Cluster Computing (2019) 22:S287–S297
https://doi.org/10.1007/s10586-018-1823-x

Chaotic social spider algorithm for load balance aware task scheduling in cloud computing

V. M. Arul Xavier[1] · **S. Annadurai**[2]

Received: 21 November 2017 / Revised: 6 January 2018 / Accepted: 11 January 2018 / Published online: 29 January 2018

Abstract
In recent years, the revolution of cloud computing has taken the IT business to greater heights with the rapid sharing of vast web resources over the internet. Proficient task scheduling and balanced task distribution is still exists as a major challenging issue in cloud computing system due to dynamic heterogeneous nature of resources and tasks. It is a NP-hard problem where the scheduler needs to find the best optimal virtual machines with minimum makespan and proper resource utilization. The major part of this problem is to design an efficient intelligent searching pattern to schedule the tasks in best virtual available machines. In this paper we propose a meta heuristic algorithm called chaotic social spider algorithm inspired by social spider to tackle the problem of task scheduling in various heterogeneous virtual machines. This paper focus on minimizing overall makespan with effective load balancing by modelling the swarm intelligence of social spider with chaotic inertia weight based random selection. The proposed algorithm prevents the local convergence and explores the global intelligent searching in finding the best optimized virtual machine for the user task among the set of virtual machines with minimum makespan and balanced resource utilization. We have made the simulation and performance evaluation using cloudsim toolkit and compared the results with other swarm intelligent based algorithms such as GA, PSO and ABC. The evaluation results show that there is a major improvement in minimizing the makespan with balanced task distribution.

Keywords Cloud computing · Task scheduling · Load balancing · Virtual machine · Social spider

1 Introduction

Cloud computing is pretty bigger, and it's going bigger every day where high computation intensive tasks can be computed on demand basis [1]. Currently, most of the Information Technology (IT) industries have been migrated on to cloud computing based frameworks to serve their clients [2]. It delivers the services via three different layers namely, "Infrastructure", "Platform" and "Application" to support the instant on-demand needs of the consumers such as data storage, computing power, high bandwidth [3]. Corporate and individual can use these services based on how they offer via Virtualization. There are numerous trending cloud providers exists such as Amazon EC2, Google, HP and IBM [4] where the resources are virtualized as per the client requirements and delivered as 'pay-per usage' service level agreement (SLA) [5]. Cloud computing has some good features such as flexible resource repository, scalable and dynamic, on-demand services, pricing based on consumption with quality of service (QoS) [6]

Cloud based services are provisioned to consumer in different levels whereas resource management plays an important role, which deals with resource pooling, configuration, and task allocation handled by services provider. The major part of resource management in cloud computing system is a task scheduler where optimization algorithms can be used to allocate the user tasks in a provisioned logical resource called as 'virtual machines (VM)' [7,8]. There are many research works were presented to enhance the per-

✉ V. M. Arul Xavier
arulvmax@gmail.com

S. Annadurai
anna_prof@hindusthan.net

[1] Department of Computer Sciences Technology, Karunya University, Coimbatore, India

[2] Department of Computer Science and Engineering, Hindusthan Group of Technical Institutions, Coimbatore, India

formance of the task scheduling process which reveals good results, in which nature inspired meta heuristic algorithms are considered as most suitable solution for the task scheduling problem [9]. The task scheduling problem is a very crucial issue in cloud computing where intelligent searching and decisions were involved to find the best optimal VM for each user tasks [10]. There are several performance parameters which affect the overall scheduling process such as makespan, cost, delay, reliability, scalability, deadline and the resource utilization [9].

We consider the makespan as the major parameter which is the overall completion time of all the tasks that is being scheduled in the VMs. We have also considered few constraints such as cost, and resource utilization since both cloud providers and the consumers must be benefited with respect their requirements [7]. For example, for the cloud providers there is a need for resource utilization with considerable profit whereas the consumers the task must be completed with minimum expenses [11]. In this regard, we focuses on intelligent searching of best fit VM which minimizes the makespan, cost and at the same time it must utilize all the available VMs with balanced load.

This paper presents chaotic social spider algorithm (CSSA) for scheduling the user tasks in VMs of cloud with balanced task distribution. We have designed this scheduling model based on the foraging behavior of social spider species. Social spider is a special kind of species where the foraging behavior creates a social community based searching model in which each spider communicates with other spiders via vibration. This vibration notifies the food location and quality of the food. This information helps other spiders to change their location correlated with best optimum and continue the searching [12]. We have mimics this natural behaviour into cloud where spiders are modelled as software based searching agent (SA) randomly located in different VMs and notify the fitness and its location to other SAs, so that searching process can properly guided to reach the global best location. This process will be repeated for each user tasks until it reaches the best optimum schedule or maximum number of iteration. During this searching process we have designed a constraint handling phase where the load balancing issue were taken to improve the performance of the whole cloud computing system by considering both cloud providers and consumers.

We organized the paper as follows; Sect. 2 describes the various kinds of related works on task scheduling and load balancing techniques. Section 3 deals with our proposed methodology with detailed search procedure. Section 4 shows the simulation setup and results discussion on performance evaluation in comparison with existing algorithms. Finally we conclude by highlighting the features of our work and future scope in Sect. 5.

2 Related works

Scheduling the tasks in cloud computing deals with binding the user tasks to connected resources based on the scheduling decisions taken by the task scheduler according to various metrics. There are several algorithms were presented for scheduling user tasks in grid and cloud computing systems. Since, the task scheduling in cloud computing is a kind of NP-complete problem [13], the previous research works shows that the heuristics based algorithms were more suitable than the traditional algorithms.

