Cluster Comput
DOI 10.1007/s10586-017-1213-9

An improved load balanced metaheuristic scheduling in cloud

M. Aruna[1] · D. Bhanu[2] · S. Karthik[3]

Received: 8 March 2017 / Revised: 16 June 2017 / Accepted: 10 July 2017

Abstract Cloud computing refers to on-demand delivery of service over internet and has application in various domains like media, research, business, bigdata analysis etc. Task scheduling is one of the prime issues in this type of environment. Various metaheuristic algorithms and hard optimization problems have been proposed for solving cloud task scheduling which is a non-deterministic polynomial or an NP. Adaptation of the scheduling strategy to the changes taking place in the environment has to be done by a good scheduler. A proposal for cloud scheduling by means of a balanced load using both firefly algorithm (FA) and particle swarm optimization (PSO) heuristics has been made. The aim is to balance the load of the entire system while at the same time bring down the makespan of a set of tasks. This new strategy for scheduling has been simulated with CloudSim tool kit package. The results of this experiment proved that the proposed FA performed better than min–min scheduling, PSO, and also the first come first serve methods.

Keywords Cloud computing · Scheduling · Load balancing algorithm · Firefly algorithm (FA)

✉ M. Aruna
aruna.cse.4388@gmail.com

D. Bhanu
bhanu.saran@gmail.com

S. Karthik
profskarthik@gmail.com

[1] Department of Computer Science and Engineering, Surya Engineering College, Erode, India

[2] Department of Computer Science and Engineering, Karpagam Institute of Technology, Coimbatore, India

[3] Department of Computer Science and Engineering, SNS College of Technology, Coimbatore, India

1 Introduction

The rapid development of processing technologies and storage technologies owing to the internet has rendered computing resources cheaper, powerful and easily available. This trend in technology has ensured the advent of a new model known as cloud computing which enables resources like storage and CPU to be used by users on demand. In the environment of cloud computing, service providers hire these storage resources and pass it on to end users. The impact of cloud computing on the industry of Information Technology over the last few years has been tremendous. Companies like Amazon, Google, and Microsoft have started providing reliable, powerful and cost efficient platforms, and many enterprises work to revamp their business models to achieve these benefits [1].

To be able to meet the requirements of the end users and to provide quality services, cloud computing should possess certain traits, namely: **On-demand self-service** it is important for a consumer to be able to access various services like storage, software, computing and so on automatically without the intervention from the provider. **Broad network access** in availing the services of cloud computing, the internet plays an indispensable role. A good, broad network range will ensure accessibility of these services by means of web enabled services like mobile phones, computers and laptops. **Resource pooling** the resources normally available are software, Virtual Machines (VM), storage and bandwidth of network. These resources when pooled and made available to users in a single location optimizes their experience. **Rapid elasticity** the selling point of cloud computing is certainly its elasticity. Even indefinite resources are accessible for any length of time. **Measured service** the cloud system should possess a metering capability for the purpose of charging users. They can also levy an optimal charge for quality of

Published online: 25 September 2017

services (QoS) based on the type and duration of the services. [2].

Cloud computing services fall under three broad categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software as-a-Service (SaaS). IaaS pertains to the delivery of big computing resources like the processing capacity, network and storage. This is also known as Hardware as a service (HaaS). PaaS makes an abstract of the infrastructure and also supports an interface of application programs to various applications of clouds. It bridges the gap between the hardware and its application. SaaS however tries to replace the applications that run in a PC. It takes away the need for a special software. This concept sells as well as cloud computing, but the network delay is a big drawback.

Virtualization is a term used to denote anything that is almost real but not real. In the given context it refers to a machine implementation that executes various programs like a real one. Through this, different services or applications can be used by the end user. There are two different types of virtualization, namely: full virtualization which denotes the installation of a complete machine on another machine. This virtual machine provides all the services rendered by the original machine so that when it is engaged the VM can be used in its place. Para virtualization denotes the hardware that permits different operating systems on the same machine while ensuring efficient use of resources like processor and memory.

Many technical challenges need to be addressed like fault tolerance, VM migration, consolidation of server and scalability and high availability; however, the main issue in VM migration is the consumption of energy. In order to achieve efficiency in utilization and processing of an infrastructure in computing, as well as to bring down energy consumption, Green cloud computing is used. The most important and commonly implemented techniques for management of the power level of the server is the DVS or the dynamic voltage scaling also known as on/off power. This is implemented for the purpose of bringing down the power consumption and ensuring that the workload is dependent on a minimum number of physical nodes, as well as the switching off of the idle nodes [3].

