Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 57 (2015) 1219 – 1227

Procedia Computer Science

Multi **objective Task Scheduling in Cloud Environment Using Nested PSO Framework**

R K Jena

Institute of Management Technology, Nagpur, 440013, India

Abstract

Cloud computing is an emerging computing paradigm with a large collection of heterogeneous autonomous systems with flexible computational architecture. Task scheduling is an important step to improve the overall performance of the cloud computing. Task scheduling is also essential to reduce power consumption and improve the profit of service providers by reducing processing time. This paper focuses on task scheduling using a multi-objective nested Particle Swarm Optimization(TSPSO) to optimize energy and processing time. The result obtained by TSPSO was simulated by an open source cloud platform (CloudSim). Finally, the results were compared to existing scheduling algorithms and found that the proposed algorithm (TSPSO) provide an optimal balance results for multiple objectives.

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

Keywords: Task Scheduling, Cloud Computing, Multi-Objective optimisation, CloudSim, PSO

1. Introduction

Cloud computing is the next generation computational paradigm. It is an emerging computing technology that is rapidly consolidating itself as the future of distributed on-demand computing [1, 2]. Cloud Computing is emerging as vital backbone for the varieties of internet businesses using the principle of virtualization. Many computing frameworks are proposed for the huge data storage and highly parallel computing needs of cloud computing [1]. On the other hand, Internet enabled business (e-Business) is becoming one of best business model in present era. To fulfill the need of internet enabled business, computing is being transformed to a model consisting of services that are commoditized and delivered in a manner similar to traditional utilities such as water, electricity, gas etc. Users can access services based on their requirements without regard to where the services are hosted or how they are delivered. Several computing paradigms have promised to deliver this utility computing [3]. Cloud computing is one such reliable computing paradigm.

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)
doi:10.1016/j.procs.2015.07.419

Cloud computing architecture typically consists of a front end and a back end connected by Internet or Intranet [3]. The front end comprises of client devices like thin client, fat client or mobile devices etc. The clients need some interface and applications for accessing the cloud computing system. The back end consists of the various servers and data storage systems. A central server is used for administering the cloud system. The central server monitors the overall traffic and fulfilling the client demands in real time. The main objective of cloud computing environment is to optimally use the available computing resources. Scheduling algorithms play an important role in optimization process. Therefore user tasks are required to schedule using efficient scheduling algorithm. The scheduling algorithms usually have the goals of spreading the load on available processors and maximizing their utilization while minimizing the total execution time [4]. Task scheduling is one of the most famous combinatorial NP complete problem problems [5]. The main purpose of scheduling is to schedule the tasks in a proper sequence in which tasks can be executed under problem specific constraints [6].

This paper presents an optimization algorithm for user job scheduling to achieve optimization of energy consumption and overall computation time. The rest of the paper is organized as, section 2 contains a literature survey about scheduling in cloud computing, section 3 describes about the model development. Section 4 discusses about multi-objective PSO Algorithm. Section 5 discusses details about experimental setup and experimental results of the proposed model and the paper concludes with conclusion in Section 6.

