (1) selection of VMs on which applications are to be executed, (2) order in which these VMs are to be operated for better utilization of resources i.e., generation of schedule. Both these steps involves several challenges related to concerned areas of scheduling like optimization of execution time, load balancing, reliability of VMs, cost of execution etc. Hence, to meet the QoS (Quality of Service) both these steps must be defined in an effective manner.

A cloud gives an illusion that enormous amount of resources are available that can be used by user on pay-per-use basis, which is called as Illusion of Infinite resources [5]. In practical world the applications are presented in the form of workflows in which tasks are dependent on each other in terms of computing and data requirements. A workflow simplifies the execution complexity and management of applications by representing process as series of steps [5]. Due to high requirement of computing resources in workflows, datacenters must choose an effective scheduling algorithm to cope with the requirements of users and improve the utilization of resources by optimizing concerning parameters. Thus, workflow scheduling came into picture due to its NP-complete nature. However, its implementation is tedious in cloud due to dynamic and heterogeneous behaviour of cloud, and also varying capabilities of VMs.

For scheduling the workflows we present a meta-heuristics technique to provide the effective scheduling that provide near optimal solution. We focus on minimizing makespan and maximizing resource utilization by developing a multi-objective fitness function which prioritizes both parameters depending upon the applications. If we consider only minimization of makespan, then tasks will select those VMs which will take minimum time to execute, but this will affect the utilization of those VMs which will take more time but results in effective utilization of resource. Thus, optimization of one parameter will affect the other. So, we must devise a multi-objective trade-off based scheduling algorithm which will minimize makespan and maximize resource utilization.

For a given workflow comprising of n tasks and available VMs are m, then the total number of possible schedule are m^n, thus generating an optimal schedule by Brute Force technique involves lot of computational time. Thus, meta-heuristics approaches are desirable to reduce the complexity by its faster rate of convergence and simpler implementation. Although these approaches does not always guarantee an optimal schedule, but it will give near optimal solution when computed over higher number of iteration.

In this paper we have used a meta-heuristic approach namely PSO to schedule the workflows. Due to its enormous popularity several researcher applied meta-heuristics approaches like Ant Colony Optimization, Genetic Algorithm [12], [13] to solve the problem of workflow scheduling, but still the problem requires attention. Our proposed approach gives better results than genetic algorithm.
In this paper we address the following aspects:

- Workflow scheduling with trade-off between makespan and resource utilization.
- Derivation of multi-objective fitness function involving two objectives of scheduling.
- Comparison of makespan and resource utilization over standard workflows with standard Genetic algorithm based workflow scheduling.

In recent years, several works [6], [7], [8] have been proposed which are related to workflow scheduling, however, none of them completely solves the problem. In [9], PSO based technique is used by Pandey et al. to schedule the workflows by considering cost as major objective in fitness function. They explained the optimization of cost by considering *ETC* (Execution Time to Compute) matrix as well as data transfer cost matrix. The main idea is to schedule tasks on VMs such that overall cost of operation is minimized but they fail to provide an efficient encoding scheme and apart from cost no other parameter is consider like makespan and resource utilization. The results were compared with Best Resource Selection (BRS) and result shows three times better cost than BRS. The schedule generated in [9] is not an optimal schedule as it is application specific.

In [10], a static cost-optimization technique for scheduling workflows based on PSO technique is explained which meets the deadline constraint and it minimizes the overall execution cost of workflows. It shows dynamic provisioning of resources in the form of leases in IaaS cloud and schedule the tasks imposing budget constraint and meeting deadline and users are cost on pay-per-use basis. However, it does not consider optimization of time and by considering cost as major objective it may happen that only those VMs are selected which has less cost resulting in poor utilization of available resources. It does not consider communication cost between VMs which shows that it assumes that a task cannot be migrated to other VMs and dependent tasks needs to be scheduled on same VM.

