International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8 Issue-3, September 2019

Reinforcement Learning Based APO-PTIRIAL for Load Balancing in Cloud Computing Environment

V. Radhamani, G. Dalin

Abstract*: *Power consumption-Traffic aware-Improved Resource Intensity Aware Load balancing (PT-IRIAL) method was proposed to balance load in cloud computing by choosing the migration Virtual Machines (VMs) and the destination Physical Machines (PMs). In this paper, an Artificial Intelligence (AI) technique called Reinforcement Learning (RL) is introduced to find out an optimal time to migrate the selected VM to the selected destination PM. RL enables an agent to find out the most appropriate time for VM migration based on the resource utilization, power consumption, temperature and traffic demand. RL is incorporated into the cloud environment by creating multiple state and action space. The state space is obtained through the computation of resource utilization, power consumption, temperature and traffic of selected VMs. The action space is represented as wait or migrate which is learned through a reward function. Based on the action space, the selected VMs are waiting or migrating to the selected destination PMs.

Index Terms*: *Load balancing, Optimized power consumption-traffic aware- improved resource intensity aware load balancing, Artificial intelligence, Reinforcement learning.

I. INTRODUCTION

One of the key enabling technologies for cloud computing environment is virtualization. It makes it feasible to run multiple applications and multiple operating systems at the same time on the same technology, so as to supply services by a virtual unit. Nevertheless, it is more difficult to allocate a massive amount of tasks to dynamic resources for distributed computing. Some nodes in the cloud may get into overload and under load state due to various factors. One of the effectual ways to solve the above problem is load balancing. It guarantees that services are distributed transparently in spite of the physical implementation and location within the cloud.

A Resource Intensity Aware Load balancing (RIAL) [2] was proposed to balance load among the machines in cloud. It assigned dynamic weights to resources based on their usage in the Physical Machines (PMs). According to the weight of the resources, the VMs were move towards lightly loaded PMs. An Improved RIAL (IRIAL) [3] was proposed which considered both heavily loaded PMs and lightly loaded PMs as destination PMs for load balancing. Because, some resources in PM are over utilized and underutilized. So, there may be a chance that the heavily loaded PM closer to PM of selected migration VM might have required resources to balance the load.

A Power communication Traffic aware-Improved Resource Intensity Aware Load Balancing (PT-IRIAL) [4] method was proposed to improve the selection of migration VMs and destination PMs that considered various factors such as power consumption, temperature and traffic measures to balance the load. For optimal selection of migration VMs and destination PMs in PT-IRIAL, different optimization algorithms such as Artificial Plant Optimization (APO) [5], Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) were introduced which effectively balance the load in cloud.

In this paper, an Artificial Intelligence (AI) technique called as Reinforcement Learning (RL) [6] is introduced to learn an optimal time for scheduling VM migration based on the resource utilization rate, communication rate, completion time, power consumption, temperature and traffic flow demand. In RL, an agent learns the optimal time to migrate VMs and allocate the actions of VMs as either wait or migrate. It is achieved through a trial and error process, where the agents are interacted with its environment and observing the state and action of VMs. Based on the action of VM, the VMs are migrated to the destination PMs at an optimal time.

II. LITERATURE SURVEY

A new model [7] was presented for VM migration between hosts to balance the load in cloud. This model used fuzzy TOPSIS algorithm to transfer VMs between cluster nodes. TOPSIS was utilized to make efficient decision and determined the most loaded servers. The fuzzy TOPSIS strategy was developed to extort a list of server from the most to the least loaded. Based on the list loads were balanced across the systems in cloud. However, the complexity of fuzzy TOPSIS is increased when the number of VMs gets increased.

A soft computing based load balancing approach [8] was proposed to select the nodes for executing a task based on the properties of the tasks.

Manuscript published on 30 September 2019
* Correspondence Author
V. Radhamani*, Department of Computer Science, Hindustan College of Arts and Science, Coimbatore, Tamilnadu, India.
G. Dalin, Department of Computer Science, Hindustan College of Arts and Science, Coimbatore, Tamilnadu, India.

