The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.53 e-ISSN: 2251-7545

A TunableWorkflow Scheduling AlgorithmBased on Particle Swarm Optimization for Cloud Computing

Jing Huang, Kai Wu, Lok Kei Leong, Seungbeom Ma, and Melody Moh
Department of Computer Science
San Jose State University
San Jose, CA, USA
Melody.Moh@SJSU.EDU

Abstract—**Cloud computing uses a great amount of heterogeneous resources to deliver countless different services to users of distinctive quality of services (QoS) requirements. Numerous diverse tasks need to be carried out to meet the vastly different QoS and budget requirements. Workflow scheduling is therefore critical for the success of large-scale cloud computing. Particle Swarm Optimization (PSO) has been adopted for workflow scheduling in cloud computing, yet most existing works focused on a single objective. This paper proposes a tunable fitness function for the PSO algorithm, based on which a workflow schedule may be selected for minimal cost or minimal makespan (completion time), or any level in between. A heuristics is further proposed to address bottleneck problems andattains a smaller makespan. Performance evaluation and complexity analysis are both presented, which show that the proposed algorithm surpasses the existing ones in both cost and makespan while maintaining a reasonable load balance and keeping the same time complexity. We believe that the tunable fitness function-based PSO have many potential applications in other soft computing and distributed computing models.**

Keywords: cloud computing, makespan, particle swarm optimization (PSO), soft computing, workflow scheduling

I. INTRODUCTION

Cloud computing is a new paradigm for distributed computing.Itdeliversa poolof abstracted, virtualizedresources, including computing power, storage, platforms and software applications over the Internet based on users' demand [1].Due to its many benefits such as elastic, scalable resource provision and cost-effectiveness, cloud computing has attracted a rapidly increasing number of users.

Cloud computing offers a great variety of services. Based on the level of services, they are generally divided into three categories: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS puts servers, storage, networks, and data center fabrics together as demanded by users. Cloud users can then install operating system images and deploy their applications based on the infrastructure. PaaS, on the other hand, provides middleware, database and development tools. It enables users to deploy applications onto a virtualized cloud platform [2]. Finally, in SaaS the complete operating environment, along with applications, management, and user interfaces, are provided to cloud users [3]. Since all these services are made available as subscription-based in a pay-per-use model, cloud computing leverages many attractive features to users, including low cost and simple management.

There are many technical challenges faced by cloud providers, such as maintaining high utilization while delivering services that arelow cost, short delay, and dynamic deployable.It is critical for cloud providers to maintain an optimal workflow scheduling and management system to meet these challenges.

A workflow is formed by a logical sequence of interdependent tasks decomposed from applications [4]. Acloud workflow system is vital for supporting large-scale e-science and e-business applications [5]. Workflow scheduling is one of the key components in a workflow management system. The scheduler decides which resources will be used, as well as which tasks will be executed on each of these resources. It allocates suitable resources to workflow tasks so that the execution can be completed while satisfyingthe QoS constraints set by users, such as execution time and cost. The workflow scheduling problem, like general scheduling problems, is NP-complete. Workflow scheduling algorithms often utilize heuristics and meta-heuristics, includingsoft computing techniques, to obtain approximated solutions.

In this paper, we adopt a workflow scheduling strategy using Particle Swarm Optimization (PSO). PSO, an applied soft computing method developed by Kennedy and Eberhart [6], is one of the latest evolutionary algorithms inspired by nature.PSO approximates an optimal solution by iteratively improving a swarm of candidate solutions, called particles. Each particle is modified iteratively by the best information from both the individual and the entire swarm. Due to the collective intelligence of these particles, the swarm is expected tomove toward the best solutions. PSO works well on most global optimal problems [6, 7]. In addition it is simple, effective, and of low computational cost.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.53 e-ISSN: 2251-7545

Makespan and cost are two main performance measurement criteria specified by cloud users and considered by workflow schedulers [8-15]. Makespan is the time from the beginningtillthe completion of the sequence of tasksina workflow. Different application schedulers may use different policies with different objectives. Some algorithms are designed to achieve minimumcost[9, 12, 14] while others strives for minimum makespan [13] or for load balance [14]. Most existing algorithms focus on achieving a single optimal criterion [12-14].

In this paper, a workflow scheduling strategy to attaina combined minimal cost and minimal makespan is introduced. Moreover, the objective is adjustable between minimal cost and minimal makespan, able to satisfy users' various quality of services (QoS) requirements.