Our paper summarized as follows. Section I give introduction to cloud computing and related works. Section II explains about workflows and scheduling problem. Overview of PSO is explained in Section III . Proposed work with illustrative example is explained in Section IV. Section VI shows update procedure. Experimental results are shown in Section VI. Section VII concludes our paper.

II. Workflow Scheduling Problem

A workflow is defined as *W* = (*TS*,*E*) where *TS* represents the set of tasks and *E* represents the set of edges, $e_{ij} = (T_i, T_j)$ where T_i is a parent task and T_j is a child tasks. Each edge $T_i \rightarrow T_j$ indicates that task T_j cannot be executed until all the dependent tasks of T_j are executed i.e., a child task cannot start executing until their parents are executed. A task executing on certain virtual machine VM_k and its dependant tasks executing on VM_l then in between VM_k and VM_l it involves certain execution time to transfer the results associated with parent task to child tasks that is known as communication overhead. If child and parent tasks executes on same VMs, then there will be no communication overhead. A sample workflow is shown in Fig. 1.

A workflow scheduling problem in IaaS cloud deals with finding an optimal schedule to execute the tasks. The main objective of workflow scheduling is to optimize overall completion time involving communication time as well as execution time on particular VM. This problem is considered as NP-complete as the search space is quite large and getting an optimal schedule in polynomial time is very challenging.

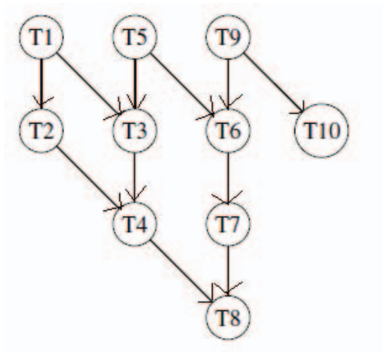

Fig. 1: Sample Workflows

III. Overview : Particle Swarm Optimization

PSO was developed as nature inspired algorithm by Kennedy and Eberhart [2] which is based on movement of birds. PSO has gained so much popularity because of its simplicity and application in several fields like wireless sensor networks, bioinformatics, image processing, data mining etc. It is used for solving several NP-hard as well as NP-complete problem like scheduling [3] and routing algorithms [4]. It is similar to other evolutionary technique like Genetic Algorithm (GA) but it does not involve generation of new individual by direct recombination rather it depends on behaviour of particles i.e., movement of birds in solution space. It does not involve special operator like mutation, crossover thus it is easier to implement, since new particles are generated based on the trajectory of other particles. Each particle keeps track of its position with respect to local best as well as global best and adjusts their position by updating their coordinates.

PSO consists of predefined initial particles (Pop_Size), where each particle represents a complete solution in the solution space. Each particle's position is represented as $S_{i,ts}$ where $1 \leq i \leq Pop\_Size$ and $1 \leq ts \leq TS$. The velocity of each particle is represented as $V_{i,ts}$. In every iteration we evaluate each particle with the help of fitness function to test the quality of solution obtained. The general representation of i^{th} particle is shown below :

$$POP_i = [S_{i,1}, S_{i,2}, \ldots.., S_{i,ts}]$$

In every iteration we find the local best position i.e., $Pbest_i$ and update particle position if new position is better than position in previous iteration. Our goal is to find *Gbest* that is considered as the global best solution. Each particle updates their velocity and position with respect to *Pbest* and *Gbest* in every iteration using Eq. 9 and Eq. 10 respectively.

IV. Proposed Approach

Our proposed approach involves certain steps that are involved in basic particle swarm optimization procedure. We propose an effective initialization procedure as well as encoding scheme to map the tasks onto VMs. Also we derive a fitness function involving two objectives transformed into linear scalar function as single objective and our goal is to optimize the fitness function. We illustrate our proposed algorithm with the help of sample workflow. The Initialization procedure is shown in Algorithm 1. Algorithm 2 represents the mapping procedure that is derived using SDT and our main workflow scheduling algorithm based on PSO is shown in Algorithm 3. The terminologies used in pseudo code of proposed approach is shown in Table I.

