International Journal of Grid and Distributed Computing
Vol.7, No.1 (2014), pp.33-42
http://dx.doi.org/10.14257/ijgdc.2014.7.1.04

Cost-Aware Scheduling Algorithm Based on PSO in Cloud Computing Environment

Gang Zhao

Teachers' skill-training Center, Mianyang Normal University,
Mianyang, Sichuan, China
2577477@qq.com

Abstract

Traditional scheduling algorithms typically aim to minimize the total time cost for processing all tasks. However, in cloud computing environments, computing capability differs for different resources, and so does the cost of resource usage. Therefore, it is vital to take into consideration the usage cost of resources. Along this line, in this paper, we proposed a modified algorithm based on PSO to solve the task scheduling problem in cloud computing environments. Specifically, by adding a cost-aware fitness function to quantify the cost of resource usage, along with the fitness function for time cost, our method can achieve the goal of minimizing both the processing time and resource usage, and therefore reach a global optimal solution. Besides, our experiment on a simulated cloud computing environment proves the efficiency of our proposed algorithm.

Keywords: *Particle Swarm Optimization (PSO), scheduling algorithm, cloud computing*

1. Introduction

Recent years have witness the development of cloud computing, which extends grid computing [1] and parallel computing [2]. The basic idea of cloud computing is to split a task into several parts over computing and/or storage resources, which make up a large scale system and send the results back to users.

Typically, the numbers of tasks and resources are extremely huge in cloud computing environment, especially for big data applications [3], the problem of task scheduling has become a major challenge. There exist some current efforts along this line. For instance, Hou *et al.*, [4] formulated the task scheduling problem as a general task graph to be executed on a multiprocessor system so that the schedule length can be minimized, and employed genetic algorithms for optimization. Wu *et al.*, [5] proposed an improved Min-Min algorithm by scheduling large tasks first. Buyya *et al.*, [6] presented a distributed computational economy as an effective metaphor for the management of resources and application scheduling, which enables the regulation of supply and demand for resources and provides an incentive for resource owners for participating in the grid.

However, existing research puts more emphasis on minimizing the processing time of tasks, while minimizing the cost of task scheduling should also be balanced. In cloud computing environments, computing capability differs for different resources, and so does the cost of usage. For example, for a resource with weak computing capability, the cost of usage is relative low; and vice versa. In this paper, we combine time and cost together for task scheduling problem, and employ a Particle Swarm Optimization (PSO) based algorithm to achieve a balance between the minimization of processing time and total cost.

ISSN: 2005-4262 IJGDC

The remain of this paper is organized as follows. Section 2 discusses some related works, and Section 3 provides the problem statement. Our method for solving scheduling problem based on PSO in cloud computing environment is proposed in Section 4. In Section 5, empirical experiments are conducted. In the end, Section 6 concludes this paper.

2. Related Work

In this section we provide a brief review about the research on task scheduling algorithms. Many researchers have proposed various scheduling algorithms, such as Min-min [5], round bin [6], genetic algorithm [7], *etc*.

There exist a great variety of studies on task scheduling algorithms. Early scheduling algorithms on multiple processors were typically developed based on the DAG and processor network model [17, 18]. Coffman *et al.*, [19] proposed a scheduling algorithm for arbitrary structured task graphs for two homogeneous processors. Ibarra and Kim [20] developed a heuristic algorithm to schedule tasks without dependencies on a homogeneous multiprocessor system, called Min-min. Braun *et al.*, [21] compared eleven heuristic methods for mapping tasks onto heterogeneous processors, with no consideration of communication delay.

PSO algorithm is adopted in the field of task mapping, resource allocation and scheduling. For example, PSO could be able to find near optimal solution for mapping all tasks in the workflow for the given set of resources [22]. Tasgetiren *et al.*, [23] proposed a PSO based algorithm for resource allocation by keeping the track of position value and fitness value calculated from a fitness function. Xue *et al.*, [24] developed an algorithm for time cost optimization on scheduling workflow applications. Unlike their work, our algorithm is proposed to optimize both time and resource usage costs for task scheduling.

Cost-aware scheduling algorithms in cloud computing environment are also studied. For example, problems related to scheduling service requests of consumers with consideration of both consumers and providers [15, 16]. We observe that existing efforts emphasize on the quality of services (QoS) from the perspective of service provisioning. However, in this paper, we aim to minimize the total cost from the perspective of providers, including both total processing time and total usage cost of resources (*i.e.*, computing or storage resources).

