ISSN 2277-3061

Pso Optimization algorithm for Task Scheduling on The Cloud Computing Environment

[1]Zahraa Tarek, [2]Magdy Zakria, [3]Fatma A. Omara
[1] Computer Science Department, Mansoura University, Egypt
[2] Computer Science Department, Mansoura University, Egypt
[3] Computer Science Department,Cairo University,Egypt

ABSTRACT

The Cloud computing is a most recent computing paradigm where IT services are provided and delivered over the Internet on demand. The Scheduling problem for cloud computing environment has a lot of awareness as the applications tasks could be mapped to the available resources to achieve better results.

One of the main existed algorithms of task scheduling on the available resources on the cloud environment is based on the Particle Swarm Optimization (PSO). According to this PSO algorithm, the application's tasks are allocated to the available resources to minimize the computation cost only.

In this paper, a modified PSO algorithm has been introduced and implemented for solving task scheduling problem in the cloud. The main idea of the modified PSO is that the tasks are allocated on the available resources to minimize the execution time in addition to the computation cost. This modified PSO algorithm is called Modified Particle Swarm Optimization (MPOS).The MPOS evaluations have been illustrated using different time, and cost parameters and their effects in the performance measures such as utilization, speedup, and efficiency. According to the implementation results, it is found that the modified MPOS algorithm outperforms the existed PSO.

Indexing terms/Keywords

Cloud computing;Task scheduling; Particle swarm optimization;Directed a cyclic graph.

Academic Discipline And Sub-Disciplines

Computer Science.

SUBJECT CLASSIFICATION

Distributed Computing Classification.

TYPE (METHOD/APPROACH)

Particle swarm optimization (PSO) algorithm, Task scheduling.

Council for Innovative Research

Peer Review Research Publishing System

Journal: INTERNATION JOURNAL OF COMPUTERS AND TECHNOLOGY

Vol. 13, No. 9

editorijctonline@gmail.com

www.ijctonline.com, www.cirworld.com

1. INTRODUCTION

The Cloud computing allows the users to use any computational resources and services of data centers (i.e., machines, network, storage, operating systems, application development environments, application programs) over the network to deploy and evaluate their applications [1]. This means that the cloud computing provides self-service provisioning, which is considered an important feature in the cloud computing[2]. On the other hand, the cloud computing services are divided into three layers; SaaS (Software as a Services), PaaS (Platform as a Services), and IaaS (Infrastructure as a Services) [3]. The Cloud computing architecture is categorized as layers, service model, and deployment model (types). By this classification, the users can easily choose the suitable cloud services and types to fit their business according to these services [1, 3].

Now a day, it is important for services (resources, applications...) to be accessed through the cloud environment because of the cloud computing benefits, such as saving cost and service availability at any time. On the other hands, Cloud computing has been emerged as a commercial reality in the field of information technology but the technology is still not fully developed [4]. There are still some topics that are needed to be focused on, as Resource management and Task scheduling.

The work in this paper is concerned with the task scheduling problem to minimize the computation cost and the total execution time of the applications using the provided resources by the Cloud service providers, such as Amazon and GoGrid3. We have achieved these features by introducing a modified Particle Swarm Optimization (MPSO) algorithm.

Particle Swarm Optimization (PSO) is a swarm-based intelligence algorithm influenced by the social behavior of animals, which is introduced by Kennedy and Eberhart [5]. Each particle has position and velocity. The position of particle at any instance of time is influenced by its personal best position (pbest) and the position of the best particle in global problem space (gbest). The performance of a particle is measured by a fitness value which is based on the problem specification.

Scheduling is the method by which threads, processes, tasks or data flows are given access (mapped) to system resources (e.g. processor time, communications bandwidth, utilization of the system) according to the users requirements [6]. A good scheduling algorithm is important as the requirement for most modern systems arises to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously). [7, 8]

The rest of the paper is organized as: Section 2 provides conducted studies. task-resource scheduling problem formulation is discussed in Section 3. The principle of the proposed task scheduling algorithm (MPOS) is discussed in Section 4. Section 5 provides the contribution of our proposed algorithm. Finally, Section 6,7 represents conclusions and future propositions.

2. CONDUCTED STUDIES

Yun Yang, Ke Liu, and Jinjun Chen [9] have proposed an Innovative transaction intensive cost-constraint scheduling algorithm which considers the cost and time. The simulation results have demonstrated that this algorithm can achieve lower cost than others while meeting the user designated deadline.

Suraj Pandey et al. [10] have proposed a heuristic task scheduling which optimizes the cost of task-resource mapping based on the solution of using particle swarm optimization (PSO) technique. PSO based mapping algorithm has much lower cost as compared to another algorithm called BRS (Best Resource Selection) based mapping. Their results show that PSO can achieve: a) as much as 3 times cost savings as compared to BRS, and b) good distribution of workload onto resources.

Ke Liu et al. [11] have presented a novel compromised-time-cost (CTC) scheduling algorithm. The CTC algorithm considers the characteristics of cloud computing to accommodate instance-intensive cost-constrained workflows by compromising the execution time and cost which are user input enabled on the fly. The simulation results has demonstrated that the CTC algorithm can achieve lower cost while meeting the user-designated deadline or reducing the mean execution time within the user- designated execution cost.

