A Flower Pollination Algorithm Based Task Scheduling in Cloud Computing

Indrajeet Gupta(✉), Amar Kaswan, and Prasanta K. Jana

Department of Computer Science and Engineering,
Indian Institute of Technology (ISM), Dhanbad, India
indrajeet7830@gmail.com, amarkaswan@gmail.com, prasantajana@yahoo.com
http://www.iitism.ac.in

Abstract. Flower pollination algorithm (FPA) is a nature inspired fascinating meta-heuristic technique, which is applicable to many real life optimization problems. Mapping of tasks on the virtual machines in cloud computing environment is a well known NP-complete problem. This paper propose a novel FPA-based algorithm to schedule the tasks on the virtual machines to minimize makespan and maximize cloud resource utilization. The proposed scheme uses an efficient pollen representation scheme and a novel multi-objective fitness function. We simulate the proposed scheme on one synthetic and two benchmark datasets of diverse configuration. The performance of the proposed scheme is compared with three other meta-heuristic based approaches, namely particle swarm optimization (PSO), genetic algorithm (GA) and gravitational search algorithm (GSA). The superiority of the proposed algorithm over other algorithms is exhibited by the simulation results.

Keywords: Flower pollination algorithm · Cloud computing · Task scheduling · Cloud utilization · Makespan

1 Introduction

Scheduling of tasks in cloud computing is a well studied problem which is NP-complete in nature. Therefore, several heuristics and meta-heuristics approaches have been proposed to address this problem that deal with independent or dependent tasks. This article describes the scheduling of independent tasks on the virtual resources and present a novel algorithm which is based on an efficient meta-heuristics approach, called flower pollination algorithm (FPA) [1]. It is noteworthy that there exist m^n possible schedules for n tasks to be mapped on m VMs. Therefore, the computational complexity to generate an optimal task-VM mapping by a brute force approach would be exponential. This also consumes huge space. Thus, a nature-inspired approach such as flower pollination algorithm (FPA) [1] can be very effective to generate a near optimal solution for the aforesaid task scheduling problem in cloud environment. Note that the FPA can produce better solution than other meta-heuristic approaches such as GA, PSO and has the highest convergence rate among them [1].

DOI: 10.1007/978-981-10-6430-2_9

The key goal of the proposed scheme is to map the tasks of a given application on the active virtual machines such that the overall application processing time namely makespan(M) is minimized and the average cloud resource utilization ($Avg.U_c$) is maximized. The proposed scheme is presented with the employed pollen representation scheme. A novel multi-objective fitness function is also derived by considering the aforesaid objectives which contradict each other. We perform rigorous experiments to compute the results of the proposed scheme using one synthetic dataset and two benchmark datasets instances. The performance results are compared with three other meta-heuristic based scheme i.e., PSO, GA and GSA. The simulated results demonstrate its superiority in terms of makespan and average cloud utilization.

Many researches have been carried out in recent years to address the task scheduling problem in [2–12,16,19,20]. Panda et al. [8] have proposed both online and off-line task scheduling algorithm to schedule the independent tasks for cloud environment. The proposed algorithms of task scheduling are shown to perform better than the cloud list scheduling (CLS) [17], Round Robin (RR) and Min-Min task scheduling algorithms [18]. In [16], the authors have used the linear programming to solve the problem of preemptive task scheduling in cloud computing. In [7], a two phase genetic algorithm with multi-parent crossover is suggested to minimize both makespan and energy consumption. However, they have not considered average cloud utilization. In [6], an improved genetic algorithm (IGA) is proposed, in which the virtual machines are selected on the basis of dividend policy. The performance of their algorithm is shown to outperform [7]. In [4], the authors have developed a PSO based algorithm for static task scheduling problem in a homogeneous cloud environment which supersedes the GA based approach. A multi-objective differential evolution (MODE) scheme is proposed in [2], as a solution to the task scheduling problem. In [5], the authors have addressed the problem of task scheduling in a grid environment and proposed a PSO based scheme to generate the task-VM mapping. The proposed algorithm is shown to produce the best solution for the task scheduling problem. Our proposed algorithm in this paper, is a novel approach in the sense that we are the first researchers to exploit the features of the meta-heuristic FPA to develop an efficient task scheduling algorithm for cloud computing. Further, there is novelty in the pollen representation and derivation of an efficient fitness function based on the two important objectives for task scheduling.

