A Simplified Particle Swarm Optimization for Job Scheduling in Cloud Computing

Article *in* International Journal of Computer Applications · April 2017

DOI: 10.5120/ijca2017913744

Ibrahim Attiya
Zagazig University
19 PUBLICATIONS 528 CITATIONS
SEE PROFILE

Xiaotong Zhang
University of Science and Technology Beijing
122 PUBLICATIONS 1,454 CITATIONS
SEE PROFILE

A Simplified Particle Swarm Optimization for Job Scheduling in Cloud Computing

Ibrahim Attiya
School of Computer and Communication Engineering,
University of Science and Technology Beijing, Beijing, China
Mathematics Department, Faculty of Science, Zagazig University, Zagazig, Egypt

Xiaotong Zhang
School of Computer and Communication Engineering,
University of Science and Technology Beijing, Beijing, China
Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing, China

ABSTRACT

Recent advances in various areas such as networking, information and communication technologies have greatly boosted the potential capabilities of cloud computing and made it become more prevalent in recent years. Cloud computing is a promising computing paradigm that facilitates the delivery of IT infrastructure, platforms, and applications of any kind to consumers as services over the internet. Although cloud computing systems nowadays provide better ways to accomplish the job requests in terms of responsiveness and scalability under various workloads, scheduling of jobs or tasks in cloud environment is still NP-complete and complex in nature due to the dynamicity of resources and on-demand user application requirements. In this paper, a simplified version of particle swarm optimization (PSO) algorithm is proposed to solve the job scheduling problem in cloud computing environment. To evaluate the performance of the proposed approach, this study compares the proposed PSO strategy with genetic algorithm (GA), by having both of them implemented on CloudSim toolkit. The results obtained demonstrate that the presented PSO algorithm can significantly reduce the makespan of job scheduling problem compared with the other metaheuristic algorithm evaluated in this paper.

General Terms

Computer Networks, Distributed Systems, Cloud Computing.

Keywords

Cloud computing, job scheduling, makespan, particle swarm optimization, resource allocation.

1. INTRODUCTION

Cloud computing is an emerging technology that facilitates the delivery of various services such as infrastructure, servers, storage and applications to consumers over the internet. More specifically, it aims to deliver such services on a pay-per-use basis. Cloud computing can be defined as “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [1]. This definition reflects the essential characteristics of cloud computing environment, for instance, on-demand self-service, accessible through broad networks like the internet, can be quickly as well as easily scaled up or down on demand, draw from unlimited pool of computing resources and involves some sort of metering capability to track usage [2].

Cloud computing has been proposed as a new computing paradigm which can offer practical solutions for solving the limitation of restricted amount of resources and significantly reduce the cost of purchasing, maintaining and managing the physical resources [3, 4]. Specifically, it is the further development of distributed computing, parallel computing and grid computing [5]. Cloud computing has gained its popularity due to its ability to facilitate the provision and use of IT infrastructure, platforms, and applications of any kind in the form of services that are electronically available on the Web. Its services are targeted to the mass, ranging from the end consumers hosting their own documents on the internet to enterprises outsourcing their whole IT infrastructure to external data centers [6]. In other words, cloud computing can serve multiple sectors effectively. Firstly, it provides great business models for small computational science and engineering research groups because these groups often do not have enough human resources and knowledge to deal with the complexity of computational and data infrastructure for their research [7]. Secondly, it offers significant benefit to IT organizations by freeing them from the low-level tasks of setting up and maintaining basic hardware and software infrastructures and thus enabling them to focus on innovation and creating business value for their services [6]. Thirdly, it offers an exciting opportunity for end users by enabling them to utilize a variety of devices, including PCs, laptops, smartphones, and PDAs to access their personal data, programs, storage, and application-development platforms over the internet, via on-demand services offered by cloud providers. Last but not the least, cloud computing enables consumers to take benefit from all of the provided technologies without the need for deep knowledge about or expertise with each one of them [2]. Moreover, the cloud aims to reduce costs and help the consumers concentrate on their core business instead of being impeded by obstacles of IT.

Central to any cloud environment is the concept of virtualization. The virtualization technology further makes cloud computing different from the traditional paradigms, and also makes cloud computing more convenient in the commercialization [8]. Virtualization provides a promising approach to separate the hardware and software resources and allows running several independent virtual servers on a single physical device [9]. Hence, all the resources are represented as virtual machine (VM) instances and allocated to cloud service customers based on the pay-as-you-go manner [3]. Each VM is an abstract unit of computing, memory and storage capacities in the cloud. The size of CPU, memory, and other resources in VMs can be configured according to the time-varying demand of consumers [8] and, accordingly the cloud resources are more effectively and efficiently utilized.

