RESEARCH ARTICLE

Energy Efficient Load Balancing Aware Task Scheduling in Cloud Computing using Multi-Objective Chaotic Darwinian Chicken Swarm Optimization

G. Kiruthiga
PG and Research Department of Computer Science, PresidencyCollege, Chennai, Tamil Nadu, India.
g_kiruthiga@yahoo.co.in

S. Mary Vennila
PG and Research Department of Computer Science, PresidencyCollege, Chennai, Tamil Nadu, India.
maryvennila13@gmail.com

Published online: 29 June 2020

Abstract – Scheduling of tasks in a cloud environment has larger influence on time and energy depletion. Different heuristic models were developed to solve the NP-hard task scheduling problem based on time. However, ideal task scheduling algorithms must also maximize energy efficiency with good load balancing and ensure better Quality-of-Service (QoS). An innovative multi-objective Chaotic Darwinian Chicken Swarm Optimization (CDCSO) system is suggested in this article to provide energy efficient QoS and load balancing aware task scheduling. The multi-objective CDCSO algorithm incorporates the chaotic and Darwinian Theory to the standard Chicken Swarm Optimization to increase its global exploration and maximize the convergence rate. This performance enhanced CDCSO algorithm models the cloud task scheduling problem as NP-hard and utilizes the optimization principles to solve them based on multiple objective parameters. The multi-objective fitness function used in CDCSO is modelled based on the objective parameters namely energy, cost, task completion time, response time, throughput and load balancing index. Based on this multi-objective function, the CDCSO effectively allocates the tasks to the suitable energy efficient, cost and time minimized Virtual machines (VMs) which are also optimally load balanced. CloudSim simulations were conducted and the obtained results illustrated that the proposed multi-objective CDCSO has provided better task scheduling with minimized energy, cost, time and optimal load balancing.

Index Terms – Cloud Task Scheduling, Multi-Objective Problem, Chaotic Darwinian Chicken Swarm Optimization, Darwinian Theory, Energy Efficiency, Load Balancing Index, Quality-of-Service.

1. INTRODUCTION

Cloud computing has attained enormous interest from the research as well as other communities due to its reliability, scalability, security, accessibility and effective cost-reduction solutions [1]. The foremost property of the cloud computing is the accessibility of resources from any geographical location through the Internet. It enables numerous business entities and science organizations to implement the commercial cloud systems for effective sharing and commercial computing profits. For providing these services to the cloud users, the providers employ multiple servers in the cloud data centers [2]. This process increases the bulk energy and also causes challenges in balancing the load and resources in the cloud hosts. These challenges have created the need for designing energy efficient and load balancing aware strategies in cloud to schedule the tasks [3].

Cloud task scheduling algorithms were primarily focused on improving the energy efficiency with reduction of cost and makespan. It also considers multiple objective factors to improve the cloud resource utilization [4]. However, it is generally hindered by the fact that the higher resource utilization increases the energy depletion and hence the cloud data center proficiency decreases. This problem is more common in the heterogeneous cloud where the resource capability varies for each cloud host [5].

This scenario is mainly due to the lack of effective balancing of the cloud load and the corresponding resources. For resolving these issues, the tasks in cloud computing applications must be scheduled such that the load and resources are balanced to ensure the energy efficiency. Hence the task scheduling algorithms must be designed such that it satisfies multiple objective parameters while scheduling the tasks in cloud hosts and ensure proper utilization of resources and efficient energy consumption. More importantly, the scheduling mechanism must analyses the load condition of specified virtual machines before proceeding to assign the tasks.

Many optimization algorithms and machine learning algorithms have been employed predominantly to schedule tasks based on multiple objectives [6]. As the focal point of this article is developing multi-objective task scheduling algorithm that ensures energy efficiency and effective resource utilization, the optimization algorithm must be selected in such a way it is competent in achieving the solution without the local convergence problem. Hybrid optimization algorithms have proven to be much effective in achieving this ideal performance. Previously, the Chaotic Quantum Whale Optimization Algorithm (CQWOA) [7] and Chaotic Quantum-Behaved Chicken Swarm Optimization (CQCSO) [8] were introduced to develop the efficient scheduling models in cloud. Nonetheless, the problem of premature convergence arises when the network size and the multi-objectives increases due to limited global solution search process. Also, the VM load balancing is not prioritized in these models and hence the resource wastage is increased. This paper has focused on resolving these problems by developing hybrid framework for multi-objective energy efficient and load balancing parameter based task scheduling model. The key strategies are: i) Considering energy and load balancing index as objectives along with the cost, throughput and time parameters to formulate multi-objective scheduling problem; ii) The development of novel enriched Chaotic Darwinian Chicken Swarm Optimization (CDCSO) algorithm by improving the standard CSO using chaos theory and Darwinian theory to improve the task scheduling behaviour. The article is written in the following order: Discussion on related studies in Section 2. Formulation and explanation about the multi-objective task scheduling problem and the proposed CDCSO based task scheduling model in section 3. Simulation and evaluation results are provided in section 4 and the conclusion in section 5.

2. RELATED WORKS

Energy efficiency and the knowledge of load balancing are considered as vital criteria for task scheduling in this research study. Most of the recent studies on task scheduling strategies have started prioritizing the energy efficiency. Azad and Navimipour [9] suggested a hybrid model which combines the cultural optimization with the standard ant colony algorithm (ACO) for developing energy and makespan based scheduling system. Although superior to the existing strategies, this approach has high resource wastage due to frequent VM migrations. Torabi and Safi-Esfahani [10] designed another hybrid model using CSO and improved raven roosting optimization (HCS-IRRO) based dynamic scheduling that decreases the energy depletion and computation cost. Zhou et al. [11] developed greedy approach of modified genetic algorithm (GA) for scheduling with reduced makespan and improved QoS. But the resource management in this approach is still poor. Abdullahi et al. [12] designed large scale scheduling model using chaotic symbiotic organisms search (CMSOS). However, the reliability issue in this model is not sufficient to satisfy the QoS performance. Elaziz et al. [13] suggested hybrid scheduling model for reduced makespan by combining the features of moth search algorithm with the benefits of differential evolution. Rajagopalan et al. [14] developed optimal task scheduling using Hybrid Firefly-Genetic algorithm to reduce time and cost along with optimal resource usage. However, these two approaches do not reduce the time complexity. Natesan and Chokkalingam[15] designed Hybrid Whale Genetic Optimization Algorithm to reduce cost and time for task scheduling. But this algorithm does not resolve the energy and reliability problem effectively.