Agarwal and Srivastava [3] has proposed a task scheduling algorithm inspired by genetic algorithm which focused on reducing response time and shows few positive results but it fails in dynamic environments where there is a need for global optimization. Vidhya et al. [14] has proposed a task scheduling algorithm using parallel particle swarm optimization (PPSO) to minimize the average execution time. Tawfeek et al. [5] proposed ant colony optimization (ACO) based task scheduling technique and showed few improvements over FCFS and Round Robin in terms of minimization makespan and degree of imbalance. Pradhan et al. [15] has introduced the task scheduling technique called Modified Round Robin Algorithm for Resource Allocation strategy in cloud computing to reduce the total turnaround time and waiting time and the evaluation results shown few improvements but gives poor throughput.

Hamad and Omara [16] proposed a genetic-based task scheduling approach (TS-GA) for allocation of application's task. The aim is to minimize makespan and cost of executing the tasks, and maximize the resource utilization. The results shows that the cost, completion time, and resource utilization of the algorithm is minimized compared to default GA and RR algorithms. Tsai et al. [6] has introduced a new strategy called improved differential evolution algorithm (IDEA) for task scheduling and resource allocation in cloud, where they combined the techniques of DEA and Taguchi method, to tackle the problem of exploration and exploitation and shown minor improvements in convergence ratio.

Keshanchi et al. [17] proposed an improved genetic algorithm (N-GA) to minimize makespan and execution time. This algorithm combined the features of evolutionary genetic algorithm and behavioral modeling approach based on model checking. The results shows that, this proposed algorithm reduces the makespan and execution time when there is small size tasks but it likely to be fails when there is a need for global optimization. Abdi et al. [18] proposed a modified PSO algorithm for task scheduling to minimize makespan where jobs are assigned based on one-to-one mapping and the shortest job was assigned to fastest processor then merged into standard PSO to improve behaviour of PSO. PSO and genetic algorithm are compared with proposed algorithm and it was seen that proposed algorithm has better makespan.

Abdullahi and Ngadi [7] presents a discrete symbiotic organism search algorithm (DSOS) for optimal scheduling the user tasks in cloud environment. It uses the mutualism, commensalism and parasitic relationship to acheive the objective function. Makespan, degree of imbalance and response time were found to be better than self adaptive particle swarm optimization (SAPSO). Jeyakrishnan and Sengottuvelan [19], has presented A hybrid strategy based bacterial swarm optimization (BSO) for resource allocation which address the problem of local convergence and introduce a solution to avoid the local converge while attain the global searching the whole search space and the results good improvements in global optimization.

Babu and Venkata Krishna [20] has introduced the Honey Bee foraging behavior based approach to balance the load on VMs of cloud environment. Their results show that makespan got reduced and the overall throughput was improved. Awada et al. [21] has presented enhanced particle swarm optimization based task scheduling where load balancing mutation process was introduced. The results were shown few improvements in minimizing the makespan, trip time and also they have address the reliability issues.

Mondal et al. [22] presented a strategy for balanced the load distribution in cloud using stochastic hill climbing algorithm to guide the balanced task distribution among the virtual machines and gives good results in local search but fails in global optimization. Zhan et al. [23] presented load balance aware genetic algorithm for task scheduling in cloud with time load balance (TLB) model to reduce the makespan. Guo-Ning et al. [24] has proposed a novel strategy called genetic simulated annealing algorithm. They have utilized simulated annealing process after the selection, crossover and mutation, to achieve better performance in local search, but fails when the user tasks increases in size.

Meta-heuristic is a kind of swarm intelligence based searching technique has been widely accepted methodology for many optimization problems. Social spider algorithm (SSA) is one of the most recent methodology inspired by the foraging behavior of social spider was proposed by Yu and Li [25] for global optimization. Generally social spider is a kind of bio species whose individuals form relatively long lasting social spider web. Each spider has a unique location in the social spider web from which its starts searching food sources. This foraging leads by random movement among the spider web for the food sources. Each spider makes a random walk towards the food sources [12].

The decision for the next movement will be predicted through the intelligent behavior of spider called *vibration* [12]. Vibration is a very prominent behavior of spider which makes notification to the other spiders about the food sources. Each vibration holds two parameters the location and intensity. Quality of the food sources will be predicted based the vibrations generated by spider. The intensity of the vibration is used by the spider to identify the quality and location of the food sources. Based on the intensity level spider can identify the best spider location with good quality of food. Spiders takes the next move based the best vibration location during foraging [12]. This process is repeated until spider finds the best feasible food locations in the spider web.

3 Proposed methodology

3.1 The system model

The cloud computing provides on-demand high performance computing services over the internet as self-service access. The infrastructure basically provides the low level services such as cloud servers, data storage, and middleware as pay-per-use basis [3]. The internal working principles of cloud model is depicted as shown in Fig. 1, which consists of cloud users, online cloud web portal, resource scheduler, and resources [26]. The resources include one or more host machines with one or more virtual machines (VMs). Each VM shares the computing power (CPU), Memory, Storage Places from the hosted machines. Every user requests is mapped to proper VM based on the resource requirement such as CPU, Memory, Storage and Network Bandwidth [26].

The working flow of this model is as follows

- Every user interacts with the cloud infrastructure and tries to access the resources through cloud web portal which ensures the authentication and authorization.