Resource management and job scheduling are the key issues of cloud computing that plays a vital role in improving the performance of the cloud system. More precisely, job scheduling is a key process for infrastructure as a service (IaaS) that is mapping the requests on resources in an efficient manner by considering cloud characteristics [10]. It takes VMs as scheduling units for allocating physical heterogeneous resources to jobs. Scheduling in cloud computing environment falls into a category of problems known as NP-hard or NP-complete problem due to some reasons such as heterogeneous and dynamic properties of resources, in addition to large solution space and large number of entry tasks with different characteristics and thus it takes a long time to find an optimal solution [11]. Although job scheduling in cloud computing has been widely studied for the last few years, there is still no algorithm that can find optimal solution within polynomial time to solve it. Therefore, instead of spending long time looking for optimal solution for scheduling in the cloud, it is desirable to find suboptimal solution, but in short period of time. Heuristic and metaheuristic-based techniques have been proved to achieve near optimal solutions within a reasonable time for such complex problems [12].

In this paper, a simplified version of particle swarm optimization-based job scheduling algorithm is proposed for scheduling of jobs in the cloud environment so as to minimize the makespan. The performance of the proposed strategy is compared to genetic algorithm through conducting an extensive simulation tests. The simulation results show that the proposed PSO technique is able to find near-optimal solutions within a reasonable time frame.

The remainder of the paper is organized as follows. Section 2 discusses related work and Section 3 formulates the scheduling problem. The description of the proposed algorithm is presented in Section 4, and in Section 5 the obtained results are presented. Finally, Section 6 concludes the paper.

2. RELATED WORK

Due to bottlenecks such as lack of scalability and elasticity, poor responsiveness and efficiency, difficulties in installation and maintenance, fault tolerance and low performance in traditional information technology (IT) frameworks, there is an urgent need to leverage modern information technology techniques and solutions to deal with the growing complexity of scientific and engineering problems. In view of this, cloud computing with its promise of provisioning virtually unlimited computing resources in a dynamic and elastic way [13], seems to be a promising alternative. With this new technological paradigm, computing resources have become cheaper, more powerful and more available than ever before [14]. However, because of cloud resources are provided as a utility, utilization of the resources is an important issue which not only influences the performance of the cloud system, but also has a direct impact on the cost issue for cloud users who run their applications, and cloud suppliers who provide the required cloud infrastructure [15, 16, 17]. Thus, scheduling the cloud resources to serve the cloud clients is considered as a major theme in cloud resource management and scheduling research [18].

Scheduling for resource utilization in cloud computing is a vital research area, which aimed at mapping the submitted tasks or jobs to the most appropriate available resources by considering some constraints such as cost, minimum makespan, task execution time, deadline, load balancing, Quality of Service (QoS), high throughput, etc. [19, 20, 17]. During the past decade, numerous researchers focus on job scheduling in the cloud environment and a large number of different algorithms have been proposed to tackle this problem [21, 22, 23]. However, due to the NP-complete nature of the problem, no feasible exact (optimal) solutions are proposed. On the other hand, metaheuristic algorithms which employ iterative strategies to find solutions in a reasonable time [24], have shown their effectiveness for solving a wide range of hard-to-solve combinatorial and multi-objective optimization problems. The most popular metaheuristic algorithms that are presented to solve NP problems are genetic algorithm (GA) [25], simulated annealing (SA) [26], tabu search (TS) [27], particle swarm optimization (PSO) [28], ant colony optimization (ACO) [29] and artificial bee colony optimization (ABC) [30]. Hence, metaheuristic techniques that are intent to find near optimal solutions are considered appropriate approaches for solving cloud scheduling issues.

In [31], the authors reviewed the advanced research and developments on resource management and task scheduling in cloud computing, and they proposed an adaptive strategy to maximize the profit earned by service providers, while it minimizes the overall cost to consumers. The work described in [32] proposes an ACO strategy to address job scheduling within a cloud. The proposed strategy is aimed to maximize scheduling throughput to accomplish all the diversified job requests according to different resources available in a cloud and minimize the makespan of cloud job scheduling. The work in [33] presents an ABC algorithm called honey bee behavior inspired load balancing, which seeks to minimize the makespan of job scheduling and balance the load across VMs in a cloud environment. In [34], the authors have exhibited a three level cloud scheduler based on swarm intelligence (SI) metaheuristics for scheduling and execution of computational mechanics applications on federated clouds. In their study, makespan and flowtime were considered as objectives. In [35] the authors have proposed a task scheduling optimizing method based on PSO to allocate jobs to resources in a cloud so as to minimize both job computation cost and job transferring time. The authors in [36] proposed a load balancing technique based on artificial neural network (ANN), which utilizes back propagation algorithm to distribute as equally as possible the workload among all the servers. In the work proposed in [37] the researchers developed a GA scheduler to schedule independent and divisible tasks in a cloud computing environment, with makespan as an objective. Furthermore, to increase the Quality of Service of the cloud system, the authors in [38] introduced an improved task scheduling algorithm to assign tasks to computing resources with the objective of maximizing the scalability and reliability of the cloud system.

In summary, by using appropriate and effective scheduling schemes, the execution time of the tasks can be minimized and the cloud resources can be fully utilized, which finally increases the availability and scalability of the entire cloud system. Our work here proposes a PSO methodology to address the issue of job scheduling in a cloud environment with makespan as an objective.

