Bio-inspired Load Balancing Algorithm in Cloud Computing

Marwa Gamal[1], Rawya Rizk[2](✉), Hani Mahdi[3], and Basem Elhady[1]

[1] Electrical Engineering Department, Suez Canal University, Ismailia, Egypt
[2] Electrical Engineering Department, Port Said University, Port Said 42523, Egypt
r.rizk@eng.psu.edu.eg
[3] Computers and Systems Engineering Department, Ain Shams University, Cairo, Egypt

Abstract. Cloud computing is a widespread computing concepts which access a huge amount of data that can be used by more clients. Therefore, load balancing between resources is an important field for scheduling tasks to achieve better performance. In this paper, a Hybrid artificial Bee and Ant Colony optimization (H_BAC) load balancing algorithm is proposed. It depends on joining the important behavior of Ant Colony Optimization (ACO) such as discovering good solutions rapidly and Artificial Bee Colony (ABC) Algorithm such as collective interaction of bees and sharing information by waggle dancing. The experimental results show that H_BAC improves execution time, response time, makespan, resource utilization and standard deviation. This improvement reaches about 40% in the execution time and response time and 30% in the makespan over the other algorithms.

Keywords: Ant Colony Optimization · Artificial Bee Colony · Bio-inspired systems · Cloud computing · Load balancing

1 Introduction

Cloud computing is a paradigm for sharing a large number of on-demand services and computing resources via a heterogeneous, broad network access [1]. Cloud computing meets numerous challenges at increasing number of users because the demand of resources sharing and usage are increased rapidly. Therefore, load balancing between resources for scheduling tasks is an important challenge.

Load balancing is the strategy of conveying the load between different resources in any system with an aim to use multiple resources with highest efficiency and minimum response time and prevent single resource overload [2]. In this way, load requires to be conveyed over the resources in cloud-based building design, so that every resource indirect the equivalent measure of work at any time. The use of load balancing led to discover new algorithms to achieve better efficiency. Load balancing must consider two main tasks, one is the resource provisioning and the other is task scheduling in disseminated environment [3].

© Springer International Publishing AG 2018
A.E. Hassanien et al. (eds.), *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2017*,
Advances in Intelligent Systems and Computing 639, DOI 10.1007/978-3-319-64861-3_54

There are basically two kinds of load balancing techniques; static and dynamic. Static algorithms work properly among homogeneous resources with low variations of load. Therefore, these algorithms are not suitable for highly varying resources in cloud environment. Dynamic algorithms are successful for load balancing among heterogeneous resources in clouds. Among dynamic load balancing algorithms, Swarm Intelligence (SI) based algorithms are investigated as a direct implementation of natural phenomena [4]. Many researches have been proposed SI based algorithms, but some of these algorithms have drawbacks such as producing overhead, causing many nodes overloaded, and getting low throughput.

The most popular algorithms in SI field are Ant Colony Optimization (ACO) and Artificial Bee Colony (ABC). The combination ACO and ABC exploits the strong purposes of these two algorithms, discovering good solutions rapidly, collective interaction, and sharing information by waggle dancing. In this paper, a Hybrid artificial Bee and Ant Colony optimization (H_BAC) load balancing algorithm is proposed. It takes into consideration monitoring the load of Virtual machines (VMs) and the decision of load balancing before scheduling tasks in VMs.

The planning of the paper further is as follows. Section 2 presents an overview of related work in task scheduling in cloud environments. Section 3, presents the proposed H_BAC algorithm. The function of the proposed algorithm is tested with the two strategies in CloudSim environment and the results are presented in Sect. 4. The paper is concluded in Sect. 5.

2 Related Work

Load balancing algorithms are divided into two categories: static and dynamic. In static load balancing, the balancing technique is done before the execution. It is done based on the probabilistic nature and no changes can be made during the execution, so time of execution-period cannot be determined exactly. In dynamic load balancing, the tasks are executed dynamically among all resources and it is necessary to monitor the current load of the system [5, 6].