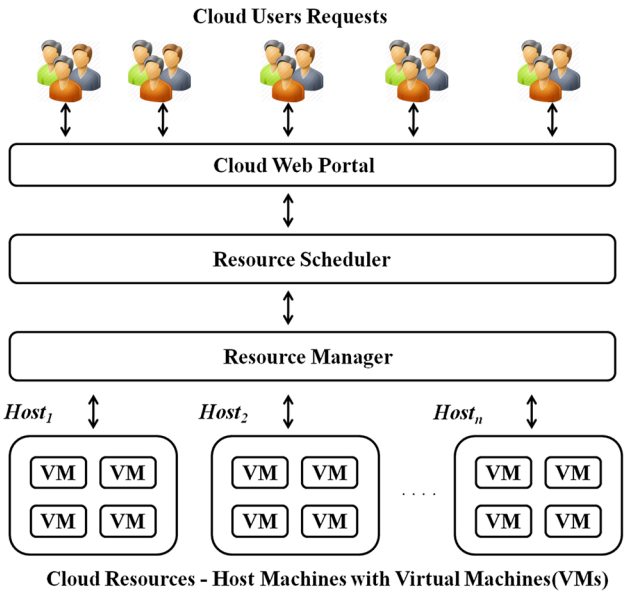

Fig. 1 Cloud computing model

- The Resource Scheduler is major component, which act as a middleware between clients and resource manager to process the client request. It further takes all the information regarding the Task requirements posted by the clients and proceeds further for resource allocation using the inbuilt scheduling algorithm.
- Finally, the resource manager plays an important role which acts as a resource information system and provides services to resource scheduler in order to allocate the user tasks in best possible VM. In order to balance the load of each host machines, resource manager has an in built resource monitor which keeps tracks of the entire loads and tackles the unbalanced condition.

3.2 The scheduling problem

In cloud computing environment, tasks are dynamically submitted by various users. It is the responsibility of the cloud scheduler to find the optimal resource i.e. cloud virtual machines (VMs) for the submitted tasks. Also, it is important to manage the load of each resource, since scheduling the tasks in overloaded resources may degrade the performance of the overall cloud computing providers [27].

In our work, the task scheduling problem is considered as a major issue, since it affects the overall performance of the cloud computing provider's in terms of quality of service (QoS). There are many factors which represent the QoS, such as execution time, transmission time, latency, resource utilization and makespan [28]. At the same time, clients needs better performance with low cost. Hence, we consider two QoS parameters named as makespan and cost as a major concern which must be optimized with proper load balancing in order to provide better performance for the cloud providers and clients.

To design the problem, we consider N number of user tasks i.e. $T_n = \{T_1, T_2, T_3, \ldots, T_n\}$ submitted to the cloud, which needs to be allocated to M number of Virtual Machines (VMs) i.e. $VM_m = \{VM_1, VM_2, VM_3, \ldots, VM_m\}$ with minimum makespan, cost and maximum resource utilization. Each VM is configured with different resource parameters such as CPU capability in millions instruction per second (MIPS), storage and network bandwidth as per the availability of hosted physical machines. Tasks are generated by users with different characteristics such as task_length, deadline, and cost and so on.

To formulate the objective function of our proposed algorithm, we consider C_{ij} is the completion time of a Task T_i, $i \in \{1, 2, 3, ..N\}$ on resource VM_j, $j \in \{1, 2, 3, ..M\}$ which can be calculated initially for all the tasks using Eq. (1) by considering the expected time to complete (ETC_{ij}) of ith task on VM_j and the waiting time W_i ith task for VM_j. Hence, the completion time C_{ij} of each task in VM is calculated using Eq. (1).

$$C_{ij} = ETC_{ij} + W_i \tag{1}$$

Moreover the cost $E_{cost_{i,j}}$ for executing the task ith on VM_j is calculated by considering the ETC_{ij} and the cost per VM for unit time which denoted as VM_{cost} which is defined in Eq. 2.

$$TE_{cost} = \sum ETC_{ij} \times VM_{cost} \tag{2}$$

The *makespan* of all the submitted tasks is the total execution time of entire tasks which can be expressed as the maximum completion time of all the tasks scheduled in all *VM*s is calculated as Eq. (3).

$$\mathrm{C}_{max} = \sum \mathrm{C}_{ij} \tag{3}$$

The aim of our proposed methodology is to make the cloud scheduler to provide optimal resource allocation for all the user tasks with minimum makespan and minimum cost as per the requirements of cloud providers and clients respectively. With the above considerations the objective function of our proposed methodology is defined as Eq. 4.

$$fitness = min\,(C_{max}) + min\,(TE_{cost}) \tag{4}$$

Also, we consider the load balancing problem in this work, in this regard the following mathematical model is considered. Load balancing is also another important issue needs to be addressed in order to provide optimal allocation. During the schedule, the VMs may be quickly overloaded as the user tasks are allocated more and more.

For balanced task distribution, the load on all the VMs needs to be monitored continuously; if it is overloaded the concern VMs needs to be avoided in order to maximize the resource utilization with the balanced VMs. To take better load balancing decision, we calculated load factor (LF) σ, which is the standard deviation of all load as per the following Eq. (5).

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(ET_i - ET)^2} \tag{5}$$

where ET_i, is the execution time of ith VM, which can be calculated as given in Eq. 6, where as ET is a total execution time of all tasks, which can be defined by Eq. 7

$$ET_i = \frac{L_{vm_i}}{CP_i} \tag{6}$$

$$ET = \sum_{i=0}^{m} ET_i \tag{7}$$

where L_{vm_i} is the load of VM_i and CP_i is the capacity of VM_i

$$L_{vm_i} = \frac{Total\ number\ of\ tasks\ on\ VM_i}{Total\ number\ of\ tasks\ executed\ in\ VM_i} \quad (8)$$

$$CP_i = CE_{N_i} * CPU_{mips_i} * VM_{bandwidth_i} * RAM_{size_i} \quad (9)$$

The capacity if ith VM is calculated via Eq. (9), where CE_{N_i} is the total number of computing elements (CE) in VM, CPU_{mips_i} is the CPU usage in terms of MIPS, $VM_{bandwidth_i}$ is the required communication network bandwidth of VM, and the required memory usage of a VM is expressed as RAM_{size_i}.