Saeed Parsa and Reza Entezari-Maleki [11] have proposed a new task scheduling algorithm called Resource-Aware-Scheduling algorithm (RASA). It is composed of two traditional scheduling algorithms; Max-min and Min-min. The main feature of the RASA algorithm is that it amalgamates the advantages of Max-min and Min-min algorithms and alleviates their disadvantages. Though, the deadline of each task, arriving rate of the tasks, cost of the task execution on each of the resource, and cost of the communication are not considered .The experimental results show that RASA algorithm outperforms the existing scheduling algorithms in large scale distributed systems.

J.Huang [12] has proposed workflow task scheduling algorithm based on the genetic algorithms (GA) model in the cloud computing environment which can fulfill the goals of the workflow task scheduling. They proved that the proposed algorithm's performance has improved perfectly analysis from algebra and the population size under the different settings, improved the efficiency of task scheduling, which can maximum satisfy the QoS (Quality Of Service) requirements of the users.

Lei Zhang et al. [13] have proposed a PSO algorithm. This proposed algorithm is similar to the genetic algorithms (GA). The aim of this algorithm is how to improve the efficiency of resource allocation and how to minimize the completion time simultaneously. It is noted that the performance of PSO usually spent shorter time to accomplish the various scheduling tasks and specifies better result comparing to the GA algorithm. Also, they have proved that the PSO algorithm can get better effect for a large scale optimization problem.

Cui Lin, and Shiyong Lu [14] have proposed an Scalable Heterogeneous Earliest-Finish-Time Algorithm (SHEFT) workflow scheduling algorithm to schedule a workflow elastically on a Cloud computing environment. The experimental results show that SHEFT is not only outperform several representative workflow scheduling algorithms in optimizing workflow execution time, but also enable resources to scale elastically at runtime.

Visalakshi and Sivanandam [15] have presented Hybrid Particle Swarm Optimization (HPSO) method for solving the Task Assignment Problem (TAP). The algorithm has been developed to dynamically schedule heterogeneous tasks on to heterogeneous processors in a distributed setup. The HPSO yields a better result than the Normal PSO when applied to the task assignment problem. The results Of PSO and HPSO is also compared with another popular heuristic optimization technique namely Genetic Algorithm (GA). The results infer that the PSO performs better than the GA.

S.Selvarani, and G.Sudha Sadhasivam [16] have proposed an improved cost-based scheduling algorithm for making efficient mapping of tasks to available resources in the cloud. The improvisation of traditional activity based costing is proposed by new task scheduling strategy for cloud environment where there may be no relation between the overhead application base and the way that different tasks cause overhead cost of resources in the cloud. This scheduling algorithm divides all user tasks depending on priority of each task into three different lists. This scheduling algorithm measures both resource cost and computation performance, it also Improves the computation/communication ratio.

Yang et al. [17] have highlighted the issue of job scheduling in cloud computing. They argued that there is no well-defined job scheduling algorithm for the cloud that considers the system state in the future .The existing job scheduling algorithms under utility computing paradigm do not take hardware/software failure and recovery in the cloud into account. To tackle this issue, they have proposed a Reinforcement Learning (RL (based algorithm that helps the scheduler to define scheduling decision with fault tolerable while maximizing utilities attained in the long term.

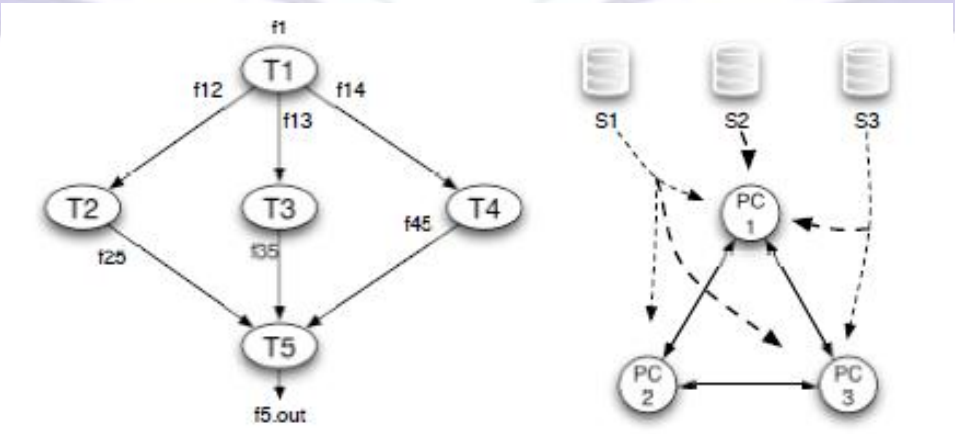
3. TASK-RESOURCE SCHEDULING PROBLEM FORMULATION

According to the task scheduling problem, the application is represented as a Directed Acyclic Graph (DAG) where nodes (or tasks) represent the needed computation and edges represent the communication between tasks. For each node in the DAG, a weight is assigned corresponding to computation cost, and weights for edges are assigned corresponding to communication cost between nodes [18].