TABLE I: Terminologies used in Pseudo Code

Notation	Definition
Initialization	Procedure that initializes each particle with randomly generated uniformly distributed number in range of (0,1] and initialize vector *VEC*
MapTaskToVM	Procedure that maps task to virtual machine
CalFitness	Procedure that calculates the fitness value of particle
Pop_Size	Population size
Min_Fit	Temporary variable that is used to initialize Gbest value.
VEC	1D array that stores the value obtained by Initialization procedure.This array is used to map each task to VM.
Ietr	Iteration number
Max_Ietr	Total number of iteration
F(i)	Fitness value of particle *i*
POP	2D array with total number of rows equal to population size and each row size equal to particle size. Matrix stores the current population
MAP	2D array that stores the current mapping between task and virtual machine

A. Initialization of Particle

Each position $S_{i,ts}$ is assigned with uniformly distributed random values by using random () function, (0<random ()≤1) where *i* represents particle number and ts represents task. Initially each tasks are numbered by their *IDs* 1≤*k*≤*TS* (*k* represents natural number) which is multiplied by total number of VMs available. Now, successive division technique is applied to get standard vector representation of tasks that is further used to generate the mapping for each tasks to VMs in each iteration. The initialization procedure is further illustrated in Algorithm 1.

B. Mapping of task to VM

After initialization, the randomly generated values assigned to $S_{i,ts}$ are then multiplied by generated vector *VEC*[*j*] to get initial mapping. The initial mapping generated by our encoding scheme represents one of the possible solution over which our fitness value is evaluated and then our PSO based scheduling is applied to get near optimal solution. The mapping procedure is further illustrated in Algorithm 2.

C. Illustration

Consider set of 10 tasks *T*= $\{T_1, T_2, ..., T_{10}\}$ represented as DAG shown in Fig. 1 to illustrate the initialization as well as encoding scheme. Let the total available VMs be 3 $\{VM_1, VM_2, VM_3\}$. For each task, execution time is associated on particular VM is represented in *ETC* matrix shown in

```
Procedure: $Initialization(POP, VEC)$
1  for $i \leftarrow 1$ to $Pop\_Size$ do
2      for $j \leftarrow 1$ to $TS$ do
3          $POP(i,j) = random()$
4      end
5  end
6  for $j \leftarrow 1$ to $TS$ do
7      $tmp = j \times VM$
8      if $tmp > VM$ then
9          while $tmp > VM$ do
10             $tmp = floor(tmp/VM)$
11         end
12     end
13     $VEC(j) = tmp$
14 end
15 $return$
```

Algorithm 1: Initialization Procedure

```
Procedure: $MapTaskToVM(POP, MAP, VEC)$
1  for $i \leftarrow 1$ to $Pop\_Size$ do
2      for $j \leftarrow 1$ to $TS$ do
3          $MAP(i,j) = ceil(POP(i,j) \times VEC(j))$
4      end
5  end
6  $return$
```

Algorithm 2: Mapping Procedure

Fig. 2, also the communication overhead among the VMs are shown in Fig. 3. Here, *CM*(*i*,*j*) = *k* shows that communication overhead from VM_i to VM_j is equal to *k* units. If two tasks dependent on each other executes on same VM then the communication overhead is zero. The *ID* representation of each task is used to generate the vector using SDT after multiplying each task by total number of available VM. The SDT is used to get the standard vector *VEC*[*j*] and this *VEC*[*j*] is used to get mapping for each solution. The representation is shown in Table II. Random values are assigned to $S_{i,ts}$ and these values are used for mapping the tasks to virtual machines as shown in Table II.