The remainder of the paper is systematized as follows. Section 2 describes the models and terminologies used in the proposed algorithm. The proposed algorithm is explained in Sect. 3. The simulation results are displayed in Sect. 4 followed by the conclusion in Sect. 5.

2 Preliminaries

2.1 Cloud and Task Model

Let us consider a cloud user's request which is expressed in the form of an application A consisting of a set of n tasks such that $A=\{T_1, T_2, \cdots, T_n\}$. The user submits his application to a cloud service provider (CSP) for the needful operations (i.e., computing, communication and storage). The CSP provides all the requested services to the users by negotiating a service level agreement (SLA).

Table 1. Notations used

Notations	Descriptions
n	Number of tasks in a application A
m	Number of active VMs
I_i	Size of the task T_i in million instructions
M_VM_i	Makespan of the i^{th} virtual machine
PS_i	Processing speed of the i^{th} virtual machine
$max(A)$	Return the maximum value form a set A
$Switch\_prob$	Switching probability of FPA

Consider an IaaS cloud model similar to the Amazon EC2, where m heterogeneous virtual machines (VMs) are deployed on a cloud. Each VM has different computing capability. Therefore, all VMs are able to execute the tasks of the given application. The key components of the assumed cloud model are as follows.

1. **Cloud user:** This is the end user of cloud resources. The cloud users initiate the various cloud service requests to execute their applications. These requests are submitted to the cloud scheduler via cloud service provider (CSP).
2. **Cloud scheduler:** Cloud scheduler is the recipient of the cloud users' service requests. The core responsibility of the cloud scheduler is to maintain status of all VMs and assign the tasks over the virtual machines so that the user's requirement is fulfilled.
3. **Cloud service provider (CSP):** The cloud service provider acts as an intermediary between the cloud users and the cloud scheduler. The CSP contains the resource pool of the active VMs. The user's application is executed on the active VMs as per the decided task scheduling algorithm.

2.2 Notations and Terminologies

Now, we define various terminologies used to derive the fitness function and analyze the simulation results as follows. The notations are shown in the Table 1 with their description.

1. **Processing speed** (PS): This is the processing speed of each VM say, VM_j which is measured in million instructions per second (MIPS) and denoted by PS_j.

2. **Execution time of a task** (ET_{ij}): Consider an application consisting of n tasks $\{T_1, T_2, T_3, \cdots, T_n\}$, where the size ($I_i$) of every task T_i is measured in million instructions (MIs). Therefore, the execution time of task T_i on virtual machine VM_j can be mathematically expressed as follows.

$$ET_{ij} = \frac{I_j}{PS_j} \tag{1}$$

3. **Makespan** (M): Let M_VM_i denote the makespan of the i^{th} virtual machine, where $1 \le i \le m$. Then, we can mathematically express the overall makespan as follows.

$$M = max\{M\_VM_i \mid 1 \le i \le m\} \tag{2}$$

4. **Average makespan** ($Avg.M\_VM$): It is the mean of the makespan of all the VMs. In other words,

$$Avg.M\_VM = \frac{\sum_{j=1}^{m} M\_VM_j}{m} \tag{3}$$

5. **Standard deviation of VM's makespan** ($Sd.M\_VM$): It is the standard deviation of virtual machines' makespan. Hence,

$$Sd.M\_VM = \sqrt{\frac{\sum_{i=1}^{m}(Avg.M\_VM - M\_VM_i)^2}{m}} \tag{4}$$

6. **Average cloud utilization** ($Avg.U_c$): It is the ratio of the average makespan ($Avg.M\_VM$) and the overall makespan (M). Therefore,

$$Avg.U_c = \frac{Avg.M\_VM}{M} \tag{5}$$

2.3 Problem Definition

We address the following problem in this paper. Given an application with a set of n independent tasks, $A = \{T_1, T_2, T_3, \cdots, T_n\}$ and given a set of m virtual machines $C = \{VM_1, VM_2, VM_3, \cdots, VM_m\}$, the problem is to determine the task-VM mapping with the following objectives:

- Objective 1: The overall makespan (M) is minimized.
- Objective 2: The average cloud utilization ($Avg.U_c$) is maximized.

