Received: June 25, 2017

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

A Hybrid Approach of Memetic and Bees Life Algorithm for Multiobjective Workflow Scheduling in Cloud

Padmaveni Krishnan[1]* **John Aravindhar[1]**

[1]*Hindustan Institute of Technology and Science, India*
* Corresponding author's Email: kpadmaveni@hindustanuniv.ac.in

Abstract: Cloud computing is a paradigm that provides platforms for scientific applications. In clouds, the users pay based on the usage and the quality of service (QoS). There are many workflow scheduling algorithms in the heterogeneous computing environment. But all these cannot be applied in cloud due to t service based resource managing method in cloud. This paper discusses a workflow scheduling problem which gives an optimized solution considering make span and deadline as two objectives. A novel hybrid algorithm of Memetic algorithm and Bee swarm optimization algorithm called Memetic Bee Optimization algorithm (MBOA) is proposed. Experiments on randomly generated workflow and real time workflows show that the schedules generated by MBOA gives more stability on most of workflow instances. The results also show that the algorithm gives significantly better solutions than existing QoS optimization algorithms.

Keywords: Cloud computing, Memetic algorithm, Bee swarm optimization algorithm, Memetic bee optimization algorithm.

1. Introduction

Cloud computing has emerged as a computing paradigm that aims at providing computing services as scalable and virtualized resources to enormous remote users having heterogeneous requirement. Cloud provides computing services like networks, storage and services allocated from the resource pool with minimal management. The services are provided so that they follow the service level agreement. The instances in SaaS are virtual machines(VMs). The VMs help the customer in getting almost unlimited access to the resources and also plays a major role in reducing the Total Ownership Cost.

Any application that has jobs and flow of data among jobs can be described by a Workflow model [1]. Workflow scheduling problem is is a problem of assigning jobs to processors in multiprocessor environment [2]. It is NP-complete and it can be represented as a Directed Acyclic Graph(DAG) in which the nodes are processes and edges are workflow among processors [3]. The edges show the data dependencies among jobs and they are directed [4]. Different workflows of Montage and Laser Interferometer Gravitational wave observation (LIGO) are explained by Luiz et al [5]. Montage application creates an image like sky that has about 17 hierarchical workflows and 900 sub workflows. LIGO represents a workflow that involves Tera bytes of data for producing the results.

Fig. 1 shows hybrid IaaS cloud having resources of private cloud and public IaaS cloud. The scheduling algorithm is run by the broker when the user submits a workflow. The broker determines the following:

i. The resources those will be used
ii. The part of workflow that will run on each cloud provider

Generally algorithms use QoS constraints and solves the problem by treating as single objective optimization problem. LOSS and GAIN [6] are algorithms that use a schedule and reassigns each job to another processor till it matches to budget. Algorithms like NSPO[7] use the Pareto Swarm

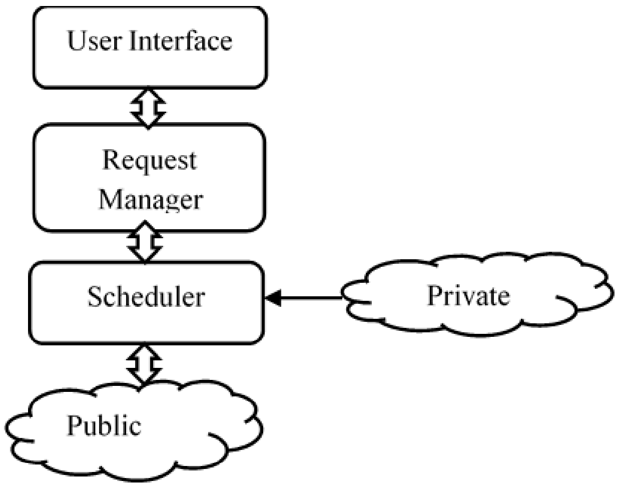


Figure.1 The hybrid IaaS cloud

Optimization algorithm (PSO) in order to generate trade off among cost and make span [8]. Multi-objective Heterogeneous Earliest Finish Time is an algorithm that extends the workflow in Amazon EC2.

Artificial bee colony algorithm is used to analyse the system reliability in machines and is verified with case examples [9]. Least squares support vector machines and particle swarm optimization is used to evaluate the system failure probability of soil slopes [10].

Even though the above algorithms can easily be applied in traditional heterogeneous environment, they are tough to apply in cloud. In this paper, an optimization algorithm for cloud workflow scheduling problem is proposed. The algorithm generates a set of schedules having different trade off between time and cost factor. The proposed algorithm is a hybrid algorithm called memetic bee optimization algorithm which has genetic algorithm and bees algorithm as its predecessors. MBOA uses real world pay-per-use pricing strategies and is based on IaaS instances. The memetic operators like encoding, evaluation function, initial population, crossover, mutation and reproduction are used. The bees algorithm is blended with memetic in the place of the selection for the next generation and also in selection of offspring for crossover.

