Journal of Computer Sciences

Original Research Paper

LPSO: Another Algorithm for Workflow Scheduling in the Cloud

[1]Toan Phan Thanh, [1]Loc Nguyen The, [2]Said Elnaffar and [3]Cuong Nguyen Doan

[1]*Hanoi National University of Education, Ha Noi, Vietnam*
[2]*American University of RAK, UAE*
[3]*Military Institute of Science and Technology, Ha Noi, Vietnam*

Article history
Received: 23-05-2016
Revised: 24-11-2016
Accepted: 11-12-2016

Corresponding Author:
Toan Phan Thanh
Hanoi National University of Education, Ha Noi, Vietnam
Email: pttoan@hnue.edu.vn

Abstract: Although workflow scheduling problem has been discussed by many researchers, a few efficient solutions have been introduced for Cloud computing. In this article, we present LPSO, a novel algorithm for workflow scheduling. Based on the Particle Swarm Optimization method, our proposed algorithm not only ensures the fast convergence but also avoid being trapped on local extrema. Our simulation experiments using CloudSim testing real scenarios reveal that LPSO is superior to formerly proposed algorithms. Moreover, the deviation between the solution found by LPSO and the optimal solution is negligible.

Keyword: Workflow Scheduling, Particle Swarm Optimization Cloud Computing

Introduction

Cloud computing emerged with the promise of securing on-demand and convenient access to shared computing resources such as storage, servers and networks. Scheduling is one of the challenges that are encountered when processing workflow tasks over geographically distributed servers. An effective solution for that problem requires a reasonably efficient scheduling algorithm in order to minimize the completion time (called *makespan*) of tasks. The rest of the article is structured as follow. Section 2 surveys some of the related research germane to workflow scheduling algorithms. Section 3 describes the communication and computation model on which Cloud tasks operate. Based on this model, section 4 presents our proposed scheduling algorithm Local-search PSO (LPSO). Section 5 describes the simulation experiments we ran using the CloudSim platform (Ullman, 1975) in order to evaluate our algorithm. Section 6 concludes the article by giving pointers to potential future work.

Related Work

Workflow Scheduling: Problems and Approaches

A workflow is a sequence of connected tasks. Scheduling tasks in the Cloud is challenge because each task needs to be mapped to an suitable server in order to satisfy some performance constraints. In general, this scheduling problem has been proved to be NP-complete (Ullman, 1975). Hence, past works mainly banked on heuristic-based solutions for scheduling workflows.

For example, Parsa and Maleki (2009) presented a Grid-based solution that minimizes the makespan of workflows. Agarwal and Jain (2014) managed to assign a suitable priority sequence number to a task using a greedy algorithm. Huang (2014) suggested a scheduling solution for workflows that is based on genetic algorithms. Pandey *et al*. (2010) proposed an effective scheduling solution (PSO_H) that reduced the execution cost. In this work, we use the simulation kit, CloudSim (Buyya *et al*., 2009) to simulate the execution of the tasks with different scheduling policy.

Using Hybrid Genetic Algorithms, Guo-Ning and Ting-Lei (2010) presented an optimized algorithm for task scheduling. The authors investigated several QoS requirements such as cost, distance, bandwidth, completion time, reliability of various types of tasks. Guo *et al*. (2012) introduced a scheduling solution in the Cloud that minimizes the overall execution and transmission time. Based on small position value rule, he proposed the PSO algorithm. Rajkumar and Mala (2012) proposed an hierarchical scheduling algorithm which helps satisfy different levels of service agreements with prompt response from the service providers. Xue and Wu (2012) proposed the hybrid PSO algorithm to minimize the execution cost of the workflow. The PSO algorithm relies on crossover and mutation of genetic algorithms in order to improve the global search. In (Liu *et al*., 2013),

the researchers presented an intelligent scheduling system for jobs that are processed in the Cloud.

The Particle Swarm Optimization Method

Kennedy and Eberhart (1995) introduced an evolutionary optimization technique called Particle Swarm Optimization (PSO). For updating the position vector, they proposed the following formula:

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

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

Where:

v_i^k, v_i^{k+1} = Velocity of particle i at iteration k and k+1

x_i^k, x_i^{k+1} = Position of particle i at iteration k and k+1

ω = Inertia weight; c_1, c_2: Acceleration coefficients

$rand_1$, $rand_2$ = Random number between 0 and 1

$pbest_i$ = Best position of particle i; $gbest$: position of best particle in a population

