Self-Adaptive PSO Memetic Algorithm For Multi Objective Workflow Scheduling in Hybrid Cloud

Padmaveni Krishnan and John Aravindhar
Department of Computer Science and Engineering, Hindustan Institute of Technology and Science, India

Abstract: *Cloud computing is a technology in distributed computing that facilitate pay per model to solve large scale problems. The main aim of cloud computing is to give optimal access among the distributed resources. Task scheduling in cloud is the allocation of best resource to the demand considering the different parameters like time, makespan, cost, throughput etc. All the workflow scheduling algorithms available cannot be applied in cloud since they fail to integrate the elasticity and heterogeneity in cloud. In this paper, the cloud workflow scheduling problem is modeled considering make span, cost, percentage of private cloud utilization and violation of deadline as four main objectives. Hybrid approach of Particle Swarm Optimization (PSO) and Memetic Algorithm (MA) called Self-Adaptive Particle Swarm Memetic Algorithm (SPMA) is proposed. SPMA can be used by cloud providers to maximize user quality of service and the profit of resource using an entropy optimization model. The heuristic is tested on several workflows. The results obtained shows that SPMA performs better than other state of art algorithms.*

Keywords: *Cloud computing, memetic algorithm, particle swarm optimization, self-adaptive particle swarm memetic algorithm.*

Received April 3, 2017; accepted May 29, 2017

1. Introduction

Large scale business applications are composed of multitasking, big data, time variant and fluctuating workloads. Cloud computing with its elastic computing property facilitates the scaling up and scaling down mechanism. The cloud provider takes care of the provisioning of resources. This paper emphases on how the heterogeneous resources are provisioned to users. The instances referred here are Virtual Machines (VMs) under execution.

The workflow scheduling in cloud environment is done in two stages. The first is the provisionisng stage in which the resources that are fit to run the task are selected. The second is the scheduling stage in which each task is mapped to the suitable resource without affecting the dependency among tasks. A workflow model is an application that has tasks and flow of data among tasks. A Workflow scheduling problem is a problem of assigning tasks to processors in multiprocessor environment [2, 14, 15]. This scheduling problem is NP-complete and it can be represented as a Directed Acyclic Graph (DAG) in which the nodes represents processes and edges represents the workflow among processors. The direction of edges represents the data dependencies among tasks and they are always directed.

Resources in cloud are allocated by the broker to the user on request. The scheduling algorithms use Quality of Service (QoS) constraints and solves the problem as single objective optimization problem. Two main algorithms like 'LOSS' and 'GAIN'[10] use a schedule and reassigns each task to another processor until it fits to budget. 'LOSS' algorithm saves larger money by assigning tasks with largest saving first. 'GAIN' algorithm starts assigning the tasks that require lest money. Some algorithms use the Pareto Swarm Optimization algorithm (PSO) for generating trade off among cost and make span. In this paper, we propose a cost minimized and deadline constrained heuristic algorithm that can be applied in cloud resource scheduling. The algorithm proposed considers the features such as dynamic provisioning and variation in performance of VM. The proposed algorithm is a hybrid algorithm called Self-adaptive Particle Swarm Memetic Algorithm which has Memetic algorithm and PSO as its predecessors. SPMA can be applied to real world pay-per-use pricing strategies and is based on IaaS instances. The algorithm gives the schedule that defines the mapping of task to VM and the time to lease and release the VM. Memetic operators like encoding, generating initial population, fitness function evaluation, crossover, mutation and reproduction are used in SPMA. The PSO algorithm is blended with memetic in selection of off springs for crossover and in the selection of population for the next generation.

The rest of the paper is organized as follows. The section 2 of paper provides a brief description of scheduling algorithms used and challenges on IaaS platforms. The section 3 provides problem definition. The section 4 explains the memetic algorithm and the way how it is merged with PSO algorithm to make

SPMA. Section 5 shows the test results and we provide concluding remarks in section 6.