3. PROBLEM STATEMENT

In order to formulate the scheduling problem in cloud computing, we briefly explain some of the key terms relevant to the problem. In cloud computing environment, all the resources are shared by using the virtualization technology over the internet. Therefore, all the resources are represented as virtual machines (VMs) and each VM is a set of computational resources with limited capacities (e.g., processing power, memory size, network bandwidth, etc.) in the cloud. In addition, each VM has its corresponding processing speed (cycles/second). The speed of each VM can be expressed in millions of instructions per second (MIPS). Such VMs are provisioned to service the jobs submitted by cloud users. A job is considered as a single set of multiple atomic tasks. Each job has its corresponding length (processing requirement) expressed in million instructions (MI). At a given moment of time, numerous jobs can be received by the cloud provider to be performed, each with different requirements. These jobs need to be arranged for execution in such a way that allows every job to be completed in time, and efficiently utilize all the available resources.

A scheduling problem can be defined as follows: Find an optimal or near optimal solution to schedule a given set of independent user jobs $J = \{J_1, J_2, \ldots, J_n\}$ to a given set of heterogeneous virtual machines $VM = \{VM_1, VM_2, \ldots, VM_m\}$ subject to a predefined set of constraints and measurements [39]. As mentioned previously, there are several criteria that are used to evaluate the efficiency and effectiveness of the schedule solution [40]. Among them, makespan is a widely used metric for measuring the quality of a schedule in cloud environment. It is defined as the completion time of the last job. Minimization of makespan implies that no job takes a long time to execute [41]. Hence, the makespan can be formulated as follows:

$$C_i = \sum_{(j\,|A_j \in i)} VME_{ij} + W_i \tag{1}$$

$$makespan = max\{C_i\} \tag{2}$$

where C_i is the time required for virtual machine i to complete all its assigned jobs; $\{j \mid A_j \in i\}$ represents the jobs assigned to virtual machine i; VME_{ij} the time it takes VM_i to complete job J_j; $i \in 1,2,\ldots,m$; $j \in 1,2,\ldots,n$; and W_i the time for which job *j* has to wait for virtual machine i to get ready.

In job scheduling, an optimal schedule will be the one that optimizes the makespan [42]. Thus, the objective of the job scheduling is to search the schedule S that minimizes the makespan while fulfilling the schedule feasibility constraints. That is, the problem is formalized as follows:

$$\begin{aligned} Obj\text{:} \quad & f(S) = min\ \ makespan\ (S) \\ & subject\ to\ S \in F_d \end{aligned} \tag{3}$$

where F_d is a feasible region in the objective space. Our job scheduling objective in this study is to obtain an ideal solution of the mentioned problem in a reasonable period of time.

4. ALGORITHM DESIGN

Particle swarm optimization (PSO) is a population-based global search swarm intelligence metaheuristic, introduced by Kennedy and Eberhart [28] in 1995. It is inspired by social behavior of organisms such as bird flocking and fish schooling. Particles, similar to individuals, not only remember their own local best positions, but also communicate with each other and record the globally best position [43]. In PSO algorithm, the swarm of particles is stochastically generated initially, and every particle position represents a possible solution to the problem. Each particle is represented by a velocity, a location in the search space and has a memory which helps it in recalling its previous best position. In each iteration, each particle adjusts its velocity based on its best position and the position of the best particle of the entire population. Particles move around in the search space based on the particles' updated position and velocity to get an optimized/enhanced solution [44, 17], meaning that PSO utilizes the velocity update and position update to guide potential solutions to "fly" toward the globally optimal region. PSO combines local search methods with global search methods attempting to balance the exploration and exploitation.

In PSO, each particle has a position represented by a position-vector $\overrightarrow{x_i}$ (i is the index of the particle), and a velocity represented by a velocity-vector $\overrightarrow{v_i}$. At each iteration, each particle alters its searching direction based on: its previous velocity $\overrightarrow{v_i}(t)$, its best position (called *personal best*) $\overrightarrow{P_i}$ it has encountered so far and the best position $\overrightarrow{P_g}$ obtained so far by all particles in the swarm (called *global best*). That is, each particle updates its velocity and position according to Eq. (4) and Eq. (5), respectively:

$$\overrightarrow{v_i}(t+1) = w\overrightarrow{v_i}(t) + c_1 r_1(\overrightarrow{p_i} - \overrightarrow{x_i}(t)) + c_2 r_2(\overrightarrow{p_g} - \overrightarrow{x_i}(t)) \tag{4}$$

$$\overrightarrow{x_i}(t+1) = \overrightarrow{x_i}(t) + \overrightarrow{v_i}(t) \tag{5}$$

where w is called the inertia weight and it plays the role of balancing the global search and local search. A large inertia weight encourages global exploration, while a smaller inertia weight encourages local exploitation [45]. Therefore, a w value that balances global and local search implies fewer iterations in order for the algorithm to converge. In the literature, there are some experimental results demonstrate that it is better to initially set the inertia parameter to a large value so as to encourage global exploration of the search space, and gradually decrease it to obtain more accurate solutions [46, 47]. r_1 and r_2 are two random numbers ranging between 0 and 1. c_1 and c_2 are two positive constants called acceleration coefficients. They represent the attraction that a particle has either towards its best position (the cognitive part) or towards the best position of its neighbors (the social part), respectively. Therefore, tuning them properly may lead to a faster convergence and may prevent the algorithm to get caught in local minima.