3 Proposed Work

In this section, we present a FPA-based scheme to address the problem defined in Sect. 2.3. We first show the pollen representation scheme used in the proposed algorithm. We then design an efficient multi-objective fitness function to incorporate in the proposed algorithm.

3.1 Pollen Representation

Each pollen (P) is represented by an n-dimensional vector, where each element (say $p_{ij} \mid 1 \leq i, j \leq n$) has a random value in the range $(0, 1)$ i.e., $0 < p_{ij} < 1$. The employed mapping function is given by Eq. 6.

$$T\_VM_{ij} = floor(p_{ij} \times m) + 1 \mid 1 \leq i \leq p\_size \text{ and } 1 \leq j \leq n \tag{6}$$

where, m denotes the total number of VMs, p_size denotes population size and n denotes the total number of tasks. This pollen representation is analogous to [15]. It is noteworthy that each pollen is able to produce entire solution for the task scheduling problem. To demonstrate this, we consider a hypothetical could-resource model where $n = 8$ and $m = 3$. The process of task-VM mapping is exhibited by considering a sample pollen as shown in Fig. 1.

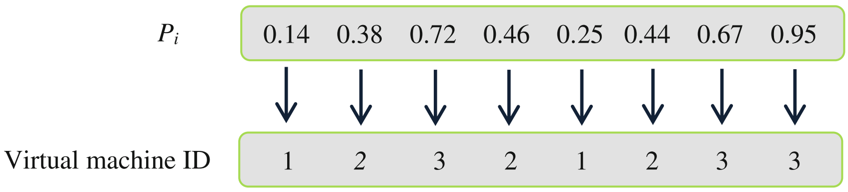


Fig. 1. Retrieval of task-VM mapping from a pollen

In Fig. 1, we can observe that the first task is mapped on virtual machine 1 since the value of p_{i1} is 0.14 and therefore, $floor(0.14 \times 3) + 1 = 1$. Remaining tasks are also mapped on the VMs following similar procedure. For instances, the tasks T_2 and T_3 are mapped onto VM_2 and VM_3 respectively using same formula given in Eq. 6.

3.2 Fitness Function

We consider two contradictory objectives to derive an efficient multi-objective fitness function for problem of independent task scheduling in cloud computing. The first objective is to minimize the makespan. The makespan can be minimized, if the standard deviation of makespan of all the VMs is minimized i.e., lower the value of $Sd.M\_VM$, lower will be the makespan. Therefore,

$$\text{Objective 1: Minimize } Sd.M\_VM \tag{7}$$

Our second objective is to maximize the average cloud utilization. Therefore,

$$\text{Objective 2: Maximize} = Avg.U_c \tag{8}$$

We combine both above-mentioned objectives into a single-objective for minimization problem by employing the weighted sum approach where the value of the first objective is multiplied with a weight value C_1 and reciprocal of the second objective is multiplied by C_2. Thus,

$$\text{Objective: Minimize}(C_1 \times Avg.M\_VM + \frac{C_2}{Avg.U_C}) \tag{9}$$

We choose this is as the single objective fitness function in our proposed algorithm. In other words,

