International Journal of Electrical and Computer Engineering (IJECE)
Vol. 8, No. 5, October 2018, pp. 3852~3859
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i5.pp3852-3859

An Effective PSO-inspired Algorithm for Workflow Scheduling

Toan Phan Thanh[1], Loc Nguyen The[2], Said Elnaffar[3], Cuong Nguyen Doan[4], Huu Dang Quoc[5]
[1,2]Hanoi National University of Education, Ha Noi, Viet Nam
[3]American University of Ras al Khaimah, UAE
[4]Military Institute of Science and Technology, Ha Noi, Viet Nam
[5]Thuong Mai University, Ha Noi, Viet Nam

Article Info

Article history:

Received Jan 21, 2018
Revised Apr 18, 2018
Accepted Apr 25, 2018

Keyword:

Minimization makespan
Optimization problem
Particle swarm optimization
cloud computing
Workflow scheduling

ABSTRACT

The Cloud is a computing platform that provides on-demand access to a shared pool of configurable resources such as networks, servers and storage that can be rapidly provisioned and released with minimal management effort from clients. At its core, Cloud computing focuses on maximizing the effectiveness of the shared resources. Therefore, workflow scheduling is one of the challenges that the Cloud must tackle especially if a large number of tasks are executed on geographically distributed servers. This entails the need to adopt an effective scheduling algorithm in order to minimize task completion time (makespan). Although workflow scheduling has been the focus of many researchers, a handful efficient solutions have been proposed for Cloud computing. In this paper, we propose the LPSO, a novel algorithm for workflow scheduling problem that is based on the Particle Swarm Optimization method. Our proposed algorithm not only ensures a fast convergence but also prevents getting trapped in local extrema. We ran realistic scenarios using CloudSim and found that LPSO is superior to previously proposed algorithms and noticed that the deviation between the solution found by LPSO and the optimal solution is negligible.

Corresponding Author:

Toan Phan Thanh,
Hanoi National University of Education,
Ha Noi, Viet Nam.
Email: pttoan@hnue.edu.vn

1. INTRODUCTION

The Cloud is a computing platform that provides convenient, on-demand access to a shared pool of configurable computing resources such as networks, servers and storage [1]. Workflow scheduling is one of the challenges that the Cloud must tackle especially if a large number of tasks are executed on the geographically distributed servers. This demands the adoption of a reasonable scheduling algorithm in order to attain a minimal completion time (called makespan).

The rest of the paper is organized as follow. Section 2 reviews some of the related works germane to workflow scheduling algorithms. Section 3 describes the computation and communication model on which Cloud tasks operate. Based on this model, Section 4 presents our proposed scheduling algorithm LPSO (Local-search Particle Swarm Optimization). Section 5 describes the experiments we conducted using the CloudSim simulation tool [2] in order to evaluate the proposed algorithm. Section 6 concludes our paper and sketches future work.

2. RELATED WORK

2.1. Approaches for workflow scheduling problems

A workflow is a sequence of connected tasks. Workflow scheduling in Clouds is challenge because each task needs to be mapped to an appropriate server while enabling that task to satisfy some performance

constraints. In general, the scheduling problem, i.e., the mapping of tasks to the computation resources such as servers, is an NP-complete problem [3]. Hence, past works banked mostly on heuristic-based solutions for scheduling workflows.

For example, S. Parsa [4] proposed a scheduling algorithm that minimizes the makespan of the workflow in the Grid environment. A. Agarwal [5] studied the greedy algorithm which assigned an appropriate priority sequence numbers to tasks. J. Huang [6] proposed a workflow task scheduling algorithm based on genetic algorithm. S. Pandey [7] presented an effective scheduling algorithm (PSO_H) to minimize the cost of the execution. R. Buyya [2] presented a brief description of CloudSim, the useful simulation toolkit that used in this paper to simulate the execution of the tasks with different scheduling policy.

J. Jintao [8] proposed a task scheduling algorithm based on service quality and the advantages of the Min-min algorithm. Guo-Ning and Ting-Lei [9] presented an optimized algorithm for task scheduling based on Hybrid Genetic Algorithms. The authors covered in their study the QoS requirements like completion time, bandwidth, cost, distance, reliability of different types of tasks. L. Guo [10] presented a model for task scheduling in Cloud to minimize the overall time of execution and transmission. L. Guo proposed the PSO algorithm (Particle Swarm Optimization) which is based on the small position value rule. R. Rajkumar [11] proposed a hierarchical scheduling algorithm that helps satisfy service level agreement with quick response from the service provider. S. J. Xue [12] proposed the hybrid PSO algorithm to minimize the cost execution of the workflow. Crossover and mutation of genetic algorithm are embedded into the PSO algorithm to improve the global search. J. Liu In *et al* [13] presented the components of an intelligent job scheduling system in cloud computing.