3.3 Chaotic social spider algorithm (CSSA) for load balance aware task scheduling

We propose an efficient approach inspired by the social spider's pray foraging process to schedule the user tasks with balanced load called CSSA. Our approach proceeds with initial system model based on the 3.2. We have considered the spider web as cloud computing environments, where each resources i.e. VM is represented as a feasible solution (food source) in the spider web. Each spider in the spider web is considered as search agent (SA) which moves freely in the solution space to find the best VM for the user tasks.

In [25] Yu and Li proposed SSA with fixed parameter scheme to solve global optimization problems. Since, cloud computing is heterogeneous and dynamic environment, we have modified the SSA in order to comply with the environment. Each SA holds memory which stores the location of the feasible solution and the fitness value of VM in a form of Broadcast Message (BM). Moreover, each SA is configured as dynamic in nature and having capability to broadcast and move to other VM location randomly anytime. The algorithm proceeds with four phases as follows.

3.3.1 Initialization phase

During this phase, the initial solution space is created with population of VMs $\{popVMs\}$ were generated. During the schedule, each SA propagates its location and the fitness of VM via broadcast messages (BMs). In our model, we configured SA as a computing agent with broadcasting facility to notify the location and the capability of VMs. Also, we use parameters such as target BM quality(BM_q^{tar}), location (L_{vm}), and the initial broadcast message quality (BM_q) are initialized to zero and initial fitness is calculated and stored for each VM location using Eq. 4.

Every BM_q of each VM comprises of its location and its quality based on the fitness of its location. For example, Let $L_i(t)$ be the location of a ith VM at time t in the solution space, the BM_q generated in every iteration when the SA to a new location. In this scenario, the general function to *represent* the BM_q of each SA can be denoted by $BM_q(t)$ and the same is calculated using Eq. (4) which is consider as the objective function for the *fitness* $f\ (vm)$ of a each SA locations.Since we are considering the spider web as a cloud web framework, the best feasible solution in the solution space is the VM which has the suitable processing requirements of user tasks. Hence, $BM_q(t)$ is computed using Eq. (10).

$$BM_q(t) = \log\left(\frac{1}{f\ (vm) - C} + 1\right) \quad (10)$$

where c is a user controlled parameter to find out the feasible fitness value, in our model we kept as very small $c \in \{0, 1\}$ in order to achieve the minimized makespan. The value of 'c' is taken as a minimization constant parameter to fine tune the performance of the scheduling objective.

In social spider foraging, the vibration is a kind of energy, so the intensity of the vibration may be attenuated over distance; But in cloud computing environments, due to high bandwidth of communication the quality of BM generated by each SA will be received by other SA without any modification.

3.3.2 Iteration phase

Once the initialization phase is completed, the algorithm further proceeds to iteration phase which comprise of subphases; fitness evaluation, broadcast message propagation, mask changing, random walk, and constraints handling.

In iteration phase, for each user tasks, $T_i, i \in \{1, 2, 3, \ldots, N\}$, the algorithm proceeds to find the best suitable $VM_{ij}, j \in \{1, 2, 3, \ldots, M\}$ in the solution space. In, iteration phase we have started with computation of, where the $BM_q(t)$ of each SA in the $\{pop\}$ is computed and the $fitness$ is evaluated using Eq. (4). Then the broadcast messages from each SA location is generated and propagated over the solution space using Eq. (10).

After receiving the broadcast messages from all the SA's, each SA proceeds to find the best quality BM_q^{best} among all the received BM and get stored. Then BM_q^{best} is compared with the current value of BM_q^{tar}, if it is greater than BM_q^{tar} it will be assigned as the new BM_q^{tar}.S_g is introduced to check whether the BM_q^{tar} is updated or not, which is initially set as

zero and it may take 0 or 1 based the updation in BM_q^{tar}. S_g is set as one if the BM_q^{tar} is not updated and set as 0 if BM_q^{tar} is changed.

3.3.3 Random VM location prediction using chaotic inertia weight

The value of S_g helps to guide the searching process that is correlated towards BM_q^{tar}, during the iteration each SA can change its location inorder to reach BM_q^{tar}, in a probability of $1 - P^{S_g}$ where P is an user controlled parameter from $\in \{0, 1\}$. Hence, each SA gets the 'New following' VM location(L_{vm}^{nf}) based on Eq. 11.

$$L_{vm}^{nf} = \begin{cases} L_{BM_q^{tar}}, S_g = 1 \\ L_R, S_g = 0 \end{cases} \tag{11}$$

where R is the random VM location which is generated from the set of VM population as follows. Here, we introduce a new random VM selection factor based on chaotic inertia weight sequences, which have been used in most of the optimization process of swarm intelligent algorithms [29,30]. In this work, we have utilized the chaotic inertia weight to formulate the random VM selection process in order reach the better fitness VM location as given below.

$$X_{n+1} = \gamma X_n (1 - X_n) \tag{12}$$

where γ the user is controlled parameter, which is used as 4 in our work and X_n is a random number generated for every iteration. During this process SA may falls into the previously visited VM location. In order to avoid that, we introduce a new weight factor 'β', called as chaotic inertia weight to guide the random VM selection process. Hence, the new random VM selection process is formulated as given in Eq. 13.

$$L_{R_{vm}}(t+1) = L_{vm}(t) + (L_{vm}(t) - L_{vm}(t-1)) \times \beta + \left(L_{vm}^{nf} - L_{vm}(t)\right) \cdot \delta \tag{13}$$

where the value for δ is the random number generated from the range (0, 1). In our work, the value 'β' is considered as weight factor and defined as given in Eq. 14.

$$\beta(t) = \left(w^{intial} - w^{final}\right) \times \left(\frac{Iter^{max} - Iter}{Iter^{max}}\right) + w^{final} \times X_{n+1} \tag{14}$$

where w^{intial} and w^{final} is the initial and final intertia weights which are user controlled values. Similarly the $Iter^{max}$ is a maximum number of iteration, where as $Iter$ is a current iteration count. Once $L_{R_{vm}}(t+1)$ is generated for each SA location the algorithm proceeds to handle the load balancing constraint in cloud.