Overall, in the particle swarm model, the particle searches the solutions in the problem space with a range $[-s, +s]$. Moreover, in order to keep the particles from flying out of the problem space, each component of $\overrightarrow{v_i}$ must be kept within the range $[-V_{max}, +V_{max}]$.

4.1 Encoding Mechanism

The first step of applying PSO to scheduling problem in the cloud is to encode the problem. In the PSO-based job scheduling scenario presented here, the dimension of the particles is the number of jobs that need to be allocated and each position of a particle indicates a mapping between the virtual machines and submitted jobs. In particular, each particle is encoded as a vector of integers and the vector length is set to n (number of jobs). In addition, each element in particle delegates a job. This element is an integer value between 1 and m (number of VMs), which represents the number of a computing resource (VM index). Moreover, the real values (if any) in the particles' positions resulting from updating particles velocities and positions are rounded off to the nearest integer.

4.2 Fitness Evaluation

The objective of this study is to minimize the makespan encountered by the job scheduling, subject to the resource constraints. As mentioned before, each particle corresponds to a candidate solution of the underlying problem. In other words, each particle represents a mapping of VM to a job. The evaluation of each particle is performed by a fitness function, which allows the comparison of the efficiency of one schedule to another, based on the objective, makespan. Thus, the fitness value of each schedule solution can be estimated using Eq. (3). The particle with minimum fitness value is considered as the best solution among the others.

4.3 The Proposed Algorithm

The PSO algorithm has gained its popularity due to its simplicity and its utilization in a wide range of applications in different domains. In addition to its fewer parameters and its fast convergence speed, PSO has proved to be both effective and fast for solving various optimization problems.

To simplify the PSO algorithm for scheduling in the cloud, we started by removing components of the algorithm and seeing how this influences the performance of PSO. We found that eliminating the personal memory term from the velocity update formula reinforces the case for PSO being mostly reliant on social interaction rather than personal experience. In detail, we eliminated the part containing the particle's previous best position by setting $c_1 = 0$ in Eq. (4). That is, the velocity updating formula becomes:

$$\vec{v_i}(t+1) = w\vec{v_i}(t) + c\ r\ (\vec{p_g} - \vec{x_i}(t)) \tag{6}$$

where c is a positive constant (called social parameter) and r is a random number in the range [0, 1]. This simplification of PSO algorithm is similar to the one that has been suggested by Kennedy in [48] who called it the “social only” PSO. Finally, the simplification made here, allows the particles to keep track only of the global best solution found so far to encourage social interaction as opposed to personal experience.

The main idea behind the proposed PSO algorithm is as follows. The proposed scheduler starts by collecting the information about the VMs and the job requests from the cloud user. It then adjusts the parameters for PSO (e.g., N (swarm size), w (inertia weight) and c (social parameter)). Then the scheduler generates a population of particles with random positions and velocities in the search space, meaning that it initializes position-vector and velocity-vector of each particle in the swarm. It then calculates the fitness value of each particle in the swarm using the fitness function. The scheduler then selects the particle with best fitness value from all particles as global best. The following steps will be repeated (for every particle) until the maximum number of iterations is met. The scheduler then updates the velocity-vector and checks that each element does not violate the velocity boundaries. It then updates the position-vector and rounds off the real values (if any) in the particle’s position. Finally, it evaluates the fitness value for each particle in the population and updates the *global best* if necessary. The algorithm stops when a termination condition is met. As a result, the best solution found so far is considered as the final solution. The pseudo-code of the proposed simplified PSO algorithm is as given in Algorithm 1.

Algorithm 1 Simplified PSO algorithm

```
Input the scheduling problem
Setup the parameters
Generate a swarm of particles with random positions and velocities
Calculate the fitness value of each particle in the swarm using Eq. (3)
Select the particle with best fitness value from all particles as global best
while termination criterion is not met do
    for each particle $i$ do
        Update the particle’s velocity using Eq. (6)
        Check the velocity boundaries for each component of velocity-vector
        Update the particle’s position using Eq. (5)
        Round off the real values in particle’s position into the nearest integer
        Evaluate the fitness of the particle using Eq. (3)
        if $F(\vec{x_i}) < F(\vec{p_g})$ then
            Update the global best
        end if
    end for
end while
Output the best particle (schedule) as the final solution.
```

5. RESULTS

In this section, we report the simulation results to assess the performance of the presented scheduling approach. We first present the details of the simulation environment and describe the data sets used in the experiments. Thereafter, we present the measured makespan obtained from studying the performance of the proposed algorithm and the other algorithm evaluated in this paper.