2. Related Work

Workflow scheduling problem in a distributed environment is a wide area and it is proven to be NP-hard and it is impossible to guarantee an optimal solution in polynomial time. In cloud model, the amount charged to a user is purely based on the quantity of resource utilized. The pricing scheme used is based on two assumptions. First, the total cost of schedule is the sum of cost of all subtasks. Second, the cost cannot be changed when the provision is under progress.

List based heuristic algorithm finds the best assignment by traversing through all available processors in every selection step but this type of search cannot be applied every time in cloud scheduling as the resources are enormous and it is not possible to do such traversals every time and it will affect the make span. One of the well-known existing heurist search algorithm is Genetic algorithm in which few genetic operators represent the mapping of task to resources by strings. But, the existing genetic approaches might not be always suitable to the cloud environment because the VM instances are not fixed. Durillo and Pordan [3] proposed a list based heuristics that can be used in cloud. This algorithm constructs an instance pool of limited size and provides the possible schedules in advance for scheduling.

Sonia *et al*. [12] proposed a fault tolerant mechanism for real time tasks in cloud computing.This concentrats in fault tolerance and resource utilization. Sahni and Vidyarthi [9] proposed a cost effective deadline constrained algorithm by calculating minimum execution time of a workflow.

Artificial bee colony algorithm is used for scheduling and reliability analysis in machines and is verified with case examples [6]. Cloud task schduling based on Ant colony optimization is proposed by Tawfeek *et al*. [13] and is compared with Fist come First Serve and Round Robin algorithm. Optimal and sub optimal resouce allocation technique is proposed by Sharkh *et al*. [11]. This gives a promising level in connection request average tardiness Least squares support vector machines and particle swarm optimization is used to generate order and to evaluate the system failure probability of soil slopes [7].

3. Problem Formulation

3.1. Workflow Definition

A workflow can be denoted by means of Direct Acyclic Graph (DAG). Here workflow WORKFLOW=(T,E) where T is the set of 'n' vertices or tasks. $T=\{T_0, T_1, \ldots, T_n\}$ and E is the set of edges or data dependencies. $E=\{(T_i,T_j)/\ T_i,\ T_j \in T\}$. Each data dependency is assigned with the weight that represents the amount of data transferred among tasks. If there is an edge from T_i to T_j, T_i is the parent of T_j and needs to wait for the completion of the parent task. The task with higher priority has to be considered first [5].

3.2. Cloud Resource Management

A virtual machine that is running is called as an instance. There are different range of instance types having different execution time of tasks and bandwidths in IaaS platform. We have assumed that a customer can demand for any number of instances of their choice. Hence the set of instances $I=\{I_0, I_1, \ldots,\}$ is infinite whereas, the set $IT=\{IT_0, IT_1, \ldots IT_m\}$ is the type of instances offered by the cloud provider and it is fixed. We can say that the processing capacity of Instance I_i is PI_i and cost per unit time for Instance I_i is CI_i. Each task fits on one instance from the available type in T based on the properties defined in each instance type.

Assumption is made such that parallel execution of the tasks is also possible. The partial utilization of the rented VM is assumed as full utilization for that unit time. For instance, if the unit time is 5 minutes and the user uses 11 minutes, the user will pay for three periods.

Cloud providers like Amazon Elastic Compute Cloud (EC2), International Business Machines (IBM), Microsoft Azure, etc., support different pricing schemes. The algorithm proposed is flexible enough to fit for any pricing model. There are 'k' different pricing models supported in our algorithm. $P=\{P_0, P_1, \ldots P_k\}$.

3.3. The Scheduling Problem

The aim is to produce few meaningful scheduling choices with different instance, type and scheduling order of tasks and to reach the optimal or near optimal solution. The pricing scheme once chosen remains unchanged till the usage is completed. The goal for the cloud provider is to use a schedule that is better among all possibilities. The hybrid cloud model is represented in the Figure 1.