Load balancing is a networking method that is used for the distribution of workload across many computers or a cluster of computers, central processing units, drives, network links or any other resources for the purpose of achieving optimum utilization of resources, increase throughput, bring down response time and to avoid overloading. As opposed to employing multiple components along with load balancing, a single component is used to improve its reliability by means of redundancy. A dedicated software and sometimes a hardware provides this service like a DNS or a Domain Name System server or a multilayer switch [4].

One of the main issues in cloud computing is load balancing which is a mechanism that undertakes the even distribution of the local workload across all nodes in the cloud to avoid conditions of overloading in some cases or completely no work in others. This helps in getting a high level of user satisfaction and improves overall performance by improving the utilization of resources. It makes sure that the resources used for computing are fairly and evenly distributed. If any of the components of the service fail, this system helps in continuing the service by making use of fairover by de-provisioning or in-provisioning applications.

Load balancing aims at improving performance by resource load balancing among central processing units, network links and disk drives to ensure optimal utilization of resources and avoidance of overload [5]. There are two types of algorithm namely static and dynamic. The former uniformly divides the traffic among servers thereby easing traffic on servers. The latter algorithm chooses suitable weights on the server by making a search on the entire network and chooses the lightest server for balancing the traffic. However, choosing the right server requires valid communication inside networks that can lead to adding of further traffic to the system.

Various metrics are considered for load balancing of cloud computing. They are: Scalability which is the algorithm's capacity to carry out load balancing for any system that has finite nodes. An improvement is required for this metric. Resource utilization is the method for checking the system's efficiency. This can be improvised at a reasonable cost by reducing response time, though acceptable delays are considered. Response Time refers to the time taken by a given algorithm belonging to a distributed system to respond. A minimisation is required for this parameter. Finally, Overhead associate computes the overhead involved in the implementation of this load balancing algorithm. This has the overheads from the movement of tasks, interprocess communication and inter-processor communication. To ensure the technique's efficiency, this should also be minimised [6].

A novel and effective load balancing algorithm is employed in this work by using PSO, as well as FA. Section II makes a review of all literature related work. Methods used are described in Section III and Section IV enumerates the results of the experiments while the work is concluded in Section V.

2 Related work

Providers of cloud service find load balancing the biggest challenge in computing environment's operation. Load balancing properly and evenly distributes the traffic and brings down the response time, ensures optimization of resources, maximizes the throughput and avoids overloading of a single resource. Pattanaik et al. [7] focused on the algorithm's dynamic load balancing and modified the PSO and first come first serve (FCFS) on the basis of their performance by means of using a Cloud Analyst Simulator. The outcomes of simulation are recorded on the basis of response time and the processing time of the datacenter and the three algorithms with the cost details of its arrival.

An introduction of a balance model that was better in terms of cloud partitioning with a mechanism to switch and choose other strategies for changing situations was made by Xu et al. [8] The game theory for the strategy of load balancing to improve the public cloud environment is used by the algorithm. Thakur and Kumar [9] made a comparison of three algorithms and their performance with various policies of different service brokers like round robin, ESCE or equally spread current execution as well as throttled load balancing. Simulation is made based on CloudSim based tools. The results have proved that this algorithm along with the service broker policy was comparatively superior. From the point of view of cost, however, the service broker policy of round robin is the most cost effective as the migration overheads are low. Ariharan and Manakattu [10] further introduced a neighbor-awareness mechanism for prediction in order to bring about an improvement in the process of selection of the nodes for the random walk. This proposed algorithm chooses the node that is least loaded from the list of neighbors. This is achieved by computation of the probability of every neighbor on the basis of the load of the neighbor perceived. So, the chances of selecting a node that is lightly loaded can be brought up and so the waiting time of the job can be brought down.

A key to solving the problems of load balance, to not just balance the load but to also meet the needs of the users, is the perfect scheduling algorithm. Pan et al. [11] made a proposal for an algorithm that is optimal load based. Here, the production of systems and scheduling of tasks to the VM can be enhanced. The finishing time for tasks within the system will be lower than that of others. Grover and Katiyar [12] made use of an approach called ABDLB or agent based dynamic load balancing where the mobile agent has an important role to play which is that of an entity of software and an independent program that is run on the network administrator's behalf. This method has the capacity to learn. On comparing this method with the traditional one it was concluded that this load balancing method brings down the cost of communication among servers, increases the load balancing rate that in turn improves throughput and the cloud's response time. Babu and Samuel [13] further proposed a method for modifying the bee colony algorithm for more efficient balancing of load in the environment of cloud. The foraging behavior of the honey bees was used in balancing the load in the VMs. The under-loaded VM was treated as the source of food and those that were overloaded as honeybees. This method also made efforts to bring down makespan and also the number of migrations of the CM. The results of the experiment displayed a noticeable improvement in the delivered QoS to the customers.