In [7–14] Bio Inspired Schedule algorithms were introduced. Researches in bio inspired nature discovered that cooperation of groups of similar agents in an environment can solve complicated problems. So many researches tended to study bio inspired nature to balance load among cloud environment such as foraging for food. Bio Inspired scheduling algorithm was divided in two categories: Evolutionary scheduling algorithm and SI based scheduling algorithm. Evolutionary Algorithm can be calculated by natural mechanisms of selection and developing. This algorithm is separated into sub-categories genetic algorithm and genetic programming. However, in many difficult cloud computing problems, evolutionary algorithms are unable to solve these complex problems efficiently. SI algorithms are inspired by the behavior of some social living creatures, such as ants, bees, birds, and fishes [7]. They have their own specific way to explore the search space of the problem.

In [8], the Fire Flies algorithm is proposed. Fire Flies are awesome natural specie which produces flash light. There are two functions of such flashes; firstly in order to attract the mating partner and to attract to potential prey. This natural phenomenon can help us in solving a large amount of complex cloud computing problem in scheduling and managing the resources. However it has some disadvantageous such as its parameters are set fixed and they do not change with the time and doesn't memorize or remember any history of better situation for each firefly.

The Cuckoo search is presented in [9] in which there is cuckoo specie which lay eggs in the nest of host birds. This mechanism helps in bluffing the host bird of the cuckoo bird. This natural phenomenon can be used in order to solve a large number of complex cloud computing problems in scheduling and managing the resources.

In [10–12] ACO Algorithm is proposed. It is a random search algorithm which inspired from the behavior of ants. It depends on foraging the food using external chemical pheromone trails for communication and return to their nest via shortest path based on the intensity of pheromone. The intensity of the pheromone depends on the quality and distance of the food. As time passes, the pheromone power begins evaporating, subsequently decreasing the quality of fascination. ACO overcomes heterogeneity since it is adaptive algorithm. In addition, it is excellent in fault tolerance and has good scalability. However, it has a lake in throughput.

ABC is based on foraging behavior of honey bee [13–16]. It consists of scout bees, employed bees, and onlooker bees. Scout bees are responsible for looking for food source randomly, employed bees share food information to the onlooker bees, and onlooker bees find the amount of nectar and calculate the probability. Finally, they return to their hive and go to the dance area to perform waggle dance. This dance is the way to share information about quality of food source. While sharing information, bees calculate the quality of food and energy waste. After that, onlooker bees choose the best one and then scout bees will back to the food source to get nectar and return to the hive. ABC performs well as system diversity increases. However, the disadvantage of ABC is that, it does not show any significant improvement in throughput, which is due to the additional queue and the computation overhead.

Combining SI algorithms such as ACO and ABC can achieve higher performance since this combination exploits the strong purposes of these two algorithms [17]. In [18], a hybrid algorithm that combines ACO and ABC is presented. However, this hybrid algorithm wasn't pointed to load balancing in its design as its parameters of calculating load were not stated. So this technique inherits the waggle dance behavior of ABC only.

In this paper, a hybrid ACO and ABC algorithm inherits the main behaviors of both these techniques together. Since the pheromone behavior of ACO is very good in discovering solutions rapidly at diversity systems, so it is adaptive to dynamic environments. In the other hand, ABC achieves global load balancing and perform well as system diversity increases by its behavior of sharing information by waggle dancing. The proposed algorithm takes into consideration monitoring the load of VMs and the decision of balancing before scheduling tasks in VMs.

3 The Proposed H_BAC Algorithm

This section presents the proposed hybrid algorithm. In H_BAC, The k-ants are responsible for finding capacity of VM_j as pheromone (τ_j) and sum of transfer time and execution time of the task on VM_j as edge weight (η_j). Bees are responsible for providing the status of the VM_j's load (Lvm_j) and deciding load balancing (LB_j). Figure 1 introduces the flowchart of the proposed H_BAC algorithm. The parameters τ_j, η_j, Lvm_j, and LB_j are stored in the knowledge base which represented as waggle dance to share other ants and bees new information. Ants calculate the probability of choosing the best VM_j to achieve load balancing as:

$$p_j^k = \begin{cases} \frac{\tau_j^\alpha . \eta_j^\beta . L_{VM_j}^\gamma . LB_j}{\sum \tau_j^\alpha . \eta_j^\beta . L_{VM_j}^\gamma . LB_j} & \text{if } j \in 1, \ldots\ldots, n \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where α, β, and γ give relative importance between pheromone edge weight and status of VM_j's Load. Edge weight (η_j) is

$$\eta_j = ET_j + TT_j \tag{2}$$

where ET_j, and TT_j are execution time and transfer time of VM_j and can be calculated as follows

$$ET_j = \frac{\sum_{i=1}^m TL_i}{Pe_{num_j} \times Pe_{mips_j}} \tag{3}$$

$$TT_j = \frac{IFS_j}{VM_{bw_j}} \tag{4}$$

where TL_i is the length of task i scheduled in VM_j, Pe_{num_j} is number of processors in VM_j, Pe_{mips_j} is million instructions per second (MIPS), VM_{bw_j} is bandwidth ability of VM_j, and IFS is the input file size of VM_j.

Pheromone (τ_j) can be calculated as follows

$$\tau_{new} = (1 - \rho)\tau_{current} + \sum_{k=1}^n \Delta\tau_j \tag{5}$$

As ρ is evaporating parameter on pheromone usually set to $0 < \rho < 1$, $\Delta\tau_j = \frac{1}{\tau_j}$ [13] and τ_0 is calculated as:

$$\tau_0 = Pe_{num_j} \times Pe_{mips_j} + VM_{bw_j} \tag{6}$$

The VM_j's load (L_{VM_j}) can be calculated by (7) and load balancing decision (LB_j) can be calculated by (8).

$$L_{VM_j} = \sum_{i=1}^m TL_i \Big/ Pe_{mips_j} \tag{7}$$

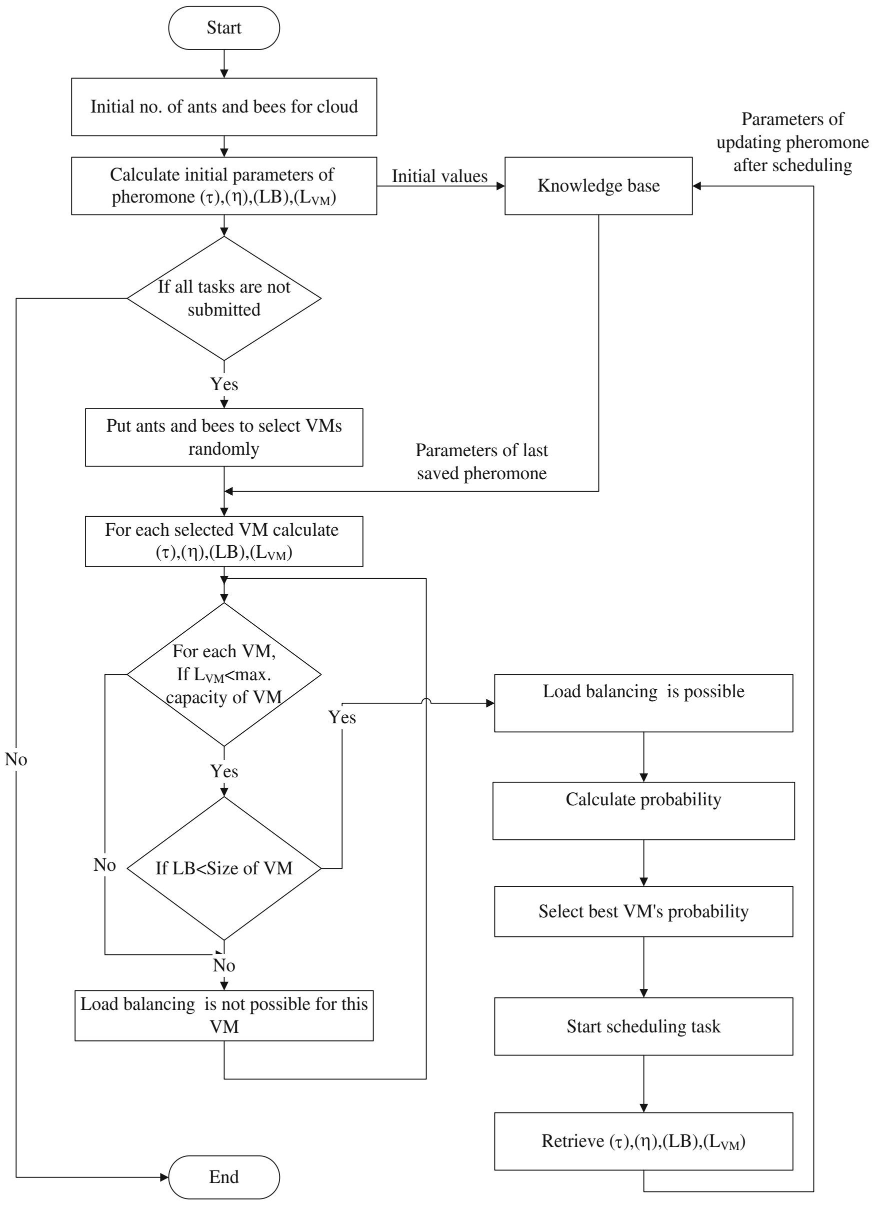
Start
Initial no. of ants and bees for cloud
Calculate initial parameters of pheromone (τ),(η),(LB),(L_{VM})
Initial values
Knowledge base
Parameters of updating pheromone after scheduling
If all tasks are not submitted
Yes
No
Put ants and bees to select VMs randomly
Parameters of last saved pheromone
For each selected VM calculate (τ),(η),(LB),(L_{VM})
For each VM, If L_{VM}<max. capacity of VM
Yes
No
If LB<Size of VM
Yes
No
Load balancing is not possible for this VM
Load balancing is possible
Calculate probability
Select best VM's probability
Start scheduling task
Retrieve (τ),(η),(LB),(L_{VM})
End