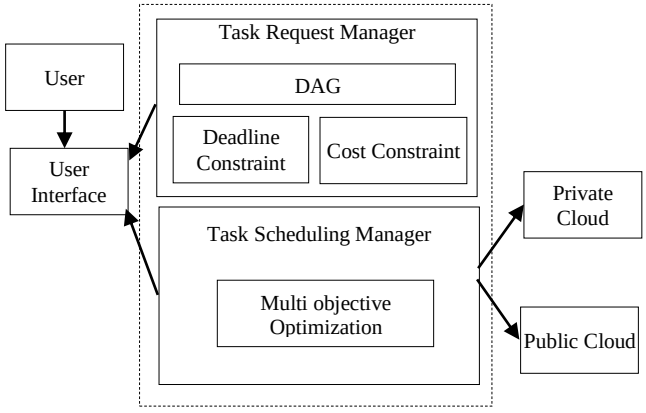


Figure 1. Hybrid cloud model.

According to the model, user submits the following as request to the request manager.

- No of task.
- The dependency among tasks.
- Properties of each task. This includes.
  - Memory.
  - No of instructions in that task.
  - No of CPUs needed.
  - Network preference if low or high.

The scheduling manager deploys the scheduling strategies under deadline and cost factors. The computing power of both private and public resource will be available with the scheduling manager.

Make span of a workflow is the total time elapsed from the start of the first task till the completion of the last job. Make span will be less if more parallel execution is done.

3.4. SPMA for Optimization

3.4.1. Memetic Algorithm

Memetic algorithm is anevolutionary population-based global search approach that can be used to optimize scheduling problems. For a heuristic search problem, the memetic algorithm generates a set of schedules. These schedules are modified using the genetic operators and the optimal solution or near optimal solution is reached within polynomial time.

3.4.2. PSO Scheduling Algorithm

Particle Swarm optimization is a population based optimization technique developed by Kennedy and Eberhart [8]. The algorithm is based on particles (ie., fish or bird) that has the capability to move around the solution space and reach to a solution. PSO has advantage of quick convergence, high precision solution and easier implementation [4]. The velocity factor says about the best solution of a particle. This is represented as Pbest and Gbest as the best solution of that particle based on the current iteration and history of the particles movement in solution space [1]. Velocity is updated for each iteration.

3.4.3. Self-Adaptive PSO Scheduling Algorithm

In self-adaptive approach the velocity updating strategy differs from the traditional PSO. There are three different velocity updating strategies Depending on the percentage of difference to the best for any particle, the strategy is selected [16].

- *Strategy* 1: Here the velocity is updated based on the difference in information of particles. It is shown in Equation

$$V_{id}(g)=x_{kd}(g)-x_{jd}(g) \quad (1)$$

$$V_{id}(g+1)=cv_{id}(g)+c[p_{id}(g)-x_{id}(g)] \quad (2)$$

Where *C* is a random no in range [0, 1]; $x_{kd}(g)$ and $x_{jd}(g)$ are d^{th} variable of two random values in g^{th} generation.

- *Strategy* 2: Here the velocity is updated based on the pbest of other particle.

$$V_{id}(g+1)= \Omega\, v_{id}(g)+cr[p_{kd}(g)-x_{id}(g)] \quad (3)$$

Where C is N(0,1);r is a uniformlu distributed real no in range [0,1]; $p_{kd}(g)$ is the pbest of a randomly selected particle. Ω is the weight or inertia. Ω says how far the previous velocities have impact on current velocity

- *Strategy* 3: This strategy is used in places where search is in smaller region. The velocity updation here is as follows.

$$V_{id}(g+1)= v_{id}(g)+0.5*cr[p_{kd}(g)-x_{id}(g)+ p_{id}(g)-x_{id}(g)] \quad (4)$$

- *Selection of Velocity Updating Strategies*: The strategies are selected based on the generation number and the difference to the gbest for any particle. A strategy count here is the number of different velocity updating strategies we have. Since we have three velocity updating strategies, the strategycount we take is 3. The strategy is selected based on the iteration count and the difference between Pbest and Gbest of particle.
- If the iteration is 1 to log(N) and the % of difference of pbest to gbest is less than N/log(n) it is likely to be closer to solution and the search space is small. So strategy 3 is selected and updation is done based on Equation (4).
- If the iteration is log(n) +1 to log(N)*log(N) and the % of difference of pbest to gbest is less than N/log(N)*log(N) the strategy 2 is selected and updation of velocity is done with Equation (3).
- For all remaining casesthe strategy 1 is selectedand updation of velocity is done with Equations (1) and (2).