Firstly, we multiply each task id by total number of *VMs* available. Then the multiplied value is successively divided by total number of *VMs* to get the number in range of total available VMs to generate the standard vector. For example: Task T_4 : *id* = 4 ; *id* × VM=12 ; *VEC*[4]= 1 (12/3=4 but 4>3 so again division is performed as 4/3 =1.33 i.e. floor value is 1). This SDT operation is performed over all tasks to generate standard vector of dimension *TS* which is used for mapping of tasks to VMs. This technique is used for generating solution vector for Pop_{Size} defined in our experimental result. The standard vector generated will be kept fixed during each iteration until our stopping criteria is satisfied, and updated

TABLE II: Mapping of tasks to VMs

Tasks	*Tasks* × *VM*	*VEC*	$S_{i,ts}$	*ceil* ($S_{i,ts}$ × *VEC*)	VMs
1	3	3	0.60	2	VM_2
2	6	2	0.35	1	VM_1
3	9	3	0.80	3	VM_3
4	12	1	0.50	1	VM_1
5	15	1	0.62	1	VM_1
6	18	2	0.15	1	VM_1
7	21	2	0.80	2	VM_2
8	24	2	0.45	2	VM_2
9	27	3	0.95	3	VM_3
10	30	3	0.25	1	VM_1

position values stored in *POP* matrix in each iteration will be multiplied by the standard vector for generating new mapping.

$$ETC = \begin{array}{c|cccc} & VM_1 & VM_2 & & VM_m \\ T_1 & w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ T_2 & w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ T_3 & w_{3,1} & w_{3,2} & \cdots & w_{3,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ T_{n-1} & w_{n-1,1} & w_{n-1,2} & \cdots & w_{n-1,m} \\ T_n & w_{n,1} & w_{n,2} & \cdots & w_{n,m} \end{array}$$

Fig. 2: ETC Matrix

$$CM = \begin{array}{c|cccc} & VM_1 & VM_2 & \cdots & VM_m \\ VM_1 & 0 & c_{1,2} & \cdots & c_{1,m} \\ VM_2 & c_{2,1} & 0 & \cdots & c_{2,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ & c_{m-1,1} & c_{m-1,2} & \cdots & c_{m-1,m} \\ VM_m & c_{m,1} & c_{m,2} & \cdots & 0 \end{array}$$

Fig. 3: Communication Overhead Matrix

D. Derivation of fitness function

The most important aspect of meta-heuristic approaches involves derivation of fitness function that satisfies the optimization problem. In case of workflow scheduling, it involves optimization of certain parameters like makespan, cost, resource utilization, energy, but in our proposed work we focus on optimization of two objectives i.e., makespan and resource utilization. As both parameters are conflicting in nature because one involves minimization of makespan and other involves maximization of resource utilization. Thus, our fitness function focuses on both these objectives as shown below:

Objective 1: Minimization of makespan subject to completion of all tasks
Mathematically makespan is represented below:

$$makespan = Q_{exe} + Q_{trans} \tag{1}$$

```
Input  : (1) A directed acyclic graph(DAG).
         (2) ETC (Expected time to compute) matrix
         (3) CM Communication cost matrix
Output: A schedule generated by algorithm.
1  $Initialization(\&POP, \&VEC)$
2  $MapTaskToVM(POP, \&MAP, VEC)$
3  for $i \leftarrow 1$ to $Pop\_Size$ do
4      $CalFitness(ETC, MAP(i), CM)$
5      $Pbest_i = POP_i$
6  end
7  for $i \leftarrow 1$ to $Pop\_Size$ do
8      if $Min\_Fit > F(Pbest_i)$ then
9          $Min\_Fit = F(Pbest_i)$
10         $Gbest = Pbest_i$
11     end
12 end
13 $Ietr = 0$
14 while $Ietr < Max\_Ietr$ do
15     for $k \leftarrow 1$ to $Pop\_Size$ do
16         Update velocity and position of $POP_k$
17         using Eq. 8 and Eq. 9
18         $MapTaskToVM(POP, \&MAP, VEC)$
19         $CalFitness(ETC, MAP(k), CM)$
20         if $F(POP_k) < F(Pbest_k)$ then
21             $Pbest_k = POP_k$
22         end
23         if $F(Pbest_k) < F(Gbest)$ then
24             $Gbest = Pbest_k$
25         end
26     end
27     $Ietr++$
28     Calculate makespan and resource utilization
       corresponding to mapping given by Gbest
29 end
```