A Directed Acyclic Graph (DAG) is represented by G= (V, E), where V = {T1, ...,Tn} is the set of tasks, and E represents the data dependencies between these tasks, whereFj,k =(Tj, Tk) ∈ E means thatthe data produced by Tj and consumed by Tk (see Fig. 1(a)) [10].

By considering a set of storage sites S = {1..., i}, a set of compute sites PC = {1, ..., j}, and a set of tasks T = {1, ..., k}. The 'average' computation time of a task Tk on a compute resource PCj for a certain size of input is considered known. Then, the cost of computation of a task on a compute host is inversely proportional to the time it takes for computation on that resource. Also, it is assumed that the cost of unit data access di,j from a resource i to a resource j is known. The access cost is fixed by the service provider (e.g. Amazon CloudFront). The transfer cost can be calculated according to the bandwidth between the sender and receiver sites. However, the cost for transferring unit data between sites, per second is one of task scheduling issues which will be considered. These costs are non-negative, symmetric, and satisfy the triangle inequality; that is, di,j = dj,i for all i, j ∈ N, and di,j + dj,k≥di,k for all i, j, k ∈ N (see Fig. 1 (b))[10].

By considering an application DAG with a set of tasks T = {1, ..., k}, a set of storage sites S = {1, ..., i}, and a set of compute sites PC = {1, ..., j}, the problem can be stated as: *"Find a task-resource mapping instance* M*, such that estimating the total cost and the total time for each compute resource* PCj*, the highest cost and also highest time among all the compute resources is minimized and load balance is achieved."*[10].



(a) **(b)**

Fig.1: DAG of an application's Tasks and Computing Resources

Cost Minimization Problem

The goal is to assign the tasks to the available compute resources to minimizing the total cost of computation and total time of completion of an application. The cost is minimized such that it completes within the time (deadline) a user specifies. The cost is determined using the following equations [10]:

$$C_{exe}(M)_j = \sum_k Wkj \qquad \forall M(K) = j \ldots\ldots\ldots \quad (1)$$

$$C_{tx}(M)_j = \sum_{k1 \in k} \sum_{k2 \in k} dM(k1), M(k2) ek1, k2 \ldots \quad (2)$$
$$\forall M(K1) = j \text{ and } \forall M(K2) \neq j$$

$$C_{total}(M)_j = C_{exe}(M)_j + C_{tx}(M)_j \ldots\ldots\ldots\ldots\ldots\ldots \quad (3)$$

$$Cost(M) = \max(C_{total}(M)_j) \qquad \forall j \in P \ldots\ldots\ldots \quad (4)$$

$$\text{Minimize } (Cost(M) \forall M) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad (5)$$

$C_{exe}(M)_j$ denoted to the total cost of all the tasks assigned to a compute resource PC_j(Eq. 1). This value is computed by adding all the node weights (the cost of execution of a task k on compute resource j) of all tasks assigned to each resource in the mapping M.

$C_{tx}(M)_j$ is considered as the total access cost (including transfer cost) between tasks assigned to a compute resource PC_j and those that are not assigned to that resource in the mapping M (Eq. 2). This value is the product of the output file size (given by the edge $weight_{ek1,k2}$) from a task k1 ∈ k to task k2 ∈ k and the cost of communication from the resource where k1 is mapped (M(k1)) to another resource where k2 is mapped (M(k2)).The average cost of communication of unit data between two resources is given by $d_{M(k1),M(k2)}$. The cost of communication is applicable only when two tasks have file dependency between them, that is when ek1, k2> 0. For two or more tasks executing on the same resource, the communication cost is zero.

For a given assignment M, the total cost $C_{total}(M)_j$ for a compute resource PC_j is the sum of execution cost and transfer cost (Eq. 3). Then, the total cost for all the assignments will be dominated by the highest cost of a compute resource (Eq. 4) ensures that all the tasks are not mapped to a single compute resource. Hence, the goal of the assignment is to minimize this cost (Eq. 5).

Time Minimization Problem

According to our modified MPOS algorithm, the total time of task execution will be introduced as another parameter should be minimized beside the cost. The goal of this modification is to assign the tasks to the compute resources such that the time of computation is minimized. The time is determined using the following equation [19]:

$$F(x) = \max 1 \leq u \leq \eth\, m \{ \text{Completion time of } (p_u) \} \ldots\ldots\ldots\ldots\ldots \quad (6)$$

Where completion time $(p_u) = \min_{ti \in I} T(t_i, p_u)$

Where I is the set of tasks assigned to P_u

Equation (6) is interpreted in our modified algorithm by the following equations:

$$C_{data}(M)_j = \sum_{k1 \in k} \sum_{k2 \in k} dM(k1), M(k2) ek1, k2 \ldots + C_{FT}(M)_j \quad (7)$$
$$\forall M(K1) = j \text{ and } \forall M(K2) \neq j$$

$$C_{ST}(M)_j = \max (C_{data}(M)_j, C_{FT}(M)_j) \ldots\ldots\ldots\ldots\ldots \quad (8)$$

$$C_{FT}(M)_j = C_{st}(M)_j + C_{exe}(M)_j \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (9)$$

$$Time(M) = \max(C_{FT}(M)_j) \qquad \forall j \in P \ldots\ldots\ldots \quad (10)$$

$$\text{Minimize } (Time(M) \quad \forall M) \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (11)$$