3.4.4. The SPMA Scheduling Algorithm

The SPMA scheduling algorithm is a hybrid version of Self-adaptive PSO and Memetic algorithm. It is shown in Algorithm 1

1. D←Dimension of Search space.
2. Generate the initial population with N particles.
3. Randomly initialize the velocity for all particles.
4. For each particle in the space, assess the fitness value.
   a. $Pbest_i$ is the best fit schedule of particle i
   b. Gbest is the best fit schedule among all particles.
   c. Perform crossover for all particles to the best schedule.
   d. Perform mutation repetitively until a better schedule is reached or the no of mutations is log(D).

e. Choose only best D particles among the available schedules after crossover and mutation operation.
f. Update the velocity and position of all particles.

5. If stopping criterion is not reached, repeat step 3 for all particles. Else output the gbest as the best schedule.

There are two main aspects in modelling the SPMA. First one is how the solution is represented as a particle and second is telling the goodness of the solution. The goodness of the solution is called as fitness function. Four main objectives of our problem are Make span, Total cost, Weightage for not missing deadline and minimal use of private cloud resource.

3.4.4.1. Total Cost

The total cost is the sum of cost involved in private cloud and cost involved in public cloud.

According to the Amazon cloud service model, the cost for public cloud resource Public Cloud Cost or (PUCOST) depends upon

- Computing Power (CP).
- Computing Cost (CC).
- Transmission Power (TP).
- Transmission Cost (TC).
- Storage Cost (SC) of the resource used.

$$\text{PUCOST} = \sum_{Ti \in I^*}^{n} cost < CPi, CCi, TPi, TCi, SCi > \quad (5)$$

A private cloud does not include the Transmission power and the transmission capacity. The cost for private cloud resource (PRCOST) depends upon

- Computing Power (CP).
- Computing Cost (CC).
- Storage Cost (SC).

$$\text{PRCOST} = \sum_{Ti \in I^*}^{n} cost < CPi, CCi, SCi > \quad (6)$$

A task Ti is a unit of task request. This is a node in the DAG graph. The cost for execution of a task Ti is calculated based on the

- Millions of Instructions (MI).
- Data Associated (DA).
- Deadline of Task (DL).

The cost for a task Ti will depend upon the Instance Type (IT) to which it is mapped to and the type of cloud it is mapped to i.e., whether public or private.

Costis calculated using the Equations (5) or (6) depending on whether private or public cloud resource is used.

$$Total\ Cost = PUCOST + PRCOST \quad (7)$$

3.4.4.2. Make Span

The Start time of any task will depend upon the finish time of its previous task and delay if any. If the task Ti belongs to instance Ij, we can say Ins (Ti)=Ij. The make span is calculated as the difference in Start Time (ST) and Completion Time (CT) of starting task and finishing task respectively.

$$\text{ST}(T_{\text{first}})=0$$

$$CT(Ti) = ST(Ti) + \frac{MI(Ti)}{CP(IT)} + \frac{DA(Ti)}{TP} \quad (8)$$

$$Make\ span = CT(T_{exit}) - ST(T_{start}) \quad (9)$$

3.4.4.3. Deadlne Missing

Another important factor is not to miss deadline for any task. This is calculatd by the difference with the deadline for each task to its completion time. If there is zero no of task missing deadline, the objective function value is 'n'. The Deadline Miss (DLMISS) is calculated by the Equation (10).