The algorithms for load balancing in the present situation should be made very efficient in allocation of requests. It should ensure that the resources are used intelligently thereby resources are neither over-utilized nor under-utilized. Joshi and Verma [14] made a proposal for a load balancing approach called IGA or Improvised Genetic Algorithm for the purpose of allocating jobs to the server or to the VM. This algorithm makes consideration for the fitness function's cost value. This strategy is simulated with the help of a MATLAB toolkit. Cho et al. [15] made a combination of ACO or Ant Colony Optimization with the PSO or Particle Swarm Optimization to find a solution to the scheduling problem of VM; this was known as ACOPS or the ACO with Particle Swarm. ACOPS makes use of information from history and this is used to predict the workload of new requests of input to be adapted in environments that are dynamic and without additional information on task. ACOPS rejects those requests that are impossible to be satisfied before the scheduling is done in order to reduce the time of computation of the procedure of scheduling. The results of the experiments show that this algorithm can maintain the load balance in an environment that is dynamic and perform better than other approaches. Dam et al. [16] made a novel suggestion in terms of a strategy for load balancing by making a search for an under-loaded node to help balance load from an overwhelmed node. A simulation tool is used by the cloud analysts to balance this. The result of the experiment was found to be positive. The proposed algorithm on a comparative basis outperformed almost all other strategies like the FCFS and also the SHC or the Stochastic Hill Climbing and other approaches of soft computing like GA and ACO. Table 1 shows the summary of related works.

3 Methodology

Scheduling is the process of allocation of tasks towards the resource available based on the qualities and needs of the task. The prime aim here is to bring about the utilization of various resources without adversely affecting the services of the cloud. The scheduling problem includes those tasks that are scheduled on the basis of some constraints in order to optimize certain functions. Normally, mapping of tasks on unlimited resources of computing falls under NP-hard problems. Techniques that are metaheuristic in nature can handle these problems and provide almost optimal solutions on time. This has gained momentum in the recent years owing to its efficiency in solving complex problems. Here the min–

Table 1 Summary of related works

Authors	Techniques	Merits	Demerits
Pattanaik et al. [7]	FCFS and PSO algorithm	Minimizing Response time, optimizing resources, maximizing throughput, and avoiding single resource overloading	Resource overloading
Xu et al. [8]	Load balancing model	Improve the efficiency	Efficiency and user satisfaction
Thakur and Kumar [9]	Load balancing algorithms	Optimistic and closest service broker policy	Overheads
Ariharan and Manakattu [10]	NARS algorithm	Improve the selection process of nodes for random walk	Random walk
Pan et al. [11]	ABC load balancing algorithm	Enhance production of the systems	Load balance problems
Grover and Katiyar [12]	ABDLB approach	Greatly reduces the communication cost of servers, improves the throughput and response time of the cloud	Load balancing system problem
Babu and Samuel [13]	Enhanced bee colony algorithm	Efficient and effective load balancing and minimize makespan	Cost effective scheduling
Joshi and Verma [14]	IGA	Fitness function	Resource discovery, fault tolerance etc.
Cho et al. [15]	ACOPS	To reduce the computing time of the scheduling procedure	VM scheduling problem
Dam et al. [16]	A novel load balancing strategy	CloudAnalyst used	High interoperability and scalability

min scheduling, the PSO algorithm, the FCFS scheduling and the load balancing scheduling of FA are enumerated.

3.1 Min–min scheduling

MET or Minimum Execution Time assigns a task to every resource that completes the task within stipulated time. However, whether the resources are at that time available or not is not considered. Assigning of task is made to such resources that do not have any minimum time for completion. Min–min start with its unassigned tasks are included and works in two phases. Firstly, calculation of MCT for all tasks is done. The time of completion is calculated. Secondly, the minimum time is chosen and tasks are assigned correspondingly [17]. The incomplete tasks are then removed from the respective makespan and a repeat of this process is made until the completion of all tasks takes place.

3.2 First come first served (FCFS) scheduling

For the purpose of parallel processing and resource targeting of those that have the lowest waiting line that is selected for the job received and this is known as FCFS scheduling. The Sim toolkit backs the FCFS scheduling plan with the interior tasks of scheduling. The app specified VMs are distributed to the hosts inside a datacenter that is cloud based within a given time by the VM. It adopts a default policy that is made in the FCFS method. The main drawback of FCFS is that it is not preventative. The long errands have to be completed before the short ones are taken up and the wait time is long. This method is also turnaround and the reaction to it is low [18].