According to equations (7-11), the dependency between each task and previous tasks is checked. If there is dependency between the tasks, so the cost of communication of unit data between two resources is given by $d_{M(k1),M(k2)}$ which is applicable only when two tasks have file dependency between them, that is when ek1, k2> 0. For two or more tasks executing on the same resource, the communication cost is zero. The value from the product of communication cost and files dependency is calculated then this result value is added to finish time of previous tasks in resource j to get $C_{data}(M)_j$ (Eq. 7).

$C_{ST}(M)_j$ is the start time of all tasks on resource j which is calculated as the max between $C_{data}(M)_j$ and finish time of the previous tasks in the same resource j (Eq. 8).

$C_{FT}(M)_j$ is the finish time of tasks on resource j which is calculated as the addition between $C_{exe}(M)_j$ in (Eq. 1) and $C_{st}(M)_j$. (Eq. 9) and this is the total time.

Then, the total time for all the assignments will be dominated by the highest time of a compute resource (Eq. 10) ensures that all the tasks are not mapped to a single compute resource. Hence, the goal of the assignment is to minimize this time (Eq. 11).

4. THE MODIFIED PARTICLE SWARM OPTIMIZATION SCHEDULING ALGORITHM

According to PSO, the population is set of particles in a problem space. Particles are initialized randomly; each particle will have a fitness value, this value evaluated by a fitness function to be optimized in each generation. Each particle knows its best position pbest and the best position so far among the entire group of particles gbest, the pbest of a particle is the best result (fitness value) so far reached by the particle, whereas gbest is the best particle in terms of fitness in an the all population. The evaluation is carried out in a loop until the results converge or until number of iterations (user specified stopping criteria) [13, 20].

The particle will have velocity, which directs the flying of the particle. Each iteration, the velocity and the position of particles will be updated as follows [20]:

$$V_i^{k+1} = W\, V_i^k + c_1\, rand_1 * (pbest^k{}_i - X_i^k) + c_2\, rand_2 * (gbest^k - X_i^k) \quad (12)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1} \quad (13)$$

As W, c1 and c2, are positive constants which represent the weight of previous velocity, the weight of the acceleration terms that pull each particle toward Pbest and gbest, respectively [20].

Where

$\mathbf{V_i^k}$	velocity of particle i at iteration k
$\mathbf{V_i^{k+1}}$	velocity of particle i at iteration k + 1
W	inertia weight between 0.9 to 0.1
$\mathbf{c_j}$	positive acceleration coefficients; j = 1, 2
$\mathbf{rand_i}$	random number between 0 and 1; i = 1, 2
$\mathbf{X_i^k}$	current position of particle i at iteration k
$\mathbf{pbest_i}$	best position of particle i
gbest	position of best particle in a population
$\mathbf{X_i^{k+1}}$	position of the particle i at iteration k + 1

The existed PSO task scheduling algorithm provides a mapping of all tasks to a set of given resources based on the model described in the following two algorithms [10].

Algorithm 1: Scheduling heuristic.

1. Calculate average computation cost of all tasks to all resources
2. Calculate average access cost (communication cost/ data size)
3. Compute PSO algorithm
4. For all ready tasks do
5. Assign tasks to resources according to PSO
6. If resource(processor) exceed limit determined for number of tasks
7. Map task to next resource with minimum cost
8. End if
9. End for
10. Update ready tasks list
11. Update communication cost between resources according to current network load
12. Compute PSO
13. Until there are unscheduled tasks

Algorithm 2: PSO algorithm.

1.Set particle dimension as equal to the size of ready tasksin {ti} # T

2. Initialize particles position randomly from PC =1, ..., j and velocity vi randomly.

3. For each particle, calculate its fitness value with respect to Cost Minimization presented by eqn. 5.

4. If the fitness value is better than the previous best pbest,set the current fitness value as the new pbest.

5. After Steps 3 and 4 for all particles, select the best particle as gbest.

6. For all particles, calculate velocity using Equation 12and update their positions using Equation 13.

7. If the stopping criteria or maximum iteration is not satisfied, repeat from Step3.

The existed PSO algorithm is modified by using two fitness functions instead of one. The first fitness function is to minimize the cost (as in the existed algorithm by equ.5), the other one is to minimize the compilation time which is presented by equ.11. Actually, these two fitness functions are implemented using different combinations (i.e., AND, sequence, and Best-To-Best operations in algorithm 2 of the existed PSO scheduling algorithm). This has been implemented by replacing step 3 in algorithm 2 using one of the following five combinations:

- ***(a)*** For each particle, calculate its fitness value with respect to ***Cost Minimization presented using eqn. 5 THEN Time Minimization presented using eqn.11.***
- ***(b)*** For each particle, calculate its fitness value with respect to ***Time Minimization presented using eqn.11 THEN Cost Minimization presented using eqn. 5.***
- ***(c)*** For each particle, calculate its fitness value with respect to ***Cost Minimization presented by eqn. 5 AND Time Minimization presented by eqn.11.***
- ***(d)*** For each particle, calculate its fitness value with respect to **Best *Cost Minimization presented by eqn.5 TO Best Time Minimization presented by eqn.11.***
- ***(e)*** For each particle, calculate its fitness value with respect to ***Best Time Minimization presented by eqn.11 TO Best Cost Minimization presented by eqn. 5.***

These combinations have been implemented one after another to justify which combination will produce good results.