The section 2 of paper highlights the commonly scheduling algorithms and challenges on IaaS platforms. This is followed by scheduling problem definition and the pricing models in section 3. The section 4 explains the memetic algorithm and the way how it is merged with bees algorithm to make MBO. Section 5 says about testing and the paper is concluded in section 6.

2. The common scheduling algorithms and challenges in IaaS platform

The workflow scheduling problem considered here assumes that the amount charged to a user is purely based on the resource utilized. POSH is an algorithm that has exponentially correlated the CPU cycles to cost. Our pricing model is based on two assumptions:

i. The total cost of task is the sum of cost of subtask.
ii. The cost cannot be changed when the service is under run.

The existing scheduling algorithms have challenges on the limitation of resource in the resource pool. List based heuristic algorithm finds the best assignment by traversing all available processors in the selection step of every task. But this cannot be applied every time in cloud scheduling since the resources are enormous and it is not possible to do such traversals every time. One of the well-known existing algorithm is Particle Swarm optimization based algorithm which defines the particle positions and velocities as matrices. Here the order is the no of tasks(n) by no of resources(m).ie.,m x n. The disadvantage here is the number of resources ‘m’ may be too large to handle.

Few genetic operators represent the mapping of task to resources by strings. But, the existing genetic approaches might not be suitable to the cloud environment always because the VM instances are not permanent since they may be allocated and deallocated anytime during execution.

Durillo et al., [11] proposed a list based heuristics that can be used in cloud. This algorithm constructs an instance pool of limited size and hosts out the possible schedules in advance.

In the algorithm proposed, the pricing criteria is considered when the fitness evaluation is made and the algorithm is designed such that it does not depend upon any fixed pricing schemes.

3. The workflow scheduling Problem

3.1 Workflow definition

A workflow can be generally represented by means of Direct Acyclic Graph (DAG). Here Work flow WFLOW=(J,C) where J is the set of ‘n’ vertices or jobs. $J=\{J_0, J_1, ..., J_n\}$ and C is the set of edges or control dependencies. $C=\{(J_i,J_j)/\ J_i,_{Jj} \in J\}$. Each control dependency is assigned with the weight that represents the quantity of data transferred among jobs. The weight assigned to the vertices are the execution time of jobs in a processor. It is denoted by *Exectime(J_i).*

Data(J_i ,J_j) denoted the transfer of data from job J_i to job J_j. The data transfer in always unidirectional. Excluding the source job, all other jobs have predecessor. Every job J_i apart from the source has its predecessor J_k provided there is an edge from J_k to J_i ie., *Pred(J_i)={J_k/(J_k,J_i)Є C }*. Since the source job will not have any predecessor, *Pred(J_{source})=Φ*

3.2 Cloud resource model

The infrastructure as service platform delivers computational resource through virtual machines. A virtual machine that is running is called as an instance. There are different range of instance types having different execution time of jobs and bandwidths in IaaS platform. It is assumed that a customer can get any number of instances. Hence the set of instances *I={I_0 ,I_1 ,….,}* is infinite whereas, the set *T={T_0 ,T_1 ,….T_m}* is the type of instances offered and it is fixed. Each job is mapped to one instance from the available type in T.

Every workflow has a user defined deadline associated with it. This deadline is the maximum time limit to complete the execution in cloud environment. If the user doesn't give the deadline, it is obtained by assigning all jobs sequentially to the fastest instance and calculate minimum execution time of the workflow

Each instance type T_i, has its own features represented by CPU Type CPU(T_i), Memory M(T_i) and Cost per time interval. It is also assumed that parallel execution of the jobs is also possible. The CPU instance plays a major role in the time taken. It specifies the configuration so that if the time taken to execute any job is half if the CPU instance is doubled. The time to run a of Job J_i on instance type T_j is represented by Eq. (1).

$$Time(Ji) = \frac{Unittime(Ji)}{CPU(Tj)} \quad (1)$$

Where $Unittime(Ji)$ is the time taken to execute the job *Ji* for unit CPU time.

Any instance type *Ti* will have its own bandwidth represented by *bw(Ti).* When the communication is among different instances, the bandwidth of both may be different. Here, the minimum among both bandwidth is considered to calculate the time so that the worst case time of communication can be calculated.

$$\text{T}ime\ communicaion(Ji, Jj) = \begin{cases} \frac{(Ji,Jj)data}{\min\{bw(InsR),bw(InsS)\}} & R \neq S \\ 0, \text{otherwise} \end{cases} \quad (2)$$

Where *InsR* and *InsS* are different instances to which *Ji* and *Jj* are scheduled.