In cases where the needed resources are not available, then the system has to wait to get it where on the other hand the algorithm will give it in parts or put a request in queue and wait to note if it is serviced. A dynamic allocation towards the constraint of deadline is followed based on the current usage. Then an allocation of data towards the requests on the basis of whichever type the request belongs to is made. In case of a constraint due to deadline, and if the request has a threshold value, it is serviced immediately, and if it is above threshold, then constraint based allocation is carried out. In case of allocation on the basis of cost constraint when there are simultaneous requests, whichever request that gives more efficiency of cost is first allocated.

The FCFS is considered more advantageous owing to its simplicity and also because it brings down the time of processing and its completion. It is very simple and allocates the CPU in the order of arrival. An assumption that the queue is managed in FIFO method or the First In First Out method is made without taking into consideration other preferences.

3.3 Particle swarm optimization (PSO)

PSO is based on the concept of bird flocking in which every particle becomes a solution to a problem space. There is a random initialisation of the particles with a particular fitness value which is calculated with the help of a fitness function which is the ideal solution of each generation. Each of the particle knows the best position which is the pbest and the ideal position until now in the entire set which is the gbest [19]. Hence, the pbest of every particle is the best and the ideal result which is calculated using fitness value whereas the gbest is the most ideal and best one among the entire population. This evaluation takes place based on repetitions. When the process of iteration optimization is taking place every such iteration of velocity and all the positions of each particle will be updated as per 1 and 2.

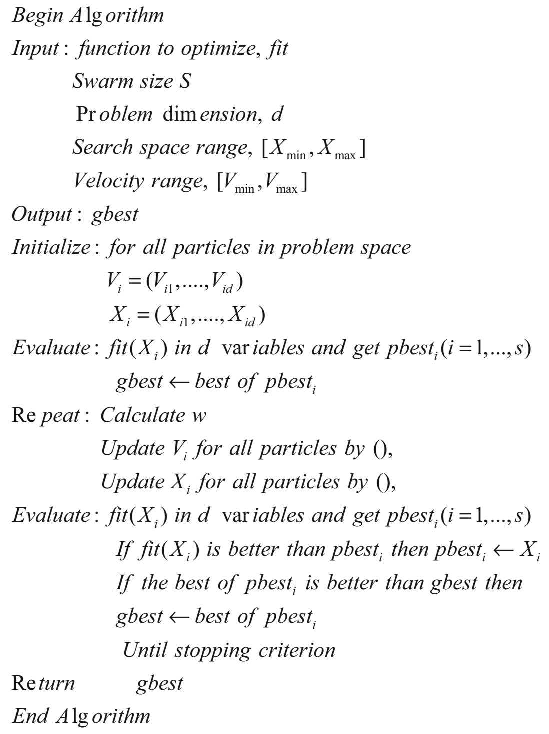
$$v_i(k+1) = w * v_i(k) + c_1 * r_1 * (pbest_i(k) - x_i(k)) + c_2 * r_2 * (gbest_i(k) - x_i(k)) \tag{1}$$

$$x_i(k+1) = k_i(k) + v_i(k+1) \tag{2}$$

In which,is the velocity of the particle **i** at the iteration k,are the random numbers that have a regular distribution within the period between 0 and 1.denote learning factors known as the cognition along with the social parameter as well as w which is inertia weight is dynamically different by application of an annealing scheme for w setting of PSO. w here is decreased and is contributed to the convergence. Generally, the setting of inertia weight w is according to equation (3).

$$w = w_{\max} - \frac{w_{\max} - w_{\min}}{itra_{\max}} \times itra \tag{3}$$

Here, w$_{max}$ denotes initial value, w$_{min}$ the final value of coefficient of weight, itra$_{max}$ the largest number of repetitions possible. The performance of this PSO algorithm is improved to a great extent by changing the inertia with an updated Eqs. (1) and (2). Here the pseudo code of PSO algorithm is shown in Fig. 1 [20].



```
Begin Algorithm
Input: function to optimize, fit
        Swarm size S
        Problem dimension, d
        Search space range, $[X_{\min}, X_{\max}]$
        Velocity range, $[V_{\min}, V_{\max}]$
Output: gbest
Initialize: for all particles in problem space
            $V_i = (V_{i1}, ...., V_{id})$
            $X_i = (X_{i1}, ...., X_{id})$
Evaluate: fit($X_i$) in d variables and get $pbest_i (i = 1, ..., s)$
             gbest ← best of $pbest_i$
Repeat: Calculate w
            Update $V_i$ for all particles by (),
            Update $X_i$ for all particles by (),
Evaluate: fit($X_i$) in d variables and get $pbest_i (i = 1, ..., s)$
             If fit($X_i$) is better than $pbest_i$ then $pbest_i \leftarrow X_i$
             If the best of $pbest_i$ is better than gbest then
             gbest ← best of $pbest_i$
              Until stopping criterion
Return      gbest
End Algorithm
```