Load balancing based task scheduling strategies resolve the problem of resource exploitation and overloading cloud hosts and results in minimized makespan. Zhan et al. [16] designed GA model LAGA for scheduling with the knowledge of the current load balanced status of the hosts. Wang et al. [17] developed another GA based model which used job traversing time and parameters for load balancing to schedule tasks with improved throughput and reduced makespan. Kaur et al. [18] introduced another GA model using load environment based scheduling with effective resource utilization. However, these GA based task scheduling models have reduced the global solution search and convergence rate. Gupta and Garg [19] suggested load balanced ACO (LB-ACO) to schedule tasks with reduced makespan. However, ACO has limitations in handling large scale tasks and hence degrades the overall QoS performance. Ebadifard and Babamir [20] introduced Particle Swarm Optimization (PSO) based task scheduling which considers load-balancing as a parameter to reduce the makespan and improved resource utilization. Although efficient for large scale tasks, this approach has issues in reliability.

Raj et al. [21] developed Spontaneous Bat Algorithm with load knowledge to schedule tasks with reduced time and cost. Still this model does have limitations in the form of poor energy management. Xavier and Annadurai [22] developed a load balance aware task scheduling using the hybrid optimization of chaotic social spider algorithm (CSSA). This algorithm overcomes the local convergence problem with effective scheduling through minimized makespan and effective resource exploitation. However, this algorithm does not provide high reliability and security while also not supporting independent tasks. From the above discussions, the key challenges in designing the multi-objective energy and load balancing aware task scheduling model are identified. They include the lack of coordinated scheduling that considers the energy and load balancing along with cost and time metrics. Also, the optimization based task schedulers have higher time complexity, slow convergence due to limited global solution search ability. These problems are considered in this research paper and proposed energy efficient load balancing aware scheduling based on multiple parameters by developing the hybrid CDCSO algorithm.

3. METHODOLOGY

3.1. Modelling the Scheduling Problem

Scheduling of tasks based on multiple constraints in cloud computing can be represented as a pareto-optimal and formulated into multi-objective optimization problem. As considering multiple constraints lead to trade-off among each of the constraints, it can be effectively solved using multi-objective optimization algorithms. CDCSO is developed as a multi-objective optimizer and hence the multi-objective fitness function must be formulated with priorities assigned to the parameters. Priorities are assigned through weights computed optimally using the proposed CDCSO. The energy and load balancing index are assigned with higher weights and the sum of all weights will be unity. Mathematically, the general multi-criteria optimization model can be defined as

$$f(x) = \sum_k w_k f(x_k); \quad 0 < k \leq n \tag{1}$$

Here $w_k$represents weights assigned and $f(x_k)$represents individual fitness function at $0 < k \leq n$. For an efficient solution, the fitness function must return a minimum value. Based on this mathematical model, the multi-objective fitness function of the proposed CDCSO for task scheduling is formulated as an NP-hard optimization problem.

$$F(x) = (w_1 \times E_{Tot}) + (w_2 \times EC_{ij}) + (w_3 \times \beta) + (w_4 \times CT) + (w_5 \times RT) + (w_6 \times Th) \tag{2}$$

Here E_{Tot} is the total energy for completion of task, EC_{ij} is the computing cost (VM resource usage cost) for task completion, β is the load balancing index, CT denotes the task completion time, RT denotes the response time, Th denotes the throughput of the VM. $w_1, w_2, w_3, w_4, w_5, w_6$ are the weights assigned to the parameters whose values are greater than 0 and the sum of all weights amounts to 1.

A system model with task set $T = \{i_1, i_2, \dots i_n\}$ with n tasks in task queue and VM set VM= $\{j_1, j_2, \dots j_m\}$ with m VMs in VM pool set, the objective parameters can be computed based on the processing time and successful task execution.

Completion time:$CT_{ij} = \sum_{i=1}^{N} Ft_i - St_i,$ (3)

Response time: $RT_{ij} = \sum_{i=1}^{N} Subt_i - Wt_i,$ (4)

Throughput:$Th_{ij} = \sum_{i=1}^{N} \frac{Succ\ tasks}{Total\ time},$ (5)

Where CT_{ij} denotes the execution time ofi-th task on j-th VM, RT_{ij} denotes the response time of i-th task on j-th VM andTh_{ij}denotes throughput of the j-th VM such that $1 \leq i \leq N; 1 \leq j \leq M$.$Ft_i$denotestask finishing time on j-th VM, St_idenotes task starting time on j-th VM, $Subt_i$denotes task submission time, Wt_idenotes task waiting time, and $Succ\ tasks$denotes successfully completed tasks on j-th VM.

Energy is estimated as the power depletion for a task executed on a VM through its resource utilization and execution time. Energy consumed E_{ij} of i-th task on j-th VM is expressed as

$$E_{ij} = P(U_{ij}) \times CT_{ij} \tag{6}$$

Here CT_{ij} is the completion time calculated using Eq. (3). U_1 denotes the minimum range of CPU utilization U, and U_2 denotes the maximum range of CPU utilization U. $P(U)$denotes the depleted power at U which is expressed by

$$P(U) = \left(P(U_1) + \left(\frac{P(U_2) - P(U_1)}{10} \times \left(U - \frac{U_1}{10}\right) \times 100\right)\right) \tag{7}$$

Ucan be computed as

$$U_{ij} = \frac{U_{iref} \times CS_{ref}}{CS_j} \tag{8}$$

Where U_{ij}denotesresource consumption ofi-th task on the j-th VM, U_{iref}denotesresource consumption ofi-th task on reference VM, CS_jdenotes j-th VM's clock speed and CS_{ref}denotes reference VM's clock speed.