3. Problem Statement

In cloud computing environment, a large scale computing task is typically split into several parts, and the mapping between tasks and resources is achieved through virtualization technologies. The problem of task scheduling is defined as mapping n independent tasks onto m heterogeneous resources, with the goal of minimizing the processing time and maximizing the resource utilization.

Suppose the set of tasks is $T = \{t_1, t_2, ..., t_n\}$, where n is the number of tasks, and t_i is either a original task or a sub-task. The set of resources is $R = \{r_1, r_2, ..., r_m\}$, where m is the number of resources. Expected Time Cost (ETC) for task t_i on resource r_j is notated as ETC_{ij}, and Expected Resource Cost (ERC) per time unit for resource r_j is notated as ERC_i. The goal of scheduling is to create a mapping between tasks and resources $f : T \rightarrow R$. In this paper, we employ a PSO based algorithm to find f , so that both total EPT and total RUC are minimized.

Note that in this study, we have the following assumptions. (1) There exists no dependence between tasks; (2) All resources are of exclusive usage and cannot be shared among different tasks. The latter assumption indicates that a resource can only be reassigned when it is freed after the completion of processing.

4. PSO Based Task Scheduling

In this section, we describe the core algorithm for scheduling problem based on PSO. We begin by introducing the basics of PSO algorithm.

4.1. PSO Basics

Particle Swarm Optimization (PSO) algorithm has become a new evolutionary computation technique because of its high-performance and flexibility. It was developed upon the analog of the social behavior of blocking birds by Kennedy and Eberhart [10, 11]. The basic idea for PSO is, if one of the individuals finds a good path to the predation location, more followers would be attracted. This helps to improve the search for best path because all population are involved.

PSO works as follows. Every particle in this swarm behavior has two characters: a position x which notates the suggested location, and a velocity v which means the speed of moving. Each particle traverses over the whole search space and remembers the best position found. Communication is made between particles so that they could adjust their locations and velocities based on solutions discovered by others. Each position is scored by a fitness function f to quantify how good the solution is. Specifically, at iteration k , particle i keep track of two values: local best position p_{LBi} and global best position p_{GB} , both of which are calculated from f . Therefore, update function is defined as follows:

$$v_i(k+1) = w * v_i(k) + c_1 * \gamma * (p_{LBi}(k) - p_i(k)) + c_2 * \gamma * (p_{GB}(k) - p_i(k)) \qquad (1)$$

$$p_i(k+1) = p_i(k) + v_i(k+1) \qquad (2)$$

where $v_i(k)$ is the velocity of particle i at iteration k , $p_i(k)$ is the current position of i at iteration k , γ is a random number between 0 and 1, c_1, c_2 are learning factors and w is an inertia weight (typically 0.1~0.9). As proved in [11], w is linearly reduced and contributes to convergence.

Indicated by Equation (1), the update of velocity is composed of three parts: (1) previous velocity ($v_i(k)$), which is called *habitual behavior*; (2) attraction toward previous best position ($p_{LBi}(k)$), known as *memory* or *self-knowledge*; (3) attraction toward global best position ($p_{GB}(k)$), called *social knowledge*. Figure 1 illustrates the basic PSO algorithm using update functions (1) and (2). Table 1 lists the notations used throughout the paper.

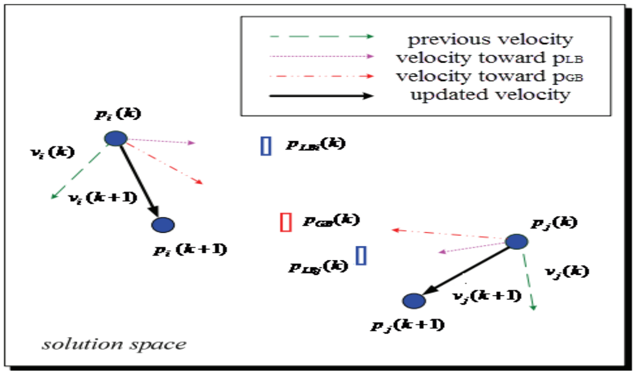


Figure 1. Illustration of PSO

Table 1. List of Notations

symbol	description
S	Population of particles
n	Number of tasks
m	Number of resources
k	iteration
r	resource
t	task
ETC_{ij}	processing time of task t_i on resource r_j
ERC_i	usage cost of resource r_j per time unit
$v_i(k)$	velocity of particle i at iteration k
$p_i(k)$	current position of i at iteration k
$p_{LBi}(k)$	history best position of particle i at iteration k
$p_{GB}(k)$	global best position at iteration k

4.2. Cost-Aware Scheduling Algorithm Based on PSO

In this section, we propose a cost-aware scheduling algorithm based on PSO.