$$DLMISS = \sum_{i=1}^{n} \text{i} * \text{x} \quad (10)$$

Where x is calculated by Equation (11)

$$x = \begin{cases} 0\ if\ DL(Ti) - CT(Ti) < 0 \\ 1,\ otherwise \end{cases} \quad (11)$$

3.4.4.4. Private Cloud Utilization

In order to have an efficient schedule, the resource needs to be provisioned in private cloud as maximum as possible. The factor is considered as as follows. The private cloud usage (PVTUSAGE) is calculated by the Equation (12)

$$PVTUSAGE = \sum_{i=1}^{n} \text{i} * \text{y} \quad (12)$$

Where y is calculated by Equation (13)

$$y = \begin{cases} 1\ if\ Private\ cloud \\ 0,\ otherwise \end{cases} \quad (13)$$

The fitness function is the combination of all the four objectives. It is calculated by Equation (14). Objective function for this problem is a combination of all four objectives mentioned ie., Obj_i is Make span, Obj_2 is

total cost, Obj_3 is Deadline missed and Obj_4 is Private cloud usage.

$$Objectivefunction = \sum_{i=1}^{4} (w_i Obj_i) \quad (14)$$

Where Obj_i is the value obtained in the objective function *i* and $0 \leq w_i \leq 1$

3.5. Encoding Procedure

The problem is encoded as following. The order of schedule is the sequence of tasks. The sequence is represented as S_1, …..S_n where S_i will start execution only after the completion of S_{i-1}. The task_instance is an array of size 'n' where the i^{th} element represents the instance type of i^{th} task. The instance_type is an array of size 'm' where the i^{th} element has the instance type of i^{th} instance. A sample DAG workflow is shown in Figure 2.

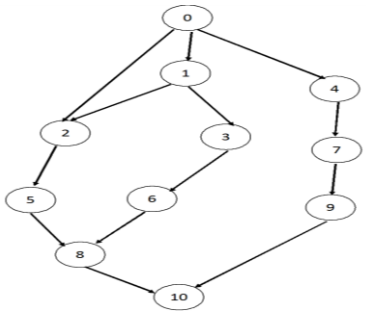


Figure 2. Example DAG workflow.

4. Spma Operators

4.1. Crossover

Crossover is a memetic operator. As the scheduling for tasks is precedence constrained. The precedence should always be maintained among the tasks. If T_k has to get the output of T_i, then T_i should precede T_k in all the possible orders generated. The crossover used is Partially Matched Crossover (PMX) shown in Algorithm 2. For two schedules 'X' and 'Y' the PMX randomly picks over two crossover points. The crossover point chosen is used for the construction of next generation schedule. The crossover is performed by considering the following facts.

- Repetition of task is not allowed in schedule.
- The dependencies among tasks are strictly maintained.

Algorithm 2: PMX and mutation to find the new chromosome

1. n= number of tasks
2. r1= random value between 0 and n-1
3. r2=random number between 0 and n-1 where r1<>r2
4. if r1 > r2, swap r1 and r2
5. Swap the allels in positions r1 and r2. Generate the new schedule
6. Check if dependency among tasks is maintained.
7. If dependency is not maintained, repeat all three previous steps
8. Check the fitness of new string generated. Above four steps until a better fit schedule is obtained or if the number of mutation done is $log n$

4.2. Mutation

This is a memetic operator that maintains alteration in one or more gene values. Mutation does an occasional random alteration of a value in a schedule with small probability as in Algorithm 3.

Algorithm 3: Mutation of off springs

1. n= number of tasks
2. r1= random value between 0 and n-1
3. r2=random number between 0 and n-1 where r1<>r2
4. if r1 > r2, swap r1 and r2
5. Substring1 ← A(r1:r2)
6. Substring1 ← B(r1:r2)
7. For all allels (Tasks) in Substring1 and Substring2
8. If the allels in substring (A,0,r1-1)and substring (A,r2+1,n-1) does not contain entries from substring (B,r1,r2)
9. Find the remaining allels in A and B and place them in Newspring1 and Newspring2 from left to right

4.3. Initial Population Generation

The initial population generation plays a major role in the convergence towards the optimal solution. For a problem size of 'N' tasks, the initial population of 'N' possible schedules are generated. Among the 'N' schedules, first three schedule (particles) are based on

- Shortest Job First (SJF).
- Ascending order of task size.
- Topological ordering based on indegree of DAG.