3.3 Scheduling problem

The leading cloud providers like Amazon EC2, IBM, Microsoft Azure, etc have different pricing schemes allowed. The algorithm designed is a generic one that flexible to fit for any pricing model. It is assumed that there are 'k' different pricing models. *P={P_0 ,P_1 ,….P_k}.* The function *COST (I_l,P_m,T_n)* will calculate the cost for the instance 'l' for the pricing model 'm' having the instance type 'n'. The IaaS model is represented in general as *S=(I,P,M)* where *M* stands for instance type

The goal is to find a schedule to execute a workflow on cloud computing environment such that total execution cost and makespan are minimized by meeting the deadline constraints. Hence the two main objectives in workflow scheduling are

i. Total Cost Minimization
ii. Makespan

3.3.1. Total cost Minimization

For a workflow WFLOW=(J,C) and an IaaS S=(J,P,M), the idea is to produce more scheduling choice having instance, type and order. The order is the vector containing the scheduling order of jobs. The problem is considered with the constraint that the pricing scheme once chosen remains unchanged till the usage is completed. Hence the goal for the scheduler is to provide a schedule that has minimal cost among all possible schedules.

$$Total\ Cost = \sum_{Ti\ \epsilon I^*}^{n} cost(I_i, P, T_i) \quad (3)$$

Here I^* is the instances used by the user, p is the pricing option which remains unchanged.

3.3.2. Makespan

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

Any schedule has Start time(*ST*) and completion time(*CT*). Start time of any job will depend upon the finish time of its previous job. If the job 'Ji belongs to instance Ij, it can be said that *Ins(Ji)=Ij*. The instance available time is mentioned in the vector *Avail(J)*. Initially it is 0 for all the available instances. When in execution, *Avail(J)* will have the time when the instance will become free for next execution.

$$ST(Jfirst)=0 \quad (4)$$

$$ST(Ji)=max\{Avail(Ins(Ji), \quad max(CT(Time(Ji)+Time\ communication(Jj,Ji))^{n}\} \quad (5)$$

If Jstart is the first job of execution and Jexit is the last job to complete the execution in the workflow,

$$Makespan=CT(Jexit)-ST(Jstart) \quad (6)$$

4. MBOA for optimization

4.1 Memetic algorithm

The memetic algorithm can be used to optimize several conflicting objectives by combining them as a single objective. The memetic algorithm is an evolutionary algorithm that simulates the natural evolution and is successful in the past. For the workflow scheduling problem, the memetic algorithm generates a set of schedules selected by the user constraints. The schedules are refined using the genetic operators and the solution is reached. The solution is generated in polynomial time and could be optimal or near optimal solution.

4.2 Bees algorithm

The Bees algorithm is also an optimization algorithm which is based on the life of Bees. The bee colony generally has a Queen bee '*Q*', lots of male bees called Drones '*D*' and several thousands of sterile females called workers 'W'. [12]. The best fit schedule in any iteration is treated as queen '*Q*'. The crossover operation in any iteration is done with queen and Drones. The worker bees are those which are worst fit. These schedules are not fit for the next iteration.

4.3 Memetic bee optimization algorithm (MBOA)

The MBOA is a hybrid version of Memetic algorithm by blending the concepts of Bees algorithm also. The flow diagram of the hybrid version is in Fig. 2.