As defined earlier, ETC_{ij} notates processing time of task t_i on resource r_j, and ERC_i means the cost of resource r_j per time unit. Therefore, the total time for processing all tasks is represented as:

$$T_{total} = \max_{r=1}^{m} \sum_{t=1}^{n_r} T(r,t) \tag{3}$$

where $T(r,t)$ means the time cost of resource r to process task t, and n_r is the number of tasks assigned on r.

Suppose the set of tasks $T = \{t_1, t_2, ..., t_n\}$ is composed of both tasks and subtasks. Notate the j -th subtask of task i as $t_{i,j}$. Therefore, the finishing time of task t is:

$$T_t = \max_{i=1}^{n_t} \sum_{j=1}^{k} T(j, i) \tag{4}$$

where $T(j, i)$ is the time cost of processing subtask j on the resource which handles task i, k is the position of resource to which subtask i is assigned for task t, and n_t is the number of subtasks of t.

On resource r, total time for processing all tasks is calculated as:

$$T_r = \sum_{i=1}^{m} T(r, i) \tag{5}$$

and total cost for processing all tasks is represented as:

$$Cost_r = \sum_{r=1}^{m} T_r \times ERC_r \tag{6}$$

Suppose the population of particles is S, and m, n are the numbers of total resources and tasks. We initialize by randomly generating S particles, and the location for i -th particle is represented as $\mathbf{p}_i = \{p_{i1}, p_{i2}, ..., p_{in'}\}$, where $1 \le n' \le n, 1 \le i \le S$, and p_{ij} means task j is assigned to node $\mathbf{p}_i$ ($1 \le p_{ij} \le m$). Velocity $\mathbf{v}_i = \{v_{i1}, v_{i2}, ..., v_{in'}\}$, where $1 \le n' \le n, 1 \le i \le S$. Initialize p_{ij} as a random integer between 1 and m, and v_{ij} as a random integer between $-(m-1)$ and $(m-1)$.

Instead of simply optimizing the processing time, we also take the usage cost of resources into consideration, which is called *cost-aware*. In this paper, we have two fitness functions corresponding to time and usage cost respectively. Define the fitness function for time cost as:

$$F_t(i) = \frac{1}{T_{total}(i)}, 1 \le i \le S \tag{7}$$

where $T_{total}(i)$ is the total time of particle i, as calculated from Equation (3).

The fitness function for usage cost is defined as:

$$F_c(i) = \frac{1}{Cost_r(i)}, 1 \le i \le S \tag{8}$$

where $Cost_r(i)$ is the total usage cost of resource r for particle i, as calculated from Equation (6).

We employ a selection strategy inspired by Genetic Algorithm [12], that is, particles with higher fitness values are selected. The less time cost and resource used, the higher score a particle gets, so that it would be more likely to be selected.

During each iteration, particles update themselves according to both previous and global best positions. Suppose previous best position for particle i is

$\mathbf{p}_{LB\,i} = \{ p_{LB\,i1}, p_{LB\,i2}, ..., p_{LB\,in}, \}$, and the global best position among all particles is $\mathbf{p}_{GB\,i} = \{ p_{GB\,i1}, p_{GB\,i2}, ..., p_{GB\,in}, \}$. At iteration k , best positions are generated using Equations (7) and (8) as follows:

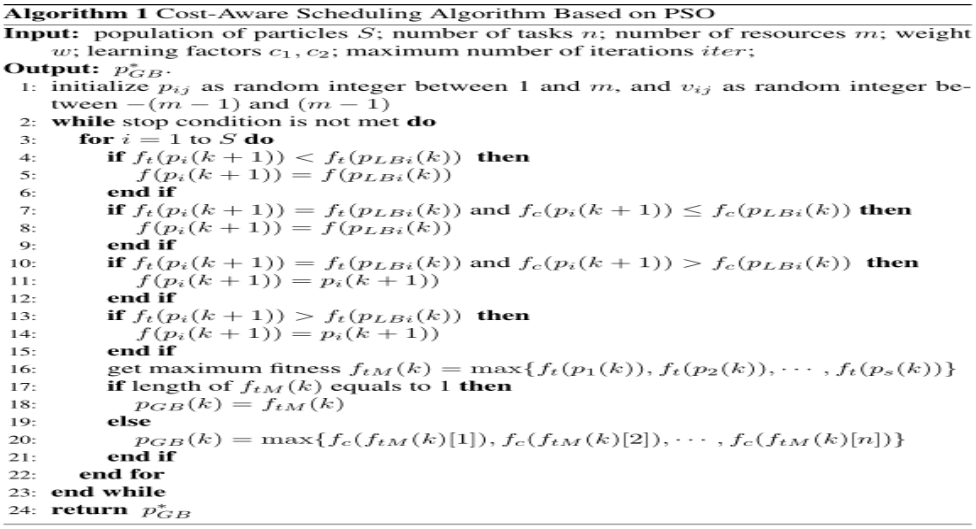
$$\mathbf{p}_{LBi}(k+1) = \begin{cases} p_{LBi}(k) & if \quad f_t(p_i(k+1)) < f_t(p_{LBi}(k)) \\ p_{LBi}(k) & if \quad f_t(p_i(k+1)) = f_t(p_{LBi}(k)) \quad \cap \quad f_c(p_i(k+1)) \le f_c(p_{LBi}(k)) \\ p_i(k+1) & if \quad f_t(p_i(k+1)) = f_t(p_{LBi}(k)) \quad \cap \quad f_c(p_i(k+1)) > f_c(p_{LBi}(k)) \\ p_i(k+1) & if \quad f_t(p_i(k+1)) > f_t(p_{LBi}(k)) \end{cases} \tag{9}$$

$$f_{tM}(k) = \max \left\{ f_t(p_{LB\,1}(k)),\ f_t(p_{LB\,2}(k)),...,\ f_t(p_{LBs}(k)) \right\} \tag{10}$$

$$\mathbf{p}_{GB}(k) = \begin{cases} f_{tM}(k), & \text{if } length\ (f_{tM}(k)) = 1 \\ \max\{ f_c(f_{tM}(k)[1]),\ f_c(f_{tM}(k)[2]),...,\ f_c(f_{tM}(k)[n'])\}, & \text{if } length\ (f_{tM}(k)) > 1 \end{cases} \tag{11}$$

Then, every particle updates its position and velocity by Equations (1) and (2). As the values of $\mathbf{p}_{\mathbf{LB}}$ and $\mathbf{p}_{\mathbf{GB}}$ are optimized after iterating, the final value of $\mathbf{p}_{\mathbf{GB}}$ is the global best solution.

Figure 2 summarizes the procedure of our proposed algorithm.



```
Algorithm 1 Cost-Aware Scheduling Algorithm Based on PSO
Input: population of particles $S$; number of tasks $n$; number of resources $m$; weight
    $w$; learning factors $c_1, c_2$; maximum number of iterations $iter$;
Output: $p^*_{GB}$.
initialize $p_{ij}$ as random integer between 1 and $m$, and $v_{ij}$ as random integer between $-(m-1)$ and $(m-1)$
while stop condition is not met do
    for $i = 1$ to $S$ do
        if $f_t(p_i(k+1)) < f_t(p_{LBi}(k))$ then
            $f(p_i(k+1)) = f(p_{LBi}(k))$
        end if
        if $f_t(p_i(k+1)) = f_t(p_{LBi}(k))$ and $f_c(p_i(k+1)) \le f_c(p_{LBi}(k))$ then
            $f(p_i(k+1)) = f(p_{LBi}(k))$
        end if
        if $f_t(p_i(k+1)) = f_t(p_{LBi}(k))$ and $f_c(p_i(k+1)) > f_c(p_{LBi}(k))$ then
            $f(p_i(k+1)) = p_i(k+1))$
        end if
        if $f_t(p_i(k+1)) > f_t(p_{LBi}(k))$ then
            $f(p_i(k+1)) = p_i(k+1))$
        end if
        get maximum fitness $f_{tM}(k) = \max\{f_t(p_1(k)), f_t(p_2(k)), \cdots, f_t(p_s(k))\}$
        if length of $f_{tM}(k)$ equals to 1 then
            $p_{GB}(k) = f_{tM}(k)$
        else
            $p_{GB}(k) = \max\{f_c(f_{tM}(k)[1]), f_c(f_{tM}(k)[2]), \cdots, f_c(f_{tM}(k)[n])\}$
        end if
    end for
end while
return $p^*_{GB}$
```

Figure 2. Procedure of Proposed Algorithm

5. Experiments

In this section, we present our experiments to evaluate the performance of our proposed method, and compare with the traditional PSO algorithm.