2. Literature Review

In cloud computing environment, user services always demand heterogeneous resources (e,g CPU, I/O, Memory etc.). Cloud resources need to be allocated not only to satisfy Quality of Service (QoS) requirements specified by users via Service Level Agreements (SLAs), but also to reduce energy usage and time to execute the user job. Therefore scheduling and load balancing techniques are very crucial to increase the efficiency of cloud setup using limited resources. Task scheduling in Cloud computing has been addressed by many researchers in the past [7-10]. In 2011, Hsu *et al.* [9] focused on energy efficiency in datacenter by using efficient task scheduling to physical servers. Heuristic based techniques have also been used in task scheduling in cloud environment. Mondal *et al.* [11] used Stochastic Hill Climbing algorithm to solve load balance in Cloud computing. Hu *et al.*[12] introduced the scheduling strategy on load balancing of PE resource in Cloud computing environment by using Genetic algorithm. It considered previous data and the current state of work in advance to the performance behavior of the system which can solve the problem of load imbalance in Cloud computing. In 2012, Wei *et al.*[13] presented Genetic algorithm for scheduling in Cloud computing to increase the system performance. Li *et al.*[14] proposed a Load Balancing Ant Colony Optimization (LBACO) Algorithm to reduce makespan in Cloud. Karaboga *et al.*, [15] presented ABC algorithm to solve the problem and find the most appropriate parameters in changing environment. Bitam *et al.* [16] proposed Bee Life algorithm for scheduling in Cloud. Mizan *et al.* [17] also solved job scheduling in Hybrid Cloud by modifying Bee Life algorithm and Greedy algorithm to achieve an affirmative response from the end users.
There are many toolkit available to simulated and measure the performance of scheduling and load balancing algorithm in cloud environment. Simulation-based approaches can evaluate Cloud computing system and application behaviors. CloudSim toolkit package is one of the mostly used simulation tool used by many researchers [18-19]. Calheiros *et al.* [18] developed the CloudSim simulation for modeling and simulation of virtualized Cloud-based datacenter environments. The simulation environment consisting up dedicated management interface for PEs, memory, storage, and bandwidth etc.

From the above discussion, it is found that most of the previous researches have focused on optimizing a single objective, but very few of them optimize more than two objectives at a time. Therefore it is a good idea to measure the effect of multiple objectives on cloud scheduling problem. To deal with these gaps, a multi-objective PSO algorithm(TAPSO) is proposed to optimize the energy and time.

3. Model Development

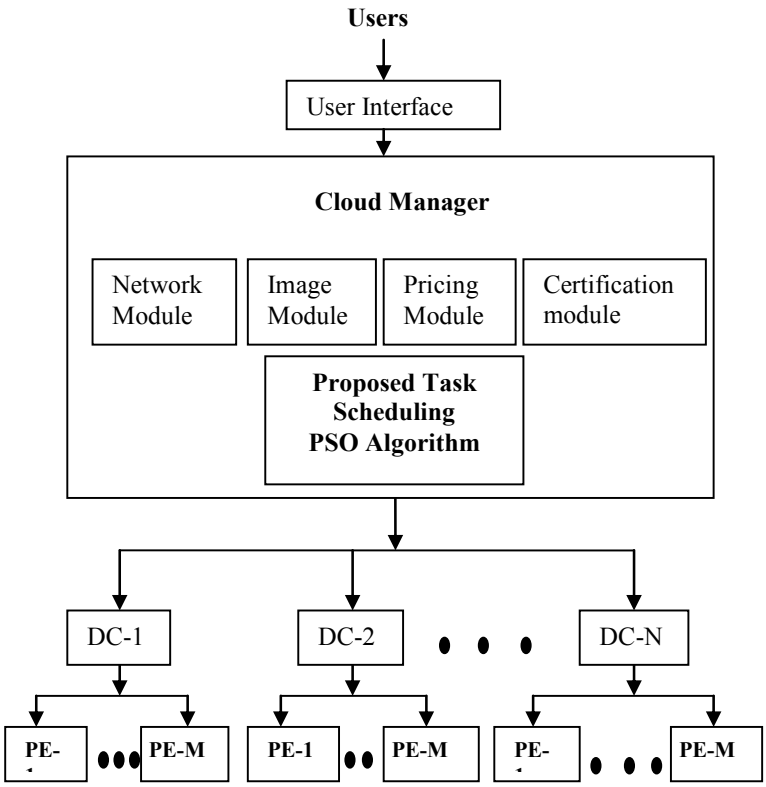

Fig. 1. Cloud Scheduling Environment

To solve the problem of resource optimization using PSO algorithm within the cloud framework, a typical cloud computing model is proposed as shown in fig. 1. The cloud system consists of many data center that are distributed geographically all over globe and are accessible using internet. Each data center consists of many computing, saving elements and other resources. Processing Elements (PEs) in each data center are connected by a high bandwidth communication network. Therefore negligible communication delay is considered in this model. In the proposed model, user can access the cloud resources using user interface. The proposed task scheduling module in the framework is responsible for efficient allocation of user tasks into different available PE with an objective to optimize energy consumption and time. In fig. 1, 'DC' indicates the Data Center and 'PE' indicate the sets of Processing Elements.