Fig. 1. The flowchart of H_BAC

$$LB_j = \begin{cases} 1 & \text{if } TLD_j < \max.VM_{capacity_j} \\ 0 & \text{if } TLD_j > \max.VM_{capacity_j} \end{cases} \tag{8}$$

where LB_j is the parameter which is used to decide if load balance is possible or not in VM_j and TLD_j is the total task length in VM_j with the length of next task which isn't scheduled.

$$TLD_j = \sum\nolimits_{i=1}^{m} TL_i + TL_{next\ task} \tag{9}$$

As in (8), load balance is possible if TLD_j of VM_j does not exceed the maximum capacity of selected VM_j and then calculate probability by (1). However, load balance isn't possible if TLD_j exceed the maximum capacity of selected VM_j.

4 Simulation Results

In this section, H_BAC algorithm was compared against ABC [10], ACO [13], and Hybrid [18] algorithms.

4.1 Simulation Environment

The proposed H_BAC algorithm has been implemented using CloudSim API 3.0.3. Table 1 shows the values of the experimental parameters that have been set for experiments [14]. 100 runs were executed of the simulator for each experiment. The readings from these 100 trials were then averaged and plotted.

4.2 Performance Metrics

The performance metrics that were used to evaluate the performance of load balancing techniques are execution time, response time, standard deviation, makespan, and resource utilization.

Execution Time
It is the quantity of total time taken for scheduling total cloudlets in VMs.

Response Time
It is the quantity of time taken between submission of asking and the initial response that's created.

Standard Deviation
Standard Deviation (SD) is calculated in order to measure the deviations of load on VMs. The smaller SD means more balanced system. It can be defined as [13]:

$$SD = \sqrt{\frac{1}{n}\sum\nolimits_{j=0}^{n} (X_j - \overline{X})^2} \tag{10}$$

Table 1. Parameters setting of cloud simulator.