3.3.4 Load balancing constraint handling

After the random VM process, every SA will be getting a new VM location $L_{R_{vm}}(t+1)$, then proceeds to address the constraints employed in the cloud. Here we consider the load balancing as the major constraint, since while allocating VM to user tasks, sometimes some VMs may falls into overloaded so the overall performance may be affected. Hence we tackle this issue by introducing Load Factor (LF)σ, for each provisioned VMs as a boundary condition which is calculated based on Eq. 5, then the new VM location is generated as given in Eq. 14.

$$L_{vm}(t+1) = \begin{cases} (\bar{\sigma} - L_{vm}(t)) \times R & if LF\left(L_{R_{vm}}(t+1)\right) \geq \bar{\sigma} \\ \left(L_{vm}(t) - \underline{\sigma}\right) \times R & if LF\left(L_{R_{vm}}(t+1)\right) \leq \underline{\sigma} \end{cases} \tag{15}$$

Here we assume, $\bar{\sigma} and \underline{\sigma}$ as over-loaded VM and under-loaded VM respectively, and defined as if the value of σ low means $\bar{\sigma}$ and the value is high means $\underline{\sigma}$. R is a random number generated as a user controlled parameter from (0, 1).

After the constraints were addressed, each SA gets the new VM location $L_{vm}(t+1)$ as given in Eq. 11 and migrated to the specified location and the algorithm further proceeds to the next iteration and repeat above mentioned steps until the stopping criteria are met. We consider the stopping criteria as maximum iteration reached with no improvement in best BM_q^{best}. Then the algorithm outputs the best VM for each task in the task list. We have also designed the above steps mentioned above (a, b, c, d) as algorithm as follows.

Algorithm: Chaotic Social Spider Algorithm (CSSA) Search Procedure

Input: User Tasks
Output: Optimized VM allocation of all user tasks submitted
Create VMList $\leftarrow [VM_1, VM_2, VM_3, \ldots VM_n]$
Create TASKList $\leftarrow [T_1, T_2, T_3, \ldots T_n]$
Set unique L_{id}(Location id) to each VMList items.
Generate SA with memory and configure the BM.
Randomly set the VM's L_{id} to each SA
Initialize the maximum number of iteration ($Iter^{max}$)
Initialize BM_q^{tar} and S_g
Set all the user controlled parameters
for each Task(t) **in** TASKList **do**
  for $i \leftarrow 0$ $\boldsymbol{to}$ $Iter^{max}$
    for each SA
      compute the fitness of each VM in each SA
      store the fitness in BM of SA
      propagate the BM
    end for
    for each SA
      Store the received BM in a BMList
      Arrange the items in BMList in fitness(BM_q) ascending order
      Select the best BM_q
      Compute $BM_q^{tar} = (BM_q^{tar} < BM_q)\ ?\ BM_q : BM_q^{tar}$
      Set $S_g \leftarrow 0$ if BM_q^{tar} is changed, otherwise set $S_g \leftarrow 1$
      if $S_g == 0$ **then**
        Choose the L_{vm}^{NF} as BM_q^{tar}
      else
        compute chaotic intertia weight($\beta(t)$) based on *Eqn.14*
        select a random VM based on *Eqn.13*
      compute the Load Factor(LF) σ using Eqn.5
      apply LF in *Eqn.15* and find the best VM
      Bind the task to best VM.
    end for
  end for
end for
Output the VM allocation schedule.

4 Simulation setup and results

To evaluate the performance of our proposed technique, we have used CloudSim framework through which we have modelled the real cloud infrastructure and simulate task scheduling process. We have evaluated the performance via several parameters such as makespan, cost, execution time, degree of imbalance, waiting time and compared with existing techniques such as Genetic Algorithm (GA), Artificial Colony Optimization (ACO), Particle Swarm Optimization (PSO), Hybrid Fuzzy K-Means++ with Clonal Selection (HFKCS) [31]. Moreover, we have used tasks distribution in a combination of small, medium, and large with specified number of tasks such as 200, 400, 600, 800, 1000. Our experiment setup parameters are shown in Table 1.

Table 1 Simulation parameters

Number virtual machines	100
Number of tasks	1000
Bandwidth	600–1200 kbps
Cost per VM	1$
MIPS	1000–2000 ms
RAM	4–8 GB
Number of physical machines	1–5
γ	4
δ	0.4
C	0.2
R	0.3
P	0.8
$Iter^{max}$	500
w^{intial}	0.9
w^{final}	0.4

Figure 2 shows the comparison of generated by the proposed algorithm with other popular algorithms such as GA, PSO, ABC, HFKCS and it clearly shows that our algorithm outperforms the other algorithms. Moreover, the above Fig. 2 shows the variation in makespan for different tasks group and an average of 14.8% improvements in our approach compared with other algorithms.