5.1 Data Sets and Experimental Settings

The simulation analysis of this study was run on a DELL PC with 2.40 GHz Intel Core i5 CPU and 4 GB of RAM. In order to evaluate the performance of the proposed algorithm, we implemented the simulations using the CloudSim simulator. CloudSim toolkit [49] is a tool for modeling and simulation of cloud computing environment. It supports dynamic creation of various types of cloudlets (jobs) and VM instances at run-time. In our experiments, we have assumed that each job/task which is submitted to the cloud may require varying processing time. VMs with different processing capacities are considered here also. The processing requirement of jobs is measured in MI, while the processing speed of VMs is measured in MIPS.

Moreover, to compare the performance of the presented approach against genetic algorithm which was aforementioned, we have utilized six different dimensions of job scheduling problem, namely, (3, 13), (5, 100), (8, 60), (10, 50), (60, 500), and (100, 1000) as used by Liu et al. in [47]. For each testing case, the notation (VM, J) is employed to indicate the number of VMs on the cloud (VM) and the number of jobs (J) to be scheduled. For example, (100, 1000) refers to the test instance with 100 VMs and 1000 jobs. Further, in order to get robust estimates for our results, we repeated each experiment 50 times and the average value of the results obtained is reported. Finally, in all experiments, the parameter settings of the evaluated scheduling algorithms are as shown in Table 1.

Table 1. Parameter settings for the evaluated algorithms

Algorithm	Parameter	Value
GA	Population size	50
	Crossover probability	0.9
	Mutation probability	0.05
	Scale for mutations	0.1
PSO	Swarm size	50
	Inertia weight w	0.72
	Social coefficient c	1.49

5.2 Results and Analysis

For the test case (3, 13), the speeds of 3 VMs are 4, 3, 2 MIPS, and the job lengths of 13 jobs are 6, 12, 16, 20, 24, 28, 30, 36, 40, 42, 48, 52, 60 MI, respectively. One of the optimal schedules for this test instance is: Allocating jobs J_3, J_7, J_8, J_{10} and J_{13} on VM$_1$, jobs J_1, J_4, J_5, J_9 and J_{11} on VM$_2$, and jobs J_2, J_6 and J_{12} on VM$_3$. In this optimal schedule: the time required for VM$_1$ to complete all its assigned jobs is 46 and the time for the other two VMs are also 46, that is the best makespan is 46. This implies that, if the workload placed on the VMs is balanced, then the resources' utilization is maximized while the makespan is minimized.

Fig. 1 shows the performance of our PSO algorithm and the GA for job scheduling problem (3, 13). The results demonstrate that the completion time of jobs for PSO is less than that of GA, meaning that the makespan obtained by the proposed PSO algorithm is better than that of GA algorithm.

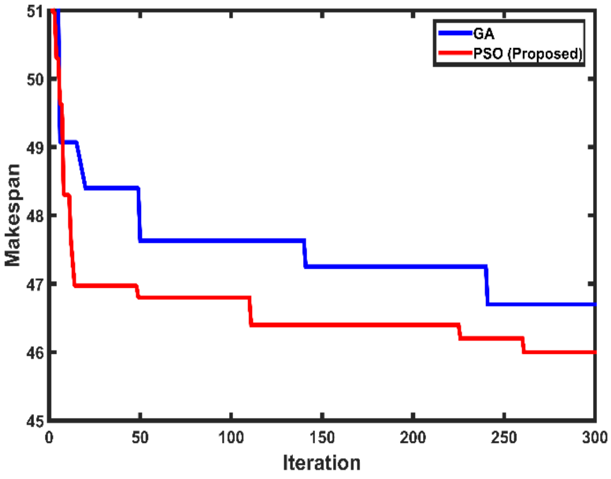

Fig. 1. Performance comparison for test case (3, 13)

Figs. 2 ~ 6 illustrate the performance of the two algorithms in terms of the makespan for the other five (VM, J) pairs, i.e. (5,100), (8, 60), (10, 50), (60, 500), and (100, 1000). For the small and middle-sized scheduling problems, the results depicted in Figs. 2, 3 and 4 show that PSO usually can find better results than the other scheduling algorithm (i.e., GA) in terms of makespan. For the large size job scheduling (60, 500) and (100, 1000) problems, the results also demonstrate that the makespan of our PSO algorithm is obviously lower than that of GA algorithm, as can be seen in Fig. 5 and Fig. 6.