The constraints considered in this algorithm are

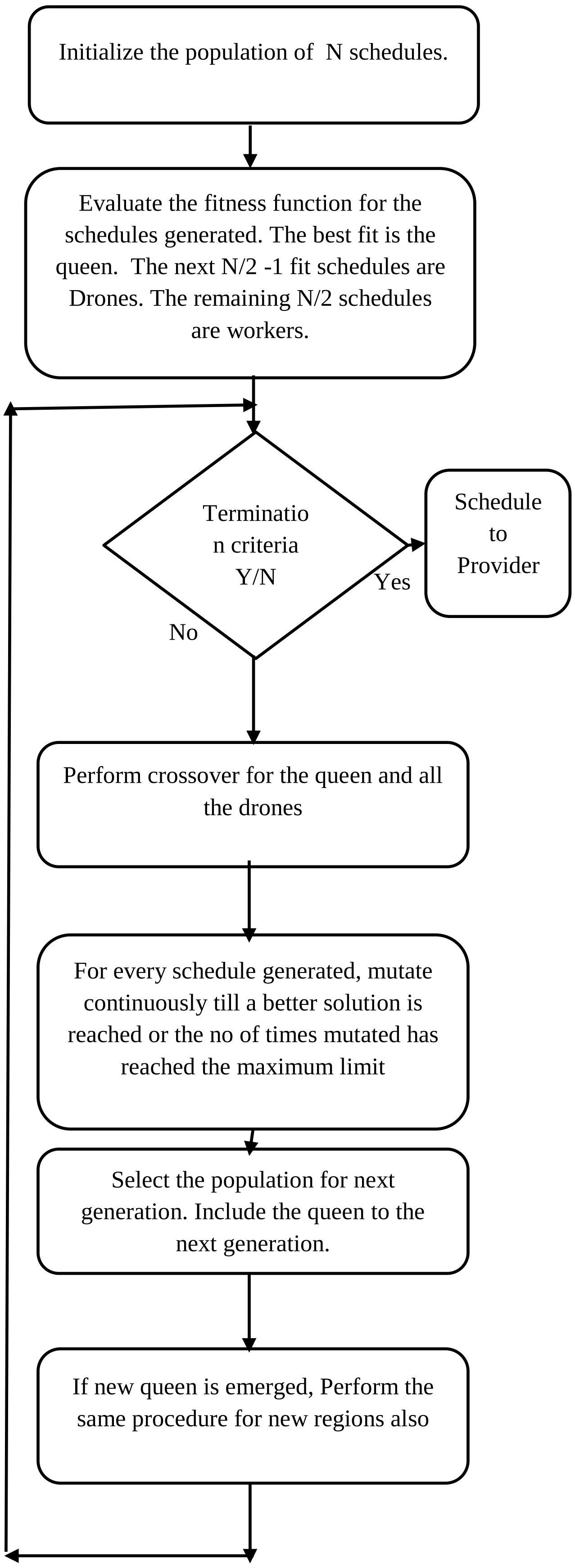

Initialize the population of N schedules.
Evaluate the fitness function for the schedules generated. The best fit is the queen. The next N/2 -1 fit schedules are Drones. The remaining N/2 schedules are workers.
Terminatio n criteria Y/N
Yes
No
Schedule to Provider
Perform crossover for the queen and all the drones
For every schedule generated, mutate continuously till a better solution is reached or the no of times mutated has reached the maximum limit
Select the population for next generation. Include the queen to the next generation.
If new queen is emerged, Perform the same procedure for new regions also

Figure.2 MBOA algorithm

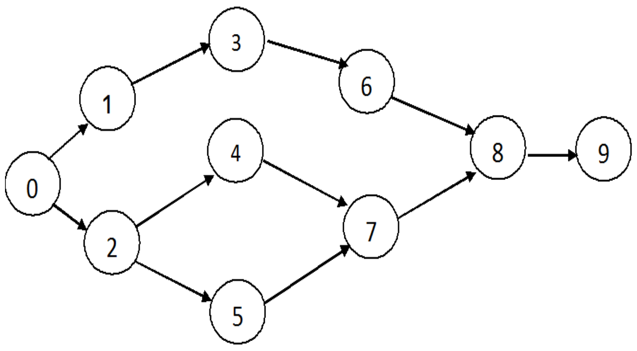


Figure.3 Example DAG workflow

i. The percentage of Drones and Workers need to be changed towards the end of solution. Towards the end, select more drones for quicker convergence to solution.
ii. New queen generation is identified by checking if the new generation has any schedule with <1% closer solution as initial queen
iii. The termination criteria is that, if one region couldn't give better solution continuously for N/5 iterations, stop proceeding with that region

4.4 Fitness function

The fitness of any possible solution is the major factor for selecting the solution. Fitness function for this problem is a combination of both objectives mentioned in section 3.3

$$Fitness = \sum_{i=1}^{2}(w_i Obj_i) \qquad (7)$$

Where Obj_1 and Obj_2 are Total cost and Makespan respectively and $0 \le w_i \le 1$.

4.5 Encoding procedure

The encoding procedure is used to map the cloud workflow as DAG. The problem encoding is done as follows. Sequence of job indices represents the Order of schedule. This can be like O_1, ...O_n where O_{i+1} will start execution only after the completion of O_i. The job_instance is an array of size 'n' where the ith element represents the instance of ith job. The instance_type is an array of size 'm' where the ith element has the instance type of ith instance. Fig. 3 is an example of DAG workflow.

Topological ordering using the following procedure:

i. Find the indegree of all vertices
ii. The first node to schedule is the node with indegree 0
iii. Remove that node and the edges corresponding to that and find the new node with indegree 0.
iv. Repeat the above step until all the nodes are ordered.

5. Memetic and bees algorithm operators

5.1 Crossover

The scheduling for jobs here is precedence constrained. The order should always be maintained in the dependencies between jobs. If J_k has to get the output of Ji, then Ji should precede Jk in all the possible orders generated. The crossover used here is Partially Matched Crossover(PMX) shown in Algorithm 1. Given two schedules 'A' and 'B' the PMX randomly picks two crossover points. This crossover point is used for the construction of next generation schedule. Here substring is a sub sequence of job in the given schedule referred with starting and ending position. The crossover is performed by considering the following facts.

i. Same job should not repeat.
ii. The dependencies among job are maintained.