$$Fit = C_1 \times Avg.M\_VM + \frac{C_2}{Avg.U_C} \tag{10}$$

3.3 FPA-based Task Scheduling Algorithm

The pseudo code of the proposed FPA-based algorithm is given in Fig. 2. In step 1, population of size p_size is initialized. In step 2 through step 4, the value of the fitness function for each pollen of the current population is computed using

```
Algorithm 1: Proposed FPA-based algorithm
INPUT: $n$, $m$, $ET$, $Switch.prob$
OUTPUT: Task-VM mapping
1.  Initialize pollen $P_k$, $1 \leq k \leq P.size$
2.  for $k = 1$ to $m$ do
3.      $Fit_k$ = Compute the value of fitness function of $j^{th}$ pollen using Eq. 10
4.  end for
5.  Initialize $Best.fit = Init\_max$
6.  Initialize $Best.pol = \emptyset$
7.  for k = 1 to m do
8.      if $Best.fit > Fit_k$ then
9.          $Best.fit = P_k$
10.     end if
11. end for
12. for $k = 1$ to $m$ do
13.     for $k = 1$ to $m$ do
14.         $temp = rand(0, 1)$
15.         if $temp > Switch.prob$ then
16.             Draw a $n$ – dimensional step $L$ following a Levy distribution
17.             Apply global pollination using $P_{k+1} = P_k + L(Best.fit - P_k)$
18.         else
19.             Draw $\varepsilon$ following a Uniform distribution
20.             Arbitrarily opt $i$ and $j$ among available solutions
21.             Apply local pollination using $P_k + 1 = P_k + \varepsilon(Pi - Pj)$
22.         end if
23.         Compute novel pollen
24.         if novel pollen is superior than current pollen then update the current pollen
25.         Identify new $Best.pol$
26.         end if
27.     end for
28. end for
```

Fig. 2. Proposed FPA-based task scheduling algorithm

Eq. 10. In step 5 and step 6, the fitness value of the best pollen is initialized to maximum value of the integer and the best pollen is initialized as an empty set respectively. In step 7 through step 11, the best pollen from the initial population is identified and stored as $Best\_pol$. In step 12 through step 27, at each iteration for every pollen, a random number between 0 and 1 is generated and if the generated value is greater than the $Switch\_prob$ than the global pollination is applied to the pollen using Levy distribution, otherwise the local pollination is applied using uniform distribution [1].

The proposed algorithm FPA is tested through extensive simulations which were carried out on an Intel Core 2 Duo processor, 2.20 GHz CPU with 4 GB RAM running on Microsoft Windows 7 platform by using MATLAB R2012a version 7.14.0.739. We evaluate the performance for the simulation results on one synthetic dataset of 6 instances and two benchmark datasets of 12 instances. For the comparison purpose, we also simulate GA, GSA and PSO by employing the same solution representation and the fitness function. In simulation, the switching probability and the size of population (p_size) is considered as 0.5 and 100 respectively.

3.4 Datasets

In simulation, we considered two benchmark datasets as given in [13] with 512 and 1024 tasks respectively. The benchmark datasets comprises of 12 diverse dataset instances. These instances are described by u_x_yyzz, where u implies uniform distribution to generate the dataset, x refers for the consistency i.e., inconsistent (i), semi-consistent (s) or consistent(c) and yyzz refers the heterogeneity of task-VM i.e., hihi, hilo, lohi, lolo respectively [13,14]. Moreover, we created one synthetic dataset using the uniform distribution. This synthetic dataset also consists of 6 different instances, where size of each instance is $n \times m$, for n tasks and m VMs.

3.5 Experimental Results

We consider makespan (M) and average cloud utilization (please refer Sect. 2.2) as the performance metrics to show the dominance of the proposed algorithm over other simulated schemes. The comparison of cloud makespan for the dataset instances of size 512×16 size and 1024×32 are shown in Figs. 3 and 4 respectively by the bar graph. We also plot the average cloud resource utilization for both the benchmark datasets as shown in Figs. 5 and 6 respectively. Similarly, for synthetic dataset, we also calculated the makespan and average cloud resource utilization as shown by Fig. 7. It can be observed that the proposed FPA-based scheme has the minimum makespan and maximum average cloud utilization for all the dataset instances in comparison with GA, GSA and the PSO based approaches.