2.2. The particle swarm optimization method

The Particle Swarm Optimization (PSO) is one of the latest evolutionary optimization techniques introduced in 1995 by Kennedy and Eberhart [14]. There are many studies which succeed PSO strategy such as [15], [16]. They proposed the formula of updating the position vector as follows:

$$v_i^{k+1} = \omega v_i^k + c_1\, rand_1 \times (pbest_i - x_i^k) + c_2\, rand_2 \times (gbest - x_i^k) \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^k \quad (2)$$

where

a. v_i^k , v_i^{k+1} : velocity of particle i at iteration k and $k+1$
b. x_i^k, x_i^{k+1} : position of the particle i at iteration k and $k+1$
c. ω : inertia weight; c_1, c_2 : acceleration coefficients
d. $rand_1$, $rand_2$: random number between 0 and 1
e. $pbest_i$: best position of particle i; $gbest$: position of best particle in a population

The goal of PSO is to find the position that minimizes the fitness function denoted by: $Fitness(gbest) \rightarrow Min$

2.3. Topological neighborhood for the PSO

The standard PSO has no neighborhood relationship, all of particles are directly connected to each other so there are no neighborhood relationships between them. The position of each particle is updated according to its personal best position (*pbest*) and the global best position among all the particles (*gbest*). However, various personal relationships, such as parent-child relationships, in real world do exist. This compelled some researchers [17] to propose topological neighborhood between particles in PSO's. Researches [17] have applied various topological neighborhoods such as the Ring neighborhood and Von Neuman neighbourhood where each particle shares its local best position among neighboring particles in the topological space. For this reason each particle is affected by the local best (*lbest*) in its local neighborhood instead of *pbest*. In PSOs that use a local best position, the formula for updating the position vector is

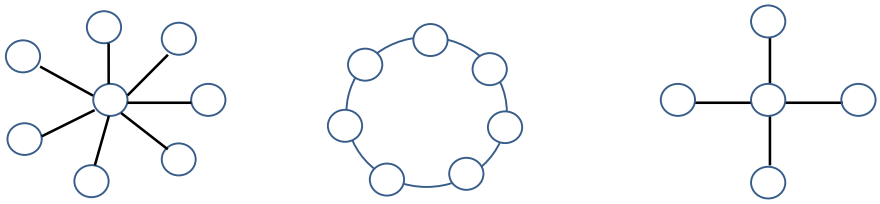
$$v_i^{k+1} = \omega \times v_i^k + c_1\, rand_1 \times (pbest_i - x_i^k) + c_2 rand_2 \times (lbest_i - x_i^k) \quad (3)$$

where $lbest_i$ is the local best position of particle i with the best fitness value among its neighbors.

As shown in Figure 1, the neighborhood relationships are determined based on each topology. For example, in the Ring topology, each particle has k neighbors. In this paper we set k=2 so each particle x_i connects directly to its left-neighbor (Left(x_i)) and its right-neighbor (Right(x_i)). Based on the Ring topology, we build a searching function described as follows

Function Ring(x_i)

Input: current position x_i
Output: x where Fitness(x) = min{Fitness(x_i), Fitness(Left(x_i)), Fitness(Right(x_i))}



(a) Star topology (b) Ring topology (c) Von neumann topology

Figure 1. Neighborhood topologies

3. PROBLEM FORMULATION

We denote the workflow as a Directed Acyclic Graph (DAG) represented by G=(V, E), where:

a. V is a set of vertex, each vertex represents a task
b. T={$T_1, T_2, \ldots, T_M$} is the set of tasks, M is the number of tasks
c. E represents the data dependencies between these tasks. The edge (T_i, T_j) ∈ E means the task T_i is the father of the task T_j, the data produced by T_i will be consumed by the task T_j.
d. The Cloud's computation resource, set of servers S = {$S_1, S_2, \ldots, S_N$}. N is the number of servers.
e. Each task T_i can be executed by any server $S_j \in$ S, and S_i has to handle whole the workload of T_i
f. The computation of task T_i denoted by W_i (flop-floating point operations)
g. P(S_i): the computation power of the server S_i (MI/s : million instructions/second)
h. The bandwidth B(S_i,S_j) between server S_i and server S_j represents by the function B(): S×S → R^+. We assume that B(S_i,S_i) = ∞ and B(S_i,S_j) = B(S_j,S_i)
i. D_{ij}: data produced by task T_i and consumed by task T_j.