Usage cost EC_{ij} of i-th task on the j-th VM is based on the task size ts_i (in MI), the processing capacity P_j (in MIPS) of the VM and the cost C_j (USD per unit time) of using the VM. It can be defined as

$$EC_{ij} = \left(\frac{ts_i}{P_j}\right) \times C_j \tag{9}$$

Load balancing index: When allocating processing VMs to task set, most of the tasks are allocated to better processing capability VMs which will get more workload than other idle VMs. To balance the load across different processing VMs, load balancing index (β) is used. β is computed to gauge the deviations of load on processing VMs as follows:

$$\beta = \sqrt{\frac{\sum_{j=1}^{m} L_j - \bar{L}}{m}} \tag{10}$$

Where L_j denotes load of j-th VM and $\bar{L}$denotes average VM load.

3.2. Multi-Objective Chaotic Darwinian Chicken Swarm Optimization for Task Scheduling

The suggested CDCSO has been developed by integrating the concepts of chaos theory and Darwinian Theory of survival to the standard CSO algorithm. The application of Darwinian Theory to optimization algorithm has been inspired from Tillett et al. [23]. Using these strategies the position update functions and the global search process are modified such that the convergence of the algorithm avoids providing the local optimum solution as global best. The proposed CDCSO is initialized similar to that of the standard CSO [24]. Fig. 1 illustrates the procedures in the proposed CDCSO algorithm for task scheduling.

First, X population of chickens are initialized from which the swarms are generated. Each swarm consists of roosters, hens, mother hens and chicks among which the rooster with highest fitness strength assumes the leader role. During the food search process, the roosters find the food independently and consume the greater quantity. The next best are the mother hens which take food for themselves and the chicks, which are completely dependent for food and security. The remaining food is shared among the remaining hens. Hence the roosters are considered as leaders, mother hens are co-leaders, hens are elders and chicks are the members assigned based on the descending fitness values in hierarchical order.

The chicken agents' population are represented based on their roles as LX (leaders), CLX (co-leaders), EX (elders) and MX (members). The position of all the X chickens is determined by the food search and their movement towards the food location. It is given as $x_{i,j}^{t}$ $(i \in [1, \dots, X], j \in [1, \dots, D])$ at time step *t* and iterations *it*. The fitness is estimated using the Eq. (2) and based on the obtained fitness value for each chicken agent, they are ranked in descending order. Depending upon the ranking order, the agents move towards or away from the other agents and inter-change their roles in hierarchical order. The movement will result in position changes which are updated using the standard position update equations in the standard CSO. In the proposed CDCSO, these equations are modified using the chaos theory. The chaotic functions using the Logistic map strategy is used to replace the uniformly distributed random number in the standard position update equations. The suggested chaotic function is given as

$$\theta_{n+1} = \xi\theta_n(1-\theta_n); \qquad (11)$$

Here ξ represents aregulatingconstraint and θ_n represent the chaotic factor. The preliminary stage of the algorithm requires fulfilling $\theta_0 \in (0,1)$ and $\theta_0 \notin (0.25, 0.50, 0.75)$. As varying values of ξ result in sequential behaviour, extensive analysis has been made to find the appropriate value. It is found that when $\xi = 4$, the system is purely chaotic and improves the efficacy of scheduling system.

Using this chaotic function, the position update of the leader is modified as

$$x_{i,j}^{t+1} = x_{i,j}^{t} \times (1 + \theta_{i,j} \times Randn(0, \sigma^2)) \qquad (12)$$

Where $\theta_{i,j}$ is calculated as the chaotic function for i-th task on j-th VM, $x_{i,j}^{t}$ is the best position until preceding iteration t, $Randn(0, \sigma^2)$denotes Gaussian function whose mean value is zero while standard deviation is represented byσ^2.

The co-leaders and the elders are moved in controlled manner and their positions are updated using the same equation

$$x_{i,j}^{t+1} = x_{i,j}^{t} + S1 \times \theta_{i,j} \times \left(x_{r1,j}^{t} - x_{i,j}^{t}\right) + S2 \times \theta_{i,j} \times (x_{r2,j}^{t} - x_{i,j}^{t}) \qquad (13)$$

Where S_1, S_2 are the social hierarchical coefficient of chickens computed as in standard CSO and $r1, r2 \in [1, \dots, N]$ are the chicken indexes.

The position of the members are updated using

Chicks $x_{i,j}^{t+1} = x_{i,j}^{t} + FL \times \theta_{i,j} \times (x_{m,j}^{t} - x_{i,j}^{t})$ (14)

Here $x_{m,j}^{t}$ is the location of the j-th member corresponding to the co-leader and $fl \in [0,2]$ is the flocking parameter.

The Darwinian theory of survival is applied at this stage by incorporating the new swarm and chick generation and deletion processes. At each step of the algorithm, the evolve process is performed in which the existing swarms are permitted to construct a new swarm which has constant probability. After the generation, selection/deletion is initiated to obtain the progressing swarms and deleting the non-progressing swarms. The evolution process is completed by evaluating the fitness in the new swarm and obtaining the optimal best positions. After updating the new positions, the swarm produces a new chicken only when the best fitness is obtained while deletes the existing chicken when worst fitness is obtained within the predetermined iterations. The procedures followed in constructing and deleting the swarms and chickens are given as follows:

3.2.1. Constructing New Swarms and Chickens

When the initial iterations result in the condition that no chicken is killed in a swarm (i.e. $N_{kill} = 0$) and the number of swarms do not exceed the maximum limit, a new swarm can be constructed with predetermined minimum population. A new swarm could be constructed with iterations by evolving two existing swarms until the maximum number of swarms is attained. Once the maximum swarms are attained, the new swarms can only be generated by killing an already existing swarm i.e. it must have $N_{kill} = 0$ to continue constructing new swarms. Thus generated new swarms will have probability $p(f/N_s)$through uniform arbitrary integer$f \in [0,1]$besides number of swarms N_s. $(1/N_s)$ is used to limit the new swarm creation when the number of swarm is approaching the maximum. The new swarm is constructing by imitating the properties of two randomly selected existing swarms considered as parents. However, the parent swarm properties are unaltered after the new swarm generation. The newly generated swarm contains the properties of two parent swarm in equal share and mostly the special dominating features on single parent may be chosen to improve the CDCSO designing. When the global best fitness is obtained for the new swarm, the process again shifts to generating another new swarm.