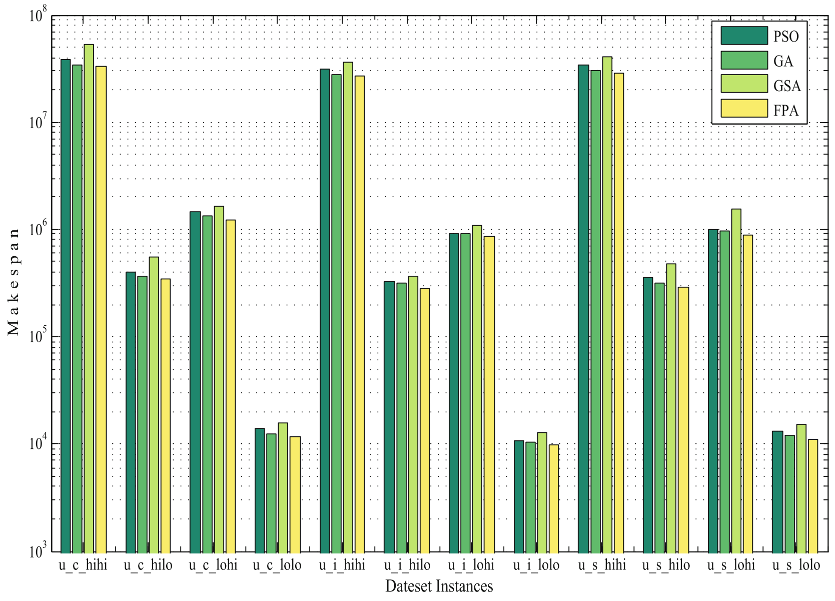


Fig. 3. Comparison of cloud makespan for 512 × 16 size dataset instances

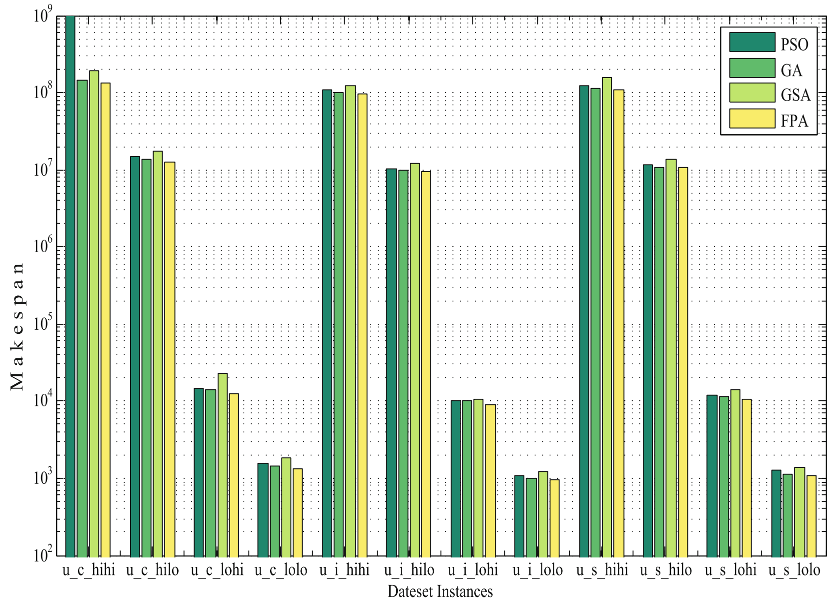


Fig. 4. Comparison of cloud makespan for 1024 × 32 size dataset instances

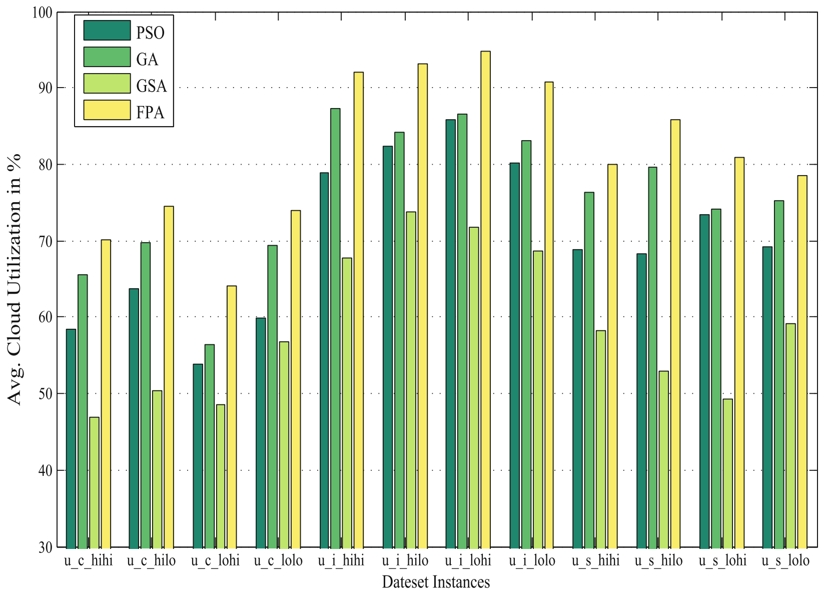


Fig. 5. Comparison of avg. cloud utilization for 512×16 size dataset instances

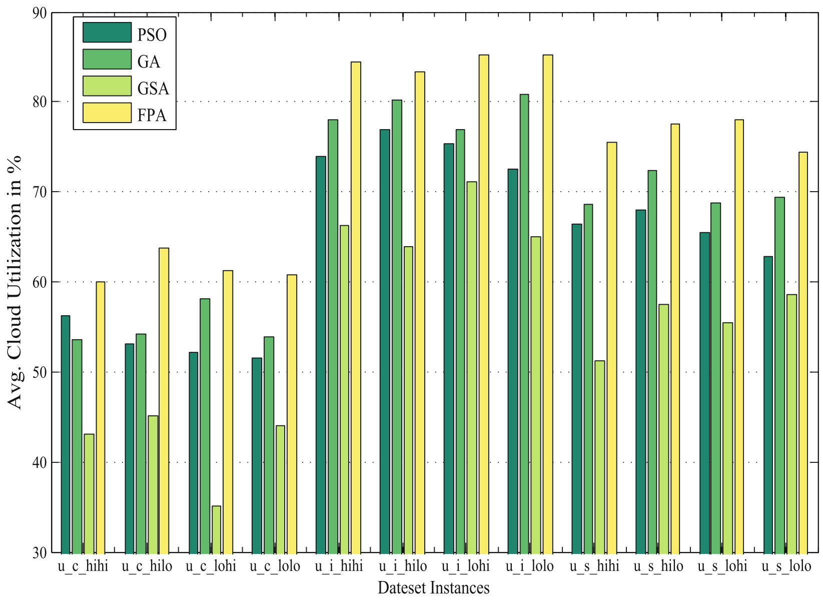
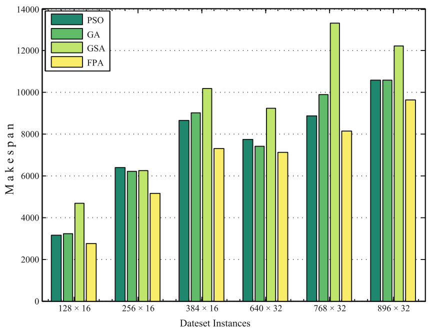
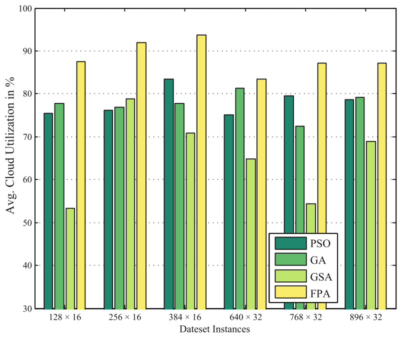


Fig. 6. Comparison of avg. cloud utilization for 1024×32 size dataset instances



(a)



(b)

Fig. 7. Comparison for synthetic dataset instances (a) cloud makespan and (b) avg. cloud utilization

4 Conclusion

We have proposed a FPA-based task scheduling scheme in a cloud computing environment. We have demonstrated an efficient scheme of pollen representation and the process to retrieve the task-VM mapping from a given pollen. We have also shown the derivation of a single objective fitness function by combining two conflicting objectives, i.e., minimization of makespan and maximization of average cloud resource utilization. Through simulation runs on one synthetic dataset and two benchmark datasets, we have shown that the proposed technique performs better than the similar meta-heuristics, namely, PSO, GA and GSA in terms of two performance metrics i.e., makespan and average cloud resource utilization. However, the proposed work considers the static behavior of the tasks and virtual machines. Our future work will be focused on dynamic workflow scheduling.