Type	Parameter	Value
Datacenter	Number of datacenters	10
	Number of hosts	5
	Type of manager	Space_shared, Time_shared
	Number of PEs per host	2–4
	Bandwidth	2000
	Host memory (MB)	2048–10240
	Datacenter cost (The cost of using processing in this resource)	10
VM	Total number of VMs	10–210
	MIPS of PE	1000–2000
	Number of PEs per VM	1
	VM memory (MB)	512–2048
	Bandwidth (Bit)	1000
	Type of manager	Time_shared
Cloudlets	Total number of tasks	200–1400
	Length of task	1000–15000
	Number of PEs per requirement	1
	Type of manager	Space_shared
H_BAC algorithm parameter setting	Number of tasks	200–1400
	Number of iterations	100
	Number of Ants	5
	Number of Honeybees	15
	Α	0.8
	Γ	0.8
	Β	0.32
	Ρ	0.1

where X_j is processing time of a VM which can be calculated as:

$$X_j = \frac{\sum_{i=0}^{k} TL_i}{Vm_{capacity_j}} \tag{11}$$

and $\overline{X}$ is mean of processing time of all VMs which can be calculated as:

$$\overline{X} = \frac{\sum_{j=1}^{n} X_j}{n} \tag{12}$$

Makespan

It is the overall completion time of task T_i on VM_j as CT_{ij} [13].

$$\text{Makespan} = \max\{CT_{ij} | i \in T_i, j, i = 1, 2, \ldots..., n \text{ and } j \in 1, 2, \ldots, m\} \quad (13)$$

Resource Utilization

It is one of the most important parameters which have to be measured for the load leveling strategy [15].

$$\text{Resource Utilization} = \frac{\text{VM demand}}{\text{range of tasks}} \quad (14)$$

5 Experimental Results

Figures 2, 3, 4, 5 and 6 show the comparison between the proposed H_BAC algorithm with ABC [10], ACO [13], and Hybrid [18] algorithms in terms of average execution time, average response time, average standard deviation, average makespan, and utilization rate; respectively. The number of VMs is fixed and equal to 100 VMs while the number of tasks is gradually increased from 200 to 1400 tasks.

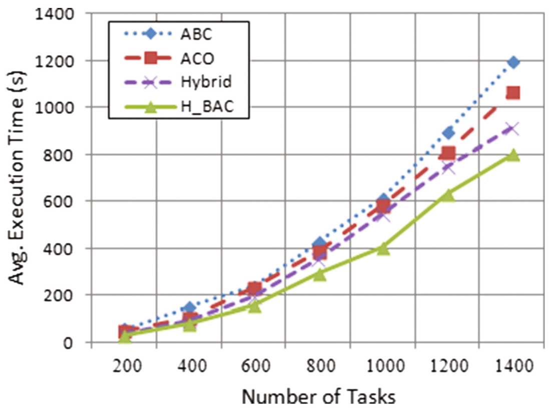

Fig. 2. Comparison of average execution time among H_BAC, ACO, ABC, and Hybrid algorithms versus the number of tasks.

Figure 2 shows the execution time of H_BAC and the other algorithms. It is shown that H_BAC improves execution time by about 36% with compared to ABC algorithm, 28% with compared to ACO algorithm, and 18% with compared to Hybrid algorithm. In Fig. 3, response time is presented. It is shown that H_BAC gets better than ABC, ACO, and Hybrid algorithms by 29%, 37%, and 18%; respectively.

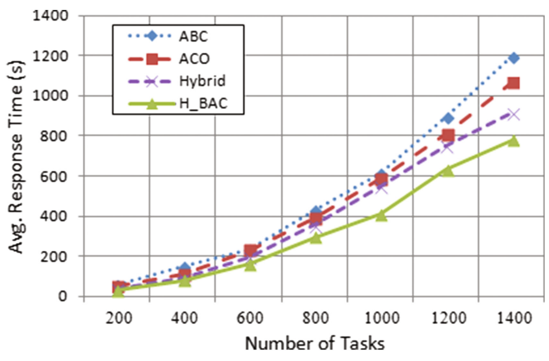

Fig. 3. Average response time of H_BAC, ACO, ABC, and Hybrid algorithms at the increased number of tasks.