The main contributions of this paper are:

1. *A model for a mapping between tasks and resources is formulated, achieving a tunable objective between cost and makespan.*
2. *A PSO-based heuristics is presented to realize the optimal mapping for the tunable objective.*
3. *The heuristics is further improved by addressing bottleneck tasks and thus reduces the makespan even more.*
4. *While most PSO papers simply use a fixed particle number in their experiments, the effect of the number of particles in the PSO performance is studied.*

The rest of the paper is organized as follows: Section 2 discusses related work. The scheduling problem formulation is described in Section 3. In Section 4 we present our scheduling algorithms using PSO. Experimental results and complexity analysis are derived in Section 5. Finally, Section 6 concludes the paper.

II. RELATEDWORK

This section discusses first the major works of workflow scheduling for grid and cloud environments. Next, we focus on those based on soft-computing approaches.

Task scheduling is an NP-Complete problem. Most efforts are therefore concentrated on heuristics and meta-heuristics. Yu and Buyya [16] studied several workflow scheduling algorithms in a grid environment, such as Min-Min, Max-Min, Heterogeneous Earliest-Finish-Time (HEFT) algorithm [17], and Greedy Randomized Adaptive Search Procedure (GRASP) algorithm [18]. Liu, et al proposed a compromised-time-cost scheduling algorithm [8]. They considered the characteristics of cloud computing to accommodate instance-intensive cost-constrained workflows by compromising execution time and cost with user input enabled on the fly. The hybrid cloud optimized cost (HCOC) algorithm, proposed by Bittencourt and Madeira [9], schedules the workflow first in a private cloud, and reschedules it onto a public cloud if the user deadline cannot be met. Xu et al proposed a scheduling strategy for multiple workflows and multiple QoS requirements [10]. The algorithm considers several factors affecting the makespan and cost of workflow; based on which a scheduling is generated to satisfy users' QoS requirements while increasing the success rate of the workflow scheduling.

Several soft-computing approaches have been adopted to solve workflow scheduling problem, including genetic algorithm (GA) [11,12] and PSO [13-15, 19], etc. These algorithms are evolutionary optimization algorithms inspired by nature. Several studies have shown that PSO-based algorithms have faster convergence and better scheduling results that GA methods [13, 19]. Zhang et al adopted a PSO-based task scheduling algorithm on a grid environment, with an objective to minimize the completion time [13]. They have found that the PSO-based method has reached results that are better than the GA approach. Pandey, et al proposed a PSO-based scheduling algorithm to minimize the total cost of workflow [14]. It takes into account both execution and transfer costs, and defines the maximum resource cost as the fitness function to achieve load balance. Wu et al proposed a revised discrete PSO scheduling algorithm, with the sum of makespan and total cost as its fitness function [15].

Note that most of existing works addressed a single fitness function (makespan[13] or cost [14]), a constrained single objective [12], or a fitness function that is the sum of makespan and cost [15]. We proposed a tunablefitness function, which may be easily adjusted according to users' priorities and QoS requirements, as described next.

III. SCHEDULING PROBLEM FORMULATION

In the following, we adopt the general model and notation used by existing works on PSO-based scheduling [14, 15]. A workflow is commonly represented by a Directed Acyclic Graph (DAG), denoted by $\square = (\square, \square)$. Let the number of tasks in workflowbe $\square$. The set of nodes $\square = \{\square_{\square}, \dots, \square_{\square}\}$represents the tasks in the workflow applications, where *n* is the total number of tasks. The set of arcs $\square = \square_{\square\square} \square 1 \le \square, \square \le \square$ represents the data dependencies amongthe tasks. An arc, $\square_{\square\square} = (\square_{\square}, \square_{\square}) \in \square$, implies that$\square_{\square}$transfers data to $\square_{\square}$. In this relationship,$\square_{\square}$is the parent task of$\square_{\square}$, and $\square_{\square}$is the child of $\square_{\square}$. The child task can be executed only after it receives data transferred from all of its parents. Fig. 1 shows a workflow example of 8 interdependent tasks. Note that any single task can have one or more children (except for the bottom nodes), and any single task can have one or more parents (except for the top node).

Suppose there are a total of $\square$ resources in the cloud environment. The resources can be denoted as $\square = \{\square_{\square}, \dots, \square_{\square}\}$. All the resources are interconnected with each other so that they can transfer data among each other.The scheduling problem is to find an optimal mapping $\square$ between tasks and resources according to some optimization objective. As mentioned before, cost is a common objective that is more concerned by user; makespan is another objective that is critical for scheduling.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.53 e-ISSN: 2251-7545

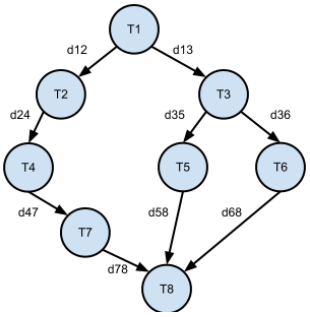

Figure 1. A workflow example with 8 tasks.