References

1. Yang, X.-S.: Flower pollination algorithm for global optimization. In: Durand-Lose, J., Jonoska, N. (eds.) UCNC 2012. LNCS, vol. 7445, pp. 240–249. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32894-7_27
2. Alkhanak, E.N., Lee, S.P., Rezaei, R., Parizi, R.M.: Cost optimization approaches for scientific workflow scheduling in cloud and grid computing: a review, classifications, and open issues. J. Syst. Softw. **113**, 1–26 (2016)
3. Tao, F., Feng, Y., Zhang, L., Liao, T.W.: CLPS-GA: a case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling. Appl. Soft Comput. **19**, 264–279 (2014)
4. Salman, A.: Particle swarm optimization for task assignment problem. Microprocess. Microsyst. **26**(8), 363–371 (2002)
5. Zhang, L., Chen, Y., Sun, R., Jing, S., Yang, B.: A task scheduling algorithm based on PSO for grid computing. Int. J. Comput. Intell. Res. **4**, 37–43 (2008)

6. Zhong, H., Tao, K., Zhang, X.: An approach to optimized resource scheduling algorithm for open-source cloud systems. In: ChinaGrid Conference (ChinaGrid), pp. 124–129. IEEE (2010)
7. Talukder, A., Kirley, M., Buyya, R.: Multi-objective differential evolution for scheduling workflow applications on global grids. Concurr. Comput. Pract. Exp. **21**(13), 1742–1756 (2009)
8. Panda, S.K., Jana, P.K.: Efficient task scheduling algorithms for heterogeneous multi-cloud environment. J. Supercomput. **71**(4), 1505–1533 (2015)
9. Panda, S.K., Jana, P.K.: Uncertainty-based QoS min min algorithm for heterogeneous multi-cloud environment. Arab. J. Sci. Eng. **41**(8), 3003–3025 (2016)
10. Gupta, I., Kumar, M.S., Jana, P.K.: Compute-intensive workflow scheduling in multi-cloud environment. In: International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 315–321 (2016)
11. Durillo, J.J., Prodan, R., Barbosa, J.G.: Pareto tradeoff scheduling of workflows on federated commercial clouds. Simul. Model. Pract. Theor. **58**, 95–111 (2015)
12. Ding, Y., Qin, X., Liu, L., Wang, T.: Efficient scheduling of virtual machines in cloud with deadline constraint. Future Gener. Comput. Syst. **50**, 62–74 (2015)
13. https://code.google.com/p/hcspchc/source/browse/trunk/AE/ProblemInstances/HCSP. Accessed 26 July 2016
14. Xhafa, F., Carretero, J., Barolli, L., Durresi, A.: Immediate mode scheduling in grid systems. Int. J. Web Grid Serv. **3**(2), 219–236 (2007)
15. Kuila, P., Jana, P.K.: Energy efficient clustering and routing algorithms for wireless sensor networks: particle swarm optimization approach. Eng. Appl. Artif. Intell. **33**, 127–140 (2014)
16. Lawler, E.L., Jacques, L.: On preemptive scheduling of unrelated parallel processors by linear programming. J. ACM (JACM) **25**(4), 612–619 (1978)
17. Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X., Gu, Z.: Online optimization for scheduling preemptable tasks on IaaS cloud system. J. Parallel Distrib. Comput. **72**, 666–677 (2012). Elsevier
18. Panda, S.K., Agrawal, P., Khilar, P.M., Mohapatra, D.P.: Skewness-based min-min max-min heuristic for grid task scheduling. In: 4th International Conference on Advanced Computing and Communication Technologies, pp. 282–289. IEEE (2014)
19. Panda, S.K., Jana, P.K.: SLA-based task scheduling algorithms for heterogeneous multi-cloud environment. J. Supercomput. **73**, 1–33 (2017)
20. Panda, S.K., Jana, P.K.: Normalization-based task scheduling algorithms for heterogeneous multi-cloud environment. Inf. Syst. Front, 1–27 (2016)