The goal of PSO is find the position that minimizes the fitness function, denoted by:

$$Fitness(gbest) \rightarrow Min$$

Topological Neighborhood for the PSO

In the original version of the PSO algorithm, all particles are directly connected to each other so there are no neighborhood relationships between them. The new position of an particle is determined based on the global best position among all the particles (*gbest*) and on its personal best position (*pbest*). However, various personal relationships, such as parent-child relationships, in real world do exist. This compelled some researchers (Zavala, 2013) to propose topological neighborhood between particles in PSO's. Researches have applied various topological neighborhoods such as the Ring neighborhood, Von Neuman neighbourhood etc (Calheiros *et al*., 2011) where each particle shares its local best position with other 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:

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

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

As shown in Fig. 1, the neighborhood relationships are determined based on each topology. For example, in the Ring topology, each particle has k neighbors. In this study 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 follow:

Function Ring(x_i)
Input: current positionx_i
Output*:* $x$where Fitness(x) = min{Fitness(x_i), Fitness(Left(x_i)), Fitness(Right(x_i))}

Problem Formulation

A workflow can be denoted as a Directed Acyclic Graph (DAG) represented by G = (V, E), where:

- V is set of vertex, each vertex represents a task
- T = {T_1, T_2,..., T_M} is the set of tasks, M is the number of tasks
- E represents the data dependencies between these tasks. The edge (T_i, T_j) ∈ E denotes that task T_i is the parent of the task T_j and that the data spawned by $T_i$are consumed by task T_j
- S = {S_1, S_2,....,S_N} is the set of N computation servers
- The workload of each task $T_i$should be fully executed by any server S_j∈S
- The computation of taskT_i denoted by W_i (flop-floating point operations)
- P(S_i) denotes the computational power of server S_i (MI/s: million instructions/second)
- The function B(S_i,S_j) represents the directed bandwidth from server S_i to server S_j represents where function B(): S×S → R^+. We assume that B(S_i,S_i) = ∞ and B(S_i,S_j) = B(S_j,S_i)
- D_{ij} denotes data produced by task T_i and consumed by task T_j

The function f(): T→S represents a scheduling plan where $f(T_i)$ is the server that processes task T_i

The above assumptions lead to:

- The execution time of task T_i is:

$$\frac{W_i}{P\left(f\left(T_i\right)\right)} \quad (4)$$

- The communication time between task T_i and task T_j is:

$$\frac{D_{ij}}{B\left(f\left(T_i\right), f\left(T_j\right)\right)} \quad (5)$$

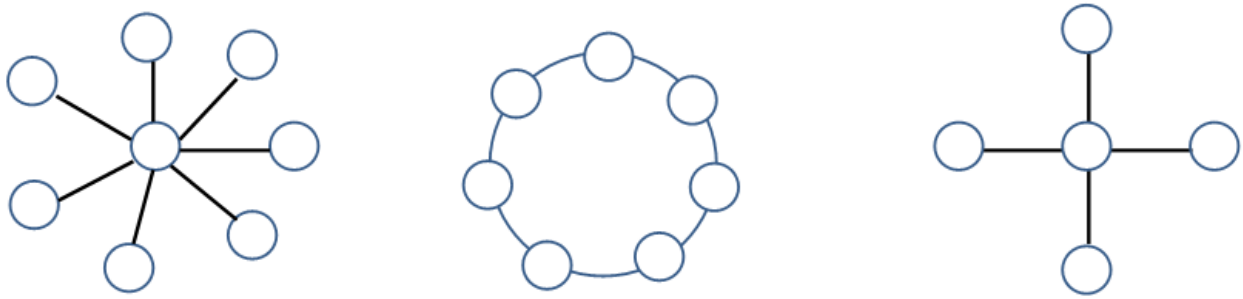

Fig. 1. Neighborhood topologies (a) Star topology (b) Ring topology (c) Von Neumann topology

Accordingly, we formally need to minimize the execution time (*makespan*) of the workflow:

$$makespan \rightarrow \min$$

where, *makespan* is the time elapsed between a task's start and finish times.

Proposed Algorithm

Escaping Local Extremum

As PSO algorithms progress, they may get trapped in a local extremum. We here propose the following method for escaping such local extrema: When the swarm gets stuck in an area around a local extremum, we combine the PSOs having topological neighborhood with a neighborhood searching function (Liu *et al*., 2007) to move particles to the new area.

Variable Neighborhood Searching Function

In order to help the swarm escape from the area around the local extrema, we designed 2 operators named Exchange and RotateRight, as illustrated in the Fig. 2 and built a Variable_Neighborhood_Searching function based on these operators.