In the following, we formulate several optimization objectives. Let $makespan(\pi)$ denote the makespan of the workflow with respect to the mapping π:

$$makespan(\pi) = \text{the length of the path from } t_1 \text{ to } t_n \text{ in } G. \quad (1)$$

The makespan of a workflow is the time duration from the beginning of the first task till the end of the last task. Since a workflow consists of interdependent tasks, both execution time and transfer time need to be considered.

Next, let $exec(r_i)$ and $trans(r_i)$ be the execution and transfer costs of resource r_i, respectively. $total(r_i)$ denotes the total cost of resource r_i:

$$total(r_i) = exec(r_i) + trans(r_i) \quad 1 \le i \le n. \quad (2)$$

Let $totalcost(\pi)$ be the total cost of the workflow *w.r.t* the mapping π:

$$totalcost(\pi) = \sum_{i=1}^{n} total(r_i). \quad (3)$$

For the objective of minimizingthe cost while balancing the load [14],the fitness function is given as:

$$fitness_1 = \max(total(r_i)), 1 \le i \le n. \quad (4)$$

The objective is to minimize $fitness_1$. The reason for not using the total cost of all the resources is to prevent from mapping all the tasks to a single, least-cost resource.

For the objective of optimizing makespan (such as the work by Zhang et al [13]), the fitness function can be defined as:

$$fitness_2 = makespan(\pi). \quad (5)$$

The objective is to minimize $fitness_2$.

In this paper we propose an objective of minimizing the *weighted* sum of total cost and makespan; the fitness function can then be defined as:

$$fitness_3 = w \cdot totalcost(\pi) + (1-w) \cdot makespan(\pi), \quad 0 \le w < 1, \quad (6)$$

where w is the weight given to thetotal cost and $1-w$ is the weight given tomakespan. This fitness function can be easily tuned by changing the value w to satisfy the various QoS requirements including budget constraints. Again the objective is to minimize $fitness_3$.

IV. Particle Swarm Optimization-based Scheduling Algorithms

This section presents the PSO method to approximate the optimal solutions specified by the fitness functions defined above. PSO is one of the latest evolutionary algorithms inspired by the social behavior of fish schooling or bird flocking [6]. Each particle corresponds to an individual bird or fish searching in a search (problem) space, and is referred to a candidate solution. The flock or swarm of particles is randomly generated initially [20]. Each particle has its own position in the space,with a fitness value corresponding to the position; it also has a velocity to determine the speed and direction by which is flies. PSO achieves an optimal solution by having a population of particles (candidate solutions), and moving these particles around in the search space according to each particle'svelocity and updated position.

Particles in the search process update themselves by tracking two best-known positions: (1) The local best position is the individual's best-known position in terms of the fitness value reached so far by the particle itself. (2) The global best position is the best position so far among all the particles in the entire population.

A. Basic Notations

Let the number of particles be n. Let x_i^t, v_i^t, $p_i^t$and g_i^t be the position, velocity, best local and best global positions, respectively, of particle *i*at iteration t. The velocity and position of particle *i* are each updated according to equations(7) and(8), respectively; and the two best positions are updated according to(9) and (10) respectively:

$$v_i^t = w v_i^{t-1} + c_1 r_1 (p_i^{t-1} - x_i^{t-1}) + c_2 r_2 (g_i^{t-1} - x_i^{t-1}) \quad 1 \le i \le n, \quad (7)$$

$$x_i^t = x_i^{t-1} + v_i^t \quad 1 \le i \le n, \quad (8)$$

$$p_i^t = min(x_i^t, p_i^{t-1}) \quad 1 \le i \le n, \quad (9)$$

$$g_i^t = min(p_1^t, \dots, p_n^t), \quad (10)$$

where, w is the inertial weight; c_1, c_2 are acceleration coefficients, and r_1, r_2 are random numbers in the range of$[0,1]$. At each iteration, the velocity is updated according to its current velocity and the local and global best positions. The position is updated based on the current position and the updated velocity. These ensure that the particles search around the local and global best positions and converge to a global best position in the limited iteration.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.53 e-ISSN: 2251-7545

In the workflow scheduling problem, a particle represents a mapping between the resources and the tasks. The dimensionof a particle is the number of tasks in the workflow. For example, consider a problem of 8 tasks and 5 resources. One possible particle for the mapping is illustrated in Fig. 2.

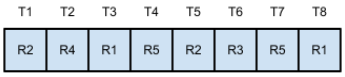

Figure 2. A sample particle (8 tasks on 5 resources).

B. The PSO Algorithm

The evaluation of each particle is performed by the fitness function, defined according to the optimization objective, as described in Section III.The high-level structure of a PSO algorithmis given in Table I, followed by a detailed description.