Fig. 1 Pseudo code of PSO algorithm

For addressing the problem of scheduling in which there are a number of users that connect to the Cloud at various times for executing their respective PSEs and each of the user requests which are the creation of the v VMs. A PSE is nothing but a set of N independent jobs that correspond to certain values for each variable of the model that is studied by PSEs.

In case of the adapted PSO algorithm, every particle shall work independently representing a VM that looks for an ideal host for being allocated. On creation of a VM every particle is initialized into a random host of a random place in its field. In every iteration, the particle will move to neighbours in the current host searching for a lower load. Every particle's velocity will be defined by the difference between the host to which the particle has been assigned previously relating to its neighbouring host. In case any of the hosts in its neighbourhood have a lower load than that of the original, the particle is moved to its neighbouring host that has a greater velocity. Taking the particles that move through the hosts of neighbourhood into consideration, the algorithm shall reach its local optimum very quickly. So every particle will make a move from the host that it is associated with one neighbour that has the minimum load in all. In case all the neighbours are found to be busier than the host, the particle will not be moved from its present host. Lastly, the particle delivers the associated VM to that of the host that has the load that is lower from among their neighbours and also completes the task.

The PSO method easily suffers from the partial optimism which causes the less exact at the regulation of its speed and the direction. The PSO method cannot work out the problems of scattering and optimization. It cannot work out the problems of non-coordinate system, such as the solution to the energy field and the moving rules of the particles in the energy field [21].

As the FA divides it population automatically into various subgroups owing to the local attraction being stronger than that of the long distance attractions. The FA will not make use of the individual best and also their explicit global best. This can bring down the drawbacks of premature convergence. Further, it will not use the velocities and therefore, the problems that are associated with the PSO velocities will be eliminated automatically. The FA will have an ability in its ability as well as it mobility to control parameters like γ. So it is seen clearly that the FA will be more efficient in the parameters of control, its local search ability, its robustness and also its elimination of any premature convergence.

3.4 Firefly algorithm (FA)

The FA has been developed by author Yang, which was based on the fireflies and their flashing characteristics. To simplify, these flashing traits have three rules [22]:

- all fireflies are assumed to be unisex to ensure that attraction takes place irrespective of their sex;
- Attractiveness is directly proportional to brightness, so in case of two fireflies, the one that is less bright will move towards the one that is more bright. As the attractiveness depends on the brightness, both decrease as the distance between them increases. If there is one firefly that is the brightest it begins to move randomly;
- The brightness of a firefly or its light intensity is based on the landscape of the to be optimised objective function.

For a problem related to maximization, its brightness is taken to be proportional to its function. All types of brightness can be defined similarly to fitness functions in BFA which is Bacterial Foraging Algorithm or GA.

There are two issues in FA. The variation of the intensity of the light and the attractiveness and its formulation. To simplify, it can be assumed that the firefly's attractiveness is decided by the light intensity or its brightness which is also decided by the encoded function of the objective. In one of the simplest forms for optimizing maximum problems the actual brightness of I of a particular firefly in a specific location x is selected as $I(x)\alpha f(x)$. But the attractiveness β is here relative, and should be viewed in the eyes of the beholder, in other words judged by other fireflies. So it should change with the distance r_{ij} between firefly i and firefly j. As the intensity of light goes down along with the distance of the source of light, it is absorbed within the media and the attractiveness has to be permitted to vary as per the degree of its absorption [23].

In simple terms, light intensity I(r) changes according to the distance r both monotonically as well as exponentially in (4).

$$I = I_0 e^{-\gamma r} \tag{4}$$

In which I_0 denotes the intensity of the original light and γ denotes the coefficient of light absorption.

The firefly's attractiveness being proportional to the intensity of light that is noticed by the fireflies adjacent to it, the attractiveness β of a firefly can be defined in (5):

$$\beta = \beta_0 e^{-\gamma r^2} \tag{5}$$

In which $\beta_0$indicates the attractiveness which is at r = 0.