The procedure for Crossover operation is given in Algorithm 1:

```
i.     Procedure Partially_Matched (Schedule A, Schedule B)
ii.    n is the number of jobs
iii.   Select a random value P1 between 0 and n-1
iv.    Select second random value P2 between 0 and n-1 where P1 ≠P2
v.     If P1 > P2, swap P1 and P2
vi.    From the Schedule A, copy he substring from position P1 to P2 as O1.
vii.   From the Schedule B, copy he substring from position P1 to P2 as O2.
viii.  For all allels(Jobs) in substring O1 and O2
ix.    If the allels in substring (A,0,P1)and substring (A,Pi+1,n-1) does not contain entries from substring (A,P1,P2) and substring (B,P1,P2), then replace substring O1 by O2,
x.     else repeat steps 3 to 9.
xi.    For string O1 and O2
xii.   Find the remaining allels in O1 and O2.
xiii.  Place them from left to right.
xiv.   Check if the dependencies among jobs are satisfied. If not repeat steps 3 to 12
```

Algorithm 1: Partially matched crossover

5.2 Mutation

The mutation is a memetic operator that maintains alteration in one or more gene values shown in algorithm 2. It is an occasional random alteration of a value in a schedule with small probability.

```
i.     Procedure Mutation( Schedule A)
ii.    n is the number of jobs
iii.   Select a random value P1 between 0 and n-2
iv.    Select a random value P1 between 1 and n-1 and P2 > P1
v.     Swap the allels in positions P1 and P2 and generate new schedule .
vi.    Check if dependency among jobs is maintained
vii.   If not repeat step 3 to 6
viii.  Check the fitness of new string
ix.    Repeat step 3 to 8 till a better fit solution is obtained or if the number of mutation done is 'n'
```

Algorithm 2: Mutation

5.3 Initial population

In any scheduling algorithm, the solution space is massive. The generation of initial population plays a major role in the convergence to the solution. For a problem size of '*n*' jobs, the initial population of '*n*' possible schedules are generated. Out of the '*n*' schedules, first four schedule are based on

i. Topological ordering
ii. Earliest Finish Time algorithm
iii. Shortest Job First
iv. Minimal Cost Ordering algorithm

The remaining *n-4* schedules are generated on random ordering. Any schedule generated will be considered only if the dependency is maintained. This is depicted in algorithm 3.

The four algorithms included in generation of initial population helps in quicker convergence towards the optimal solution. For every schedule generated, the instance-type and job to instance are mapped accordingly.

```
i.     Procedure initial population
ii.    n is the number of jobs
iii.   m is the number of instance types
iv.    Generate the first schedule by topological ordering based on indegree of any vertex.
v.     Generate the second schedule by heterogeneous earliest finish time algorithm.
vi.    Generate the third schedule by Shortest job first algorithm
vii.   Generate the fourth schedule based on the descending order of cost .
viii.  For i=1 to n-4
ix.    Generate a random schedule 'I' so that the dependency is maintained.
x.     Find the fitness of each schedule. The best fit is the queen. The next N/2 -1 fit schedules are Drones. The remaining N/2 schedules are workers.
xi.    End procedure
```

Algorithm 3: Initial population

5.4 Complexity analysis

The complexity of memetic operators crossover and mutation are $O(n^2)$ and O(n) respectively for n jobs. The checking of dependency and fitness is O(n). The time complexity of evaluation and generation each generation is $O(n^2)$. For a graph of 'n' vertices, a maximum of n^2 edges exist. For 'g' iterations, the overall complexity is $O(gn^2)$. Excluding the evolution, four schedules in initial population are generated based on algorithms. The complexity of these also needs to be included. The HEFT, Topological ordering, SJF and MCO have $O(n^2)$ complexity each. The overall complexity is $O(4n^2)$ + $O(gn^2)$. Which is $O(gn^2)$ in general.

6. Testing

6.1 Experimental workflow

The MBOA algorithm is assessed using standard workflows defined in Pegasus workflow management systems[13]. The workflow models used are Montage, Epigenomics, Cybershake, Sipht and Inspiral. [14]. It is shown in Fig. 4

6.2 Algorithms compared

MBOA is compared with two algorithms namely Particle Swarm optimization(PSO)[15] and Genetic algorithm(GA) [16].

PSO is an evolutionary algorithm based on particles (ie.,bird or fish) [17]. The particles have the ability to move around the solution space and reach to a solution [18].

GA is a meta-heuristic algorithm based on evolutionary ideas of natural selection and genetics.

[19]It provides solution to optimization problems with its initialization, selection, and generic operators[20].

6.3 Experimental setup

The algorithm is implemented in Cloudsim3.0. We assume only six types of instances taken from Amazon as sample. The processing time for each job is estimated based on their processing capacity and the instance.

The average bandwidth to transfer intermediate data between instances is comparable to the average bandwidth provided by Amazon elastic block store (20kbps). Our experiment has considered a billing period of 10 minutes and the acquisition delay is estimated as 1 minute.