5. SIMULATION AND ANALYSIS OF RESULTS

In this section, the metric of the experiment setup, comparison, and results are presented.

5.1 Experimental Environment

The modified task scheduling algorithm has been written by java programming language using eclipse program in Intel(R) Core(TM)2 Duo CPU in 1.60GHZ of processor and 2.50 GB of RAM . The experimental setup of the PSO algorithm considers that the iterations =20, and the number of execution = 30.

5.2 Experimental Results

Three matrices are used to store the results for:

a) Average computation cost of each task on each resource (TP-matrix),

b) Average communication cost per unit data between compute resources (PP-matrix), and

c) Input input/ output Data Size of each task (DS-matrix).

The values for PP-matrix resemble the cost of unit data transfer between resources given by Amazon Cloud Front [21] .It is assumed that PC1 to be in US, PC2 in Hong Kong (HK) and PC3 in Japan (JP), respectively. The PP-matrix's values could be proposed randomly for every repeated experiment, but these values are kept constant during our MPSO task scheduling implementation. While, the values for TP-matrix are given by the Amazon EC2's pricing policy for different classes of virtual machine instances is used [22]. Each task has its own Data Size matrix (DS). The sum of all the values in the DS matrix varies according to the size of data (e.g., 64-1024 MB). According to Figure 1(a), if x is the output data size of task T1, then tasks T2, T3, and T4 receive x data as input and produce x data as output. Finally, task T 5 consumes 3x data and produces 6x data. These matrices are depicted in Table 1 [10].

$$TP[5\times3]= \begin{array}{c} \\ T1 \\ T2 \\ T3 \\ T4 \\ T5 \end{array} \begin{array}{c} \begin{array}{ccc} PC1 & PC2 & PC3 \end{array} \\ \begin{bmatrix} 1.23 & 1.12 & 1.15 \\ 1.17 & 1.17 & 1.28 \\ 1.13 & 1.11 & 1.11 \\ 1.26 & 1.12 & 1.14 \\ 1.19 & 1.14 & 1.22 \end{bmatrix} \end{array}$$

TP [i, j] = Cost of execution of Ti at PCj

(EC2 price of resources for High CPU instance)

(Example matrix values are in the range $1.1 − $1.28/hr)

$$PP\ [3\times 3]= \begin{array}{c} \\ PC1 \\ PC2 \\ PC3 \end{array} \begin{array}{c} \begin{array}{ccc} PC1 & PC2 & PC3 \end{array} \\ \begin{bmatrix} 0 & 0.17 & 0.21 \\ 0.17 & 0 & 0.22 \\ 0.21 & 0.22 & 0 \end{bmatrix} \end{array}$$

PP[i, j] = Cost of communication between PCi &PCj

(Values in $/MB/second)

$$DS_{T2,T3,T4}[2\times 2]= \begin{array}{c} \\ i/p \\ o/p \end{array} \begin{array}{c} \begin{array}{cc} f1 & f2 \end{array} \\ \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix} \end{array}$$

$$DS_{T5}\ [2\times 2] = \begin{array}{c} \\ i/p \\ o/p \end{array} \begin{bmatrix} 15 & 30 \\ 15 & 30 \end{bmatrix}$$

row1 = i/p file sizes, row2 = o/p file sizes

Table 1: TP, PP, and DS matrices and their values

Figures (2-6) represent the experimental results of our MPOS algorithm by considering five combinations of the time and cost fitness functions which are defined from (a) to (e).

By applying the time and cost fitness functions according to combination (a) for our MPOS algorithms, the results are depicted in Figure 2, and the performance parameters are presented in Table 2.