Each scheduling plan can be represented by the function f(): T→S where $f(T_i)$ is the server which handles the task T_i.

Under the above assumptions, we may compute:

a. The execution time of the task T_i is

$$\frac{W_i}{P(f(T_i))} \tag{4}$$

b. The communication time between the task T_i and T_j is

$$\frac{D_{ij}}{B(f(T_i), f(T_j))} \tag{5}$$

Formally, we seek to minimize the execution time of the workflow: *makespan* → min
where the execution time, called *makespan*, is the time difference between the start and finish of a sequence of workflow's tasks.

4. PROPOSED ALGORITHM

4.1. Escaping local extremum

During their execution, PSO-based algorithms may get trapped in local extrema. Our proposed idea to escape such local extrema is as follows: when the swarm falls into the area around the local extrema, we combine the PSOs in order to have a topological neighborhood with a neighborhood searching function [18] that moves particles to a new area.

Variable Neighborhood Searching Function in order to help the swarm escape from the area around the local extrema, we devised two operators named Exchange and RotateRight, as illustrated in Figure 2, and built a Variable_Neighborhood_Searching function based on these operators.

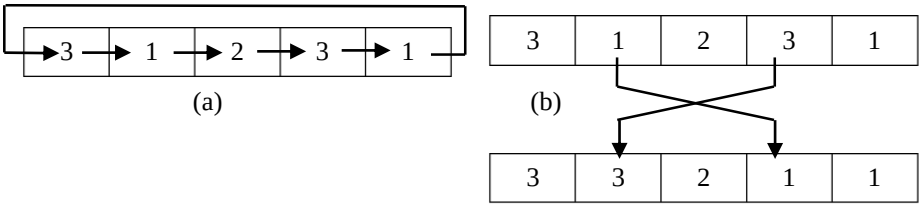


Figure 2. Operator rotateright (a) and Operator exchange (b)

Function Variable_Neighborhood_Searching ( )

Input: position vector *xi*
Output: position vector *xk* : Fitness(*xk*) < Fitness(*xi*)
Begin
1. *t* := 0;
2. while (Fitness(*xk*) > Fitness (*xi*) and (t < *Max_Iteration*)
3. *r* := random [1,*M*];
4. *xi* := RotateRight(*xi*, *r*);
5. *rand1* := [1,*M*]; *rand2* := [1,*M*];
6. *xk* := Exchange (*xi*, *rand1*, *rand2*);
7. if Fitness(*xk*) < Fitness(*xi*) then return *xk* else return *xi*;
8. *t* := *t*+1;
9. end while

End.

Note: If the function cannot find a better position than the current position(x_i) within the *Max_Iteration* limit, x_i is returned.

4.2. The LPSO algorithm

The LPSO algorithm can be described as follows:

Algorithm LPSO ( )

Input: T, S, size of workload W[1×*M*], P[1×*N*], B[*N*×*N*], D[*M*×*M*], the constant K, the deviation ε, the number of particle *NoP*
Output: the best position *gbest*
Begin
1. For *i*:=1 to *NoP* do
2. x_i:= random vectors; v_i:= random vectors;
3. end for
4. *t*:= 0 ;
5. While (the deviation of *gbest* > ε) Do
6. for *i*:=1 to *NoP* do
7. Compute new position x_i
8. end for
9. for *i*:=1 to *NoP* do
10. Update $pbest_i$;
11. end for
12. Update *gbest*;
13. for *i*:=1 to *NoP* do
14. $lbest_i$:= Ring(x_i) ;
15. end for
16. for *i*:=1 to *NoP* do
17. Update v_i^k and compute x_i ;
19. end for
20. *t*++ ;
21. if (the deviation of *gbest* ≤ ε after *K* generations) then *gbest*:= Variable_Neighborhood_Searching *(gbest)*;
23. End while;
24. Return *gbest;*

End.