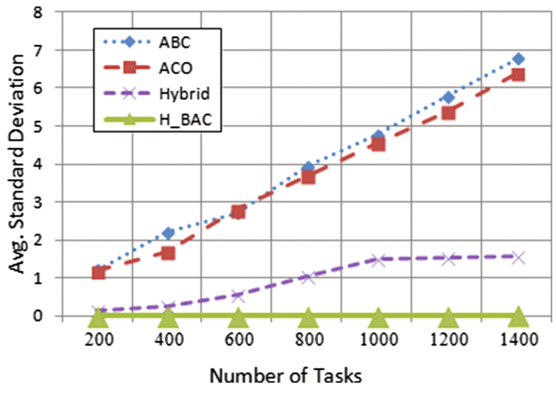

Fig. 4. Comparison of average standard deviation among H_BAC, ACO, ABC, and Hybrid algorithms at different number of tasks.

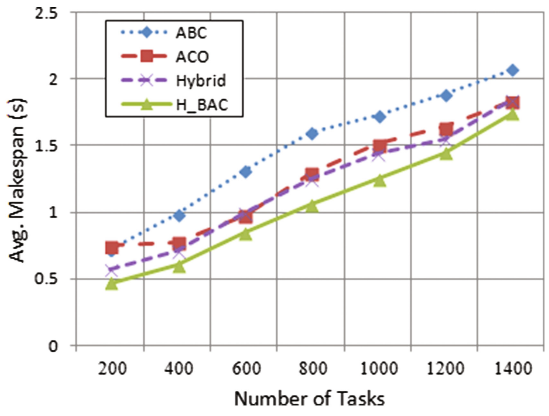

Fig. 5. Comparison of average makespan among H_BAC, ACO, ABC, and Hybrid algorithms at the increased number of tasks.

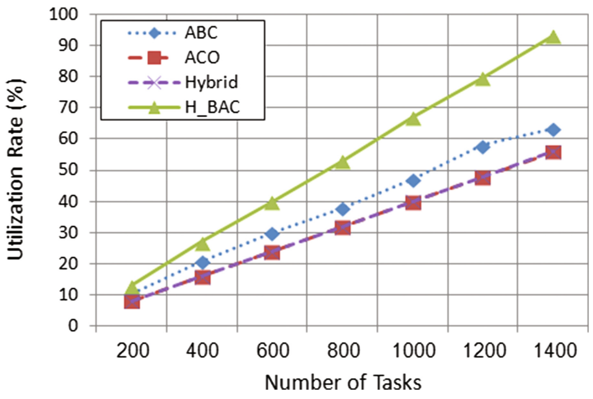

Fig. 6. Utilization rate of H_BAC, ACO, ABC, and Hybrid algorithms versus the number of tasks.

Figure 4 presents makespan of the four algorithms. Makespan of the proposed H_BAC algorithm is decreased by 30%, 18%, and 13% with compared to ABC, ACO, and Hybrid algorithms; respectively. The standard deviation is introduced in Fig. 5. It is clear that H_BAC realizes about zero standard deviation. It achieves about 99.6% improvement with compared to ABC, ACO, and Hybrid algorithms since it adds constraints in order to calculate and decide load balance for each VM. Then, H_BAC is the most balanced between the other algorithms.

An important parameter used in this work to check the load balancing strategy is utilization rate. In Fig. 6 utilization rate is presented. It is shown that the utilization rate of H_BAC is improved by 27%, 40%, 39% with compared to ABC, ACO, Hybrid algorithms; respectively. This is due to H_BAC adds two constraints at scheduling cloudlets in VMs; one for computing load balancing in VMs and the other to monitor total task length for deciding if the load balance is possible or not. These constraints give more accurate results in selecting the suitable VM and don't risk the balance of the system.

6 Conclusion

In this paper, a new algorithm is proposed to find load balancing for task scheduling in cloud computing. H_BAC inherits the main behaviors of both ACO and ABC algorithms. It takes into consideration the parameter of monitoring the load of VM and the decision of load balancing before scheduling tasks in VMs. H_BAC has been tested in large system to calculate the performance at various metrics. H_BAC decreases execution time, response time and makespan and verifies that it is the most balanced algorithm over ACO, ABC, and Hybrid algorithm. H_BAC uses two constraints in order to select the most suitable VM in the process and then guarantee the load balancing of the system. This leads to improve utilization rate.