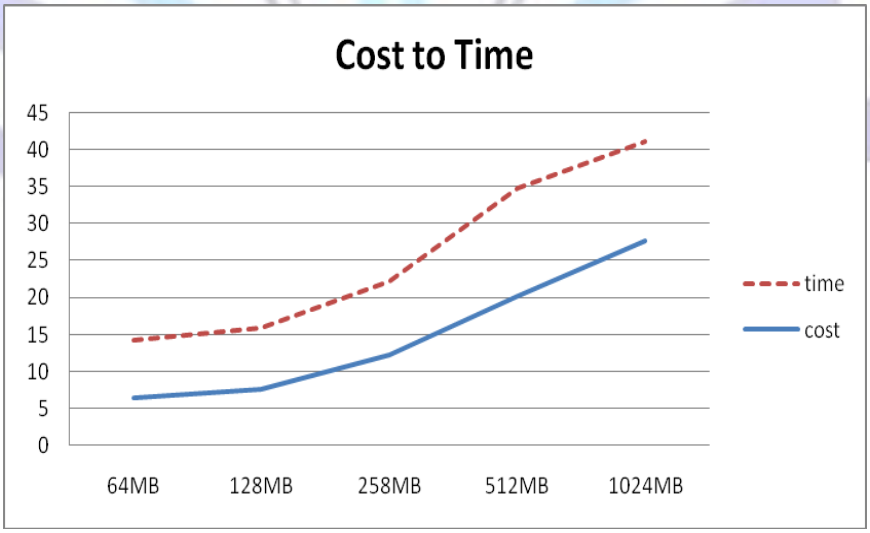


Fig.2: Results when cost first then time

Measure \ Data size	64	128	256	512	1024
utilization	0.418838	0.423256	0.433781	0.407002	0.436762
speedup	0.64505	0.599928	0.502294	0.343816	0.370005
efficiency	0.215017	0.199976	0.167431	0.114605	0.123335

Table2.performance parameters for each data size

According to the results in Figure 2, it is found that by applying the cost fitness function then time fitness function, the average cost=14.86 and the average time=10.82.

By applying the time function and then cost fitness function according to combination (b) for our MPOS algorithm, the results are depicted in Figure3, and the performance parameters are presented in Table 3.

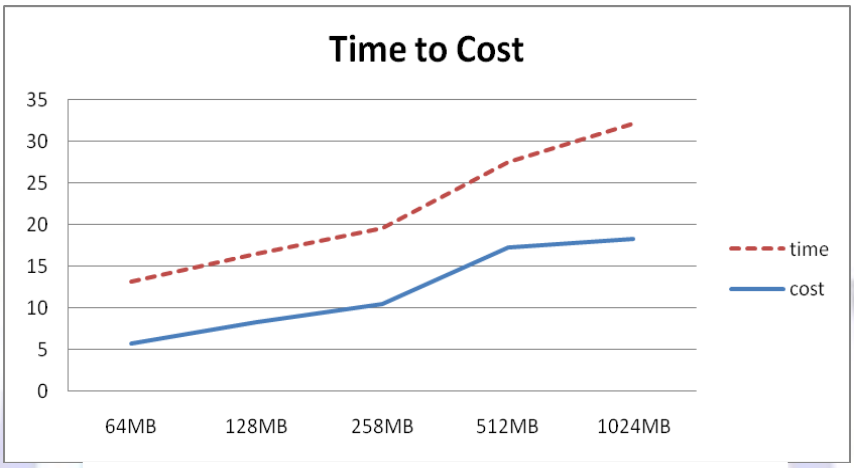


Fig.3: Results when time first then cost

Measure \ Data size	64	128	256	512	1024
utilization	0.442999	0.449285	0.419856	0.41489	0.398963
speedup	0.675007	0.61589	0.54939	0.486035	0.360473
efficiency	0.225002	0.205297	0.18313	0.162012	0.120158

Table3. performance parameters for each data size

According to the results in Figure 3, it is found that by applying the cost fitness function then time fitness function, the average cost= 12.05 and average time= 9.76.

By applying the time and cost fitness functions according to combination (c) for our MPOS algorithms, the results are depicted in Figure 4, and the performance parameters are presented in Table 4.

Fig. 4: Results of both cost and time

Data size / Measure	64	128	256	512	1024
utilization	0.433987	0.536272	0.488025	0.440362	0.481839
speedup	0.662486	0.801496	0.715649	0.647612	0.745342
efficiency	0.220829	0.267165	0.23855	0.215871	0.248447

Table4. performance parameters for each data size

According to the results in Figure 4, it is found that by applying the cost and time fitness functions, the average cost = 6.20 and average time = 7.04.

By applying the time and cost fitness functions according to combination (d) for our MPOS algorithms, the results are depicted in Figure 5, and the performance parameters are presented in Table 5.

Fig.5: Results when best time lead to best cost

Data size / Measure	64	128	256	512	1024
utilization	0.468638	0.454404	0.448058	0.46963	0.497977
speedup	0.680457	0.672616	0.672917	0.701033	0.722404
efficiency	0.226819	0.224205	0.224306	0.233678	0.240801

Table5. performance parameters for each data size

According to the results in Figure 5, it is found that by applying the best cost fitness function then the best time fitness function, the average cost = 6.15 and average time = 7.25.

By applying the time and cost fitness functions according to combination (e) for our MPOS algorithms, the results are depicted in Figure 6, and the performance parameters are presented in Table 6.