In each iteration, the LPSO updates the position vectors of particles based on *gbest* and *lbest* using formulas (2) and (3). If the deviation of *gbest* less than ε during *K* continuous generations, this means that the swarm is trapped in a local extremum area, and hence the function *Variable_Neighbourhood_Searching( )* should be called. This function moves (migrates) the swarm to a new area and produces a new generation.

If *gbest* is not improved significantly, i.e. the deviation of *gbest* is still less than ε after *K* continuous migrations upon calling the function *Variable_Neighbourhood_Searching( )*, LPSO halts. In our experiments, we set *K*=100 and ε = 0.21. In the best case, LPSO can find the absolute position upon calling the function *Variable Neighbourhood Searching( )* *K* times, leading to spawning K^2 generations.

In the worst case, LPSO always finds a better position after the function *Variable_Neighbourhood_Searching( )* is executed without getting trapped in a local extrumum, rendering LPSO an exhaustive search. Our default threshold for number of generations is 300. The LPSO stops upon reaching this threshold.

5. RESULTS AND DISCUSSION

We conducted some experiments in order to compare the performance of the LPSO algorithm with others, namely the PSO_H [7] and Random [19]. Our experimental setup consists of a computer with Intel Core i5 2.2 GHz, RAM 4GB, and Windows 7 Ultimate. The experiments were carried out using the CloudSim simulation package, the packet library Jswarm [20] and Java.

5.1. Problem instance

We use both random and real world instances in our experiments using the following data sets:

a. The computation power of the servers and the bandwidth of connections between servers are collected from Cloud firms such as Amazon [21] and their Web site (exp. http://aws.amazon.com/ec2/pricing)
b. The sets of working data are collected from the Montage project [22]

We denote the ratio of the number of edges and the number of vertexes of graph G as follows:

$$\alpha = \frac{|E|}{M \times (M-1)/2}$$

5.2. Configuration parameters

The Cloud's configuration parameters are choosen as follows:

a. Server's computation power: from 1 to 250 (million instructions/s)
b. Connection bandwidth B: from 10 to 100 (Megabit/s)
c. Communication data D: from 1 to 10000 (Megabit)
d. $\omega = 0.729$; $c_1 = c_2 = 1.49445$; $K = 30$, Deviation $\varepsilon = 0.21$,
e. Number of particles *NoP*=25 ; $\varepsilon = 0.21$; α : from 0.2 to 0.7

5.3. Results

Each problem instance was executed 30 times continuously. The results summarized in Table 1 show that the mean value (listed in column *Mean*) and standard deviation value (listed in column *STD*) of LPSO are better than those of PSO_H [7] and Random [19] in most of the cases. When the number of servers (*N*) and the number of tasks (*M*) are relatively large (i.e. larger scale cloud), for example *M*=20 and *N*=8; *M*=25, *N*=8; *M*=50, *N*=8, LPSO outperforns PSO_H and Random with respect to all metrics: mean, standard deviation and best value (listed under column *Best*).

Since the number of server (*N*) is a finite integer number, the elements of the position vector (denoted by $x_i^k[t]$) must be integer numbers (t =1..*M*) too. In Equation (2), the value of the left hand side x_i^{k+1} is an integer number while the value of the right hand side ($x_i^k + v_i^k$) is a real number. Pandey [7] resolved this situation by rounding the real value of the right hand side to the nearest integer. For example, if $x_i^k[t] + v_i^k[t] = 3.2$ then task T_t gets assigned to server S_3. If $x_i^k[t] + v_i^k[t] = 3.8$ then T_t gets assigned to server S_4. Inevitably, this introduces some sort of randomness in the assignment of servers in the PSO_H algorithm [7], and hence it can not maintain the diversification of swarm. For this reason, PSO_H often gets trapped in local extrema.

Alternatively, we propose a novel method in which we assign the left hand side x_i^{k+1} to the server whose computation power is the closest to ($x_i^k + v_i^k$).

$$x_i^{k+1}[t] \leftarrow j \text{ if } \left| P(S_j) - (x_i^k[t] + v_i^k[t]) \right| \le \left| P(S_r) - (x_i^k[t] + v_i^k[t]) \right| \qquad \forall S_r \in S \; ; \; t = 1,2 \; .. \; M$$

In other words, the new particle's position is the one which renders the task to be assigned to the server that has the closest computation power to the real value computed from the position vector. The results described in Table 1 show that the mean value (the *Mean* column) and standard deviation value (the *STD* column) of LPSO are better than those of PSO_H [7] and Random [19] in most of the cases. The solutions of LPSO are smaller than the solutions of PSO_H with a value difference varying from 1% to 12%. The LPSO's standard deviations are smaller than the PSO_H's with a value difference varying from 53% to 84%. These

results show that LPSO is stable and better than both the PSO_H [7] and Random [19]. Table 2 shows the comparison the makespan of LPSO with other algorithms for Montage workflows (seconds).