Function Variable_Neighborhood_Searching ( )
Input: position vector x_i
Output: position vector x_k : Fitness(x_k) < Fitness(x_i)
Begin
1. t: = 0;
2. while (Fitness(x_k) > Fitness (x_i) and (t <*Max_Iteration*)
3. r: = random [1,M];
4. x_i: = RotateRight(x_i, r);
5. $rand_1$: = [1,M]; $rand_2$: = [1,M];
6. x_k: = Exchange (x_i, $rand_1$, $rand_2$);
7. if Fitness(x_k) < Fitness(x_i) then return x_k else return x_i;
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.

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 deviation ε, the number of particle *NoP,* the constant K
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 (t ≤ number of iterations) Do
6. for i: = 1 to *NoP* do
7. Compute new positionx_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 andcompute 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.

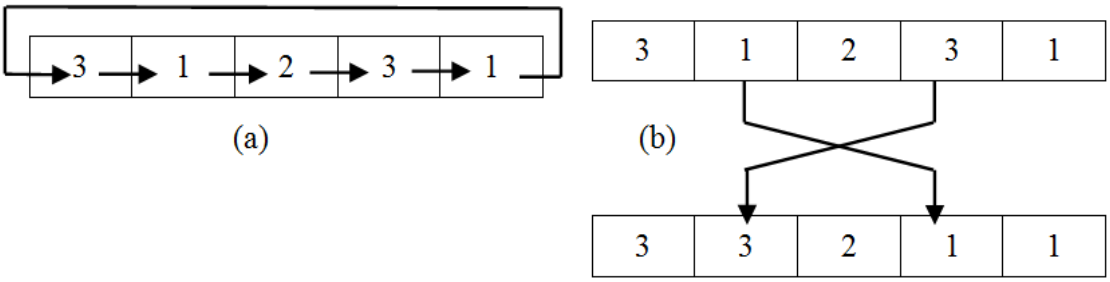

Fig. 2. Operator RotateRight (a) and Operator Exchange (b)

At 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 stuck in a local extremum 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 less than ε after K continuous migrations upon calling the function *Variable_Neighbourhood_Searching()*, LPSO halts. In our experiments, we set K = 30, ε = 0.21. In the best case, LPSO can find the absolute position upon calling the function *Variable Neighbourhood Searching()K* times.

In the case of LPSO fall into a trap of the local extremum, LPSO will escape from the area around the extremum by calling the function *Variable_Neighbourhood_Searching( )*.

Results and Discussion

We conducted some experiments in order to compare the performance of the LPSO algorithm with others, namely the PSO_H (Pandey *et al*., 2010) and Random (Mitzenmacher and Upfal, 2005). Our experimental setup consists of a computer with RAM 4GB, Intel Core i5 2.2 GHz and Windows 7 Ultimate. We used Java, the simulation tool CloudSim and the library Jswarm (Calheiros *et al*., 2011) to conduct our simulation experiments.

Problem Instances

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

- The computation power of the servers and the bandwidth of connections between servers are collected from Cloud providers such as Amazon (Vliet and Paganelli, 2011) and their web site (exp. http://aws.amazon.com/ec2/pricing)
- The sets of working data are collected from Montage project (http://montage.ipac.caltech.edu)

Based on the number of servers, N and the number of tasks, M, we split the instances into 6 groups:

Group 1: M =10, N = 3; Group 2: M = 10, N = 5; Group 3: M = 20, N = 5;
Group 4: M = 20, N = 8; Group 5: M = 25, N = 8; Group 6: M = 50, N = 8;

The ratio of the number of edges to the number of vertexes of graph Gcan be formulated as follows:

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

Configuration Parameters

The Cloud's configuration parameters are chosen as follows:

- Server's computation power: From 1 to 250 (million instructions/s)
- Connection bandwidth B: From 10 to 100 (Megabit/s)
- Communication data D: From 1 to 10000 (Megabit)
- ω = 0.729; $c_1 = c_2$ = 1.49445; K = 30, Deviation ε = 0.21
- Number of particles *NoP* = 25 ; ε = 0.21 ; α: from 0.2 to 0.7
- Our default threshold for number of generations is 300. Once the algorithm exceeds this threshold, the execution is terminated. Both of LPSO and PSO_H are executed with the number of generations is 300, after that their best results are recorded and listed in the Table 1

Results

Each problem instance was executed 30 times continuously. The results described 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 (Pandey *et al*., 2010) and

Random (Mitzenmacher and Upfal, 2005) 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 is better than 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 *et al*. (2010) 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_Halgorithm (Pandey *et al*., 2010) and hence it cannot maintain the diversification of swarm. For this reason, PSO_H often gets trapped in local extrema.