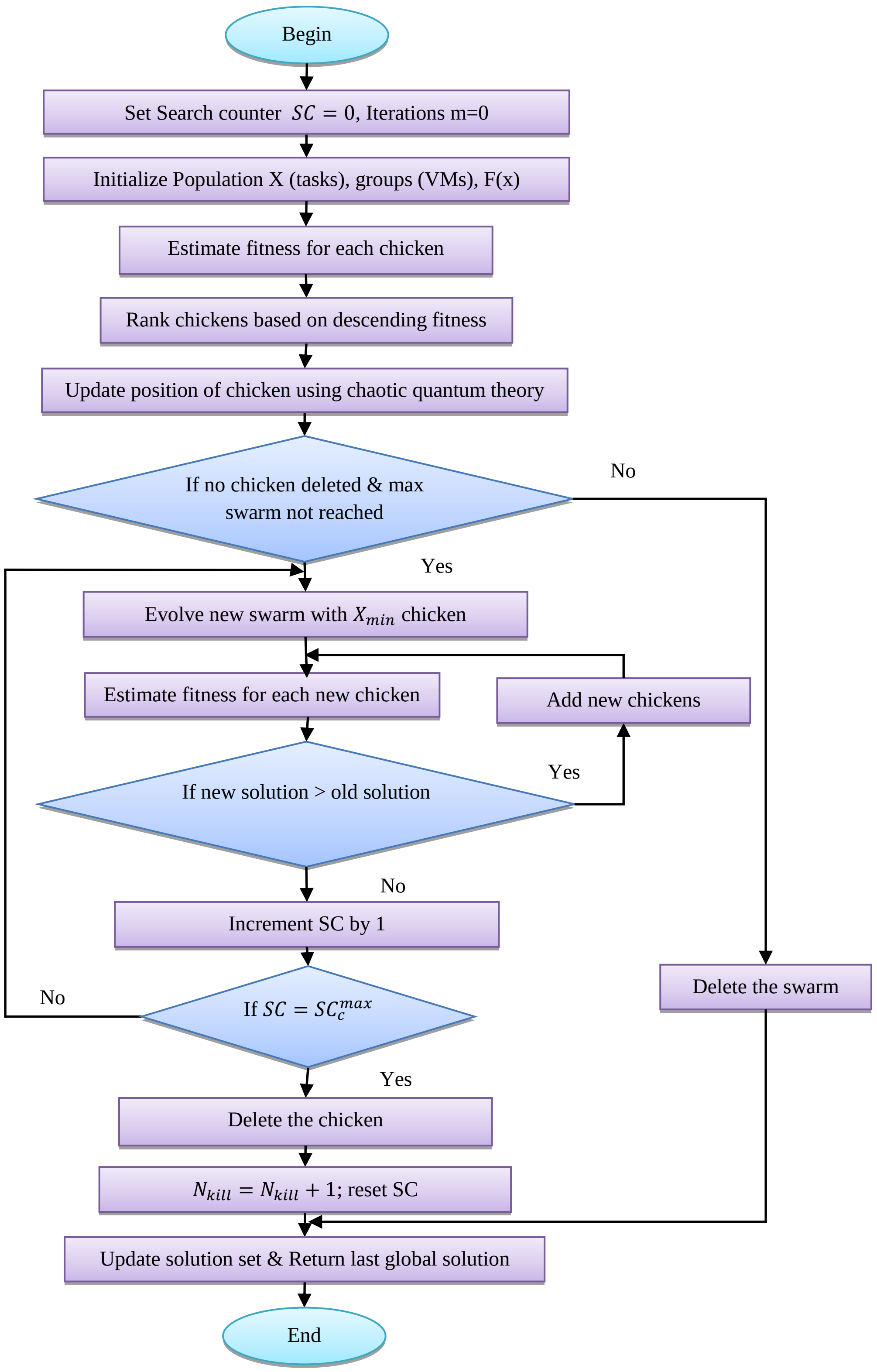
Begin
Set Search counter $SC = 0$, Iterations m=0
Initialize Population X (tasks), groups (VMs), F(x)
Estimate fitness for each chicken
Rank chickens based on descending fitness
Update position of chicken using chaotic quantum theory
If no chicken deleted & max swarm not reached
No
Yes
Evolve new swarm with X_{min} chicken
Estimate fitness for each new chicken
Add new chickens
Yes
If new solution > old solution
No
Increment SC by 1
Delete the swarm
No
If $SC = SC_c^{max}$
Yes
Delete the chicken
$N_{kill} = N_{kill} + 1$; reset SC
Update solution set & Return last global solution
End

Figure 1. Flowchart of proposed CDCSO Algorithm

3.2.2. Removing Swarm

A swarm is eliminated only when the total chickens in the swarm is less than a pre-defined quantity. The swarm's chicken population X is bounded such that, $X_{min} \leq X \leq X_{max}$. Thus the swarm is removed from the solution set when $X < X_{min}$.