We have considered six sizes of scientific workflow in our experiments small (approx. 20 Jobs), medium (approx. 50 Jobs), large (approx. 100 Jobs), xlarge(approx. 200 Jobs), 2xlarge(approx. 500 Jobs) and 4xlarge(approx.1000 Jobs). We have conducted each experiment 10 times.

To evaluate the performance of the MBOA, we need the deadline of each workflow. The deadline is obtained from any of the following

i. The user
ii. By assigning all jobs sequentially to the fastest instance and calculate minimum execution time of the workflow.

The deadline is also recorded as the Fastest time. Assigning the jobs sequentially to the slowest instance gives the Slowest Time. Testing was done using four different deadline intervals of 1,2,3 and 4. Deadline interval is calculated by finding the difference between fastest time and the slowest time and then dividing it by five. Say, if the slowest time is 50 ns and the fastest time is 100 ns, the difference is 50 ns. Here the interval is 10ns. The deadline interval 1 is 60ns, deadline interval 2 is 70 ns and so on.

The IaaS parameters used for different virtual machines are mentioned in the table 1 (source *www.aws.amazon.com*). Here L, M and represents the Low, Moderate and High in Network Preference.

7. Results and analysis

Graphs are plotted to show the deadlines met. Comparison graph is generated based on the percentage of fitness given by each algorithm.

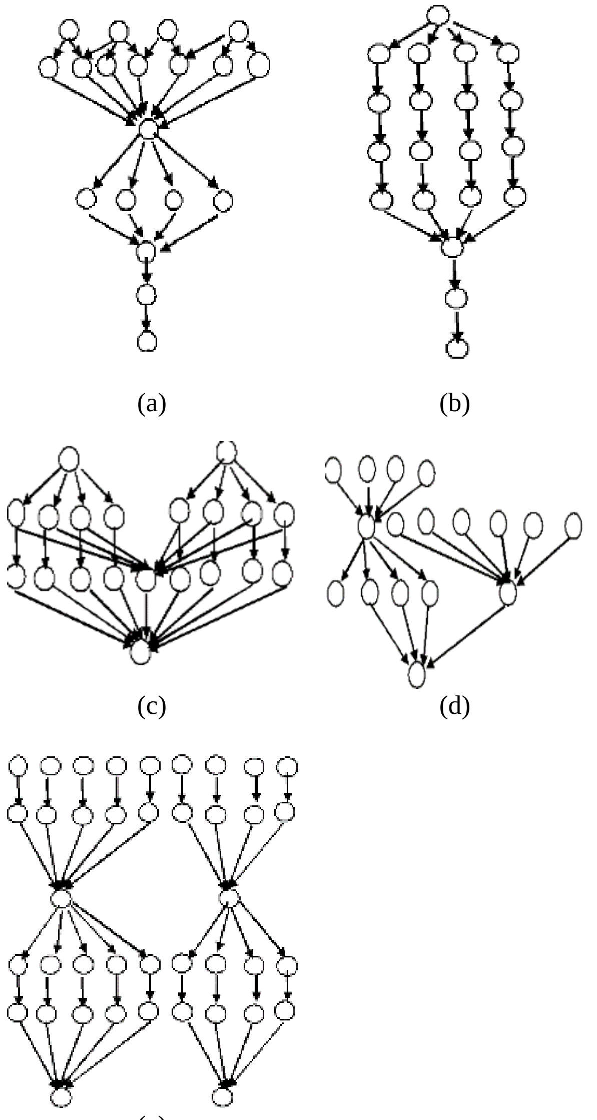

(a) (b) (c) (d) (e)

Figure.4 Workflow models: (a) Montage, (b) Epigenomics, (c) Cybershake, (d) Sipht, and (e) Inspiral

Table 1. IaaS parameters

Inst type	CPU	Memory (GiB)	PIOP Opzd	N/W Pref	Price
Db.m1. small	1	1.7	-	L	\$0.044
Db.m1. medium	1	3.75	-	M	\$0.087
Db.m1. large	2	7.5	Yes	M	\$0.175
Db.m1. xlarge	4	15	Yes	H	\$0.350
Db.m2. 2xlarge	4	34.2	Yes	M	\$0.420
Db.m2. 4xlarge	8	68.4	Yes	H	\$0.840

The fitness is compared among three algorithms namely

- Genetic Algorithm(GA)
- Particle swarm Optimization(PSO)
- MBOA

For the testing purpose it is assumed with 0.5 for w_i in Eq. (7).

In case of the Montage workflow, MBOA gave a better result than GA and PSO. The GA gave better result for deadline interval 3 and 4. MBOA achieved 100% result in deadline interval 3 and 4. It gave 92% and 96% result in deadline interval 1 and 2 respectively. Comparing the results for Epigenomics, it is inferred that MBOA gives a better result than the other two algorithms.

In the Cybershake workflow model, it is found that all the three algorithms showed a decently better performance in deadline 1 and 2. MBO gave 100% performance for deadline 3 and deadline 4.