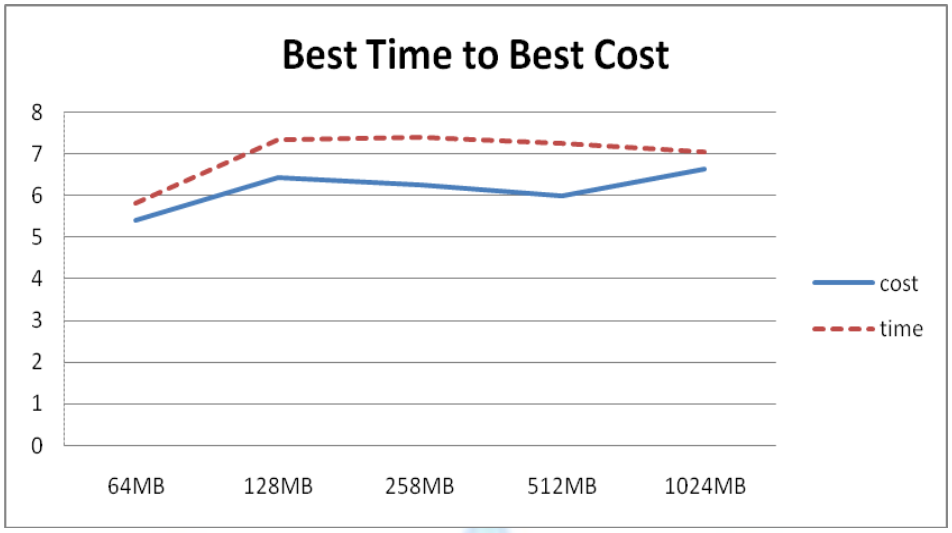


Fig.6: Results when best cost to best time

Data size / Measure	64	128	256	512	1024
utilization	0.562646	0.463143	0.44587	0.456203	0.486767
speedup	0.856996	0.680025	0.675311	0.68899	0.709757
efficiency	0.285665	0.226675	0.225104	0.229663	0.236586

Table6. performance parameters for each data size

According to the results in Figure 6, it is found that by applying the best cost fitness function then the best time fitness function, the average cost= 6.15 and average time= 6.98.

According to the experiments results (see Figures (2-6)) using five combinations of the time and cost fitness functions which are defined from (a) to (e), we note that the average cost and average time are reduced and performance measures (i.e., utilization, speedup, efficiency) is increased in each experiment. So, by applying combination (e) for our MPOS algorithm would produce good results with respect to the total cost and total computation time minimization.

The implementation results of our modified MPSO using the five combinations (a-e) with respect to the existed PSO algorithm are presented in Figure 7.

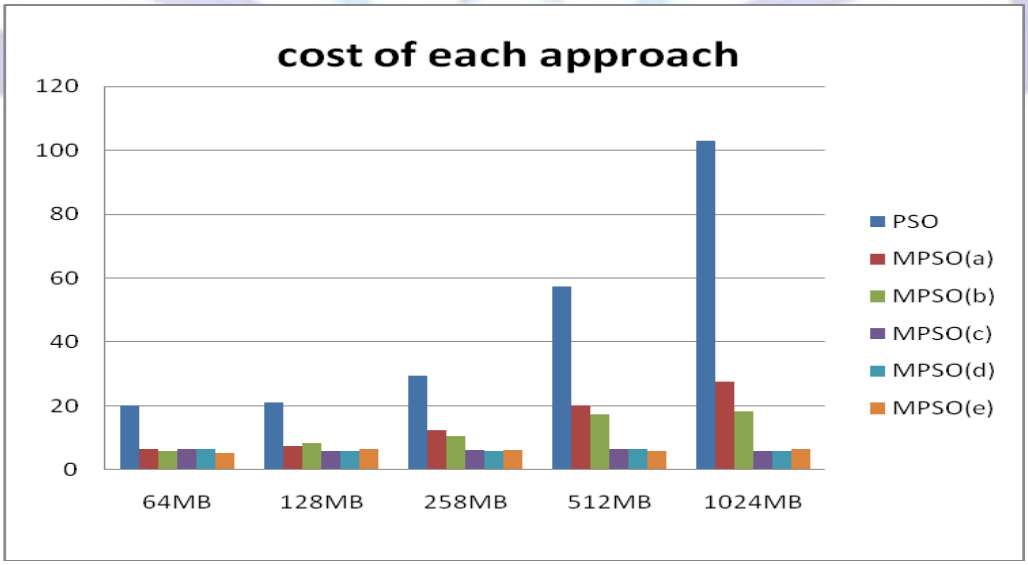


Fig.7: Average cost of each combination of MPSO and existed PSO Algorithms

With respect to combination (a):

Cost reduction of our MPSO algorithm relative to the existed PSO algorithm =

14.86/46.176* 100 = 32%

With respect to combination (b):

Cost reduction of our MPSO algorithm relative to the existed PSO algorithm =

(12.05/ 46.176)*100 =26%

With respect to combination (c):

Cost reduction of our MPSO algorithm relative to the existed PSO algorithm =

(6.2/46.176)*100= 13.4%

With respect to combination (d):

Cost reduction of our MPSO algorithm relative to the existed PSO algorithm =

(6.15/46.176) *100=13.3%

With respect to combination (e):

Cost reduction of our MPSO algorithm relative to the existed PSO algorithm =

(6.15/46.176) *100=13.3%

According to the results in Fig. 7 and the computed cost reduction, we note that our modified MPSO algorithm is always outperformed the existed PSO algorithm.

Generally, by considering the computation time of tasks besides the cost for allocating tasks to the available resources produces better results than that considering the cost only.