3.1. Problem Formulation

In the proposed model, a cloud application is considered as a collection of user tasks that carry out a complex computing task using cloud resources. During the scheduling process, the user tasks are assigned to the available data centers (DC's) ($D_1, D_2, D_3 \ldots . D_M$) . Each data center is associated with $< m >$. 'm' is the number of available Processing Elements (PEs) to execute user tasks. Each data center has set of Processing Elements $\{P_1, P_2 \ldots P_m\}$ to compute user's task. Each Processing Elements is associated with a duplet < s, p > . 's' and 'p' denotes the execution speed and power consumption of each Processing Elements respectively. Each User Job is represented as a as a Directed Acyclic Graph (DAG), denoted as $G(V, E)$ (figure 2). The set of nodes $V = \{ T_1 \ldots, T_n \}$ represents the tasks in user job, the set of arcs denotes precedence constraints and the control/data dependencies between tasks. An arc is in the form of $< T_i, T_j > \in E$, where T_i is called the parent task and T_j is the child task. The data produced by T_i is consumed by T_j . It is assumed that a child task cannot be executed until all of its parent tasks have been completed. In a given task graph, a task with no parent is referred as an *entry task*, and one

without any child is called an *exit task*. In this model only one entry and one exit tasks node is considered. Therefore two dummy tasks T_{entry} and T_{exit} is added in the beginning and at the end of the DAG having zero execution time respectively (fig. 2).

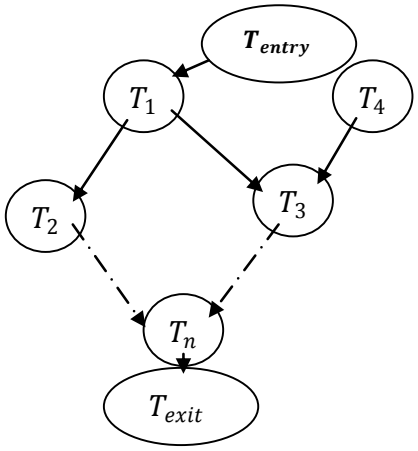

Fig. 2. Task Graph

Each vertex V in the DAG is associated with a value $< l >$, 'l' represents the length of the task in Million Instruction (MI). The problem of this model is how to optimally schedule user jobs to the Processing Elements available in the cloud under different data center. All the PEs is considered homogeneous, unrelated and parallel. Scheduling is considered as non preemptive, which means that the processing of any task can't be interrupted.

3.2. Objective Function

Suppose user job U_i is assigned to Data center D_j and T_j (a set of tasks of user job (U_i)) is assigned to a Processing Element (P_j). If the time require executing T_j using P_j is denoted by Γ_j. The finishing time of T_j can be expressed as:

$$Finish(T_j) = \text{start}(T_j) + \Gamma_j \quad (1)$$

So, the total time spend to complete the user job by D_j (Makespan_j) can be defined as:

$$\text{Makespan}_j = \max\{Finish(T_j)\} \quad (2)$$

Where $T_{j=1\ldots n}$ the tasks are assign to D_j

The Energy consumption to compute the user job (U_i) by Datacenter D_j is calculated as follows:

$$E_j = \sum_{k=1}^{N} (\Gamma_k \times p_k) \quad (3)$$

The objective functions of this proposed model can be expresses as:

Minimize $\text{Makespan}_j \quad \text{j} = 1..\text{M}$ (4)