While comparing the performance in the Sipht workflow, PSO gave a better performance than GA. But MBOA outperformed both.

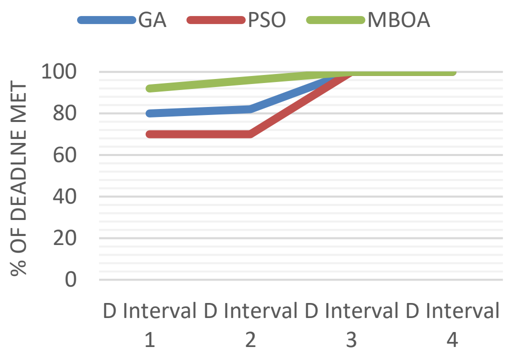
In the Inspiral workflow model, MBOA showed the best of 92 % for interval 1 & 2 and 100% for interval 3 &4. GA came closer to MBOA having 80% accuracy in interval 1 & 2 and 100% for interval 3 & 4.

Comparing the values and graph in Fig. 5, the other two algorithms totally outperform GA. Out of the other two algorithms MBOA has proven better than PSO.

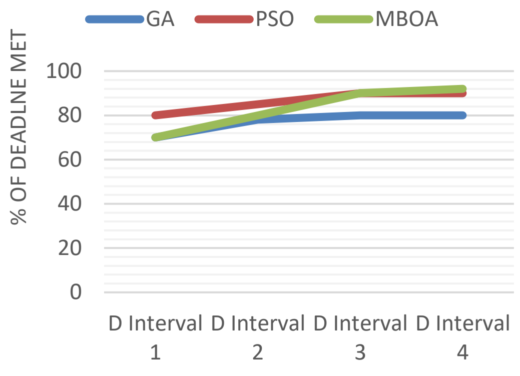
8. Further analysis

The performance was analysed for different intances. Inference says that the solution was generated with lower makespan most of the time. As future work, analysis can be done with hybrid algorithms of ACO or Modified PSO algorithm with genetic algorithm An important point is the impact on the selection of initial population. The initial four algorithms selected has a major difference in time for the convergence to the solution. It helps in faster convergence towards solution.

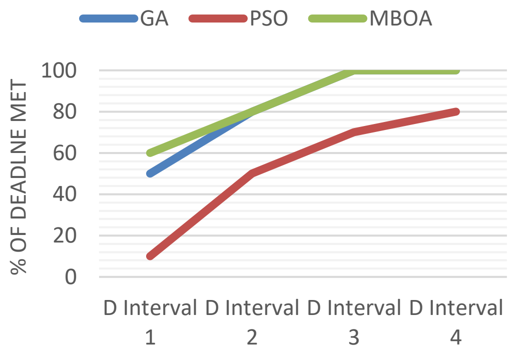
When the algorithm was checked with the input condition of one VM of every type in each job. It was found that the algorithm takes higher execution time for longer jobs when compared to shorter jobs. When seeing the computational complexity, MBOA algorithm is better than the other algorithms like PSO.



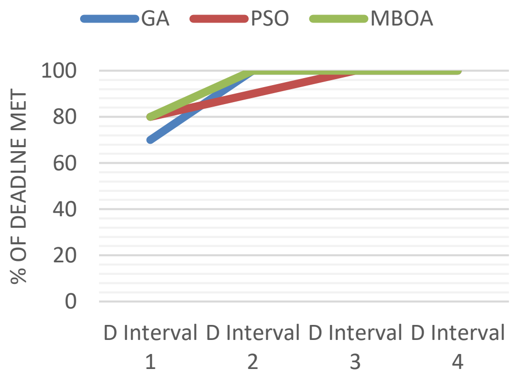
(a)



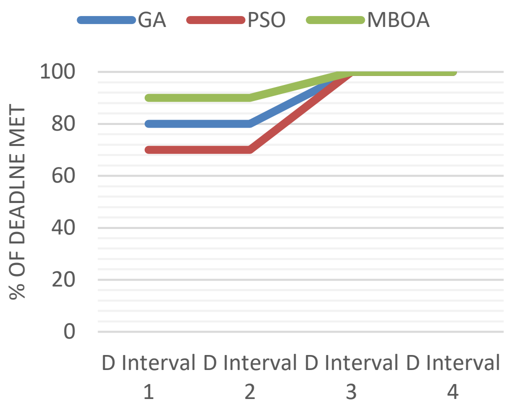
(b)



(c)



(d)



(e)

Figure.5 Deadline for real-world workflows: (a) Montage, (b) Epigenomics, (c) Cybershake, (d) Sipht, and (e) Inspiral

9. Conclusion and future work

Many scheduling algorithms are available for multiprocessor architectures in cloud environment. But most of these have difficulty when directly applied in cloud. The algorithm developed overcomes the issues in the real-world cloud computing models.