6. CONCLUSION

In this paper, a modified task scheduling heuristic based on Particle Swarm Optimization (PSO) is introduced and implemented. This modified algorithm is called MPSO. The aim of the modified PSO is to minimize the total cost and time of execution of application workflows on Cloud computing environments, where the total cost of execution is obtained by varying the communication cost between resources and the execution time of compute resources. The main principle of our MPSO algorithm is that two fitness functions, cost and time, are introduced. According to the comparative results, it is found that our MPSO algorithm outperforms the existed PSO algorithm.

7. FUTURE PROPOSITIONS

As part of our future work, we would like to integrate PSO based heuristic into our workflow management system to schedule workflows of real applications like LCC for HSP problem [23].

References

1. A. A. Soror, U. F. Minhas, A. Aboulnaga, K. Salem, P. Kokosielis, and S. Kamath, "Deploying Database Appliances in the Cloud.," IEEE Data Eng. Bull., vol. 32, No. 1, pp. 13-20, 2009.
2. A.jangra, and T.Saini. "Scheduling Optimization in Cloud Computing." International Journal of Advanced Research in Computer Science and Software Engineering,IJARCSSE 3 (April 2013): 62-65.
3. C. Computing. (2010). Handbook of Cloud Computing. [online]. Available: http://www.springerlink.com/index/10.1007/978-1-4419-6524-0
4. Cui Lin, Shiyong Lu," Scheduling ScientificWorkflows Elastically for Cloud Computing" in IEEE 4th International Conference on Cloud Computing, 2011.
5. http://tracer.uc3m.es/tws/pso/basics.html
6. http://aws.amazon.com/ec2
7. http://aws.amazon.com/cloudfront/
8. J. Kennedy and R. Eberhart. Particle swarm optimization.In IEEE International Conference on Neural Networks, volume4, pages 1942–1948, 1995.
9. J.Huang. "The Workflow Task Scheduling Algorithm Based on the GA Model in the Cloud Computing Environment." *Journal of Software* 9, No 4 (Apr 2014): 873-880.
10. K.SUNITHA, MRS. P V SUDHA. "AN EFFICIENT TASK SCHEDULING IN DISTRIBUTED COMPUTING SYSTEMS BY IMPROVED GENETIC ALGORITHM." International Journal of Communication Network Security, Volume-2, Issue-2, 2013.

11. L. C. Qi Zhang, Raouf Boutaba, "Cloud computing: state-of-the-art and research challenges " Journal of Internet Services and Applications, vol. 1, No. 1, pp. 7-18, May 2010.
12. Lei Zhang, et al. "A Task Scheduling Algorithm Based on PSO for Grid Computing." *International Journal of Computational Intelligence Research* 4,No.1 (2008): 37–43.
13. Mrs.S.Selvarani1; Dr.G.Sudha Sadhasivam, improved cost-based algorithm for task scheduling in Cloud computing ,IEEE 2010.
14. Mell, P., Grance, T., ―The NIST Definition Cloud Computing‖, Version 15, 10-7-09. National Institute of Standard and Technology, Information technology Laboratory 2009.
15. Radulescu & v.Gemund," Fast and effective task scheduling in heterogeneous systems," In: 9th Heterogeneous computing, Workshop, 2000, p. 299–238.
16. Rong-Jiang Ma, Nan-Yang Yu and Jun-Yi Hu. Application of Particle Swarm Optimization Algorithm in the Heating System Planning Problem, The Scientific World Journal Volume 2013 (2013), 11 pages.
17. Suraj Pandey, Linlin Wu, Siddeswara Guru, and Rajkumar Buyya. "A Particle Swarm Optimization (PSO)-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments." *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA), Perth, Australia.* April 20-23, 2010.
18. Saeed Parsa and Reza Entezari-Maleki," RASA: A New Task Scheduling Algorithm in Grid Environment" in World Applied Sciences Journal 7 (Special Issue of Computer & IT): 152-160, 2009.
19. Ucar Bora et al."Task assignment in heterogeneous computing systems,"J Parallel Distrib Comput, 2005, 66, 32–46.
20. Visalakshi, and Sivanandam. "Dynamic Task Scheduling with Load Balancing using Hybrid Particle Swarm Optimization." *International Journal of Open Problems in Computer Science and Mathematics, ICSRS Publication* 2– No. 3 (Sep 2009).
21. Y. Yang, et al. An Algorithm in SwinDeW-C for Scheduling Transaction- Intensive Cost-Constrained Cloud Workflows, Proc. of 4th IEEE International Conference on e-Science, 374-375, Indianapolis, USA, December 2008.
22. Yang, et al. An utility- based job scheduling algorithm for cloud computing considering reliability factor. Proceedings of the 2011 International Conference on Cloud and Service Computing, Dec. 12-14, IEEE Xplore Press, Hong Kong, pp: 95-102.
23. Zhao Rizos Henan, Sakellariou Rizos, "An investigation into rank function of the heterogeneous earliest finish time (HEFT) algorithm," University of Manchester, UK: Department of ComputerScience;2003.