First, the position and velocity of all the particles are randomly initialized. If the iteration stopping criterion is not met, the algorithm repeatedly does the following: for each particle, it first calculates its fitness value using one of the fitness functions given in (4-6), and then updates its local best positions using (9). Then, it calculates the global best position among all the particles using (10). It updates thevelocity and position of all the particles using (7) and (8). Finally, when the stopping criterion is met, the global best position is the optimal mapping.

TABLE I. THE PSO ALGORITHM

```
1:  Initialize particles' position and velocity randomly.
2:  While stopping criterion is not satisfied do
3:      For each particle do
4:          Calculate its fitness value using the fitness function.
5:          Update its local best position.
6:      End For
7:      Update the global best position.
8:      For each particle do
9:          Update its velocity and position.
10:     End For
11: End While
12: Return the global best position.
```

After computing the mapping using PSO, the scheduling algorithm dispatches the ready tasks into each of the resources. A ready task is defined to beone that hascompletely received the entire data transferred from all of its parent tasks.

C. Bottleneck Reduction

Next, we present a heuristics to further reduce the makespan, as illustrated in Table II and described below. Since all the ready tasks assigned to a specific resource are independent, it will speed up the workflow by scheduling first the "bottleneck" task; i.e., the task having most descendants. Thus, in Step 3 the ready tasks are sorted according to the number of descendants. If there is a tie, the one with a short execution time will be given a high priority to execute first; this is done in Step 5 where tasks are sorted according to the execution time.

TABLE II. IMPROVEMENT: BOTTLENECK REDUCTION ALGORITHM

```
1: For each resource do
2:   For all the ready tasks in the resourcedo
3:     Sort tasks in descending order of the number ofdescendants.
4:     For ready tasks having the same number of descendantsdo
5:       Sort tasks in ascending order of execution time.
```

V. PERFORMANCE EVALUATION AND COMPLEXITY ANALYSIS

In this section, we first describe the four PSO algorithms and the cloud experiment setup. Next, the performance results are presented. Finally, the timecomplexity of four algorithms is analyzed.

A. PSO Algorithms

TheJSwarmpackage is extended for conducting the PSO experiments[21]. The number of iterations in the PSO algorithm is set to be 100. Four algorithms are evaluated, as summarized in Table III.

TABLE III. FOUR PSO ALGORITHMS

Algorithm	Objective	Fitness Function	Strength
1 [18]	Minimize the maximal cost	(4)	Load balance
2	Minimize the makespan	(5)	Low makespan
3 (Proposed)	Minimize the weighted sum of total cost and makespan	(6)	Tunable
4 (Proposed)	Minimize the weighted sum of total cost and makespan, with *bottleneck reduction*	(6)	Tunable, with Bottleneck-reduction, minimal cost & min. makespan

B. Cloud Experiment Setup

CloudSim 3.0 is used to configure cloud environment and simulate the execution of workflow [22-23]. It is is a toolkit for modeling and simulation of cloud computing environments. A data center (shown in Fig. 3) consisting of one switch and four hosts each having two VM (Virtual Machines) is configured in CloudSim. Note that the ports of a given switch each havea different bandwidth. The allocation of VM to hosts uses the default FCFS algorithm in CloudSim. For each VM on the same host, the time-shared policy is used such that two VM can run concurrently. For each task on the same VM, the space-shared policy is used such that tasks in one VM are executed sequentially.

The millions of instructions per second (MIPS) and execution cost of each VM is given in Table III; the data transfer cost between different VMs is shown in Table IV. The prices are by referring to the pricing policy of Amazon

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.53 e-ISSN: 2251-7545

EC2's pricing policy. In addition, the execution cost of a task is in proportion to the task's millions of instructions (MI) requirement and the MIPS of the VM. The data transfer cost is in proportion to the data size and the bandwidth between VMs where the data are transferred.

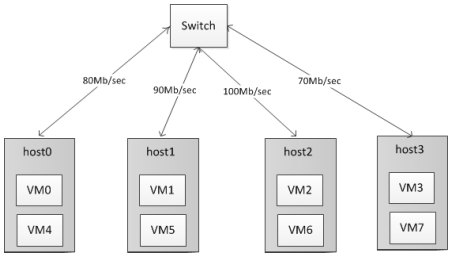

Figure 3. Experimental datacenter infrastructure.