Minimize $E_j \quad \text{j} = 1..\text{M}$ (5)

Subject to:

1. The user job must finish before deadline (d_i)
2. Each user job can be allocated to only one Data center.

4. **Multi-objective Optimization**

In multi-objective optimization (MO), there are several objectives to be optimized. Thus, there are several solutions which are not comparable, usually referred to as Pareto-optimal solutions. A multi-objective minimization problem with 'n' variables and 'm' objectives can be formulated, without loss of generality, as:

$$min\ y = f(\bar{x}) = \min(f_1(\bar{x}), f_2(\bar{x}), \ldots., f_m(\bar{x})) \quad (6)$$

Where $\bar{x} = (x_1, x_2 \ldots., x_n)$ and $y = (y_1, y_2 \ldots, y_m)$

In most cases, the objective functions are in conflicts, so that is not possible to reduce any of the objective functions without increasing at least one of the other objective functions. This is known as the concept of pareto-optimality. In order to deal with the multi-objective nature of task scheduling problem, a multi-objective PSO based framework was proposed.

4.1. PSO Approach

Problems with multiple objectives are present in a great variety of real-life optimization problems. In these problems there are several conflicting objectives to be optimized and it is difficult to identify the best solution. Despite the considerable diversity of techniques developed in the operations research field to tackle these problems, their intrinsic complexity calls for alternative approaches. Over the last decades, heuristics that find approximate solutions have attracted great interest. From these heuristics, Multi-Objective Evolutionary Algorithms (MOEAs) have been found to be very successful to solve multi-objective optimization problems. Another technique that has been adopted in the last years for dealing with multi-objective optimization problems is Particle Swarm Optimization (PSO) [20-21], which is precisely the approach adopted in the work reported in this paper.
The PSO algorithm was first proposed by J. Kennedy and R. Eberhart in 1995 [22-23] and it was successfully used in several single-objective optimization problems. PSO is based on the behavior of communities that have both social and individual conducts, similar to birds searching for food. PSO is a population-based algorithm. Each individual (particle) represents a solution in a n-dimensional space. Each particle also has knowledge of its previous best experience and knows the global best experience (solution) found by the entire swarm. Particles update their exploration directions (their weights) using the following equations:

$$v_{i,j} = w \times v_{i,j} + c_1 \times r_1 \times (p_{i,,j} - x_{i,j}) + c_2 \times r_2 \times (p_{g,j} - x_{i,j} \quad (7)$$

$$x_{i,j} = x_{i,j} + v_{i,j} \quad (8)$$

Where 'w' is the inertia factor influencing the local and global abilities of the algorithm, $x_{i,j}$ is the velocity of the particle 'I' in the j^{th} dimension, c_1 and c_2 are weights affecting the cognitive and social factors, respectively. r_1 and r_2 ~υ (0,1), p_i stands for the best value found by particle 'i' (pbest) and p_g denotes the global best found by the entire swarm (gbest). After the velocity is updated, the new position i in its j^{th} dimension is calculated. This process is repeated for every dimension and for all the particles in the swarm.

In order to use PSO for multi-objective optimization problems, the PSO algorithm is hybridized with some concepts taken from the EAs field such as a mutation operator, and with concepts commonly used in MOEAs, such as a selection based on Pareto dominance and mechanisms to produce a good spread of solutions. The nested multi-objective PSO (MOPSO) used in this paper is presented below:

Algorithm MOPSO( )
{
Initialize External Archive (Æ) // Æ =∪ *E*
For j = 1 to M (*M* is the size of particle swarm)
Initialize S_j & V_j // Initialization of each particle swarm and its velocity
For k=1 to L // L is the number of iteration
{
For j = 1 to M
E[j] = PSO (S_j)// E [j] is the archive for particle swarm S_j
Update the archive (Æ) of non-dominated solutions

Select leader particle from the archive (Æ)//
Update velocity
Update position
}
Return (Non-dominated solution)
}