Computational cost is a overall cost involved in completion of user cost, which affects the overall performance. In Fig. 3 shows the comparative analysis of operational cost involved in the proposed algorithm with other algorithms. The projected results reveal that, when the no.of tasks are 200 the CSSA have reduced the overall cost to in an average of 33.66% than GA, PSO, ABC, HFKCS. As we increased the no.of tasks, it can be observed that there is a slight improvement in minimization of cost, when compared to ABC and HFKCS and there are huge improvements between CSSA with GA and PSO. When the no.of tasks, reaches to 1000, CSSA shows an average of 13.5% improvements than other algorithms.

Figure 4 shows the comparison of resource utilization in CSSA with other algorithms, which is the overall usage of all resources for the user tasks; The graph experimental that, our proposed algorithm gives slight improvements in resource utilization when compared to ABC and HFKCS. Also it is observed that there is a huge improvement when it is compared with GA and PSO. Hence, CSSA is more suitable to revenue growth of cloud service providers.

Figure 5 shows that the average response time of CSSA for the user tasks and it is compared with other algorithms. Response time is a kind of metric to evaluate the time taken to give the response to the user. It is indicated that, there is much improvements in minimizing the response times as

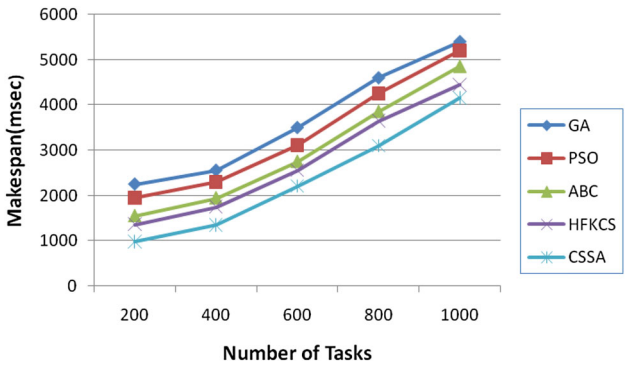

Fig. 2 Makespan

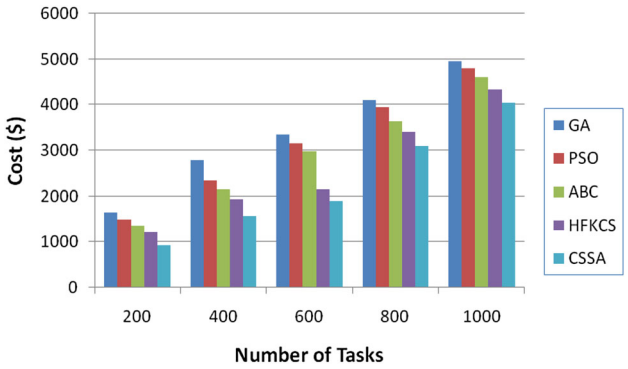

Fig. 3 Computational cost

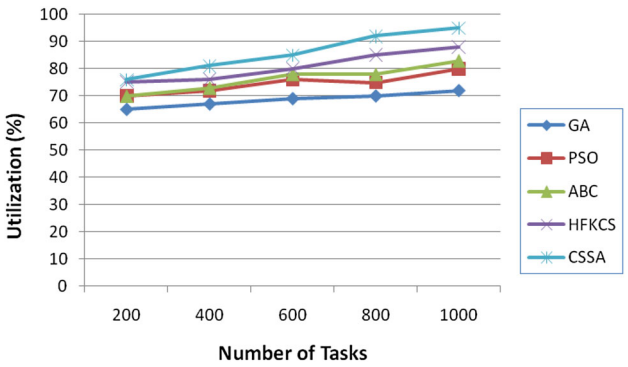

Fig. 4 Resource utilization

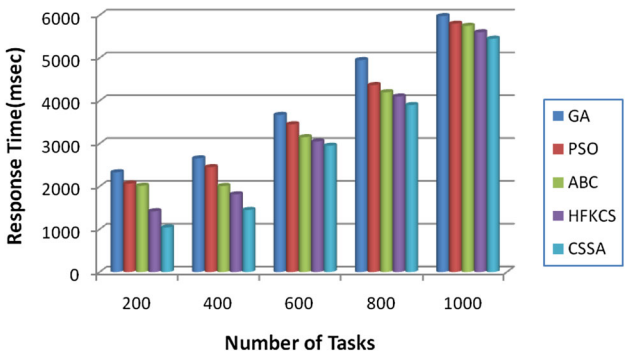

Fig. 5 Response time

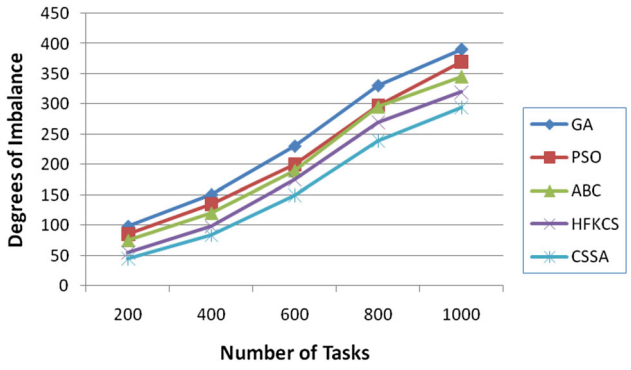

Fig. 6 Degree of imbalance

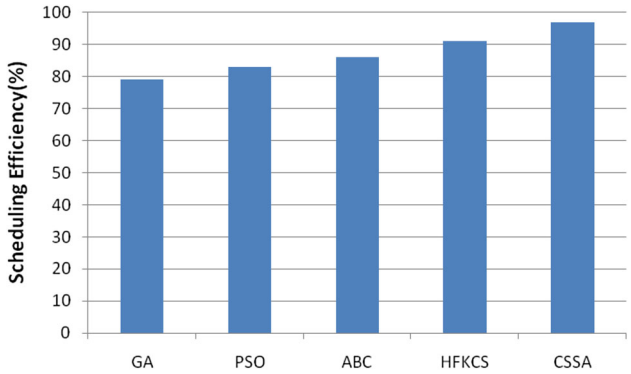

Fig. 7 Scheduling efficiency

we increased the number of tasks. Particularly there is huge improvements produced by CSSA when compared with GA and PSO and also it indicates slight improvements with ABC and HFKCS as well.