TABLE IV. MIPS AND EXECUTION COST OF EACH VM

VM	MIPS	Execution cost (cents/MI)
0	1.011	0.03361
1	1.004	0.03333
2	1.013	0.03444
3	1.000	0.03278
4	0.990	0.03111
5	1.043	0.03528
6	1.023	0.03472
7	0.998	0.03167

TABLE V. TRANSFER COST (CENTS/MB) BETWEEN EACH VM

VM	0	1	2	3	4	5	6	7
0	0	0.17	0. 20	0.20	0.21	0.21	0.18	0.18
1	0.17	0	0.20	0.20	0.21	0.21	0.18	0.18
2	0.20	0.20	0	0.17	0.22	0.22	0.19	0.19
3	0.20	0.20	0.17	0	0.22	0.22	0.19	0.19
4	0.21	0.21	0.22	0.22	0	0.17	0.20	0.20
5	0.21	0.21	0.22	0.22	0.17	0	0.20	0.20
6	0.18	0.18	0.19	0.19	0.20	0.20	0	0.17
7	0.18	0.18	0.19	0.19	0.20	0.20	0.17	0

The workflow with 96 tasks is used in experiment (Fig. 4). Each task has its own MI; data transfers in megabyte (MB) among tasks are also specified.

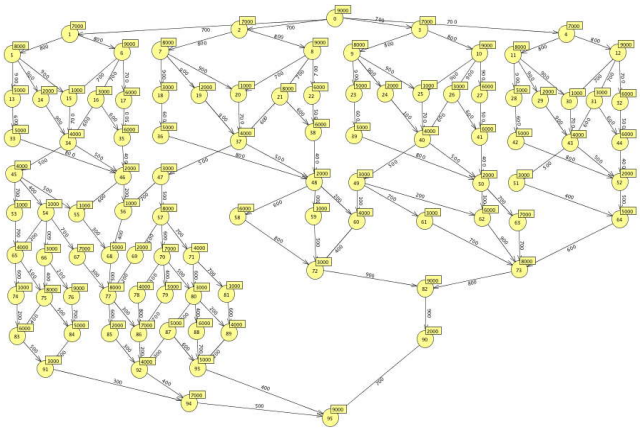

Figure 4. Experimental workflow (96 tasks).

C. Experimental Results

This section describes the results. All the results are the average of 30 independent executions. □ = 0.5 for Algorithms 3 and 4 unless otherwise specified. Except for Section 1), the number of particles = 500 is used.

1) The Effect of the Number of Particles

The number of particles may influence the performance by a varyingdegreedepending on the problem being optimized [7]. In a cloud environment, the total number of tasks to be executed is usually large; i.e., the particle dimension in the PSO is large. Existing results use a small number of particles: 25 (for5 tasks [14]) and 30 (for 50-300 tasks [15]). We believe that using a larger number of particles is more desired when there are a large number oftasks(96 tasks in our experiments), so we conduct some experiments to investigatethiseffect.

Fig. 5 shows the cost of Algorithms 1 and 4, and Fig. 6 the makespan of Algorithms 2 and 4. It is clear that the number of particles affect the optimization results, especially when the particle number is small (smaller than 200). When the particle number increases beyond 500, the effect gradually diminishes.

Using larger number of particles generally improves optimization results, but also increases the time complexity of the PSO algorithm (see Section V.D for complexity analysis), we therefore choose a compromise and use 500 for the rest of the experiments.

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.53 e-ISSN: 2251-7545

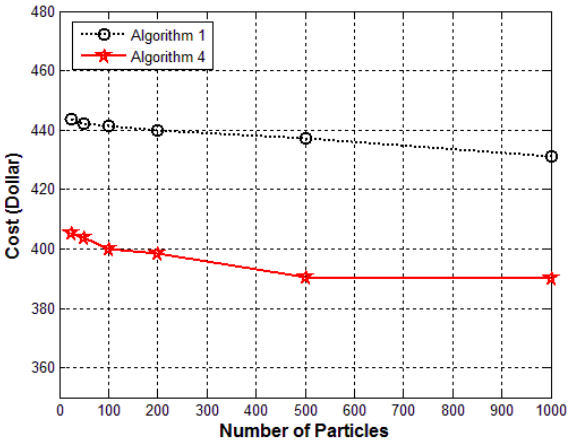

Figure 5. Cost vs. the number of particles.

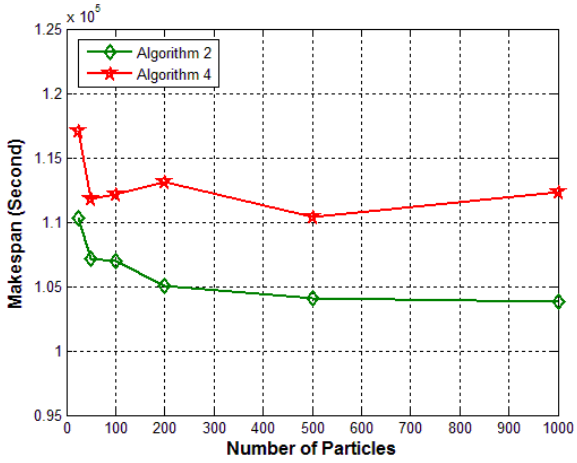

Figure 6. Makespan vs. the number of particles.