Algorithm 3: Main Procedure

$$Q_{exe} = \sum_{k=1}^{TS} ETC(k, Z(k)) \tag{2}$$

$$Q_{trans} = \sum_{T_i \in T} \sum_{T_j \in T} CM(Z(T_i), Z(T_j)) \times Adj(T_i, T_j) \tag{3}$$

$$Z(T_i) \neq Z(T_j) \tag{4}$$

$$Adj(T_i, T_j) = \begin{cases} 1, & \text{if } T_i \rightarrow T_j \in E \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

Objective 2: Maximization of resource utilization (RU) subject to usage of available VMs

$$RU = (\sum_{i=1}^{VM} (time(VM_i))/makespan)/Total\_VM \tag{6}$$

Eq. 2 represents total execution time where $Z(i) = j$ shows task i is mapped to VM_j. Eq. 3 shows communication overhead among tasks. $Adj(T_i, T_j)$ shows adjacency matrix. We had used weighted sum approach to prioritize both the objectives. Eq. 7 calculates fitness value for each particle. In Eq. 7, α acts as weighting parameter which defines the priority of makespan and resource utilization depending upon application requirement. Thus, the fitness function involving both parameters is shown below:

$$F(i) = \alpha \times makespan + (1 - \alpha) \times (1/RU) \tag{7}$$

V. Update Velocity and Position

After generation of initial solution, PSO updates the velocity and position of every particle using Eq. 8 and Eq. 9 respectively. It may happen that during update of velocity and position values may goes out of bound of range (0,1] hence, mapping of tasks to VM will not be bounded to total availability of VMs. To overcome this problem we used following technique:

$$S'_{i,ts} = \begin{cases} 0.001 & \text{if } S'_{i,ts} < 0 \\ 0.999 & \text{if } S'_{i,ts} > 1 \end{cases} \tag{8}$$

Eq. 8 represents the updated position of particle. After updation of position of particle we generate the mapping using standard vector to evaluate fitness value of each particle.
From Eq. 9 and Eq. 10 we update position and velocity of each particle during each iteration until our stopping criteria is satisfied. In our proposed approach stopping criteria is predefined number of iteration. The *Gbest* obtained after meeting the stopping criteria represents the near optimal schedule for the tasks mapped to VMs.

$$V_{i,ts}(t) = w \times V_{i,ts}(t-1) + c_1 \times r_1 \times (S_{Pbest_{i,ts}} - S_{i,ts}(t-1)) + c_2 \times r_2 \times (S_{Gbest_{ts}} - S_{i,ts}(t-1)) \tag{9}$$

$$S_{i,ts}(t) = S_{i,ts}(t-1) + V_{i,ts}(t) \tag{10}$$

VI. Experimental Results

We have performed rigorous experiments of our proposed algorithm using Windows 7 Home Basic platform consisting of 4GB RAM with 2.2 GHz processor. The results were tested over some standard workflows that are used in several scientific applications. The results are simulated and compared with Genetic Algorithm based workflow scheduling by considering same fitness function.

A. Experimental Paremeters

The parameters used in our experiments are shown in Table III. In the list shown w indicates inertial weight in range of [0,1] which signifies friction. c_1 and c_2 represents cognitive acceleration constant and social acceleration constant to signify the bird's acceleration in search space. r_1 and r_2 are random numbers generated using random () function in range of (0,1]. These parameters are variable in nature which are tuned according to nature of experiments.