Alternatively, we introducea new method. The left hand side x_i^{k+1} will be assigned to the server whose computation power is the closest to ($x_i^k + v_i^k$)

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

In other words, the new position of a particle is the one that makes a task get assigned to a server whose computation power is the closest to the real value computed from the position vector. The results described in Table 1 shows that the mean (under the *Mean* column) and standard deviation (under the *STD* column) of LPSO are better than those of PSO_H (Pandey *et al*., 2010) and Random (Mitzenmacher and Upfal, 2005) 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 (Pandey *et al*., 2010) and Random (Mitzenmacher and Upfal, 2005).

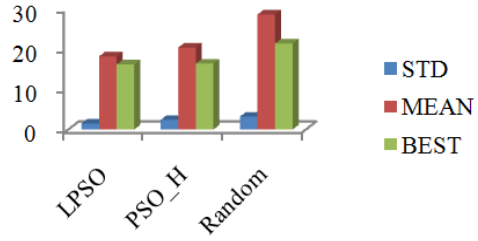

Fig. 3. M = 10, N = 3

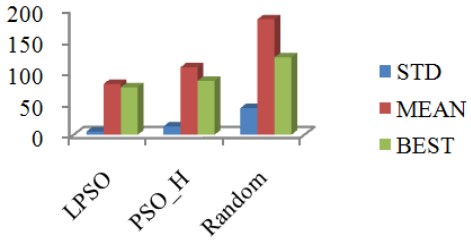

Fig. 4. M = 10, N = 5

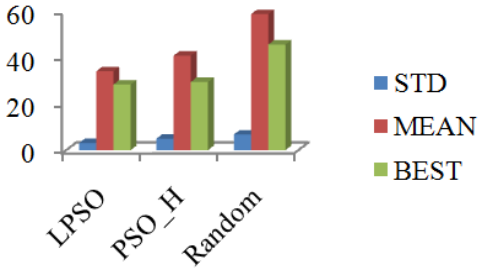

Fig. 5. M = 20, N = 5

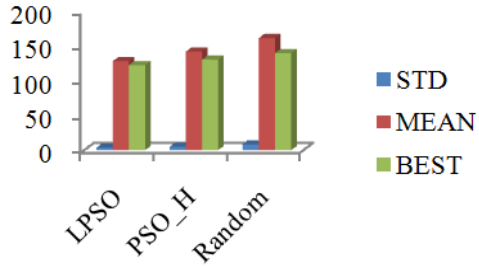

Fig. 6. M = 20, N = 3

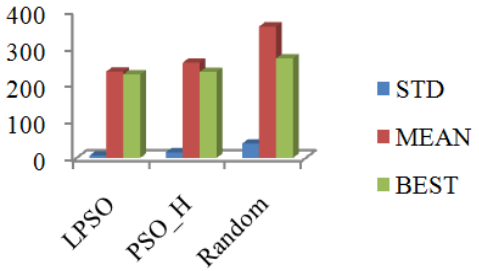

Fig. 7. M = 25, N = 8

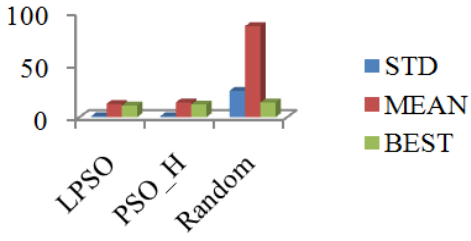

Fig. 8. M = 50, N = 8

Table 1. Comparison of makespan between LPSO and other algorithms

			LPSO			PSO_H			RANDOM		
M	N	α	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

Figure 3-8 depict the performance of the three algorithms: Proposed algorithm LPSO, PSO_H (Pandey *et al*., 2010) and Random (Mitzenmacher and Upfal, 2005) where the vertical axis represents the makespan (seconds) of the schedule. For each instance, we evaluate the mean value (column *MEAN*), standard deviation value (column *STD*) and the best position vector (listed in column *BEST*), At the first instance, LPSO even found the optimal solution.

Conclusion

Minimizing the makespan is the ultimate objective of any scheduling algorithm. In addition to that objective, our proposed algorithm also avoid being trapped on local extrema. The contributions of our paper are:

- Building a novel approach, represented by the function *Variable_Neighbourhood_Searching*, to help optimization algorithms escape from a local extremum
- Proposing a novel scheduling algorithm called LPSO by augmenting the PSO strategy with the *Variable Neighbourhood Searching* function

The simulation results suggest that LPSO is superior to its predecessor especially when LPSO operates in a larger Cloud with respect to the number of tasks and servers are larger. As a future work, we plan to enhance the LPSO algorithm in order to process bigger instances in a reasonable makespan.