2) The Cost Performance

We vary the tasks' MI by multiplying different proportion values.Fig. 7 shows the total cost of the four algorithms. The cost of Algorithms 3 and 4 are lower than Algorithms 1 and 2, while the cost of Algorithm 2 is the highest. This is reasonable since Algorithm 1 minimizes the maximalof the individual costsrather than the total cost (to prevent all the tasks executing on a single resource), while Algorithm 2 minimizes the makespan. Algorithms 3 and 4 aim to minimize the sum of total cost and makespan, both achieves the lowest cost.The Bottleneck reduction in Algorithm 4 does not change the mapping and therefore does not affect the cost.

The makespan of the four algorithms is shown in Fig. 8. It is clear, and with obvious reason, that Algorithms 1 and 2 have the highest and the lowest makespan, respectively. Note that Algorithm 4 attains a makespan that is very close to Algorithm 2, and is even lower than Algorithm 2 with MI proportion = 1.8. *This demonstrates that Algorithm 4 has achieved concurrently both minimal total cost and minimal makespan.*

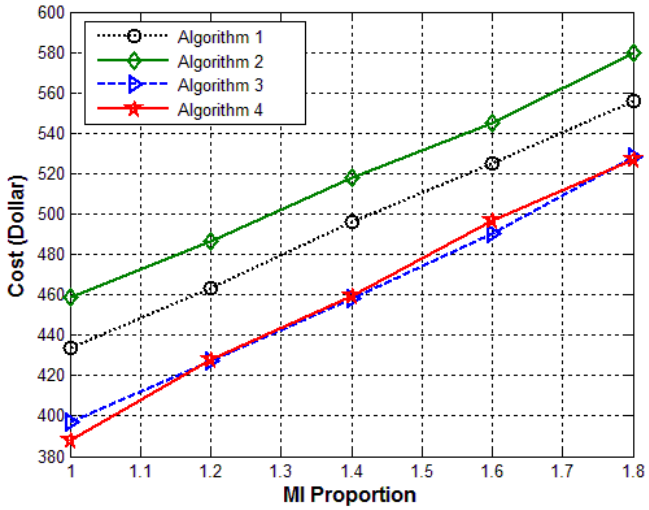

Figure 7. Cost vs. MI (Millions of Instructions).

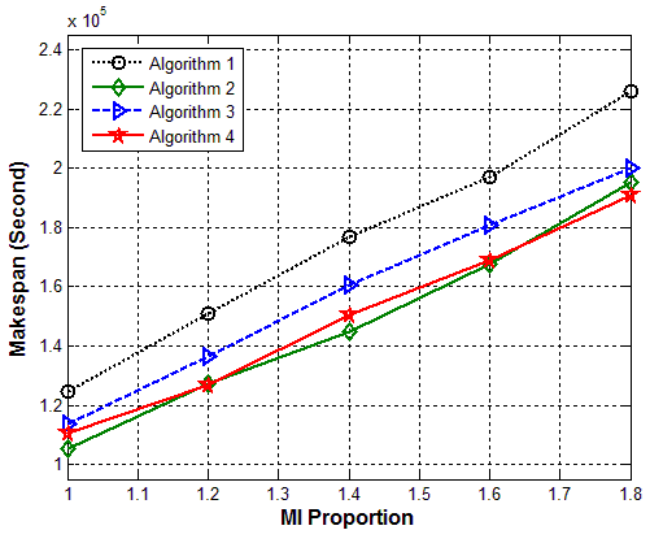

Figure 8. Makespan vs. MI (Millions of Instructions).

3) The Load Balance Performance

Fig. 9 shows the average and the sample standard deviation (SSD) of the number of tasks per VM. Note that since there are 96 tasks and 8 resources (VM), the average number of tasks per VM is 12. The smaller the SSD, the more balanced the load distribution. Observed that Algorithm 1 has achieved the most balanced load. Algorithms 3 and 4 have slightly higher SD.

To achieve a tunable PSO algorithm with excellent load balance, the following (the fourth) fitness function may be defined:

$$\text{[illegible]} = \text{[illegible]}(\text{[illegible]}) + (1-\text{[illegible]})\ \text{[illegible]}(\text{[illegible]}),\ 0 \le \text{[illegible]} < 1, 1 \le \text{[illegible]} \le \text{[illegible]}, \quad (11)$$

whichis to minimize the weighted sum of one maximum resource cost (like Algorithm 1) and makespan (like Algorithm 2). In other words, this fitness function simply

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.53 e-ISSN: 2251-7545

combines equations (5) and (6) with weighted values. We namethe algorithm using fitness function (11) as Algorithm 5, and the one using fitness function (11) with bottleneck reduction as Algorithm 6. Their load balance performance is also shown in Figure 9.