The other n-3 particles are generated on random ordering. Any schedule generated will be considered only if the dependency is maintained.

4.4. Complexity Analysis

Complexity of memetic operators crossover and mutation are $O(n^2)$ and $O(n)$ for n tasks. The dependency check and fitness evaluation, require only conditional judgments. Hence, the complexity of optimization for this is linear, namely, O(n). Any graph of 'n' vertices could have a maximum of n^2 edges. considering 'h' iterations, the overall complexity is $O(hn^2)$. The complexity of initial three algorithms also needs to be included. The SJF, Topological ordering and ascending order of task size has $O(n^2)$ complexity each. The complexity of Self-adaptive PSO operations include the comparison of particle fitness value in two stages. This is O(n). Updating velocity is O(n). The overall complexity is $O(n)+O(4n^2) + O(hn^2)$ Which is $O(hn^2)$ in general.

5. Testing

5.1. Experiment Setup

The performance is verified using the cloud simulation software Cloudsim 3.0. Three data centers A, B and C are created with different VM setup. as in Table 1. The task parameters are in Table 2.

Table 1. Parameter setup of VM.

Parameter	Instance setup in A	Instance setup in B	Instance setup in C
Number of CPU	1	1	2
CPU computing capacity	200 MIPS	400 MIPS	800 MIPS
RAM	1GB	4GB	4GB
Bandwidth	2M/s	4 M/s	4 M/s
Storage	4G	4G	8G

Table 2. Task parameters.

Parameter	Setup of Tasks
Length	[400,800]MIPs
File	[200,400]MB
Output Size	[20,40]MB

The data center A and B are private cloud and the data center C is public cloud.

The cloud tasks are randomly created as Cloudlet in Cloudsim. The number of tasks is varied from 100-800.

The price of public cloud includes an additional parameter called transmission cost. The cost of public cloud is depicted in the Table 3.

Table 3. Cost setup.

Parameter	Cost
Computing cost	0.07 $ per hour
Storage price	0.05 $ per GB
Transmission cost	0.04 $ per GB

5.2. Evaluation Metrics

The SPMA algorithm is compared with three algorithms.

- IaaS Cloud Partial Critical Path (IC-PCP).
- Genetic algorithm.
- PSO.

Four parameters are used to evaluate the performance of the scheduling strategies:

- The make span.
- The cost.
- The violation of deadline.
- The utilization of private cloud resources.

Violation of deadline is taken as a penalty. The transmission cost is less in case of private cloud and even negligible in certain cases and high whenever a resource from public cloud is used.

6. Results and Analysis

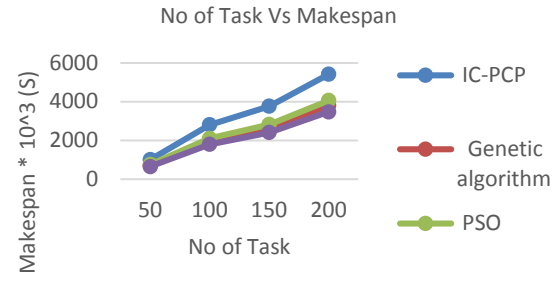
The result represents average performance of 200 tasks done 10 time and compared with three algorithms as mentioned in section V B. The performance of SPMA is the best of all. This is because of the self-adaptive approach and its convergence towards the solution.

Comparison graph is generated based on the four objectives. In the No of Tasks Vs Make span graph it was found that, IC-PCP was totally outplayed by the other algorithms. SPMA gave an accuracy of 4 to 5 % better than PSO.