3.2.3. Removing a Chicken

A search counter is created and the minimum and maximum values are pre-determined. If the search counter value is greater than the maximum value, the corresponding chicken will be removed. At the beginning of creating a new swarm, the search counter, SC is started from zero and it keeps on adding 1 whenever there is no improvement in chicken' fitness value. The chicken that constantly result in worst fitness even after multiple evolution, is deleted and the number of evolutions that resulted in non-favourable fitness is monitored usingSC. The threshold SC_c^{max} is fixed as the maximum critical threshold for this counter. The chicken that exceeds SC_c^{max} is removed from the group and search counter, $SC$is reset to a value closer toSC_c^{max}instead zero in order to reduce the time to monitor the fitness improvement. This reduction will help in improving the swarms' fitness without degrading the delay tolerance. This reset value of the search counter is selected depending upon theN_{kill}, given as follows

$$SC_c(N_{kill}) = SC_c^{max}\left[1 - \frac{1}{N_{kill}+1}\right] \quad (15)$$

Based on this counter values, the chicken swarms are generated and deleted to obtain the most optimal VMs for allocating the tasks. The task scheduling mechanism using the CDCSO algorithm is given in algorithm 1.

```
Define the Set of Tasks, VMs
Population initialization, X chickens
Set initial position, other parameters
Establish the fitness function using Eq. (2)
Set Iterations $it = 0$, Search counter SC=0
For each chicken agent
  Calculate the individual fitness values
  While ($it < it_{max}$)
      If ($it$ %% $G == 0$)
          Assign the swarm with tasks and VMs
          Compare and move the agent towards global best
          Rank the agents in hierarchical order based on fitness
          Until termination conditions achieved
      End if
  For i = 1 : N applying chaos theory
      If i == leader, then
          Update position via Eq. (12)
      Else If i == co-leader or elder, then
          Update position via Eq. (13)
      Else If i == member, then
          Update position via Eq. (14)
      End if
      Evolve the swarm to obtain new solutions
      If $N_{kill} = 0$ and $N_s < N_s^{max}$ then
          Create fresh swarm with minimum chicken population
          Evaluate the fitness for each chicken in new swarm
          If the $new\ fitness > previous\ best,$
              Add new chicken to the swarm
              Until $X < X_{max}$
          Else
              Update $SC = SC + 1$
              If $SC = SC_c^{max}$ then
                  Delete the chicken
                      $N_{kill} = N_{kill} + 1$
                  Reset the $SC$ using Eq. (15)
                  End if
          End if
      Else if $N_{kill} \neq 0$ and $X < X_{min}$
          Delete the swarm
          Update the solution set
      End if
      Repeat the evolving process until no new swarm can be formed
  End for
      $it = it + 1$
      End while
```

```
        Return best VMs for allocating tasks
End for
```

Algorithm 1: CDCSO Task Scheduling

4. EXPERIMENTAL RESULTS

The suggested CDCSO for scheduling of cloud tasks is evaluated through simulation software of CloudSim 3.0.3 and the performance evaluation is achieved over a set of independent tasks generated from the Planet lab workload traces. Planet lab workload consists of CPU utilization data of hundreds of VMs from more than 500 servers positioned around the world. The tests are performed on Intel ® core i5 processor with 1.8GHz frequency CPU, 8GB RAM and Windows 10 operating system using Eclipse and JDK 1.8. The CloudSim parameter settings are given in Table 1.

Unit	Parameter	Value
Data center	Quantity	1
	Type	Heterogeneous
	Link delay (milliseconds)	10-100
	Bandwidth (Gbps)	1-10
Host	Quantity	5
	Cores	1-4
	Host RAM (MB)	4096
	Host Storage (MB)	1000000
	Host bandwidth (bps)	10000
VM	Quantity	20-100
	CPU (MIPS)	1000-10000
	RAM (MB)	512
	Bandwidth (bps)	1000
	Cores per VM	1
Task	Quantity	1000
	Task length (MI)	200-1000
	Task size	200-600
	Iterations	5 - 25

Table 1 Simulation Parameter Settings

The performance comparisons are made in terms of energy, cost, load balancing index, task completion time, response time, throughput, CPU utilization (CPU) and Bandwidth utilization (BW). CPU and BW are computed as given in [25].

$$CPU = \frac{T_{MIPS} . CPU_{MIPS}}{1000 . T_{ms}} \quad (16)$$

$$BW = \frac{\tau_v \times 100}{BW_v \times \psi_v} \quad (17)$$

Here T_{MIPS} denotes calculated task's MIPS length; $\mathrm{CPU}_{\mathrm{MIPS}}$ denotes allotment of the CPU MIPS; T_{ms} denotesimplementation time of task on a VM in milliseconds. BW_{v} denotes bandwidth allocation; τ_{v} denotes data transfer capacity; ψ_{v} denotes VM lifetime.

The performance outcomes of the suggested CDCSO algorithm are given in Table 2 for 20 VMs and varying tasks of 25 to 100.

Number of tasks	Energy (J)	Cost ($)	Load balancing index	Task completion time (ms)	Throughput (bps)	Response time (seconds)	CPU Utilization (%)	Bandwidth utilization (%)
25	0.5217	36	0.07144	4.8	145.020	0.002916	28.279	7.86
50	0.7878	63	0.1072	7.19	158.913	0.004374	42.7746	8.628
75	1.38456	83	0.1429	9.6	290.699	0.005832	56.554	11.874
100	2.2738	110	0.1786	12.0	325.926	0.00729	71.042	9.558

Table 2. Performance Results of CDCSO (VMs= 20)

Number of tasks	Energy (J)	Cost ($)	Load balancing index	Task completion time (ms)	Throughput (bps)	Response time (seconds)	CPU Utilization (%)	Bandwidth utilization (%)
200	7.89	198	0.3214	21.59	572.592	0.0131	77.955	15.8087
400	12.878	230	0.3572	24.0	742.716	0.01458	80.4116	20.2654

600	13.405	254	0.4286	28.79	804.88	0.0175	82.8683	24.722
800	18.558	274	0.4858	36.0	946.419	0.02187	86.1	28.178
1000	25.793	300	0.5429	43.19	1199.7	0.02624	89.6487	31.743

Table 3. Performance Results of CDCSO (VMs= 100)

The performance outcomes of the suggested CDCSO algorithm are given in Table 3 for 100 VMs and varying tasks of 200 to 1000.