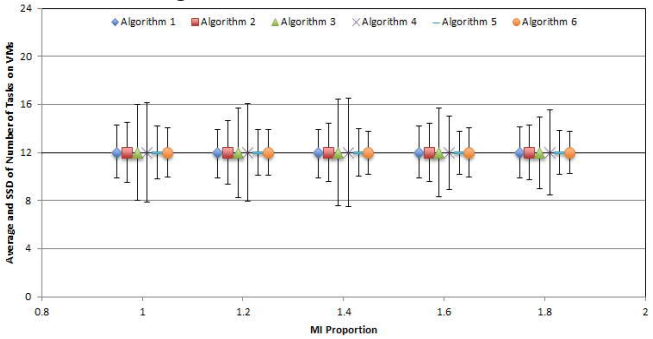

Figure 9. Load balance: Average and SSD vs. MI (Millions of Instructions).

It is clear that, using the new tunable fitness function, Algorithms5 and 6achieve the most balanced load.This is because fitness function (11) used in Algorithms 5 and 6 considers maximum resource cost instead of total cost as in fitness function (6) used in Algorithm 3 and 4, the load of each resource is thus more balanced. Note that the costs of Algorithms 5 and 6 are between Algorithm 1 and 2, which are higher than Algorithms 3 and 4.The makespan of Algorithms 5 and 6 are also higher than Algorithm 3 and 4, but lower than Algorithm 1.

4) Tuning theWeight Value (□)

One chief advantage of the proposed Algorithms 3 and 4,is that the weight value (□) can be tuned. Fig. 10 shows the cost and makespan results while varying □. As □,the weight placed on optimizingthe cost, increases, the cost decreases while the makespan increases. Thus, □ may be tuned to achieve a degree of compromise between cost and makespan according to different QoS requirements. (Note that the point of intersection may move when the scale of the two vertical axis change.)

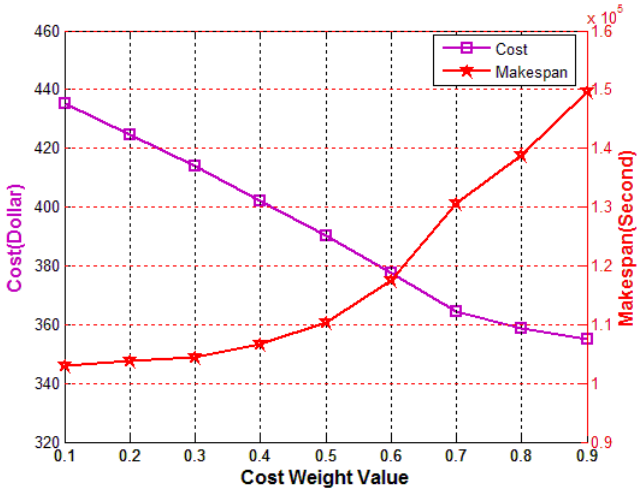

Figure 10. Cost, Makespan vs. weight (□) values of Algorithm 4.

D. Complexity Analysis

While Algorithms 3 and 4 perform better than Algorithms 1 and 2 in terms of combined cost and makespan, as shown above, we analyze and compare the time complexity of the four algorithms. Let N be the number of particles, Lbe the number of iterations, $n$be the number of tasks, and $e$be the number of edges in the DAG (i.e., the number of data transfers needed among tasks) in the PSO algorithm. The time complexity of the four algorithms is summarized in Table VI. Clearly the four algorithms have comparable complexities. Algorithm 4, which improves over Algorithm 3, does not need a larger time complexity.

TABLE VI. TIMECOMPLEXITYANALYSIS

Algorithm	1 [18]	2	3 (Proposed)	4 (Proposed)
Fitness function	O (n^2)	O (e) ≤ O (n^2)	O ($n^2 + e$) = O (n^2)	O ($n^2 + e$) = O (n^2)
For N particles	O (Nn^2)	O (Ne) ≤ O (Nn^2)	O (Nn^2)	O (Nn^2)
For L iterations	**O(LNn^2)**	**O(LNe)**≤ O (LNn^2)	**O (LNn^2)**	O (LNn^2)
Bottleneck reduction	N.A.	N.A.	N.A.	O(LNn^2+$n$log$n$) = **O (LNn^2)**