References

1. Endo, P.T., Rodrigues, M., Gonçalves, G.E., Kelner, J., Sadok, D.H., Curescu, C.: High availability in clouds: systematic review and research challenges. J. Cloud Comput. Adv. Syst. Appl. **5**(1), 5–16 (2016)
2. Saber, W., Rizk, R., Moussa, W., Ghonem, A.: LBSR: Load balance over slow resources. In: International Conference on Computer Applications & Technology (ICCAT), Cairo, Egypt (2017)
3. Kumar, V.V., Revathi, R., Rajkumar, M.N.: An assessment on various load balancing techniques. Int. J. Adv. Inf. Commun. Technol. **1**(8), 667–670 (2014)
4. Patil, A., Gala, H., Kapoor, J.: Dynamic load balancing in cloud computing using swarm intelligence algorithms. Int. J. Comput. Appl. **130**(15), 15–21 (2015)
5. Moharana, S.S., Ramesh, R.D., Powar, D.: Analysis of load balancers in cloud computing. Int. J. Comput. Sci. Eng. **2**(2), 101–108 (2013)
6. Katoch, S., Thakur, J.: Load balancing algorithms in cloud computing environment: a review. Int. J. Recent Innov. Trends Comput. Commun. **2**(8), 2151–2156 (2014)
7. Singh, G., Kaur, A.: Bio inspired algorithms: an efficient approach for resource scheduling in cloud computing. Int. J. Comput. Appl. **116**(10), 16–21 (2015)
8. Thilagavathi, D., Thanamani, A.S.: Scheduling in high performance computing environment using firefly algorithm and intelligent water drop algorithm. Int. J. Eng. Trends Technol. **14** (1), 8–12 (2014)
9. Mandal, T., Acharyya, S.: Optimal task scheduling in cloud computing environment: meta heuristic approaches. In: Proceedings of the 2nd International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, pp. 24–28 (2015)
10. Tawfeek, M., El-Sisi, A., Keshk, A., Torkey, F.: Cloud task scheduling based on ant colony optimization. Int. Arab J. Inf. Technol. **12**(2), 129–136 (2015)
11. Nishant, K., Sharma, P., Krishna, V., Gupta, C., Singh, K.P., Nitin, Rastogi, R.: Load balancing of nodes in cloud using ant colony optimization. In: International Conference of Computer Modelling and Simulation (UKSim), Cambridge, pp. 3–8 (2012)
12. Pacini, E., Mateos, C., Garino, C.G.: Balancing throughput and response time in online scientific clouds via ant colony optimization (sp2013/2013/00006). Adv. Eng. Softw. **84**, 31–47 (2015)
13. Babua, L.D.D., Krishnab, P.V.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. **13**(5), 2292–2303 (2013)
14. Kruekaew, B., Kimpan, W.: Virtual machine scheduling management on cloud computing using artificial bee colony. In: The International Multiconference of Engineers and Computer Scientists (IMECS), Hong Kong, vol. I, pp. 18–22 (2014)
15. Rathore, M., Rai, S., Saluja, N.: Load balancing of virtual machine using honey bee galvanizing algorithm in cloud. Int. J. Comput. Sci. Inf. Technol. **6**(4), 4128–4132 (2015)
16. Saravanan, S., Venkatachalam, V., Malligai, S.T.: Optimization of SLA violation in cloud computing using artificial. Int. J. Adv. Eng. **1**(3), 410–414 (2015)
17. Singh, S., Vivek, T.: Implementation of a hybrid load balancing algorithm for cloud computing. Int. J. Adv. Technol. Eng. Sci. **3**(1), 73–81 (2015)
18. Madivi, R., Kamath, S.: An hybrid bio-inspired task scheduling algorithm. In: Proceedings of the 5th International Conference on Computing Communication and Networking Technologies (ICCCNT), China, pp. 1–7 (2014)