TABLE III: Values of Parameter used

Parameter	**Value**
Pop_Size	20
c_1	2.05
c_2	2.05
w	0.5314
Max_Ietr	500

B. Experimental Workflows

To test the efficacy of our proposed approach we simulate our results on standard workflows as shown in Fig. 4 that are published by Pegasus. These workflows are commonly used in several areas like in the field of Bioinformatics, Astronomy, Physics, Earthquake Science etc. The common workflows we used are Montage, Cybershake, Epigenomics, Inspiral, Ligo [11].

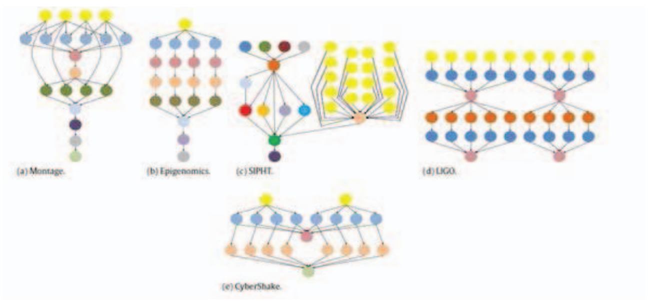

Fig. 4: Scientific Workflows

C. Experimental Analysis

The proposed approach has been simulated and tested by varying different parameters of PSO and with different α values and average results are shown. We find that our fitness function selects those mapping which maintain trade-off between makespan and resource utilization. We divide the workflows in two categories i.e., small (10-50 tasks) and medium (50-200 tasks) on the basis of number of tasks involved. From the results we find that our proposed approach gives better result for all types of workflows i.e., 100% better results are obtained through our proposed approach.

From Fig. 5 and Fig. 6, we find that for small sized and medium sized workflows, the makespan obtained using PSO based approach is lesser than GA based approach. For all the workflows the difference in makespan is significant and can be seen easily seen. Thus, our proposed approach gives 100% better results than GA based approach.

From Fig. 7 and Fig. 8, we find that the resource utilization obtained from PSO based approach is maximized as compared to GA based approach. The results clearly signified that our proposed approach outperform GA in all the cases by giving higher resource utilization.

Hence, from the results we find that the trade-off maintained by PSO based approach for $\alpha = 0.5$ gives better makespan and resource utilization as compared to existing approach.