In Fig. 6 we have analyzed the degrees of imbalance, since during the schedule VM likely gets overloaded when the user tasks are scheduled in cloud resources. A degree of imbalance is a kind of metric to evaluate the load of virtual machines. In graph we can observe that, our algorithm produces very minimal degree imbalance when compared to other algorithms. As the numbers of tasks are increased, CSSA gives the reduced degree of imbalance thereby tasks are uniformly distributed without affecting the performance of the resource.

Finally, the above Fig. 7, it can be observed that more improvements in overall efficiency in CSSA than GA, PSO, ABC, and HFKCS. It clearly shows that, the CSSA produces 97% of efficiency whereas other algorithms produce 79, 83, 86, and 91% respectively. Table 2 shows the overall performance comparisons of our proposed approach with existing algorithms.

With the evaluation of results, it can be observed that CSSA produces more benefits in terms of makespan optimization and other associated parameters. Moreover, it clearly addresses the load balancing issues with minimal degree of imbalance than other algorithms. Moreover, this method provides better cost optimization, which will be benefited for the consumer who wants to use the cloud services.

5 Conclusions

In this paper, we have proposed CSSA for load balance aware task scheduling in cloud computing environments. This algorithm was inspired by the foraging behaviour of social spider species and its relationships. We have modelled the process of foraging suitable to cloud computing environments with modifications to find best optimized virtual machine for the user tasks. The proposed algorithm was simulated via cloudsim and the various performance parameters were evaluated. The main objective was minimizing the makespan of the cloud scheduling process thereby improving the throughput of the cloud system. In this regard, we have measured various parameters such as makespan, computational cost, response time, average response time, degree of imbalance among various VMs. Our experiments result shows that, the CSSA produce the makespan minimization to 14.8% when compared to GA, PSO, ABC and HFKCS for 100–1000 numbers of tasks. Moreover, GA and PSO more likely to local convergence, but the CSSA more suitable for global search and it prevents local convergence. In our we have also addressed the load balancing while mapping the tasks to VM, the degree of imbalance results shows that CSSA outperforms the other algorithms. Also CSSA more suitable for consumer since it reduces the overall cost of computation. In overall, the scheduling efficiency was improved with CSSA when compared to other algorithms.

Table 2 Performance improvements of CSSA

	Makespan (%)	Cost (%)	Utilization (%)	Response time (%)	Imbalance (%)	Efficiency (%)
GA	2.01	1.02	10.09	12.04	2.04	79
PSO	6.05	9.03	12.22	14.05	4.05	83
ABC	8.88	12.88	17.33	19.04	4.66	86
HFKCS	11.22	15.22	18.01	22.04	5.01	91
CSSA	14.05	18.55	19.09	24.55	5.99	97

In future work, other performance parameters such as security, reliability may be included so that security threats and trust nodes can be identified. Moreover, we further extending this work so as to compatible with independent tasks.