Results obtained in Table 2 for the CDCSO algorithm when considering 20 VMs with varying tasks illustrate that the performance of CDCSO has linearly improved. The energy, cost and load balancing indexes are increased with the increasing number of tasks. Similarly, the time and resource utilization metrics are also increased, thus indicating the increasing scheduling process. It is also noted that the energy consumption is increased directly proportional to the resource utilization (CPU and bandwidth) of the VMs. The performance values suggest that the proposed CDCSO has performed better with the smaller number of tasks. Results in Table 3 for the CDCSO algorithm, also indicate that the proposed CDCSO has achieved better results. This is significantly important considering that the workload is increased (200 to 1000 tasks). As evident from the literature studies, the performance of the task scheduling and load balancing algorithms tends to degrade gradually when the workload increases. The proposed CDCSO has provided consistent performance even at larger workload and hence it is considered as one of the better performing scheduling algorithms.

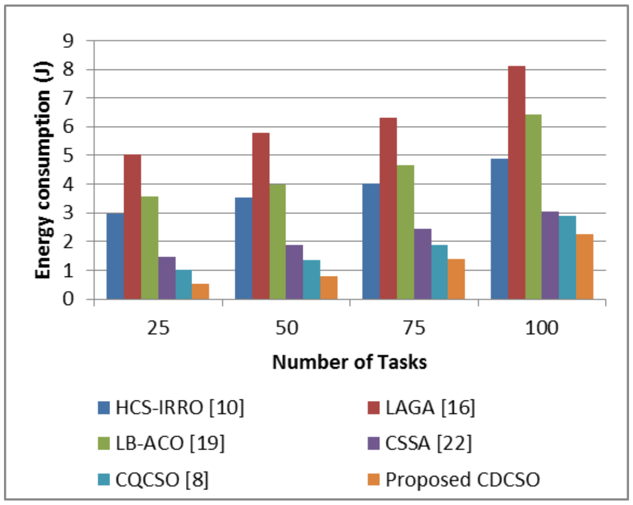

Figure 2 Energy Evaluation

To validate the competence of the suggested CDCSO algorithm, the results are equated with the prevailing optimization based energy and load aware task scheduling strategies. The existing task scheduling models considered for performance comparison are HCS-IRRO [10], LAGA [16], LB-ACO [19], CSSA [22] and CQCSO [8]. Comparisons are made for these methods in terms of the six objective parameters under the scenario of 20 VMs and number of tasks varying from 25 to 100.

Figure 2 illustrates the comparison of energy consumption of the suggested CDCSO and the prevailing scheduling algorithms. The suggested CDCSO algorithm has less energy consumption than the other algorithms due to the optimized selection of VMs based on multiple objectives especially the energy and load balancing index which improves the effective resource utilization.

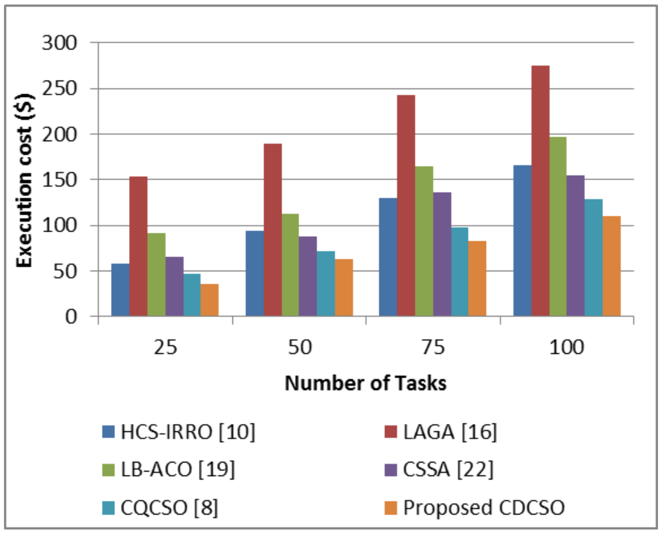

Figure 3 Cost Evaluation

Figure 3 compares the suggested CDCSO performance against the prevailing scheduling algorithms in terms of cost ($). The cost of CDCSO is minimum than the other methods due to the effective utilization of resources which is not considered significantly in other prevailing algorithms. The cost reduction also impacts the usage of this model in business entities and other commercial industries to minimize the expenditure.

Figure 4 displays the comparison of load balancing index values of the suggested CDCSO against the prevailing scheduling algorithms. Load balancing index must be low to ensure the reduced functioning of the idle VMs and thus reducing the energy consumption and resource wastage.

CDCSO has provided with the lowest β values under the constraint environment even at larger workloads.