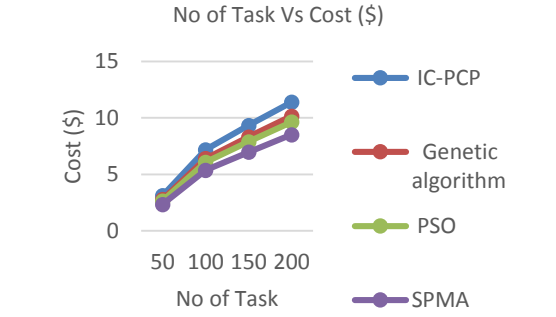
Considering the cost minimization parameter, Genetic algorithm and PSO performed with a close % of accuracy. Whereas the SPMA gave better result as the number of jobs increase.

While comparing the utilization of private cloud, the SPMA gave a better utilization of private cloud. The PSO also performed closer till the task count was 150. This is because the impact of self-adaptive approach which selects the necessary strategy in updating of velocity.

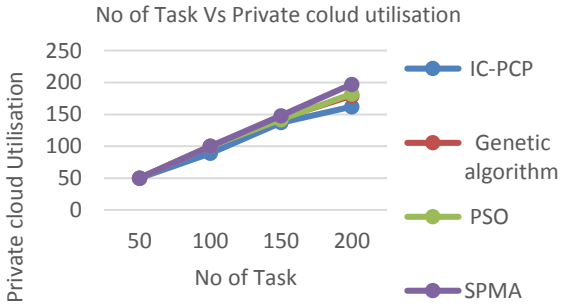
While comparing the Figures, we can say that IC-PCP is out performed by the other three algorithms. Out of the other three algorithms SPMA has proven better than GA. Comparative graph is in Figure 3.



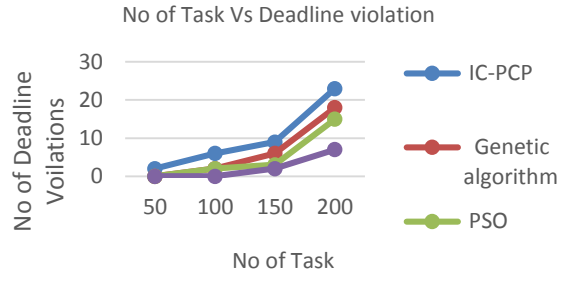
a) Makespan analysis.



b) Cost analysis.



c) Private cloud utilisation.



d) Deadline violation.

Figure 3. Analysis graphs.

7. Further Analysis

The algorithms performance was analyzed for different instances. Results says that the solution was generated with lower make span and minimized cost most of the time. The private cloud was also used to the maximum. As a future work, the algorithm could be compared with other optimization techniques available.

A significant point is the impact on the selection of initial population. The initial three algorithms selected helps in faster convergence towards solution.

When checked with the input condition of one virtual machine of every type in each task, it was found that the algorithm takes more execution time for longer

tasks when compared to shorter tasks. Overall SPMA algorithm performs better than other algorithms. Further analysis can be done with hybrid algorithms of ACO or PSO algorithm with bees algorithm.

8. Conclusions and Future Work

Many scheduling algorithms are available for heterogeneous cloud environment. Most of these algorithms have difficulty when directly applied in cloud. SPMA overcomes the issues as it uses the real-world cloud computing model. The pricing could be varied according to the need of cloud provider.

To provide a solution to the multi-objective cloud scheduling problem, a DAG based encoding is used. Different instances of tasks and their types are considered here. The memetic operators like evaluation function, crossover and mutation are merged along with the PSO algorithm.

The future work can be a hybrid algorithm of Bees algorithm or any other optimization algorithm using more than one pricing scheme. The resource allocation algorithm can also include the priorities among task for scheduling. The work can be extended with privacy metrics so that the scheduling locations can be kept safe.

References

[1] Al-Maamari A. and Omara F., "Task Scheduling Using PSO Algorithm in Cloud Computing Environments," *International Journal of Grid Distribution Computing*, vol. 8, no. 5, pp. 245-256, 2015.