Algorithm PSO(S_j)
{ // S_j : represents the set of user tasks of j[th] particle allocated to different data center (D_t) ,
t = 1,2…P
For t = 1 to P //P is the number of available data center in Cloud environment
{
For i = 1 to N (N is the size of particle swarm)
{
Initialize S[i]
Initialize the velocity V of each particle V[i]
Initialize the Personal Best *pBest* of each particle: $pBest[i]$ = S[i]
Evaluate objectives of each particle: *Evaluate S[i]*
Initialize the Global Best particle (*gBest*) with the best one among the 'N' particles:
g$Best$ = *Best particle found in S*
} // end of loop 'i'
Add the non-dominated solutions found in S into *EA[t]* // *EA[t]* is the External Archive storing the pareto front for the task assign to data center D_t
Initialize the iteration number (k) = 0
Repeat until k > G // (G is the maximum number of iterations)
{
For i = 1 *to N* (swarm size)
{
Randomly select the global best particle for S from the External Archive *EA[t]* and store its position in *gBest*.
Calculate the new velocity V[i] according to (7)
Compute the new position of S[i] according to (8)
If ($t < G * PMUT$) *then* // (*PMUT* is the probability of mutation)
Perform mutation on S[i]
Evaluate S[i] *using (2) and (3)*
Update the personal best solution of each particle S[i]
Update the External Archive *EA[t]*
} // end of loop 'i'
}
Retain the best pereto solution in *EA[t]*
} // end of loop 't'
Return (Min { EA[t].Makespan$_{t=1..P}$}, Sum{ EA[t].Energy$_{t=1..P}$})
}

5. **Implementation and Result**

The multi-objective PSO (MOPSO) was implemented using C++ object oriented framework. PSO related parameters used in the experiment are shown in table 2. CloudSim-3.0.1 is used to evaluate the scheduling of MOPSO. The experiments consist of 20 datacenters and 180-360 tasks under the simulation platform. The parameters setting on the proposed algorithm is shown in table 1.

Table 1. Workload Parameters

Type	Parameters	Values
Datacenter	Number of Datacenter	20
	Number of PE per Datacenter	10-20
Processing Elements(PEs)	Speed of PE	1000-200000MIPS
	Power Consumption	0.28-3.45kW
Task	Total Number of Tasks	180-360
	Length of Tasks	5000-15000 Million Instruction

Table 2. Parameter values for PSO

Iterations	External File size	Mutation operator	Particles	C1 =C2	W
5000	600	0.5	30	1.5	0.5

Several experiments and with different parameter setting were performed to evaluate the efficiency and efficacy of MOPSO algorithm. Most of the past research schedule tasks based on earliest finish time, earliest starting time or the high processing capabilities. All these algorithms selected resources based on its performance was denoted as "best resource selection" (BRS) approach. In this research, results of MOPSO were compared with BRS and Random Scheduling Algorithm (RSA). The BRS aims to maximize the number of scheduled applications, while the RSA randomly assigns the applications to the cloud.