A Stochastic Hill Climbing approach was utilized for distribution of incoming jobs to the VMs or servers. The stochastic hill climbing approach was a loop that constantly migrated in the direction where the maximum resources are available, which is uphill. This loop was stop when it reached a peak where no neighbour has a maximum optimization of available resources. However, it has high computational complexity.

A Genetic Algorithm (GA) based load balancing strategy [9] was proposed for load balancing in cloud computing. Each chromosome in the population of GA, balancing the load of cloud infrastructure based on the make span of a given task set. The crossover and mutation processes were carried out in GA to optimally distribute the task set among the system in cloud. However, the GA has convergence problem.

A Honey Bee Behavior inspired Load Balancing (HBB-LB) [10] was proposed to balance load in cloud. With the consideration of waiting time of the tasks in the queue, HBB-LB balanced the priorities of tasks. Based on the positive and negative signals among the bees the tasks were allocated to the nodes in the cloud. The two circumstances required a type of correspondence among honey bee swarms. However, the HBB-LB algorithm has lack of scalability problem.

An Energy-aware multi-objective hybrid fruitfly optimization algorithm [11] was proposed to balance load in cloud. It was based on the replicated annealing to enhance optimization accuracy and convergence rate of fruitfly optimization algorithm. Each swarm of fireflies in hybrid fruitfly optimization algorithm moved in various directions to go behind uniform distribution and then combined the replicated annealing to revise the solution and current location for load balancing. However, a threshold value in hybrid fruitfly optimization algorithm influences the workload assignment.

III. PROPOSED METHODOLOGY

In this section, Reinforcement Learning (RL) based determination of optimal time to migrate the selected VMs to the destination PMs for load balancing is described in detail. The selected migration VMs based on APO have to migrate to the selected destination PMs at an optimal time for effective load balancing in cloud. The RL agent is used as a judgment support during live migration in cloud environment. Q-Learning is used to calculate the state-action pair of VMs that can make predictions gradually by bootstrapping the present estimates onto previous estimates.

A. Reinforcement learning-optimized power consumption traffic aware-improved resource intensity aware load balancing

Network Model

Consider a network has N Physical Machines (PMs) which provide as a resource collection in the cloud. Assume P_x represents PM $x\ (x = 1,2, \dots N)$ and the number of VMs hosted by P_x is denoted as n_x, number of VMs are denoted as $V_{xy}(y = 0,1,2, \dots, n_x)$. Let l_{xy} be the Euclidean distance of each candidate in P_x with ideal VM and ideal communicate rate T_{xy}, Pow_{xy} be the power consumption by x PM while migrating y VMs, $Temp_{xy}$ denotes the temperature produced while migrating y VM to x PM and STR_{xy} denotes the traffic flow between VM's in PM. A cost matrix $Cost_{xy}$ of each candidate VM_{xy} in P_x is given as,

$$Cost_{xy} = Min\left(l_{xy}, Pow_{xy}, Temp_{xy}, STR_{xy}\right) \quad (1)$$

A cost matrix of each candidate PM P_x from the ideal PM is constructed as,

$$Cost_{p,xy} = Min\left(l_{p,xy}, Pow_{p,xy}, Temp_{p,xy}, STR_{p,xy}\right) \quad (2)$$

Based on the above cost matrix, Artificial Plant Optimization (APO) selected the optimal migration VMs and destination PMs.

Reinforcement Learning

RL is used as a decision system where the RL agent can make a decision the most appropriate time to transfer a VM to the destination PM depending on the current resource utilization rate, communication rate, completion time, power consumption, temperature and traffic flow in the cloud. Through trial and error process, the agents in RL learn by communicating with its environment and examining the resulting reward signal. The RL is modeled as Markov Decision Processes (MDP) which is a tuple $\langle S, A, P, R\rangle$. An agent of RL maps an action $a \in A$, to state $s \in S$, which then migrates to future state $s' \in S$ with a probability P. When an action a is executed in state s it will transition to s' with a probability P which is defined as follows:

$$P^{a}_{s,s'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \quad (3)$$

The agent obtains a scalar reward r_t which can be either positive or negative. The reward space represented as any current action a_t, state s_t and along with any next state s_{t+1}. Eq. (4) shows the expected value of next reward.