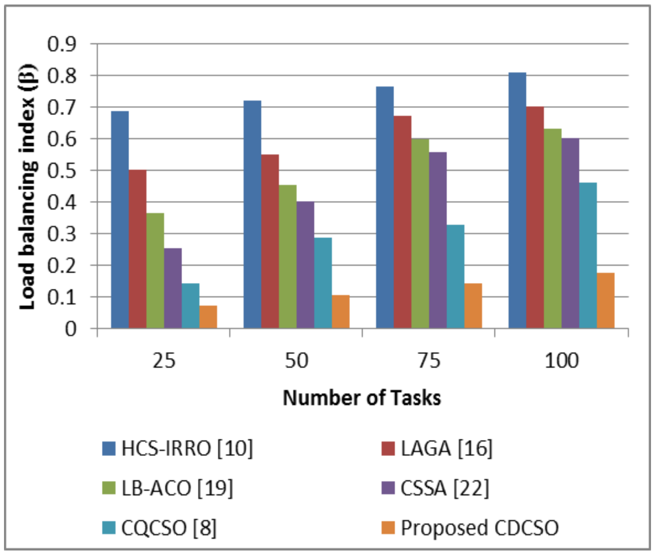

Figure 4 Load Balancing Index Evaluation

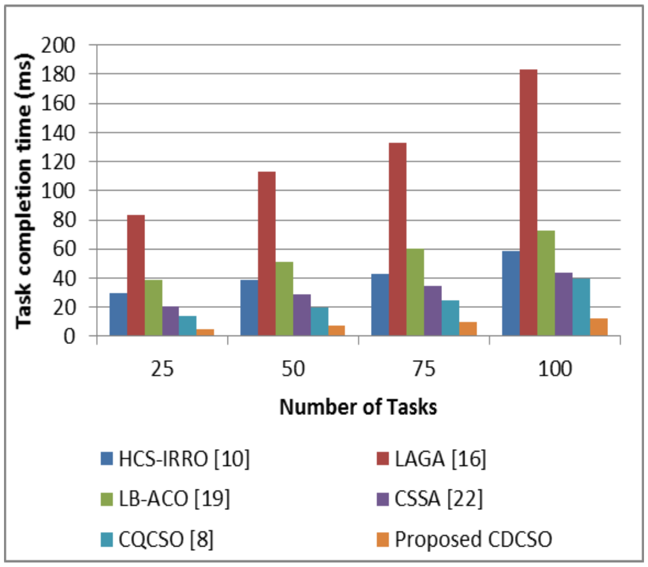

Figure 5 Evaluation of Task Completion Time

Figure 5 displays the suggested CDCSO and the prevailing scheduling algorithms in terms of completion time. The suggested CDCSO optimization based task scheduling algorithm has outperformed other algorithms with minimum time due to the optimal and load balanced task scheduling ensuring the delay-free and faster execution of tasks.

Figure 6 illustrates the comparison of CDCSO and the prevailing task scheduling algorithms in terms of response time. The suggested CDCSO outperformed the other compared algorithms. This performance directly reduces the delay of execution and the resources wasted for queuing and this improvement can be attributed to the faster convergence of the CDCSO algorithm.

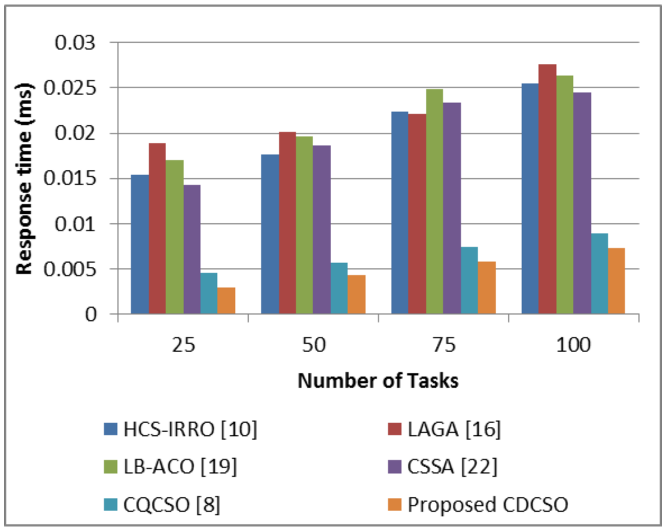

Figure 6 Evaluation of Response Time

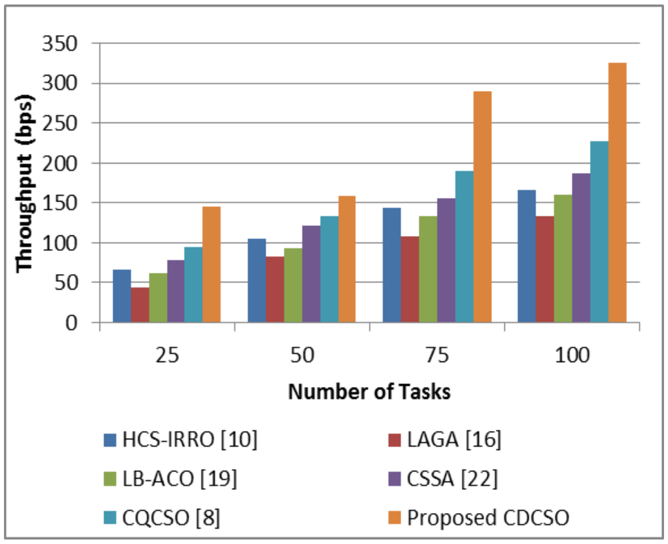

Figure 7 Throughput Comparison

Figure 7 compares the Throughput of the CDCSO and the compared optimization based task scheduling algorithms. From the figure, it can be justified that the suggested CDCSO has improved throughput than the other compared algorithms due to the load balanced and energy aware scheduling which has significantly reduced the complexity and overheads.

The obtained performance improvement of the suggested CDCSO model over the existing models is because of the increased global search ability and convergence rate of CDCSO. Although the existing models worked efficiently for smaller cloud networks, their convergence rate decreased for the larger cloud. The suggested CDCSO through the chaotic initialization and the Darwinian process of maintaining the swarm and chickens through evolution has improved its searching capabilities and convergence rate that improved the task scheduling performance. The another major factor that influence this performance improvement is the utilization of the load balancing index metric for analyzing the VM status for scheduling which is not performed in existing models. Thus, it is concluded from the simulation results that the suggested CDCSO significantly improved the scheduling performance better than the existing models.

5. CONCLUSION

Energy efficiency and load balancing are considered as the primary constraints for task scheduling in this paper. Based on these primary and other secondary constraints, the task scheduling is performed using a novel hybrid algorithm called CDCSO developed by integrating the chaos theory and Darwinian Theory to the standard chicken swarm optimization. The experimental results were obtained and compared with that of prevailing optimization based task scheduling algorithms for assessing the efficiency of the suggested CDCSO. The evaluations indicated the CDCSO algorithm based task scheduling has achieved better performance than the existing algorithms with reduced energy consumption, cost, load balancing index, task completion time and response time while also increasing system throughput. In future, the proposed algorithm can be improved by including more QoS parameters without increasing the complexity and overhead. The other possible direction is to extend the proposed CDCSO to perform fault tolerance aware and security aware task scheduling.

REFERENCES

[1] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., &Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 25(6), 599-616.

[2] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing—The business perspective. Decision support systems, 51(1), 176-189.

[3] Grobauer, B., Walloschek, T., & Stocker, E. (2010). Understanding cloud computing vulnerabilities. IEEE Security & privacy, 9(2), 50-57.

[4] Arunarani, A. R., Manjula, D., &Sugumaran, V. (2019). Task scheduling techniques in cloud computing: A literature survey. Future Generation Computer Systems, 91, 407-415.