The actual distance that exists between two fireflies is defined by using the Cartesian distance in (6):

$$r_{ij} = |x_i - x_j| = \sqrt{\sum\nolimits_{k=1}^{d} \left(x_{i,k} - x_{j,k}\right)^2} \tag{6}$$

Firefly i is hereby attracted towards j, the more attractive firefly, its measure can be defined as (7):

$$\Delta x_i = \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha\varepsilon_i, x_i^{t+1} + \Delta x_i \tag{7}$$

In Δx_i which is an equation, the first term denotes attraction, which is the limitation where the value is near to zero or is too large. If γ approaching zero ($\gamma \rightarrow 0$), its attractiveness and its brightness become constant $\beta = \beta_0$. In other words, a firefly is noticed in any position so a global search can be easily made. If γ which is nearing infinity or is too large ($\gamma \rightarrow \infty$), then the attractiveness as well as its brightness may decrease and the movements of the firefly may become random. Two asymptotic behaviours can be used for the implementation of FA. When the second is a terms or randomization which is the parameter for randomizing it. Here, the ε_i can be replaced by ran -1/2 that is the random number generated from 0 to 1.

3.5 Proposed load balance scheduling based on firefly algorithm

The approach that is proposed makes a trigger to a method for the generation of a good strategy for load balancing to be done in the cloud network for the purpose of node scheduling. This process is executed by FA. The process of scheduling pertains to a node set that has the lowest load possession. Alternatively, the nodes that contain the least load are taken for making extra process by the network of the cloud. In this approach, VM along with three servers which in turn has three nodes is considered. Every node here has different attributes as defined by the process of scheduling.

VM's overview is decided by the approach proposed. This consists of three steps for the generation and the scheduling of the VM method proposed by balancing of nodes. They are listed below [24]:

Generation of population Population means the group of numbers that is required by the cloud system based on the needs of users. Once there is a request in the server, it brings a node that is free and available. The process is on the basis of the nodes, nine in number, mapped in accordance to their time for processing and scheduling and a list is made. This list is taken into consideration for the proposed approach as its initial population.

Calculation of scheduling index (SI) This is one of the main factors that affect the process of scheduling. Therefore, before it is done, a discussion on the parameters used for initial population has to be made. According to the FA's definition, an attraction between the fireflies has to exist denoted by nodes. This attraction has to be based on the affinity of one to another and it is controlled by the parameters of the decision which has been proposed for the approach. These proposed parameters are considered to be the attributes for the nodes in this approach.

The equation for attraction in accordance to the FA definitions is as below (8):

$$attr(n_i) = \frac{p_i}{cpu_i + mem_i} \tag{8}$$

In this, $attr(n_i)$ denotes the attraction that exists between the node and that of the request because the node is regarded in the attraction request is high. P_i here represents the time taken to process a given node. cpu_i denotes the rate of cpu of the node and finally mem_i denotes the rate of memory of the nodes. SI can therefore be derived from the formula mentioned above or from the one below in (9):

$$SI = \sum_{i=1}^{n} \frac{p_i}{cpu_i + mem_i} \tag{9}$$

The SI along with its total sum of nodes in a given queue of scheduling is shown. According to this equation all queues of scheduling is calculated and a list is prepared.

Selection of the node containing minimum load This process of node selection has been inspired from FA as the firefly that is least distinct has similar characteristics. As inspired from this theory, a calculation of the distance between nodes in the queues is computed [20]. Even before the calculation is made, the node with least $attr(n_i)$ values is found. This is considered to be the pivot point for this queue and least distinct nodes are thus calculated. The basis for calculation of the distance values of the nodes is the Cartesian distance which is as shown in (10):

$$Dist = \sqrt{\sum_{j=1}^{k} (n_i - n_j)^2} \tag{10}$$

The expression Dist and the n_i as the node selected and n_j as the node to be compared is considered. As soon as all distance values are computed between all the node values, they are duly rearranged on the basis of the lowest node to the pivot node. This is sorted out on the list of schedule on the basis of the SI value. The top most queue in the list is taken to be the ideal scheduling queue.