$$R^{a}_{s,s'} = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \quad (4)$$

The aim of finding solution for MDPs is to determine a policy, by maximizing collected rewards. A model-free RL algorithm such as Q learning is utilized to estimate the optimal policy. Q learning estimates the state-action pair $Q(s_t, a_t)$. It makes prediction by bootstrapping the current estimates onto preceding estimates. Followed by every state-action-reward state transition experience, Q- learning computes a Q-value as,

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_t + \gamma(MaxQ(s_{t+1}, a_{t+1}) - Qst,at \right. \quad (5)$$

In Eq. (5), α denotes the learning rate it finds how quickly an agent learns and γ control agent's degree of myopia. Both α and γ ranges from 0 to 1. If the α value is chosen close to 0, then it guarantees that the most recent information obtained is utilized. If the α value is chosen close to 1, then it infers no learning will take place. If the γ value is chosen close to 1, then it emphasizes greater weight on future rewards. If the γ value is chosen to 0, then it considers only the most recent rewards. $MaxQ(s_{t+1}, a_{t+1})$ restores the greatest estimate for the future state-action pair. After the calculation of Q-value, it is store in the agent's Q-Matrix.

The RL is incorporate into the cloud environment by defined a multiple state and action space which is represented as $\langle s, d, a\rangle$. The Reinforcement Learning-Optimized Power consumption Traffic aware-Improved Resource Intensity Aware Load Balancing strategy has two agents for its state space.

The current cost matrix of VM represents the first state space. It permits the first state space to be defined as $s_{tVM}, s_{tPM} \in S = \{0,100\}$. The state of the selected VM s_{tVM} is calculated as,

$$s_{tVM} = \sum_{v=1}^{n} F(VM_n) \quad (6)$$

In Eq. (6), the current VM selected by APO is denoted as VM_n, F is the function that computes the Euclidean distance of VM_n in P_x with ideal VM and ideal T_{xy}, power consumption by x PM while migrating VM_n and temperature produced while migrating VM_n (Radhamani, V. & Dalin, G., 2018b). n is all possible VMs that can be migrated.

The state of the selected PM s_{tPM} is calculated as,

$$s_{tPM} = \sum_{v=1}^{n} H(PM_n) \quad (7)$$

In Eq. (7), PM_n is the current PM selected by APO, H is the function that calculates the Euclidean distance of $PM\ P_x$ from the ideal PM [5], power consumption by PM and temperature produced while migrating VM_n to PM_n [5]. n is all possible PMs in the network.

The second state space d_{tVM} and d_{tPM} are defined as the current VMs traffic and PMs traffic respectively. The d_{tVM} is calculated using STR_{xy} and d_{tPM} is calculated using $STR_{p,xy}$ [5]. Both the VMs traffic and PMs traffic has three categories as increase, decrease and level. When $d_{tVM}, d_{tPM} = increase$, the power consumption, Euclidean distance and temperature demand of VMs and PMs has increased compared to the preceding step. If $d_{tVM}, d_{tPM} = decrease$, then the power consumption, Euclidean distance and temperature demand of VMs and PMs has decreased when compared to the preceding time step. If $d_{tVM}, d_{tPM} = level$, then the traffic of VMs and PMs is the same level of demand in terms of power consumption, Euclidean distance and temperature demand as it was in the previous step.

Either migrates or wait is the action space a_t. When the RL agent selects the migrate action, then the selected VM by APO migrate to the selected destination PM. On the other hand, if the RL agent selects to wait then no migration will happen in that time-step. If the RL agent selects the action migrate, then the selected VM by APO migrate to the selected destination PM. After defining the multiple state space, optimal action is determined to perform when an agent is in a certain s_{tVM}, s_{tPM} and d_{tVM}, d_{tPM}. Through a reward function, the optimal action can be learned.