Figure 3, Figure Figure 4 Figure 5 and Figure 6 depict the performance of the three algorithms: proposed algorithm LPSO, PSO_H [7], and Random [19] where the vertical axis represents the makespan of the schedule in seconds. For each instance, we compare the best position vector (column *BEST*), the mean value (column *MEAN*) and standard deviation value (column *STD*). At the first instance, LPSO was even able to find the optimal solution.

Table 1. Comparison the Makespan of LPSO with other Algorithms for Random Workflows (Seconds)

M	N	α	LPSO			PSO_H			RANDOM		
			Best	Mean	STD	Best	Mean	STD	Best	Mean	STD
10	3	0.26	16.2	18.2	1.5	16.4	20.4	2.4	21.4	28.6	3.2
10	5	0.26	75.6	81.0	5.0	86.0	107.5	13.2	123.3	184.1	42.4
20	5	0.15	28.5	34.2	3.1	29.6	41.0	5.0	45.8	59.0	6.8
20	3	0.31	122.7	128.4	3.6	130.6	142.1	4.8	140	161.8	8.4
25	8	0.3	228.4	236.1	6.1	235.1	260.3	15.0	271.9	359.0	39.9
50	8	0.3	11.1	12.6	0.8	12.1	14.0	0.9	13.9	87.1	25.2

Table 2. Comparison the Makespan of LPSO with other Algorithms for Montage Workflows (Seconds)

M	N	LPSO			PSO_H			RANDOM		
		Best	Mean	STD	Best	Mean	STD	Best	Mean	STD
20	5	421.4	437.7	9.3	440.1	461.1	10.9	450.2	540.2	44.6
20	5	118.7	123.4	3.3	122.8	132	5.4	143.8	156.9	9.0
25	8	228.4	236.1	6.1	235.1	260.3	15.0	271.9	359.0	39.0
25	3	311.6	312.5	0.5	311.7	315.4	4.0	324.4	389.3	43.9
50	8	91.1	101.7	5.5	95.0	108.0	6.3	110.5	196.8	32.8