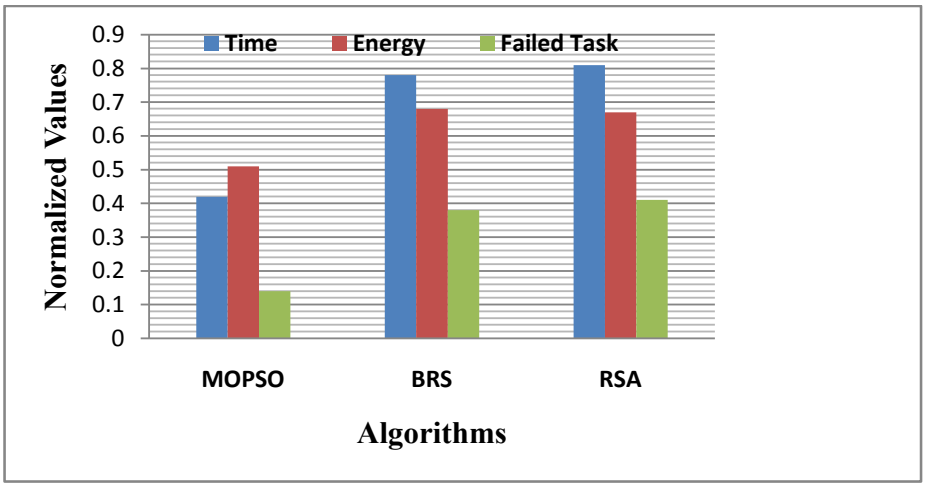

Fig. 3. Comparisons of different approaches

Fig. 3 showed a comparison of results between MOPSO, BRS and Random Scheduling Algorithm(RSA). The proposed algorithm (MOPSO) reduced 30% of energy consumption and 25% of time (Makespan) in compare to other scheduling algorithm. The figure-3 also showed that MOPSO also reduced the number of failed tasks, which generally increase the profitability of the cloud environment.

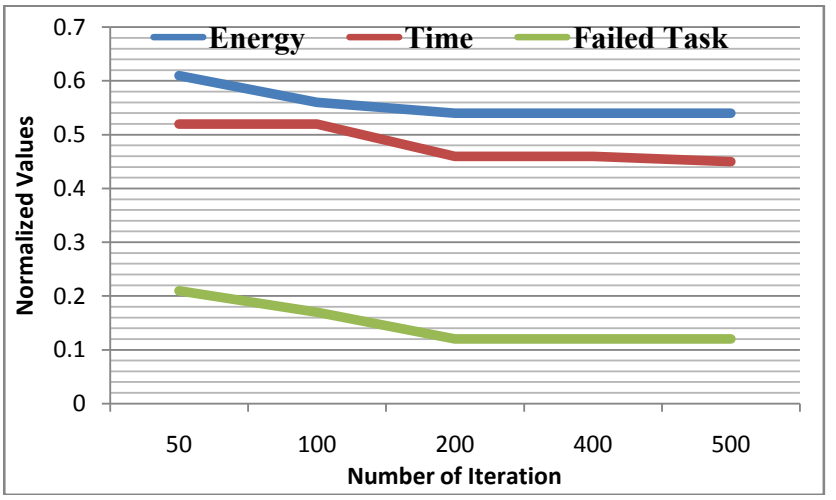

Fig. 4. Optimal values with respect to number of Iteration

The fig. 4 showed the effect of optimal solutions with respect to increased number of iteration. The increased number of iteration improves the quality of solution up to a certain limit. The solution doesn't change much after that. Result showed (Figure-4) after 2000 iterations the quality of solution doesn't improve significantly. Again the number of maximum iteration depends on the complexity of the scheduling i.e the number of user job, number of data center etc.

6. **Conclusion**

This paper presented multi-objective PSO based optimization algorithm which can solve the task scheduling problem under the computing environment, where of the number of data center and user job changes dynamically. But, in changing environment, cloud computing resources needs to be operated in optimally manner. Therefore, multi-objective nested PSO based algorithm was suitable for cloud computing environment because the algorithm was able to effectively utilize the system resources to reduce energy and makespan. The experimental results illustrated that the proposed methods (MOPSO out-performed the BRS and RSA. For further studies, the optimization model should add more objectives (bandwidth, load balancing, cost etc) and should focus more robust algorithm.