The optimal action can be learned through a reward function. The main intention of RL is to maximize the rewards through mapping states to actions. To maximize the rewards, a recurrent interaction at discrete time-steps between the cloud environment and agents is required. The RL agent obtains a representation of the cloud environment in the type of present state s_{tVM}, s_{tPM} and d_{tVM}, d_{tPM}. It permits the RL agent to choose and return an action a_t, according to the ϵ-greedy policy π it is following. The ϵ-greedy policy is utilized to ensure an agent can search the entirety of the multiple state-space based on the value of ϵ. The reward function $R(s_{tVM}, s_{tPM}, d_{tVM}, d_{tPM}, a_t)$ is found out as the action a_t, for the current $s_{tVM}, s_{tPM}, d_{tVM}, d_{tPM}$ state. Subsequently the environment returns a new representation of the future VM and PM state s_{tVM+1} and s_{tPM+1}. Then estimate future power consumption, Euclidean distance and temperature demand of VM and PM d_{tVM+1} and d_{tPM+1} while also getting a numerical reward r_t based on the preceding action a_t undertaken.

In the RL- Power consumption Traffic aware-Improved Resource Intensity Aware Load Balancing strategy, an RL agent receive a negative reward when it chooses to wait. This negative reward is assigned based on the time spent in an over-utilized state which described by the violations in the Service Level Agreement (SLA). An RL agent receives a negative reward when it chooses to migrate. This negative reward is assigned according to the total delayed time taken to migrate multiple VMs from source PM to destination PM.

Service Level Agreement Violation Reward

A Service Level Agreement Violation (SLAV) reward is calculated when the RL agent makes a decision to choose the action wait. It is computed by the total time a PM spend in an over-utilized state. The SLAV is calculated as,

$$SLAV = \sum_{h=1}^{o} SLAV(h_n) \quad (8)$$

In Eq. (8), h_n represents the current over-utilized state, the total time already spent in an over-utilized state is denoted as $SLAV$ and the current set of resources in PMs in an over-utilized state is denoted as o. Then the $SLAV$ reward is the ratio of the $SLAV$ and current weight direction w_{dt}. It ensuring convergence to an optimal π.

$$R_{SLAV} = \frac{w_{dt}}{SLAV} \quad (9)$$

Delayed Reward

When migrating VMs, a CloudSim's network flow model is considered as the delayed reward. It utilized point to point interaction for data from source s to destination d entity which is termed as the flow and it is denoted as $flow = sizeflow; s; d$, where $sizeflow$ denotes the number of bytes in the flow. The bandwidth that is available between two entities is denoted as bw, the latency is represented as $latency$, the duration of a single network flow is calculated which is given as follows:

$$delay(source, dest, VM_n) = latency + sizeflow/bw \quad (10)$$

In Eq. (10), VM_n is the current VM being migration from source PM $source$ to destination PM $dest$. The total migration time T_{MT} is calculated by accumulating all of the delay times for each VMs migrated from a particular PM.

$$T_{MT} = \sum_{v=1}^{n} delay(source, dest, VM_n) \quad (11)$$

The delay function computes the time of migration for the current VM_n to be migrated from source PM source to destination PM $dest$ and n is the set of VMs to be migrated.

After the calculation of T_{MT}, the T_{MT} is divided into the current weight w_{dt} to ensure a converges to an optimal π.

$$R_{Delay} = \frac{w_{dt}}{T_{MT}} \quad (12)$$

In the RL- PT-IRIAL, one over-utilized PM h_n is selected at each time-step, the state s_t is calculated and forwarded for migration. The agent decide their actions a_t as migrate or wait based on the current d_{tVM} and d_{tPM}. The reward r_t is calculated based on the following two scenarios.

Scenario 1: An agent will be punished according to the R_{SLAV} penalty when the agent decided to wait during the low network congestion. The agent would have been benefit to choose to migrate in this situation.

Scenario 2: If an agent makes a decision to migrate when the resources are low and network congestion is high, it will obtain a negative reward according to the R_{Delay}. Nevertheless in this scenario, should the agent decide to wait, it will obtain a reward based on the R_{SLAV} penalty, for the sustained time the PM is waiting for relief. In this scenario the optimal action is for the agent to wait.

Once an action is performed, the RL agent's q-value $Q(s_{tVM}, s_{tPM}, d_{tVM}, d_{tPM}, a_t)$ is calculated and the q-matrix is updated based on the states s_{tVM}, s_{tPM}, the VMs traffic d_{tVM} and PMs traffic d_{tPM} and the action a_t performed. By considering the current traffic, power consumption, Euclidean distance and temperature demand, the RL agent can decide the most opportune time to migrate a VM, to ensure it receives the greatest possible reward.