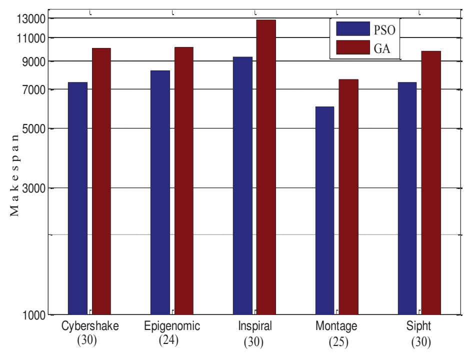

Fig. 5: Makespan comparison for small size workflows

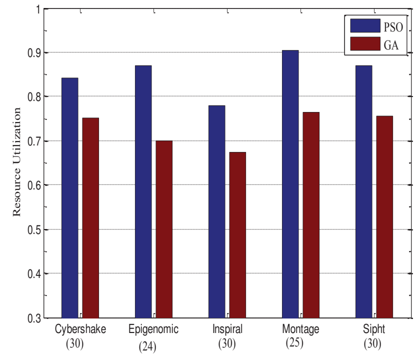

Fig. 7: Utilization comparison for small size workflows

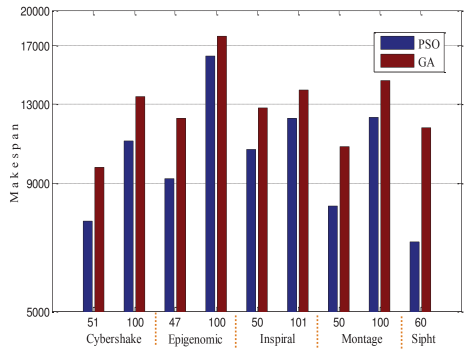

Fig. 6: Makespan comparison for medium size workflows

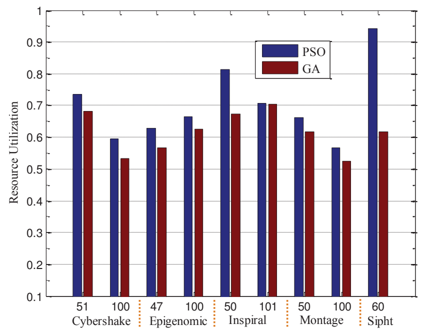

Fig. 8: Utilization comparison for medium size workflows

VII. Conclusion and Future Works

In this paper we have presented a multi-objective workflow scheduling algorithm based on PSO technique. We have presented an effective encoding scheme with weighted linear transformed fitness function that involves two parameters of scheduling, i.e., makespan and resource utilization. We have rigorously tested our proposed algorithm on several scientific workflows with different weighting values to show the efficacy of our approach over other meta-heuristic approach like genetic algorithm. The experimental results have shown that PSO based workflow scheduling outperforms GA over different performance metrics. In future we will consider dynamic behaviour of workflows and we will also extend our algorithm for large number of tasks.

References

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

[2] Kennedy, J.; Eberhart, R., "Particle swarm optimization," in Neural Networks, 1995. Proceedings., IEEE International Conference on , vol.4, no., pp.1942-1948 vol.4, Nov/Dec 1995 doi: 10.1109/ICNN.1995.488968

[3] B. Yu, X. Yuan and J. Wang, "Short-term hydro-thermal scheduling using particle swarm optimization method", Energy Conversion and Management, vol. 48, no. 7, pp. 1902-1908, 2007.

[4] P. Kuila and P. Jana, "Energy efficient clustering and routing algorithms for wireless sensor networks: Particle swarm optimization approach", Engineering Applications of Artificial Intelligence, vol. 33, pp. 127-140, 2014.

[5] Fakhfakh F, Kacem HH, Kacem AH. "Workflow Scheduling in Cloud Computing: A Survey", InEnterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW), 2014 IEEE 18th International 2014 Sep 1 (pp. 372-378). IEEE.

[6] Ding, Youwei, Xiaolin Qin, Liang Liu, and Taochun Wang. "Energy efficient scheduling of virtual machines in cloud with deadline constraint." Future Generation Computer Systems 50 (2015): 62-74.

[7] Alkhanak, Ehab Nabiel, Sai Peck Lee, Reza Rezaei, and Reza Meimandi Parizi. "Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: A review, classifications, and open issues." Journal of Systems and Software 113 (2016): 1-26.

[8] A. Talukder, M. Kirley and R. Buyya, "Multiobjective differential evolution for scheduling workflow applications on global Grids", Concurrency and Computation: Practice and Experience, vol. 21, no. 13, pp. 1742-1756, 2009.

[9] Pandey S, Wu L, Guru SM, Buyya R. "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments", InAdvanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on 2010 Apr 20 (pp. 400-407). IEEE.

[10] M. Rodriguez and R. Buyya, "Deadline Based Resource Provisioningand Scheduling Algorithm for Scientific Workflows on Clouds", IEEE Transactions on Cloud Computing, vol. 2, no. 2, pp. 222-235, 2014.

[11] Bharathi S, Chervenak A, Deelman E, Mehta G, Su MH, Vahi K. "Characterization of scientific workflows", InWorkflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on 2008 Nov 17 (pp. 1-10). IEEE.

[12] Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, Dan Wang, "Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization", CHINAGRID, 2011, 2013 8th ChinaGrid Annual Conference, 2013 8th ChinaGrid Annual Conference 2011, pp. 3-9, doi:10.1109/ChinaGrid.2011.17

[13] Xiaoguang Wang, "A Genetic Algorithm for Task Scheduling Based on User Overall Satisfaction", CYBERC, 2012, 2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 2013 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery 2012, pp. 527-530, doi:10.1109/CyberC.2012.97