VI. CONCLUSIONS AND FUTURE WORK

Most of the existing works using PSO for workflow scheduling in the cloud environment use a single, fixed fitness function. In this work a tunable fitness function has been proposed, which may give different weights to cost minimization and to makespan minimization. Furthermore, as existing works carry out experiments mostly using a small number of particles (25-30), we investigated the effect of the number of particles in PSO-based algorithms, and chose a suitable large number (500) in the experiments. With an additional heuristics that deals with bottleneck tasks, the proposed PSO-based algorithm has achieved both minimal cost and minimal makespan comparing with two existing algorithms. A similar tunable PSO-based algorithm has also

The International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 3, No. 3, Special Issue:
The Proceeding of International Conference on Soft Computing and Software Engineering 2013 [SCSE'13],
San Francisco State University, CA, U.S.A., March 2013
Doi: 10.7321/jscse.v3.n3.53 e-ISSN: 2251-7545

been proposed that achieved the best load-balance.Future work may include exploring discrete PSO, experimenting the proposed algorithm on real-life cloud environments, and applying the idea of tunable objective functions on other soft computing and distributed computing models.

REFERENCES

[1] I. Foster, Z. Yong, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," Proc. computing environments workshop, pp. 1-10, 2008.

[2] K. Hwang, J. J. Dongarra, and G. C. Fox, Distributed and cloud computing: from parallel processing to the internet of things. Elsevier, Morgan Kaufmann, 2012.

[3] B. Sosinsky, Cloud computing bible. Wiley, 2011.

[4] L. F. Bittencourt, E. R. M. Madeira, and N. L. S. da Fonseca, "Scheduling in hybrid clouds," IEEE Communications Magazine, vol. 50, no. 9, pp. 48-55, 2012.

[5] Z. Wu, X. Liu, Z. Ni, D. Yuan, and Y. Yang, "A market-oriented hierarchical scheduling strategy in cloud workflow systems," Journal of Supercomputing, 2011.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proc. IEEE International Conference on Neural Networks, vol. 4, pp. 1942-1948, 1995.

[7] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," Proc. Swarm Intelligence Symposium, pp. 120-127, 2007.

[8] K. Liu, Y. Yang, J. Chen, X. Liu, D. Yuan and H. Jin, "A compromised-time-cost scheduling algorithm in SwinDeW-C for instance-intensive cost-constrained workflows on cloud computing platform," International Journal of High Performance Computing Applications, vol. 24, no.4, pp. 445-456, 2010.

[9] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," Journal of Internet Services and Applications, vol. 2, no. 3, pp. 207-27, 2011.

[10] M. Xu, L. Cui, H. Wang, and Y. Bi, "A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing," Proc. IEEE International Symposium on Parallel and Distributed Processing with Applications, pp. 629-634, 2009.

[11] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," Scientific Programming, vol. 14, nos. 3/4, pp. 217-230, 2006.

[12] J. Yu and R. Buyya, "A budget constrained scheduling of workflow applications on utility grids using genetic algorithms", Proc. Workshop on Workflows in Support of Large-Scale Science, pp. 1-10, 2006.

[13] L. Zhang, Y. Chen, R. Sun, S. Jing, and B. Yang. "A task scheduling algorithm based on pso for grid computing," International Journal of Computational Intelligence Research, vol. 4, no.1, pp. 37-43, 2008.

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

[15] Z. Wu, Z. Ni, L. Gu, and X. Liu, "A revised discrete particle swarm optimization for cloud workflow scheduling," Proc. International Conference on Computational Intelligence and Security, pp.184-188, Dec. 2010.

[16] J. Yu and R. Buyya, "Workflow scheduling algorithms for grid computing", Metaheuristics for Scheduling in Distributed Computing Environments, Springer, 2008.

[17] H. Topcuouglu, S. Hariri and M. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," IEEE Transactions on Parallel and Distribution Systems, vol. 13, no. 3, pp. 260-274, 2002.

[18] R. Sakellariou and H. Zhao, "A hybrid heuristic for DAG scheduling on hetero-geneous systems," Proc. 13th Heterogeneous Computing Workshop (HCW 2004), April 26, 2004.

[19] A. Salman, "Particle swarm optimization for task assignment problem," Microprocessors and Microsystems, vol. 26, no. 8, pp. 363-371, 2002.

[20] P. Yin, S. Yu, and Y. Wang, "A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems," Computer Standards and Interfaces, vol. 28, no. 4, pp. 441-450, 2006.

[21] P. Cingolani. (2005, June 20). JSwarm-PSO [Online]. Available: http://jswarm-pso.sourceforge.net/

[22] R. N. Calheiros1, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software – Practice and Experience, vol. 41, pp. 23-50, 2011.

[23] S. K. Garg and R. Buyya, "NetworkCloudSim: modelling parallel applications in cloud simulations," Proc. Fourth IEEE International Conference on Utility and Cloud Computing, pp. 105-113, 2011.