Cloud computing provides high performance computing resources on demand for solving large scale scientific problems. To execute the large scale scientific application, cloud provider needs to have appropriate provisioning algorithm which helps the cloud provider in minimizing the cost and makespan. Towards this, MBOA is proposed.

For giving a solution to the multi-objective cloud scheduling problem, an encoding scheme that represents the scheduling criteria having the different instances of jobs and their types are modelled. The memetic operators like evaluation function, crossover and mutation are used along with the bees algorithm. The experiment is checked with the actual pricing model in Amazon EC2 and the results are promising.

The future work can be a hybrid algorithm of PSO or any other optimization algorithm using more than one pricing scheme. The issues like termination delay of virtual Machines can be considered. Further work can be done in execution of workflow deployed in different regions and the data transfer cost among different data centre can be considered.

References

[1] J. Yu, M. Kirley, and R. Buyya, “Multiobjective planning for workflow execution on grids”, In: *Proc. of the 8th IEEE/ ACM International conference on Grid computing*, pp 11-17, 2007.

[2] W. N. Chen and J. Zhang, “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] F. Zhang, J. Cao, K.Hwang , C. Wu,” Ordinal optimized scheduling of scientific workflows in elastic compute *clouds*”, In: *Proc. of the 3rd IEEE International conference on cloud computing technology and science*, pp. 9-17, 2011.

[4] S. Ghanbari, M. Othman, “A priority basedJob scheduling *algorithm* in cloud computing”, In: *Proc. of the International conference on Advances science and contemporary engineering*, pp 778-785, 2012.

[5] L. F. Bittencourt, E. R. M. Madeira, and Nelson L. S. da Fonseca, *University* of Campinas “Scheduling in Hybrid Clouds Cloud computing: Networking and communications challenges”, *IEEE communications Magazine*, September, pp.42-47, 2012.

[6] R. Sakellaiou, H. Zaho, and E. Tsiakkouri, M.Dikaiakos, “Scheduling workflows with budget constraints”, *Integrated research in GRID computing*, pp 189-202, 2007.

[7] X. Zuo, “Self adaptive learning PSO – Based deadline constrained task scheduling for Hybrid IaaS cloud”, *IEEE Transactions on Automation Science and Engineerin*”, Vol. 11, No. 2, April 2014.

[8] R. Garg and A.K. Singh, “Multiobjective workflow grid scheduling based on discrete particle swarn optimization”, *Swarn, Evolutionary and Memetic computing*, pp 183-190, 2011.

[9] F. Kang and J.J. Li, “Artificial Bee Colony Algorithm Optimized S upport Vector Regression for System Reliability Analysis of Slopes“, *Journal of Computing in Civil Engineering*, Vol. 30, No. 3, pp.1-13, 2016.

[10] K.F. Kang, J.S. Li, and J.J Li, “System reliability analysis of slopes using least squares support vector machines with particle swarm optimization“, *Neurocomputing*, Vol. 209, No. C, pp 46-56, 2016.

[11] J. Durillo and R. Pordan, “Multiobjective workflow scheduling in Amazon EC2”, *Cluster computing*, Vol.17, No.2, pp.169 – 189, 2014.

[12] S.S. Bitam, “Bees life algorithm for job scheduling in cloud computing” *International conference on communications and information technology*, pp186-191, 2012.

[13] S. Abrishami, M. Naghibzadeh, and D. Epema, "Deadline constrained workflow scheduling algorithms for IaaS clouds", *Future generation computer systems*, Vol.23, No.8, pp 1400-1414, 2012.

[14] S.Bharathi, A. Chervanak, E. Deelman, and G.Mehta, "Characterization of scientific workflows", In: *Proc. of the 3rd workshop on workflows in support of large scale*, pp 1-10, 2008.

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

[16] Z. Zhu and G. Zang, "Evolutionary multi-objective workflow scheduling in cloud", *Transactions on parallel and distributed systems*, pp.1-14, 2015.

[17] M.M.A. Rodriguez and R. Buyya, "Deadline based resource provisioning and scheduling algorithm for scientific workflows on cloud", *IEEE Transactions on cloud computing*, Vol. 2, No. 2, pp.222-235, 2014.

[18] A.A.I. Awada, N.A. El-Hefnawyb, and H.M.A. kaderc, "Enhanced Particle Swarm Optimization For Task Scheduling In Cloud Computing Environments", In: *Proc. of the International Conference on Communication, Management and Information Technology*, pp 920-929, 2015.

[19] O.O. Udomkasemsub, L. Xiaorong, and T. Achalkul, "A multiple objective workflow scheduling framework for cloud data analytics", In: *Proc. of the 24th IEEE International Joint Conference in Computing Sciences and Software Engineering* , pp 391-398, 2012.

[20] Y.Y. Ge and G. Wei, "GA based task scheduler for the cloud computing systems", In: *Proc. of the International Conference on Web Information Systems and Mining*, pp 181-186, 2010.