[2] Chen W. and Zhang J., "An Ant Colony Optimization Approach to A Grid Workflow Scheduling Problem with Various Qos Requirements," *IEEE Transaction System Man Cybern*, vol. 39, no. 1, pp. 29-43, 2009.

[3] Durillo J. and Pordan R., "Multi Objective Workflow Scheduling in Amazon EC2," *Cluster Computing*, vol. 17, no. 2, pp. 169-189, 2014.

[4] Garg R. and Singh A., "Multiobjective Workflow Grid Scheduling Based on Discrete Particle Swarm Optimization," *in Proceedings of International Conference on Swarm, Evolutionary, and Memetic Computing*, Visakhapatnam, pp. 183-190, 2011.

[5] John S. and Mohamed M., "A Network Performance Aware QoS Based Workflow Scheduling for Grid Services," *The International Arab Journal of Information Technology*, vol. 15, no. 5, pp. 894-903, 2018.

[6] Kang F. and Li J., "Artificial Bee Colony Algorithm Optimized Sup Port Vector Regression for System Reliability Analysis of Slopes," *Journal of Computing in Civil Engineering*, vol. 30, no. 3, pp. 04015040, 2016.

[7] Kang F., Li J., and LiJ J., "System Reliability Analysis of Slopes Using Least Squares Support Vector Machines with Particle Swarm Optimization," *Neurocomputing*, vol. 209, pp. 46-56, 2016.

[8] Kennedy J. and Eberhart R., "Particle Swarm Optimization," *in Proceedings of the 6th IEEE International Conference in Neural Network*, Perth, pp. 1942-1928, 1995.

[9] Sahni J. and Vidyarthi D., "A Cost-Effective Deadline-Constrained Dynamic Scheduling Algorithm for Scientific Workflows in a Cloud Environment," *IEEE Transactions on Cloud Computing*, vol. 6, pp. 2-18, 2016.

[10] Sakellaiou R., Zaho H., Tsiakkouri E., and Dikaiakos M., *Integrated Research in GRID Computing*, Springer, 2007.

[11] Sharkh M., Shami A., and Ouda A., "Optimal and Suboptimal Resource Allocation Techniques in Cloud Computing Data Centers," *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 6, no. 1, pp. 1-17, 2017.

[12] Soniya J., Sujana J., and Revathi T., "IEEE Dynamic Fault Tolerant Scheduling Mechanism for Real Time Tasks in Cloud Computing," *in Proceedings of International Conference on Electrical, Electronics, and Optimization Techniques*, Chennai, pp. 124-129, 2016.

[13] Tawfeek M., El-Sisi A., Keshk A., and Torkey F., "Cloud Task Scheduling Based on Ant Colony Optimization," *The International Arab Journal of Information Technology*, vol. 12, no. 2, pp. 129-137, 2015.

[14] Yu J., Kirle y., and Buyya R., "Multiobjective Planning for Workflow Execution on Grids," *in Proceedings of the 8th IEEE/ ACM International Conference on Grid Computing*, Austin, pp. 10-17, 2007.

[15] Zhang F., Cao J., Hwang K., and Wu C., "Ordinal Optimized Scheduling Of Scientific Workflows in Elastic Compute Clouds," *in Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science*, Athens, pp. 9-17, 2011.

[16] Zuo X., Zhang G., and Tan W., "Self Adaptive Learning PSO- Based Deadline Constrained Task Scheduling for Hybrid IaaS Cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564-573, 2014.

Padmaveni Krishnan received ME degree in computer Science from Madurai Kamaraj University in 2002. She is currently an Assistant Professor in Hindustan Institute of Technology and Science and her main interest are in virtualization and scheduling in cloud.

John Aravindhar received PhD degree in Data mining from Hindustan University. He is currently an Associate Professor in Computer Science Department of Hindustan Institute of Technology and Science. His area of interest are Data mining and cloud.