5.1. Experimental Settings

In order to evaluate the performance of our proposed algorithm, we employ Matlab to produce matrices ETC and ERC. First, we use CloudSim [13] to simulate the cloud computing environment. We compare our modified algorithm with the basic PSO algorithm. We run the experiments 100 times and get the average results as performance metrics.

Parameter settings are made in light of Bin *et al.*, [14], shown as Table 2. We use the same parameter settings for both algorithms. The number of subtasks for each task is [30, 70], and the expected processing time ranges from 1 to 30 seconds. The termination condition is either 1) maximum number of iterations is reached, or (2) total time and usage costs remain unchanged for continuous 60 executions.

Table 2. Parameter Settings

item	value
Population of particles S	150
Number of tasks n	50
Number of resources m	50
Weight w	0.9
Learning factor c_1	2
Learning factor c_2	2
Maximum number of iterations	200

5.2. Performance Results

Total time and usage costs for 50 tasks are shown in Figures 3 and 4 respectively.

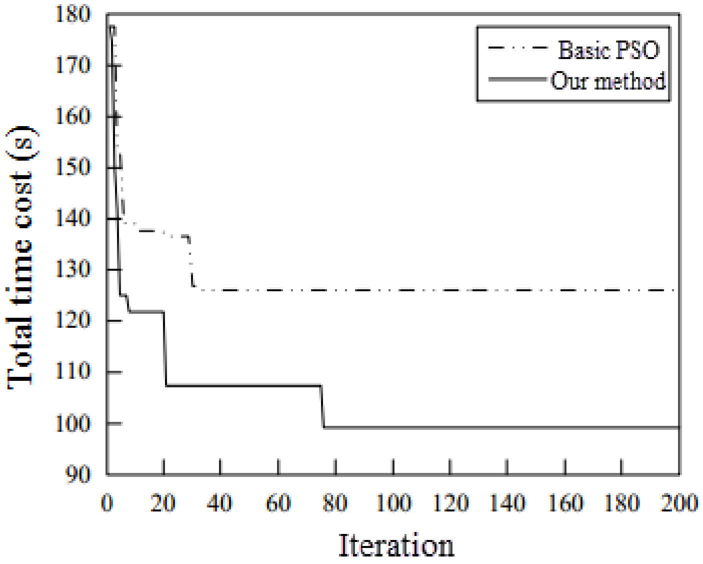


Figure 3. Total Time Cost for 50 Tasks

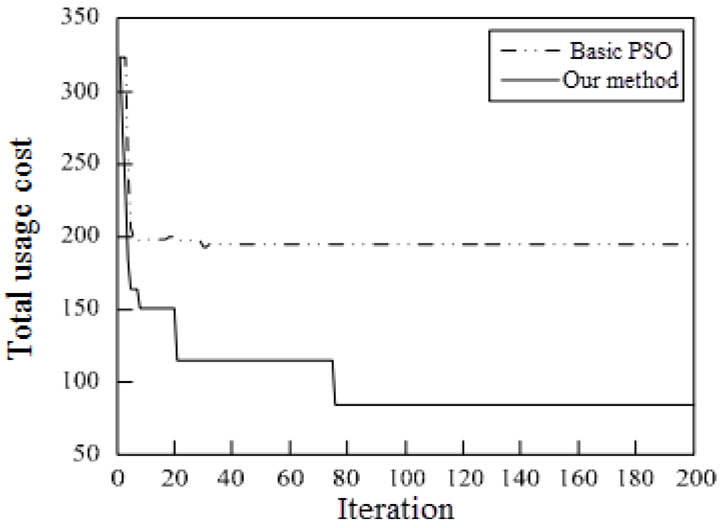


Figure 4. Total Usage Cost for 50 Tasks

From Figures 3 and 4, we have two observations. (1) The processing at first several iterations for both PSO and our method are almost identical, which means our method is not worse than PSO. (2) As the iterations grow, total costs are decreasing, and our method outperforms the basic PSO algorithm.