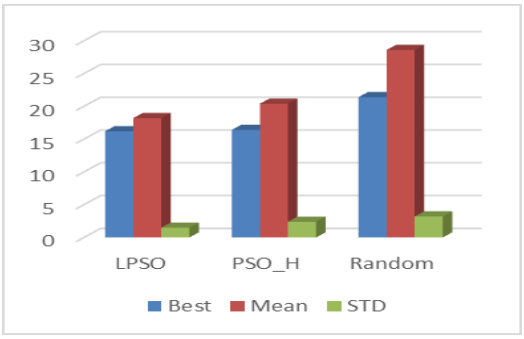


Figure 3. M=10, N=3

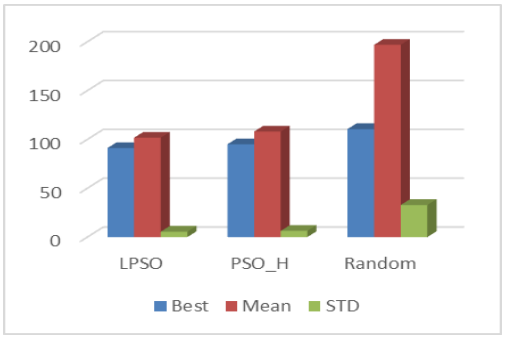


Figure 4. M=20, N=3

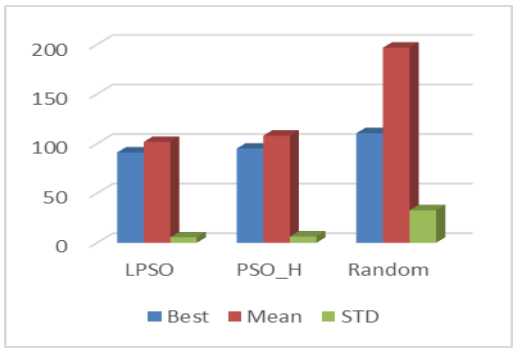


Figure 5. Montage, M=20, N=5

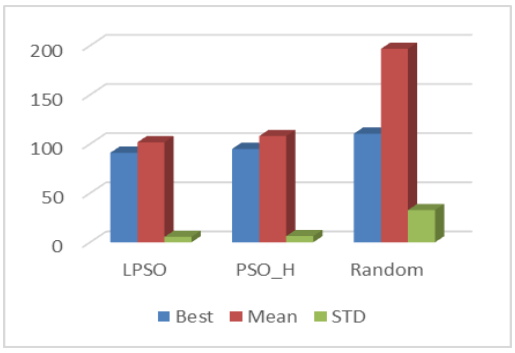


Figure 6. Montage, M=50, N=8

6. CONCLUSION

The ultimate goal of any scheduling algorithm is to minimize the execution time. In this work, we showed is advantageous as it avert getting trapped in local extrema. The contributions of our paper are:

a. Building a novel approach, represented by the function *Variable_Neighbourhood_Searching*, to help optimization algorithms escape from a local extremum.
b. Proposing a new scheduling algorithm named LPSO by incorporating the PSO strategy and function *Variable Neighbourhood Searching*.

The experimental results show that LPSO is superior to its predecessor especially when LPSO works in a larger scale Cloud. In the future, we wish to investigate how to improve the LPSO algorithm in order to solve bigger instances within a reasonable makespan.

REFERENCES

[1] Lao Zhihong, Larisa Ivascu, "*Cloud Computing Resource Dynamic Optimization Considering Load Energy Balancing Consumption*", *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 14, no. 2A, pp. 18-25, ISSN: 1693-6930, 2016.
[2] R. Buyya, R. Calheiros, "Modeling and Simulation of Scalable Cloud Environment and the Cloud Sim Toolkit: Challenges and Opportunities", *Proceedings of the High Performance Computing and Simulation Conference HPCS*, ISBN: 978-1-4244-4907-1, IEEE Press, New York, USA, pp. 1-11, 2009.
[3] J. D. Ullman, "NP-Complete Scheduling Problems", *Journal of Computer and System Sciences*, vol. 10, no. 3, pp. 384-393, 1975.
[4] S. Parsa and R.E. Maleki, "RASA: A New Task Scheduling Algorithm in Grid Environment", *International Journal of Digital Content Technology and its Applications*, vol. 3, no. 4, 2009.
[5] A. Agarwal and S. Jain, "Efficient Optimal Algorithm of Task Scheduling in Cloud Computing Environment", *International Journal of Computer Trends and Technology*, vol. 9, 2014.
[6] J. Huang, "The Workflow Task Scheduling Algorithm Based on the GA Model in the Cloud Computing Environment", *Journal of Software*, vol. 9, 2014.
[7] S. Pandey, *et al*., "A Particle Swarm Optimization (PSO)-based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", *Proc. of 24th IEEE International Conference on Advanced Information Networking and Applications (AINA)*, pp. 400-407, 2010.
[8] Jiao Jintao, *et al*., "Research on Batch Scheduling in Cloud Computing", *TELKOMNIKA (Telecommunication, Computing, Electronics and Control)*, vol. 14, no. 4, pp. 1454-1461, ISSN: 1693-6930, 2016.
[9] G. Guo-Ning and H. Ting-Lei, "Genetic Simulated Annealing Algorithm for Task Sched-uling based on Cloud Computing Environment", *Proceedings of International Conference on Intelligent Computing and Integrated Systems*, pp. 60-63, 2010.
[10] L. Guo, *et al*., "Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm", *Journal of Networks*, vol. 7, no. 3, pp. 547-552, ISSN 1796-2056, 2012.
[11] R. Rajkumar and T. Mala, "Achieving Service Level Agreement in Cloud Environment using Job Prioritization in Hierarchical Scheduling", *Proceeding of International Conference on Information System Design and Intelligent Application, Springer Verlag Berlin Heidelberg*, vol. 132, pp. 547-554, ISBN 978-3-642-27442-8, 2012.
[12] S.J. Xue and W. Wu, "Scheduling Workflow in Cloud Computing Based on Hybrid Particle Swarm Algorithm", *Indonesian Journal of Electrical Engineering*, vol. 10, pp. 1560-1566, 2012.
[13] J. Liu, *et al*., "An Intelligent Job Scheduling System for Web Service in Cloud Computing", *Indonesian Journal of Electrical Engineering*, vol. 11, pp. 2956-2961, 2013.
[14] J. Kennedy and R.C. Eberhart, "Particle swarm optimization", *Proceeding of IEEE International Conference on Neural Networks*, pp. 1942-1948, 1995.
[15] K. Lenin, "Embellished Particle Swarm Optimization Algorithm for Solving Reactive Power Problem", *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, vol. 5, no. 3, pp. 192-198, 2017.
[16] Salam Waley Shneen, *et al*., "Advanced Optimal PSO, Fuzzy and PI Controller with PMSM and WTGS at 5Hz Side of Generation and 50Hz Side of Grid", *International Journal of Power Electronics and Drive System (IJPEDS)*, vol. 7, no. 1, pp. 173-192, ISSN: 2088-8694, 2016.
[17] A. E. M. Zavala, "EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation IIA Comparison, A Comparison Study of PSO Neighborhoods", *Springer Verlag Berlin Heideberg*, pp. 251-295, ISBN 978-3-642-32725-4, 2013.
[18] H. Liu, *et al*., "A Novel Variable Neighborhood Particle Swarm Optimization for Multi-objective Flexible Job-Shop Scheduling Problems", *Proc. of 2nd International Conference on Digital Information Management (ICDIM '07)*, vol. 1, pp. 138-145, 2007.
[19] M. Mitzenmacher and E. Upfal, "Probability and Computing: Randomized Algorithms and Probabilistic Analysis", Cambridge University Press, ISBN-13:858-1000053552, 2005.
[20] R. N. Calheiros *et al*., "CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", *Software- Practice and Experience*, vol. 41, no. 1, pp. 23-50, John Wiley & Sons, Inc. USA, 2011.
[21] J. V. Vliet and F. Paganelli, "Programming Amazon EC2", O'Reilly Media, ISBN: 1449393683, 2011.
[22] http://montage.ipac.caltech.edu