Reinforcement Learning- Optimized Power consumption Traffic aware-Improved Resource Intensity Aware Load Balancing Algorithm

1. For each selected PMs and VMs
2. Assign Action ← 0, 1
3. Select a_t from Actions using π
4. Perform action (wait or migrate)
5. Observe $future\ l_{xy}, future\ Pow_{xy}, future\ Temp_{xy}, future\ STR_{xy}, future\ l_{p,xy}, future\ Pow_{p,xy}, future\ Temp_{p,xy}, future\ STR_{p,xy},$ reward
6. Calculate Q

$$Q(s_{tVM}, s_{tPM}, d_{tVM}, d_{tPM}, a_t) \leftarrow Q(s_{tVM}, s_{tPM}, d_{tVM}, d_{tPM}, a_t) + \alpha\left[r_t + \gamma MaxQ\left((s_{tVM+1}, s_{tPM+1}, d_{tVM+1}, d_{tPM+1}, a_{t+1}\right) - QstVM,stPM,dtVM,dtPM,at \right. \quad (13)$$

7. Update $Q - Matrix$

IV. RESULTS AND DISCUSSION

The efficiency of existing Optimized PT-IRIAL (OPT-IRIAL) method and proposed RL-OPT-IRIAL method are tested in terms of communication cost reduction, number of migrations and performance degradation. A CloudSim simulator [12] is used in this experiment. The learning parameters α, γ and ϵ of RL are set as 0.8, 0.8 and 0.05 respectively.

A. Communication Cost Reduction

The communication cost reduction is the difference between the communication cost observed at a certain time point and the initial cost of al VMs.

Table 1 shows the comparison of communication cost reduction of OPT-IRIAL and RL-OPT-IRIAL methods under different timings.

Table 1: Comparison between OPT-IRIAL and RL-OPT-IRIAL in terms of communication cost reduction

Time (hrs)	OPT-IRIAL	RL-OPT-IRIAL
8	234	247
16	287	299
24	322	341

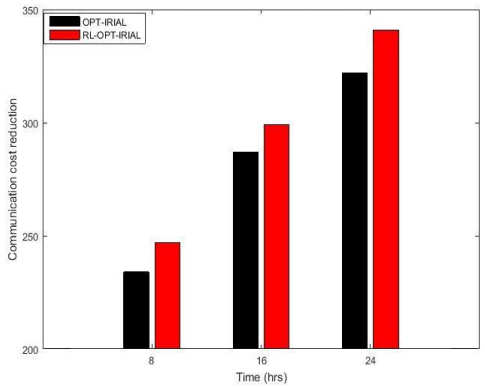

Fig.1: Comparison between OPT-IRIAL and RL-OPT-IRIAL in terms of communication cost reduction

Fig. 1 shows the comparison between OPT-IRIAL and RL-OPT-IRIAL in terms of communication cost reduction. The time in terms of hours is taken in X-axis and the communication cost reduction is taken in terms of Y-axis. When time is 24 hours, the communication cost reduction of RL-OPT-IRIAL method is 5.9% higher than OPT-IRIAL method. From this analysis, it is understand that the proposed RL-OPT-IRIAL has better communication cost reduction than the OPT-IRIAL.

B. Number of Migrations

The number of migration represents the migration of VMs at an optimal time from one PM to the selected destination PM.

Table 2 shows the comparison of number of migration by using OPT-IRIAL and RL-OPT-IRIAL methods for different number of VMs.