Besides, Figure 5 depicts the total time cost with different numbers of tasks. We can observe that the number of tasks plays an important role on performance. Specifically, the more tasks we have, the more difficult to reach a optimal solution.

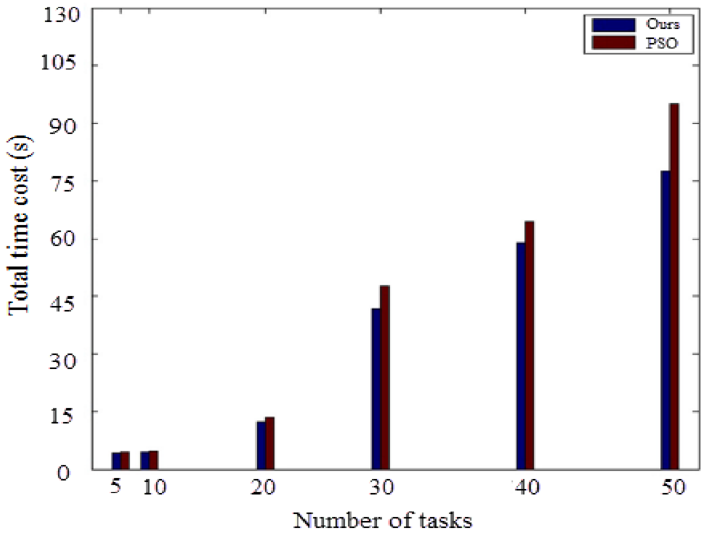


Figure 5. Total Time Cost with Different Numbers of Tasks

Figure 6 shows the total time cost with different numbers of resources (*i.e.*, processors). As we can see, performance is also affected by the number of resources. Specifically, the more resources available, the easier to find a optimal solution.

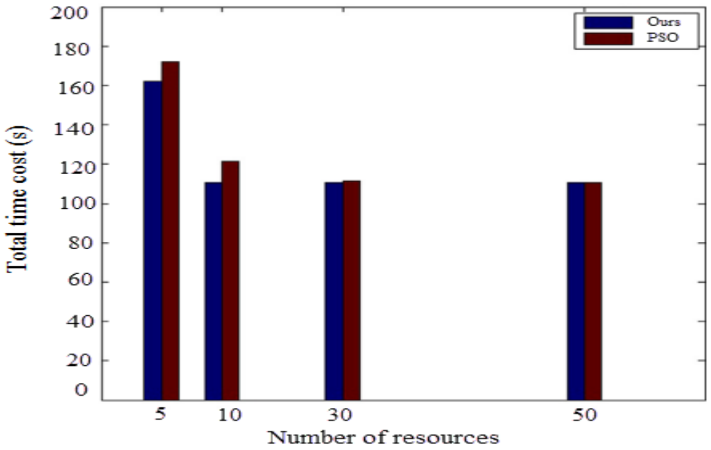


Figure 6. Total Time Cost with Different Numbers of Resources

From above experiments, we can see that traditional PSO algorithm simply emphasizes on the time cost only, which leads to lost of excellent varieties during iterations. Accordingly, the speed of convergence is accelerated; however, the solution is typically a local optimal instead of a global one, which contributes to a larger cost of both time and usage. In this paper, we add a cost-aware fitness to quantify the cost of resource usage, so that the total time cost is reduced, and resource usage is optimized as well.

6. Conclusion

Traditional scheduling algorithms in cloud computing environment fail to take the resource usage cost into consideration, but simply aim to minimize the total time cost. In this paper, we proposed a modified algorithm based on PSO to solve the task scheduling problem in cloud. Specifically, by adding a cost-aware fitness to quantify the cost of resource usage, along with the fitness for time cost, our method can minimize both the processing time and resource usage and reach a global optimal solution. Besides, we conducted empirical experiments in a simulated cloud computing environment. The results indicated that our algorithm outperforms the traditional PSO algorithm. In future work, we will try to investigate the influence of data distribution on scheduling, especially for skew data distribution.

Acknowledgements

The authors would like to thank anonymous reviewers for their useful comments and language editing which have greatly improved the manuscript.

References

[1] F. Berman, G. Fox and A. J. G. Hey, “Grid computing: making the global infrastructure a reality”, Wiley. Com, vol. 2, **(2003)**.