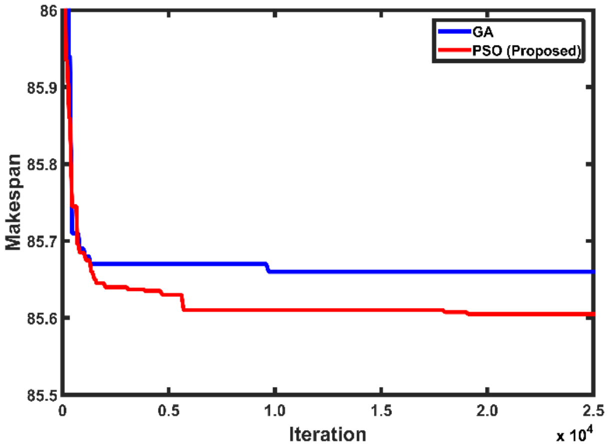

Fig. 2. Performance comparison for test case (5, 100)

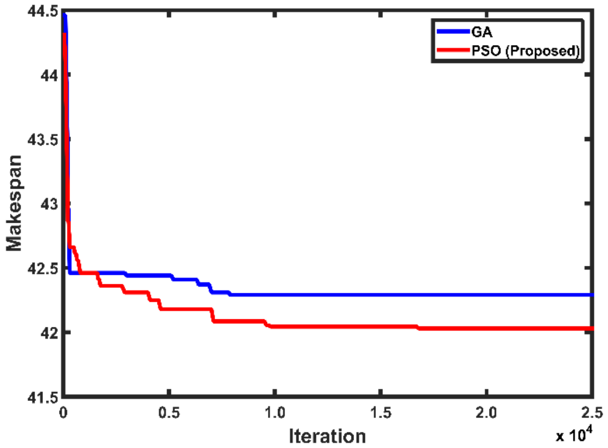

Fig. 3. Performance comparison for test case (8, 60)

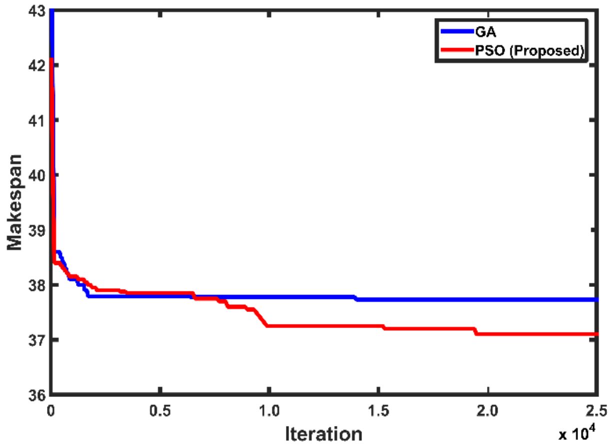

Fig. 4. Performance comparison for test case (10, 50)

To summarize, the results depict that the proposed PSO algorithm is able to handle large sized scheduling problems and has the ability to minimize the makespan of job scheduling in the cloud. From the results, it is also clear that the presented PSO algorithm converges faster than the other metaheuristic GA algorithm.

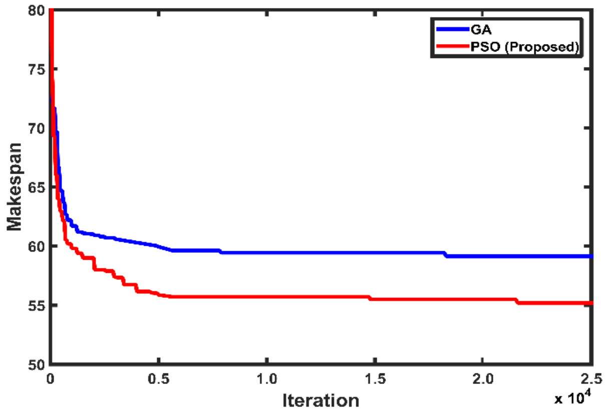

Fig. 5. Performance comparison for test case (60, 500)

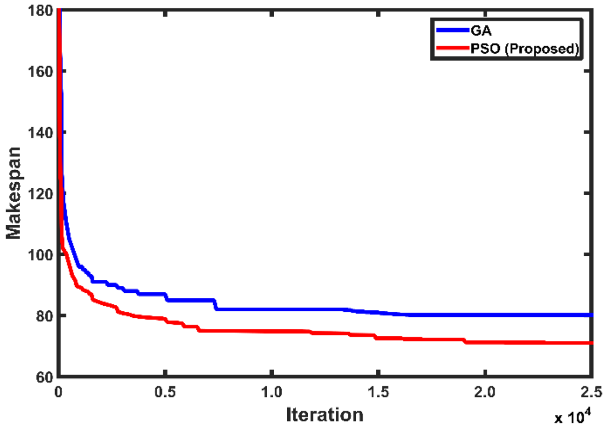

Fig. 6. Performance comparison for test case (100, 1000)

6. CONCLUSIONS AND FUTURE TRENDS

With cloud computing, we can focus on design, simulation, analysis, and discovery, instead of spending much time building, configuring and maintaining complex IT infrastructures. In cloud computing, the resources should be utilized effectively and consequently scheduling should consider the resource utilization to decrease the execution time and thereby increasing the throughput of the system. In this paper, a simplified particle swarm optimization-based job scheduling algorithm was implemented for scheduling of jobs in cloud environment in order to minimize the makespan. The performance of the proposed method was compared to genetic algorithm through carrying out extensive simulation tests and different settings. Simulation results for a large variety of test cases show that the proposed method is able to find near optimal solutions in a reasonable time. Moreover, it significantly outperforms the considered GA method in terms of makespan, specifically in the middle and large sizes of scheduling problems. In the future, our research works can address other important factors like the flowtime, overall execution cost and load balancing during the scheduling of tasks and jobs.