References

1. Aceto, G., Botta, A., de Donato, W., Pescapè, A.: Cloud monitoring: a survey. Int. J. Comput. Netw. **57**(9), 2093–2115 (2013)
2. Buyya, R., Yeoa, C.N., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision hype and reality for delivering computing as the 5th utility. Fut. Gener. Comput. Syst. **25**, 599–616 (2009)
3. Agarwal, M., Srivastava, G.M.S.: An efficient approach to genetic algorithm for task scheduling in cloud computing environment. Int. J. Inf. Technol. Comput. Sci. **10**, 74–79 (2012)
4. Bölöni, L., Turgut, D.: Value of information based scheduling of cloud computing resources. Fut. Gener. Comput. Syst. **71**, 212–220 (2017)
5. Tawfeek, M.A., El-Sisi, A., Keshk, A.E., Torkey, F.A.: Cloud task scheduling based on ant colony optimization. In: 2013 8th International Conference on Computer Engineering & Systems (ICCES), pp. 64–69
6. Tsai, J.-T., Fang, J.-C., Chou, J.-H.: Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. Comput. Oper. Res. **40**(12), 3045–3055 (2013)
7. Abdullahi, M., Ngadi, M.A.: Symbiotic organism search optimization based task scheduling in cloud computing environment. Fut. Gener. Comput. Syst. **56**, 640–650 (2016)
8. Akbar, M.F., Munir, E.U., Rafique, M.M., Malik, Z., Khan, S.U., Yang, L.T.: List-based task scheduling for cloud computing. In: 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 652–659 (2016)
9. Kalra, M., Singh, S.: A review of metaheuristic scheduling techniques in cloud computing. Egypt. Inf. J. **16**(3), 275–295 (2015)
10. Pacini, E., Mateos, C., Garino, C.G.: Balancing throughput and response time in online scientific clouds via ant colony optimization. Int. J. Adv. Eng. Softw. **84**, 31–47 (2015)
11. Karthikeyan, P., Chandrasekaran, M.: Dynamic programming inspired virtual machine instances allocation in cloud computing. J. Comput. Theor. Nanosci. **14**, 551–560 (2017)
12. Uetz, G.W.: Foraging strategies of spiders. Trends Ecol. Evol. **7**(5), 155–159 (1992)
13. Kumari, V., Kalra, M., Singh, S.: Independent task scheduling in cloud environment using Big Bang-Big Crunch approach. In: 2nd International Conference on Recent Advances in Engineering & Computational Sciences (RAECS) (2015)
14. Vidhya, M., Sadhasivam, N.: Parallel particle swarm optimization for task scheduling in cloud computing. Int. J. Innov. Res. Sci. Eng. Technol. **4**(6), 136–140 (2015)
15. Pradhan, P., Behera, P.K., Ray, B.N.B.: Modified round robin algorithm for resource allocation in cloud computing. In: International Conference on Computational Modeling and Security (CMS 2016), Procedia Computer Science, vol. 85, pp. 878–890 (2016)
16. Hamad, S.A., Omara, F.A.: Genetic-based task scheduling algorithm in cloud computing environment. Int. J. Adv. Comput. Sci. Appl. **7**(4), 550–556 (2016)
17. Keshanchi, B., Souri, A., Navimipour, N.J.: An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. J. Syst. Softw. **124**, 1–21 (2017)
18. Abdi, S., Motamedi, S.A., Sharifian, S.: Task scheduling using modified PSO algorithm in cloud computing environment. In: International Conference on Machine Learning, Electrical and Mechanical Engineering (ICMLEME'2014) Jan. 8–9, Dubai (UAE) (2014)
19. Jeyakrishnan, V., Sengottuvelan, P.: A hybrid strategy for resource allocation and load balancing in virtualized data centers using BSO algorithms. Wirel. Pers. Commun. **94**, 2363–2375 (2017)
20. Dhinesh Babua, L.D., Venkata Krishna, P.: Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. **13**, 2292–2303 (2013)
21. Awada, A.I., El-Hefnawya, N.A., Abdel kaderb, H.M.: Enhanced particle swarm optimization for task scheduling in cloud computing environments. Procedia Comput. Sci. **65**, 920–929 (2015)
22. Mondal, B., Dasgupta, K., Dutta, P.: Load balancing in cloud computing using stochastic hill climbing-a soft computing approach. Procedia Technol. **4**, 783–789 (2012)
23. Zhan, Z.-H., Zhang, G.-Y., Gong, Y.-J., Zhang, J.: Load balance aware genetic algorithm for task scheduling in cloud computing. In: Simulated Evolution and Learning 10th International Conference, pp. 15–18 (2014)
24. Guo-Ning, G., Ting-Lei, H.: Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. In: Proceedings of International Conference on Intelligent Computing and Integrated Systems, pp. 60–63 (2010)
25. Yu, J.Q., Li, V.O.: A social spider algorithm for global optimization. Int. J. Appl. Soft Comput. **30**, 614–627 (2015)
26. Martinez, G., Zeadally, S., Chao, H.-C.: Editorial: cloud computing service and architecture models. Inf. Sci. **258**(10), 353–354 (2014)
27. Ghom, E.J., Rahmani, A.M., Qader, N.N.: Load-balancing algorithms in cloud computing: a survey. J. Netw. Comput. Appl. **88**, 50–71 (2017)
28. Abdelmaboud, A., Jawawi, D.N., Ghani, I., Elsafi, A., Kitchenham, B.: Quality of service approaches in cloud computing: a systematic mapping study. J. Syst. Softw. **101**, 159–179 (2015)
29. Park, J.B., Jeong, Y.W., Shin, J.R., Lee, K.Y.: An improved particle swarm optimization for nonconvex economic: dispatch problems. IEEE Trans. Power Syst. **25**(1), 156–166 (2010)
30. Shengsong, L., Min, W., Zhijian, H.: Hybrid algorithm of chaos optimization and SLP for optimal power flow problems with multimodal characteristic. Proceedings of the institution of Electrical Engineers, Generation, Transmission and Distribution **150**(5), 543–547 (2003)
31. Arul Xavier, V.M., Annadurai, S.: HFKCS: hybrid fuzzy K-means++ with clonal selection algorithm for task scheduling and load balancing in cloud computing. Int. J. Appl. Eng. Res. **10**(20), 20140–20156 (2015)

V. M. Arul Xavier has completed B.Tech. degree in Information Technology from Velammal Engineering College, Madras University, Chennai and received the M.E. Degree in Computer Science and Engineering from Noorul Islam College of Engineering, Anna University, Chennai. He is currently working as Assistant Professor in Department Computer Science and Engineering at Karunya University, Coimbatore. Presently, he is doing Ph.D. under Anna University, Chennai. His research interest includes Grid Computing, Cloud Computing, Wireless Sensor Networks, and Mobile Computing.

S. Annadurai received B.E. degree in Electronics and Communication Engineering and M.E. degree in Power System from the University of Madras, Chennai and M.E. degree in Computer Science and Engineering from Bharathiyar University, Coimbatore. He completed his Ph.D. degree from Anna University, Chennai. He is currently working as a Professor and Advisor at Hindusthan Group of Technical Institutions, Coimbatore. He has contributed more than hundred research papers in different areas of computer science and engineering such as Automatic Speech Recognition, Neural and Fuzzy Neural Networks, Image Processing and Computer Networks. He is a life member of Professional bodies such as CSI and ISTE. He was a chairperson and organizing committee of many national and international conferences held in India and abroad. His research interests include Image Processing, Computer Networks, and Fuzzy Neural Networks, Cloud Computing and Automatic Speech Recognition.