Acknowledgment

This research has been supported by the Hanoi National University of Education and Military Institute of Science and Technology (Vietnam) and the American University of RAK (UAE).

Authors Contributions

Toan Phan Thanh is the principle investigator in this research which is part of his PhD work. Dr. Loc Nguyen The is his PhD academic advisor and directly supervising this work. Dr. Said Elnaffar helped with algorithm design and formalism, CloudSim simulation, and the write up of the manuscript. Dr. Cuong Nguyen Doan oversaw the work and provided advices.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Agarwal, A. and S. Jain, 2014. Efficient optimal algorithm of task scheduling in cloud computing environment. Int. J. Comput. Trends Technol., 9: 344-349.

Buyya, R., R. Ranjan and R.N. Calheiros, 2009. Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. Proceedings of the International Conference on High Performance Computing and Simulation, Jun. 21-24, IEEE Xplore Press, pp: 1-11.
DOI: 10.1109/HPCSIM.2009.5192685

Calheiros, R.N., R. Ranjan, A. Beloglazov, C.A.F. De Rose and R. Buyya, 2011. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software Pract. Exp., 41: 23-50.

Guo, L., S. Zhao, S. Shen and C. Jiang, 2012. Task scheduling optimization in cloud computing based on heuristic algorithm. J. Netw., 7: 547-552.
DOI: 10.4304/jnw.7.3.547-553

Guo-Ning, G. and H. Ting-Lei, 2010. Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. Proceedings of the International Conference on Intelligent Computing and Integrated Systems, Oct. 22-24, IEEE Xplore Press, pp: 60-63.
DOI: 10.1109/ICISS.2010.5655013

http://montage.ipac.caltech.edu

Huang, J., 2014. The workflow task scheduling algorithm based on the GA model in the cloud computing environment. J. Software, 9: 873-880.

Kennedy, J. and R.C. Eberhart, 1995. Particle swarm optimization. Proceeding of the IEEE International Conference on Neural Networks, Nov. 27-Dec. 1, IEEE Xplore Press, pp: 1942-1948. DOI: 10.1109/ICNN.1995.488968

Liu, H., A. Abraham and C. Grosan, 2007. A novel variable neighborhood particle swarm optimization for multi-objective flexible job-shop scheduling problems. Proceedings of the 2nd International Conference on Digital Information Management, Oct. 28-31, IEEE Xplore Press, pp: 138-145. DOI: 10.1109/ICDIM.2007.4444214

Liu, J., X. Luo, B. Li, X. Zhang and F. Zhang, 2013. An intelligent job scheduling system for web service in cloud computing. Indonesian J. Electr. Eng., 11: 2956-2961. DOI: 10.11591/telkomnika.v11i6.2118

Mitzenmacher, M. and E. Upfal, 2005. Probability and Computing: Randomized Algorithms and Probabilistic Analysis. 1st Edn., Cambridge University Press, Cambridge, ISBN-10: 0521835402, pp: 352.

Pandey, S., L. Wu, S.M. Guru and R. Buyya, 2010. A Particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications, Apr. 20-23, IEEE Xplore Press, pp: 400-407. DOI: 10.1109/AINA.2010.31

Parsa, S. and R.E. Maleki, 2009. RASA: A new task scheduling algorithm in grid environment. Int. J. Digital Content Technol. Applic., 3: 91-99. DOI: 10.4156/jdcta.vol3.issue4.10

Rajkumar, R. and T. Mala, 2012. Achieving service level agreement in cloud environment using job prioritization in hierarchical scheduling. Proceeding of the International Conference on Information System Design and Intelligent Application, (DIA' 12), Springer, pp: 547-554. DOI: 10.1007/978-3-642-27443-5_63

Ullman, J.D., 1975. NP-complete scheduling problems. J. Comput. Syst. Sci., 10: 384-393. DOI: 10.1016/S0022-0000(75)80008-0

Vliet, J.V. and F. Paganelli, 2011. Programming Amazon EC2: Survive your Success. 1st Edn., O'Reilly Media, Inc., Sebastopol, ISBN-10: 1449305261, pp: 186.

Xue, S.J. and W. Wu, 2012. Scheduling workflow in cloud computing based on hybrid particle swarm algorithm. Indonesian J. Electr. Eng., 10: 1560-1566. DOI: 10.11591/telkomnika.v10i7.1452

Zavala, A.E.M., 2013. EVOLVE-A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation IIA Comparison, A Comparison Study of PSO Neighborhoods. 1st Edn., Springer, Verlag Berlin Heideberg, ISBN-13: 978-3-642-32725-4