7. ACKNOWLEDGMENTS

The work of this paper is sponsored by the National High Technology Research and Development Program (863 Program) of China (Grant No. 2014AA041801-2). The authors would also like to thank the support of Beijing Key Laboratory of Knowledge Engineering for Materials Science.

8. REFERENCES

[1] P. M. Mell and T. Grance, “SP 800-145. The NIST Definition of Cloud Computing,” National Institute of Standards & Technology, Gaithersburg, MD, United States, 2011.

[2] I. Attiya and X. Zhang, “Cloud Computing Technology: Promises and Concerns,” *Int. J. Comput. Appl.*, vol. 159, no. 9, pp. 32–37, Feb. 2017.

[3] D.-K. Kang, S.-H. Kim, C.-H. Youn, and M. Chen, “Cost adaptive workflow scheduling in cloud computing,” in *Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication - ICUIMC ’14*, 2014, pp. 1–8.

[4] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: State-of-the-art and research challenges,” *J. Internet Serv. Appl.*, vol. 1, no. 1, pp. 7–18, 2010.

[5] B. P. Rimal, E. Choi, and I. Lumb, “A Taxonomy and Survey of Cloud Computing Systems,” in *2009 Fifth International Joint Conference on INC, IMS and IDC*, 2009, pp. 44–51.

[6] R. Buyya, S. Pandey, and C. Vecchiola, “Cloudbus Toolkit for Market-Oriented Cloud Computing,” in *Proceedings of the 1st International Conference on Cloud Computing*, M. G. Jaatun, G. Zhao, and C. Rong, Eds. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 24–44.

[7] H.-L. Truong and S. Dustdar, “Cloud computing for small research groups in computational science and engineering: current status and outlook,” *Computing*, vol. 91, no. 1, pp. 75–91, Jan. 2011.

[8] X. Liu, Z. Zhan, K. Du, and W. Chen, “Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach,” in *Proceedings of the 2014 conference on Genetic and evolutionary computation - GECCO ’14*, 2014, pp. 41–48.

[9] Z. Xiao, W. Song, and Q. Chen, “Dynamic resource allocation using virtual machines for cloud computing environment,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, 2013.

[10] A. Ghorbannia Delavar and Y. Aryan, “HSGA: A hybrid heuristic algorithm for workflow scheduling in cloud systems,” *Cluster Comput.*, vol. 17, no. 1, pp. 129–137, 2014.

[11] M. Kalra and S. Singh, “A review of metaheuristic scheduling techniques in cloud computing,” *Egypt. Informatics J.*, vol. 16, no. 3, pp. 275–295, Nov. 2015.

[12] S. R. Shishira, A. Kandasamy, and K. Chandrasekaran, “Survey on meta heuristic optimization techniques in cloud computing,” in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016, pp. 1434–1440.

[13] S. N. Srirama, O. Batrashev, P. Jakovits, and E. Vainikko, “Scalability of parallel scientific applications on the cloud,” *Sci. Program.*, vol. 19, no. 2–3, pp. 91–105, 2011.

[14] M. G. Avram, “Advantages and Challenges of Adopting Cloud Computing from an Enterprise Perspective,” *Procedia Technol.*, vol. 12, pp. 529–534, 2014.

[15] H. Zhang, P. Li, Z. Zhou, and X. Yu, "A PSO-Based Hierarchical Resource Scheduling Strategy on Cloud Computing," in *Trustworthy Computing and Services: International Conference, ISCTCS 2012, Beijing, China, May 28 -- June 2, 2012, Revised Selected Papers*, Y. Yuan, X. Wu, and Y. Lu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 325–332.

[16] W. Li, J. Tordsson, and E. Elmroth, "Modeling for dynamic cloud scheduling via migration of virtual machines," in *Proceedings - 2011 3rd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2011*, 2011, pp. 163–171.

[17] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A Survey of PSO-Based Scheduling Algorithms in Cloud Computing," *J. Netw. Syst. Manag.*, vol. 25, no. 1, pp. 122–158, Jan. 2017.

[18] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches," *ACM Comput. Surv.*, vol. 47, no. 4, pp. 1–33, 2015.

[19] R. Nallakumar, N. Sengottaiyan, and S. P. K. S, "A Survey on Scheduling and the Attributes of Task Scheduling in the Cloud," *Int. J. Adv. Res. Comput. Commun. Eng.*, vol. 3, no. 10, pp. 8167–8171, 2014.

[20] C. Lin and S. Lu, "Scheduling scientific workflows elastically for cloud computing," in *Proceedings - 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011*, 2011, pp. 746–747.