[2] V. Kumar, A. Grama, A. Gupta and G. Karypis, “Introduction to parallel computing”, Redwood City: Benjamin/Cummings, vol. 110, **(1994)**.

[3] D. Agrawal, S. Das and A. El Abbadi, “Big data and cloud computing: current state and future opportunities”, Proceedings of the 14th International Conference on Extending Database Technology, ACM, **(2011)**.

[4] E. S. H. Hou, N. Ansari and H. Ren, “A genetic algorithm for multiprocessor scheduling, Parallel and Distributed Systems”, IEEE Transactions, vol. 5, no. 2, **(1994)** February, pp. 113-120.

[5] M.-Y. Wu, W. Shu and H. Zhang, "Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems", Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th. IEEE, **(2000)**.

[6] R. Buyya, "Economic-based distributed resource management and scheduling for grid computing", arXiv preprint cs/0204048, **(2002)**.

[7] F. A. Omara and M. A. Mona, "Genetic algorithms for task scheduling problem", Journal of Parallel and Distributed Computing, vol. 70, no. 1, **(2010)**, pp. 13-22.

[8] V. Sarkar and J. Hennessy, "Compile-time partitioning and scheduling of parallel programs", ACM, vol. 21, no. 7, **(1986)**.

[9] L. Bianco, "Scheduling multiprocessor tasks on a dynamic configuration of dedicated processors", Annals of Operations Research, vol. 58, no. 7, **(1995)**, pp. 493-517.

[10] J. Kennedy and R. C. Eberhart, "Particle swarm intelligence", Procs of the International Conference on Neural Network, **(1995)**.

[11] Y. Shi and R. Eberhart, "A modified particle swarm optimizer", Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on. IEEE, **(1998)**.

[12] D. Whitley, "A genetic algorithm tutorial", Statistics and computing, vol. 4, no. 2, **(1994)**, pp. 65-85.

[13] R. N. Calheiros, R. Ranjan, C. A. F. De Rose and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services", arXiv preprint arXiv: 0903.2525, **(2009)**.

[14] L. Bin and L. Wenfeng, "Simulation based optimization for parameter selection in PSO", Computer Engineering and Applications, vol. 47, no. 33, **(2011)**, pp. 30-35.

[15] C. Yan, H. Luo, Z. Hu, X. Li and Y. Zhang, "Deadline Guarantee Enhanced Scheduling of Scientific Workflow Applications in Grid", Journal of Computers, vol. 8, no. 4, **(2013)**, pp. 842-850.

[16] D. Dutta and R. C. Joshi, "A genetic: algorithm approach to cost-based multi-QoS job scheduling in cloud computing environment", Proceedings of the International Conference & Workshop on Emerging Trends in Technology, ACM, **(2011)**, pp. 422-427.

[17] J. Bruno, E. G. Coffman Jr, and R. Sethi, "Scheduling independent tasks to reduce mean finishing time", Communications of the ACM 17, no. 7, **(1974)**, pp. 382-387.

[18] G. and N. Harold, "An almost-linear algorithm for two-processor scheduling", Journal of the ACM (JACM), vol. 29, no. 3, **(1982)**, pp. 766-780.

[19] Ass Prof E. G. Coffman Jr. and R. L. Graham, "Optimal scheduling for two-processor systems", Acta Informatica, vol. 1, no. 3, **(1972)**, pp. 200-213.

[20] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on non-identical processors", Journal of the ACM, vol. 24, no. 2, **(1977)**, pp. 280-9.

[21] T. D. Braun, H. J. Siegal, N. Beck, L. L. Boloni, M. Maheswaran, A. I. Reuther and J. P. Robertson, "A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems", Heterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth, IEEE, **(1999)**, pp. 15-29.

[22] 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, **(2008)**, pp. 37-43.

[23] M. Fatih Tasgetiren, Y.-C. Liang, M. Sevkli and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem", European Journal of Operational Research, vol. 177, no. 3, **(2007)**, pp. 1930-1947.

[24] S.-J. Xue and W. Wu, "Scheduling Workflow in Cloud Computing Based on Hybrid Particle Swarm Algorithm", TELKOMNIKA Indonesian Journal of Electrical Engineering, vol. 10, no. 7, **(2012)**, pp. 1560-1566.

Author

Gang Zhao], Graduated from Southwest University in China in the year 2000, majoring in Science of Computer. Now, he is a lecturer in Mianyang Normal University and focuses on the teaching of computer basic as well as the application technology of database.