4 Results and discussion

In this section, the min–min scheduling, FCFS, PSO and firefly methods are evaluated using load balancing as the objective. The CloudSim is a generalized and extensible framework of simulation that permits simulation, seamless modeling, and experimentation of the infrastructures of emerging cloud computing application services and infrastructures. By means of using the Cloud Sim the researchers have developed another application service in an easy set up environment that is controlled. On the basis of the results of evaluation of results it has been reported by the CloudSim, that further fine tuning of the performance of service is possible. The advantages of using this for the initial performance in testing are (1) time effectiveness: this takes very little time for the implementation of the Cloud based application as well as testing the environment and (2) flexibility and applicability: the developers may be able to model as well as test the application services and their performance in the Cloud environments that are heterogeneous (Amazon EC2, Microsoft Azure) using very little programming and effort of deployment. CloudSim with Matlab were used in the experiments with simulation parameters shown in Table 2. The makespan and degree of imbalance are as shown in Tables 3 and 4.

From Table 3, it can be observed that the min–min scheduling has higher makespan by 6.59, 4.41 and 5.28% for FCFS, by 11.23, 8.36 and 11.26% for PSO and by 16.09, 13.15 and 15.44% for firefly when compared with 100, 300 and 500 number of tasks.

Table 2 Simulation parameters

Number of datacenter	15
Number of host in datacenter	4–8
Number of VMs	50
MIPS of processing element	1500–2500
Number of PE per VM	5–10
VM RAM	1024–4096
Number of tasks	200–1000
Length of task	5000–15,000 MI
Number of Pes requirement	2–5

Table 3 Makespan

Number of tasks	Min–min scheduling	FCFS	PSO	Firefly
100	47	44	42	40
200	104	98	93	89
300	162	155	149	142
400	219	207	199	190
500	272	258	243	233

Table 4 Degree of imbalance

Number of tasks	Min–min scheduling	FCFS	PSO	Firefly
100	3.8	3.7	3.6	3.4
200	3.6	3.5	3.3	3.1
300	3.9	3.8	3.6	3.4
400	4	3.8	3.7	3.4
500	3.5	3.4	3.5	3.4

Table 5 Execution time for each algorithm before and after load balancing

Number of tasks	Min–min scheduling	FCFS	PSO	Firefly
Execution time—before load balancing				
100	54	51	48	46
200	120	113	107	103
300	185	178	170	163
400	254	239	229	217
500	315	296	277	267
Execution time—after load balancing				
100	53	49	47	45
200	117	110	105	99
300	182	172	167	160
400	246	231	225	214
500	306	291	270	262

From Table 4, it can be observed that the min–min scheduling has higher degree of imbalance by 2.66, 2.59 and 2.89% for FCFS, by 5.4, 8% and same value for PSO and by 11.11, 13.69 and 2.89% for firefly when compared with 100, 300 and 500 number of tasks. Table 4 shows the execution times before load balancing and after load balancing for all the four algorithms.

From table 5 it can be seen that load balancing significantly improves the execution time by more than 5%.

5 Conclusion

The scheduling algorithm for cloud computing services is computed on the basis of focus given to load balancing. A schedule based on load balancing is formulated for the approach proposed. This approach is inspired by FA owing to its attracting features. This is developed in three different sets. Firstly, a population generation is made from cloud network. Secondly, an SI is calculated and finally a optimization of the scheduled list is done using FA. The results of the experiment show that the makespan of min–min scheduling is greater by 6.59, 4.41 and 5.28% in case of FCFS, by 11.23, 8.36 and 11.26% in case of PSO and by 16.09, 13.15 and 15.44% in case of firefly when 100, 300 and 500 tasks are considered. Both the FA and the PSO suffer from problems of local optima. Further work is done by means of exploring an efficient algorithm for load balancing that maintains a better balance from among the parameters and further helps in the achievement of green computing. So any work in future is completed by making use of the memetic techniques for handling the limitations of the algorithms that are meta-heuristic.