[21] M. L. Pinedo, *Scheduling: Theory, algorithms, and systems: Fourth edition*. Boston, MA: Springer US, 2012.

[22] D. M. Lei, "Minimizing makespan for scheduling stochastic job shop with random breakdown," *Appl. Math. Comput.*, vol. 218, no. 24, pp. 11851–11858, 2012.

[23] S. S. Kim, J. H. Byeon, H. Yu, and H. Liu, "Biogeography-based optimization for optimal job scheduling in cloud computing," *Appl. Math. Comput.*, vol. 247, pp. 266–280, 2014.

[24] C. W. Tsai and J. J. P. C. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *IEEE Syst. J.*, vol. 8, no. 1, pp. 279–291, 2014.

[25] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[26] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science (80-.).*, vol. 220, no. 4598, pp. 671–680, May 1983.

[27] F. Glover, "Future Paths for Integer Programming and Links to Artificial Intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, May 1986.

[28] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948.

[29] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 53–66, Apr. 1997.

[30] A. Walker, J. Hallam, and D. Willshaw, "Bee-havior in a mobile robot: the construction of a self-organized cognitive map and its use in robot navigation within a complex, natural environment," in *IEEE International Conference on Neural Networks*, 1993, pp. 1451–1456 vol.3.

[31] F. Pop and M. Potop-Butucaru, "ARMCO: Advanced topics in resource management for ubiquitous cloud computing: An adaptive approach," *Futur. Gener. Comput. Syst.*, vol. 54, pp. 79–81, Jan. 2016.

[32] S. Banerjee, I. Mukherjee, and P. K. Mahanti, "Cloud Computing Initiative using Modified Ant Colony Framework," *Eng. Technol.*, vol. 56, no. 8, pp. 221–224, 2009.

[33] L. D. Dhinesh Babu and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Appl. Soft Comput. J.*, vol. 13, no. 5, pp. 2292–2303, 2013.

[34] E. Pacini, C. Mateos, C. G. Garino, C. Careglio, and A. Mirasso, "A bio-inspired scheduler for minimizing makespan and flowtime of computational mechanics applications on federated clouds," *J. Intell. Fuzzy Syst.*, vol. 31, no. 3, pp. 1731–1743, 2016.

[35] X. Wang, C. S. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Futur. Gener. Comput. Syst.*, vol. 27, no. 8, pp. 1124–1134, Oct. 2011.

[36] S. A. A. A. Nada M. Al Sallami Ali Al daoud, "Load Balancing with Neural Network," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 10, pp. 138–145, 2013.

[37] C. Zhao, S. Zhang, Q. Liu, J. Xie, and J. Hu, "Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing," in *2009 5th International Conference on Wireless Communications, Networking and Mobile Computing*, 2009, pp. 1–4.

[38] S. Selvarani and G. S. Sadhasivam, "Improved cost-based algorithm for task scheduling in cloud computing," in *2010 IEEE International Conference on Computational Intelligence and Computing Research*, 2010, pp. 1–5.

[39] C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A Hyper-Heuristic Scheduling Algorithm for Cloud," *IEEE Trans. Cloud Comput.*, vol. PP, no. 99, pp. 1–1, 2014.

[40] I. Attiya, X. Zhang, and X. Yang, "TCSA : A Dynamic Job Scheduling Algorithm for Computational Grids," in *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, 2016, pp. 408–412.

[41] J. A. Torkestani, J. Akbari Torkestani, and J. A. Torkestani, "A new approach to the job scheduling problem in computational grids," *Cluster Comput.*, vol. 15, no. 3, pp. 201–210, 2011.

[42] S.-S. Kim *et al.*, "Optimal job scheduling in grid computing using efficient binary artificial bee colony optimization," *Soft Comput.*, vol. 17, no. 5, pp. 867–882,

2013.

[43] A. Y. S. Lam and V. O. K. Li, “Chemical Reaction Optimization for Task Scheduling in Grid Computing,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 10, pp. 1624–1631, 2011.

[44] Eberhart and Yuhui Shi, “Particle swarm optimization: developments, applications and resources,” in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*, 2001, vol. 1, pp. 81–86.

[45] Y. Shi and R. C. Eberhart, “Empirical study of particle swarm optimization,” in *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, 1999, vol. 3, pp. 1945–1950.

[46] J. Kennedy and R. Mendes, “Population structure and particle swarm performance.pdf,” *Proc. 2002 Congr. Evol. Comput. CEC 2002*, pp. 1671–1676, 2002.

[47] H. Liu, A. Abraham, and A. E. Hassanien, “Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm,” *Futur. Gener. Comput. Syst.*, vol. 26, no. 8, pp. 1336–1343, 2010.

[48] J. Kennedy, “The particle swarm: social adaptation of knowledge,” in *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC ’97)*, 1997, pp. 303–308.

[49] R. N. Calheiros, 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,” *Softw. Pract. Exp.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.