Reference

1. J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters, Sixth Symposium on Operating System Design and Implementation (OSDI'04). Dec. 2004, p. 1-13.
2. Rajkumar Buyyaa, Chee Shin Yeo, Srikumar Venugopal, James Broberg, Ivona Brandic . Cloud Computing and Emerging IT platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. *Journal Future Generation Computer Systems*. 2009; **25** : 6.
3. Liu, K. Scheduling Algorithms for Instance Intensive Cloud Workflows. *Ph.D. Thesis*, Swinburne University of Technology, Australia, 2009.
4. A. Y. Zomaya, and Y. The Observations on using genetic algorithms for dynamic load-balancing. *IEEE Transaction on Parallel and Distributed Systems*; 2001; **12:** 899-911
5. Yu, J., and Buyya, R. Workflow Scheduling Algorithms for Grid Computing,; In Xhafa F, Abraham A (eds), *Metaheuristics for scheduling in distributed computing environments,* Springer, Berlin, 2008.
6. XIAO Zhi-Jiao, CHANG Hui-You,YI Yang. An Optimization Method of Workflow Dynamic Scheduling Based on Heuristic GA; *Computer Science*; 2007;**34**:2.

7. S.Sindhu, and S. Mukherjee. Efficient task scheduling algorithms for cloud computing environment. In *High Performance Architecture and Grid Computing, Communications in Computer and Information Science*; 2011; **169:** 79-83.
8. S. T. Maguluri, R. Srikant, and L. Ying. Stochastic models of load balancing and scheduling in cloud computing clusters; In *Proc. IEEE Infocom;* 2012. p. 702-710.
9. Y. C. Hsu, P. Liu, and J. J. Wu Job sequence scheduling for cloud computing. In *Int. Conf. on Cloud and Service Computing (CSC 2011)*; 2011. p. 212-219 .
10. Y. Fang, F. Wang, and J. Ge. A task scheduling algorithm based on load balancing in cloud computing. In *Web Information Systems and Mining, Lecture Notes in Computer Science; 2010***, 6318**:271-277.
11 B. Mondal, K. Dasgupta, and P. Dutta. Load balancing in cloud computing using Stochastic Hill Climbimg-A soft computing approach. In *Procedia Tachnology*, 2004; **4**: 783-789.
12. J. Hu, J. Gu, G. Sun, and T. Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In *3rd Int. Symp. on Parallel Architectures, Algorithms and Programming(PAAP).* 2010. p. 89-96.
13 Y. Wei, and L. Tian. Research on cloud design resources scheduling based on genetic algorithm. In *2012 Int. Conf. on Systems and Informatics (ICSAI 2012)*. 2012. p. 2651-2656.
14. K. Li, G. Xu, G. Zhao, Y. Dong, and D. Wang. Cloud task scheduling based on load balancing Ant Colony Optimization. In *6th Annual ChinaGrid Conf.*; 2011. p. 3-9.
[15] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga. A comprehensive survey: artificial bee colony (ABC) algorithm and applications. In *Artificial Intelligence Review 2012*, *Springer Science Business Media B.V*. 2012.
16. S. Bitam. Bees life algorithm for job scheduling in cloud computing. In *Conf. on Computing and Information Technology (*ICCIT *2012)*; 2012.pp. 186-191.
17. T. Mizan, S. M. R. A. Masud, and R. Latip. Modified bees life algorithm for job scheduling in hybrid cloud. *Int. Journal of Engineering and Technology(IJET).* 2012; **2**: 974-979.
18. R. N. Calheiros, R.Ranjan, C. A. F. D. Rose, and R. Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software : Practice and Experience.* 2011; **41**:23-50.
19. R. N. Calheiros, R.Ranjan, C. A. F. D. Rose, and R. Buyya. CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv preprint arXiv: 0903.2525*, 2009.
20. J. Kennedy and R. Eberhart. Swarm Intelligence. Morgan Kaufmann Publishers, California,USA, 2001.
21. J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Congress on Evolutionary Computation* . **2**: 1671-1676.
22. J. Kennedy and R. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks, Piscataway, NJ.* 1995.IEEE Sevice Center. p. 1942-1948.
24. C. Coello Coello and M. Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *Congress on Evolutionary Computation,* Piscataway; NJ. 2002. IEEE Service Center, p. 1051-1056.