[5] Lin, W., Peng, G., Bian, X., Xu, S., Chang, V., & Li, Y. (2019). Scheduling Algorithms for Heterogeneous Cloud Environment: Main Resource Load Balancing Algorithm and Time Balancing Algorithm. Journal of Grid Computing, 17(4), 699-726.

[6] Singh, H., Tyagi, S., & Kumar, P. (2020). Scheduling in Cloud Computing Environment using Metaheuristic Techniques: A Survey. In Emerging Technology in Modelling and Graphics (pp. 753-763). Springer, Singapore.

[7] Kiruthiga, G. & Mary Vennila, S. (2019). An Enriched Chaotic Quantum Whale Optimization Algorithm Based Job scheduling in Cloud Computing Environment. International Journal of Advanced Trends in Computer Science and Engineering, 6(4),1753-1760.

[8] Kiruthiga, G. & Mary Vennila, S. (2020). Multi-Objective Task Scheduling using Chaotic Quantum-Behaved Chicken Swarm Optimization (CQCSO) in Cloud Computing Environment.International Conference On Evolutionary Computing And Mobile Sustainable Networks,

[9] Azad, P., &Navimipour, N. J. (2017). An energy-aware task scheduling in the cloud computing using a hybrid cultural and ant colony optimization algorithm. International Journal of Cloud Applications and Computing (IJCAC), 7(4), 20-40.

[10] Torabi, S., & Safi-Esfahani, F. (2018). A dynamic task scheduling framework based on chicken swarm and improved raven roosting optimization methods in cloud computing. The Journal of Supercomputing, 74(6), 2581-2626.

[11] Zhou, Z., Li, F., Zhu, H. et al. An improved genetic algorithm using greedy strategy toward task scheduling optimization in cloud environments. Neural Comput & Applic 32, 1531–1541 (2020).

[12] Abdullahi, M., Ngadi, M. A., Dishing, S. I., & Ahmad, B. I. E. (2019). An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment. Journal of Network and Computer Applications, 133, 60-74.

[13] Elaziz, M. A., Xiong, S., Jayasena, K. P. N., & Li, L. (2019). Task scheduling in cloud computing based on hybrid moth search algorithm and differential evolution. Knowledge-Based Systems, 169, 39-52.

[14] Rajagopalan, A., Modale, D. R., &Senthilkumar, R. (2020). Optimal Scheduling of Tasks in Cloud Computing Using Hybrid Firefly-Genetic Algorithm. In Advances in Decision Sciences, Image Processing, Security and Computer Vision (pp. 678-687). Springer, Cham.

[15] Natesan, G., &Chokkalingam, A. (2020). Multi-Objective Task Scheduling Using Hybrid Whale Genetic Optimization Algorithm in Heterogeneous Computing Environment. Wireless Personal Communications, 110(4), 1887-1913.

[16] Zhan, Z. H., Zhang, G. Y., Gong, Y. J., & Zhang, J. (2014). Load balance aware genetic algorithm for task scheduling in cloud computing. In Asia-Pacific Conference on Simulated Evolution and Learning (pp. 644-655). Springer, Cham.

[17] Wang, T., Liu, Z., Chen, Y., Xu, Y., & Dai, X. (2014). Load balancing task scheduling based on genetic algorithm in cloud computing. In 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing (pp. 146-152). IEEE.

[18] Kaur, K., Kaur, N., & Kaur, K. (2018). A novel context and load-aware family genetic algorithm based task scheduling in cloud computing. In Data Engineering and Intelligent Computing (pp. 521-531). Springer, Singapore.

[19] Gupta, A., &Garg, R. (2017). Load balancing based task scheduling with ACO in cloud computing. In 2017 International Conference on Computer and Applications (ICCA) (pp. 174-179). IEEE.

[20] Ebadifard, F., &Babamir, S. M. (2018). A PSO‐ based task scheduling algorithm improved using a load‐ balancing technique for the cloud computing environment. Concurrency and Computation: Practice and Experience, 30(12), e4368.

[21] Raj, B., Ranjan, P., Rizvi, N., Pranav, P., & Paul, S. (2018). Improvised Bat Algorithm for Load Balancing-Based Task Scheduling. In Progress in Intelligent Computing Techniques: Theory, Practice, and Applications (pp. 521-530). Springer, Singapore.

[22] Xavier, V. A., &Annadurai, S. (2019). Chaotic social spider algorithm for load balance aware task scheduling in cloud computing. Cluster Computing, 22(1), 287-297.

[23] Tillett, J., Rao, T., Sahin, F., & Rao, R. (2005). Darwinian particle swarm optimization. Accessed from https://scholarworks.rit.edu/other/574

[24] Meng, X., Liu, Y., Gao, X., & Zhang, H. (2014). A new bio-inspired algorithm: chicken swarm optimization. In International conference in swarm intelligence, Springer, Cham, pp. 86-94.

[25] Mosleh, M. A., Radhamani, G., Hazber, M. A., &Hasan, S. H. (2016). Adaptive cost-based task scheduling in cloud environment. Scientific Programming, 2016.

RESEARCH ARTICLE

Authors

G. Kiruthiga completed B.Sc. in Electronic Science (2001) from Bharathidasan University, Tiruchirappalli, India and Master of Computer Applications (2004) from SASTRA University, Thanjavur, India. She completed her M.Phil (Computer Science) in the year 2006 from Annamalai University, Chidambaram and currently pursing Ph.D as Part-Time in Computer Science from Presidency College, Chennai. She is currently working as an Associate Professor in Department of Computer Applications, Guru Nanak College, Chennai. Her areas of interest include Computer Networks, Cloud Computing and Bio inspired Algorithms.

Dr. S. Mary Vennila completed B.Sc. in Physics (1986) and M.Sc in Computer Science (1990) from Bharathidasan University, Tiruchirappalli, India. She obtained her M.Phil (Computer Science) in the year 2002 and Ph.D in Computer Science (2009) from Mother Teresa Women's University, Kodaikanal. She is currently working as an Associate Professor and Head of the PG and Research Department of Computer Science, Presidency College, Chennai, India. Her areas of interest include Networks, Grid computing, Cloud security and Data mining.