References

1. Zhang, Q., Cheng, L., Boutaba, R.: Cloud computing: state-of-the-art and research challenges. J. Internet Serv. Appl. **1**(1), 7–18 (2010)
2. Sajid, M., Raza, Z.: Cloud computing: issues challenges. In: International Conference on Cloud, Big Data and Trust, vol. 20, no. 13, pp. 13–15 (2013)
3. Kaur, P., Kaur, P.D.: Efficient and enhanced load balancing algorithms in cloud computing. Int. J. Grid Distrib. Comput. **8**(2), 9–14 (2015)
4. Haryani, N., Jagli, D.: Dynamic method for load balancing in cloud computing. IOSR J. Comput. Eng. **16**(4), 23–28 (2014)
5. Kashyap, D., Viradiya, J.: A survey of various load balancing algorithms in cloud computing. Int. J. Sci. Technol. Res. **3**(11), 115–19 (2014)
6. Saranya, D., Maheswari, L.S.: Load balancing algorithms in cloud computing: a review. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **5**(7), 1107–1111 (2015)
7. Pattanaik, P.A., Roy, S., Pattnaik, P.K.: Performance study of some dynamic load balancing algorithms in cloud computing environment. In: IEEE 2nd International Conference on Signal Processing and Integrated Networks (SPIN), pp. 619–624 (2015)
8. Xu, G., Pang, J., Fu, X.: A load balancing model based on cloud partitioning for the public cloud. Tsinghua Sci. Technol. **18**(1), 34–39 (2013)
9. Thakur, V., Kumar, S.: A comparison of select load balancing algorithms in cloud computing. IUP J. Comput. Sci. **9**(1), 7 (2015)
10. Ariharan, V., Manakattu, S.S.: Neighbour aware random sampling (NARS) algorithm for load balancing in cloud computing. In: IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), pp. 1–5 (2015)
11. Pan, J.S., Wang, H., Zhao, H., Tang, L.: Interaction artificial bee colony based load balance method in cloud computing. In: Genetic and Evolutionary Computing, pp. 49–57. Springer, New York (2015)
12. Grover, J., Katiyar, S.: Agent based dynamic load balancing in Cloud Computing. In: IEEE International Conference on Human Computer Interactions (ICHCI), pp. 1–6 (2013)
13. Babu, K.R., Samuel, P.: Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud. In: Innovations in Bio-Inspired Computing and Applications, pp. 67–78. Springer, New York (2016)
14. Joshi, G., Verma, S.K.: Load balancing approach in cloud computing using improvised genetic algorithm: a soft computing approach. Int. J. Comput. Appl. **122**(9) (2015)
15. Cho, K.M., Tsai, P.W., Tsai, C.W., Yang, C.S.: A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing. Neural Comput. Appl. **26**(6), 1297–1309 (2015)
16. Dam, S., Mandal, G., Dasgupta, K., Dutta, P.: Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing. In: IEEE Third International Conference on Computer, Communication, Control and Information Technology (C3IT), pp. 1–7 (2015)

17. Priyadarsini, R.J., Arockiam, L.: Performance evaluation of min-min and max-min algorithms for job scheduling in federated cloud. Int. J. Comput. Appl. (0975–8887) **99**(18), 47–54 (2014)
18. Kaur, R., Kinger, S.: Analysis of job scheduling algorithms in cloud computing. Int. J. Comput. Trends Technol. **9**(7), 379–386 (2014)
19. Pacini, E., Mateos, C., García Garino, C.: Dynamic scheduling based on particle swarm optimization for cloud-based scientific experiments. CLEI Electron. J. **17**(1), 3–3 (2014)
20. Azir, D.I.E.: Scheduling jobs on cloud computing using firefly algorithm. Doctoral dissertation, University of Science and Technology (2015)
21. Selvi, V., Umarani, D.R.: Comparative analysis of ant colony and particle swarm optimization techniques. Int. J. Comput. Appl. (0975–8887) **5**(4) (2010)
22. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. Int. J. Bio-Inspired Comput. **2**(2), 78–84 (2010)
23. Baskaran, M., Sadagopan, C.: Synchronous firefly algorithm for cluster head selection in WSN. Sci. World J. (2015). doi:10.1155/2015/780879
24. Florence, A.P., Shanthi, V.: A load balancing model using firefly algorithm in cloud computing. J. Comput. Sci. **10**(7), 1156 (2014)

M. Aruna received her Bachelor and Master degree in Engineering from Anna University, Chennai. She has 12 years of teaching experience from reputed Engineering Colleges and currently, she is working as an Associate Professor. Her research interests include Cloud Computing, Software Engineering and Software Testing. She has published papers in reputed Conferences and Journals.

D. Bhanu received her Master degree in Engineering and Ph.D. degree from Anna University, Chennai. She has 18 years of teaching and research experience in reputed Engineering Colleges and currently holds the additional responsibility of the Vice Principal. She has published papers in the reputed Conferences and Journals and is the member of various Professional bodies. Her research interests include Data Mining, Cloud Computing, Internet of Things and Service Oriented Architecture.

S. Karthik has received his Master Degree in Engineering and Ph.D. degree from Anna University, Chennai. He is presently Professor & Dean, Department of Computer Science & Engineering, SNS College of Technology. His research interests include Network Security, Web Services and Wireless Systems. In particular, he is currently working in a research group developing new Internet security architectures and active defense systems against DDoS attacks. He has published papers in International Journal and Conference papers. He is an active member of IEEE, ISTE and Indian Computer Society.