BIOGRAPHIES OF AUTHORS

Toan Phan Thanh received Bachelor and M.S. degree in School of Information and Communication Technology, Hanoi University of Science and Technology, Viet Nam, in 1998 and 2001. He worked at Hanoi National University of Education, Viet Nam from 2003. His research interests include Cloud Computing, Computer Network and Software Engineering.

Loc Nguyen The received Bachelor and M.S. degree in School of Information and Communication Technology, Hanoi University of Science and Technology, Viet Nam, in 1998 and 2001, respectively. He received Ph.D. degree in School of Information Science, Japan Advanced Institute of Science and Technology, Japan, 2007. He worked at Hanoi National University of Education, Viet Nam from 1997 and is currently a professor, vice dean of the Department of Information Technology. His research interests include Computer Network and Computer Graphics.

Dr. Elnaffar received his Ph.D. in Computer Science from Queen's University (ON, Canada) 2004. He got his M.Sc. in computer science from Queen's University in 1999. He worked as an Adjunct Assistant Professor in the School of Computing at Queen's University (September-December 2004). From 2000 until 2004, Dr. Elnaffar worked as a Research Associate at the IBM Centre of Advanced Studies (CAS) in Canada. He worked as an Associate Professor (2005-2014) in the College of Information Technology, UAE University (UAE). Presently, he is an Associate Professor of Computer Science in the School of Engineering at the American University of RAK (UAE). Dr. Elnaffar is also an entrepreneur and helped many firms with his industrial experience and professional consultation. Dr. Elnaffar's research has been funded by numerous governmental organizations, and industrial agencies. His work is published in several international journals and IEEE/ACM Conferences of Web services, Grid computing, Systems performance and Evaluation, and applied AI. He is the winner of the Teaching Award in 2008 (UAE University) and the best research paper in computing education (2013).

Cuong Nguyen Doan received Bachelor in Le Qui Don University, Viet Nam. He received Ph.D. degree in Saint- Peterburg Electrotechnical University, Russia in 2006. He worked at Institute of Information Technology Military Institute of Science and Technology Ha Noi, Viet Nam from 2007. His research interests include Data mining and Software Engineering.

Huu Dang Quoc received Bachelor and M.S. degree in School of Information Technology, Vietnam National University, Ha Noi, Viet Nam, in 2000 and 2015. He worked at Thuong Mai University, Ha Noi, Viet Nam from 2006. His research interests include Computer Network and Software Engineering.