Table 2: Comparison between OPT-IRIAL and RL-OPT-IRIAL in terms of number of migrations

No. of VMs	OPT-IRIAL	RL-OPT-IRIAL
2500	0.42	0.31
3000	1.39	1.3
5000	1.64	1.57

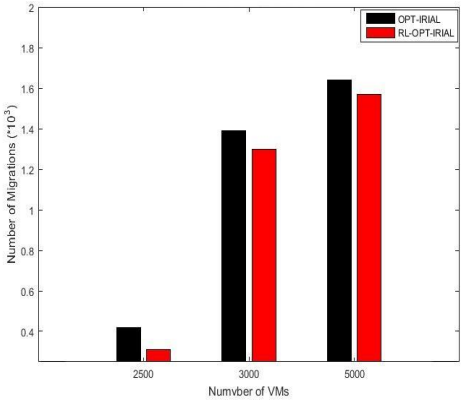

Fig.2: Comparison between OPT-IRIAL and RL-OPT-IRIAL in terms of number of Migration

Fig. 2 shows the comparison between OPT-IRIAL and RL-OPT-IRIAL in terms of number of migration. When the number of VMs is 5000, the number of migration of RL-OPT-IRIAL method is 4.3% less than OPT-IRIAL method. From this analysis, it is understand that the proposed RL-OPT-IRIAL has less number of migrations than the OPT-IRIAL.

C. Performance Degradation

When a VM is being migrated to selected destination PM at an optimal time, its performance is degraded. The performance degradation is calculated based on the migration of VMs. Table 3 shows the comparison of performance degradation by using OPT-IRIAL and RL-OPT-IRIAL methods for the different number of VMs.

Table 3: Comparison between OPT-IRIAL and RL-OPT-IRIAL in terms of performance degradation

No. of VMs	OPT-IRIAL	RL-OPT-IRIAL
2500	0.04	0.03
3000	0.5	0.42
5000	2.54	2.4

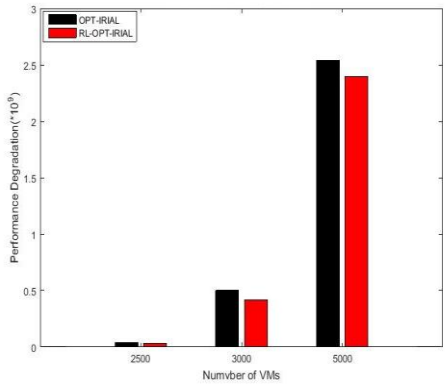

Fig.3: Comparison between OPT-IRIAL and RL-OPT-IRIAL in terms of performance degradation

Fig. 3 shows the comparison between OPT-IRIAL and RL-OPT-IRIAL in terms of performance degradation. When the number of VMs is 5000, the performance degradation of RL-OPT-IRIAL method is 5.5% less than OPT-IRIAL method. From this analysis, it is understand that the proposed RL-OPT-IRIAL has less performance degradation than the OPT-IRIAL.

V. CONCLUSION

In this paper, Reinforcement Learning- Optimized Power consumption Traffic aware-Improved Resource Intensity Aware Load balancing method is proposed to learn the most optimal time to migrate the selected VMs to the selected PMs. Based on this proposed RL-OPT-IRIAL method, load balancing problem in the cloud computing is solved. The RL-OPT-IRIAL learns the optimal time to migrate the selected VMs to the selected PMs based on the cloud environment. Hence, communication cost reduction is increased and the number of migration and performance degradation is reduced significantly. As a result, this method can be helpful in real time applications to balance the load in cloud computing.

REFERENCES

1. R. Gao and J. Wu, "Dynamic load balancing strategy for cloud computing with ant colony optimization", Future Internet, vol. 7, no. 4, pp. 465-483, 2015.
2. H. Shen, "RIAL: Resource intensity aware load balancing in clouds", IEEE Trans. Cloud Comput., vol. PP, no. 99, pp.1-14, 2017.
3. G. Dalin and V. Radhamani, "IRIAL-an improved approach for VM migrations in cloud computing", Int. J. Adv. Technol. Eng. Explor., vol. 5, no. 44, pp. 165-171, 2018.
4. V. Radhamani and G. Dalin, "PCA-TA-IRIAL: Power Consumption Aware- Traffic Aware- IRIAL a Novel Unified Approach for Green and Load Balanced Computing in Cloud", IEEE 3rd International Conference on Engineering and Technology (ICETECH'18). 2018.
5. V. Radhamani and G. Dalin, "Selection of migration VMs and destination PMs using an optimization algorithms in PCA-TA-IRIAL approach for green and load balanced cloud computing", ARPN J. Eng. Appl. Sci., in press.
6. M. Duggan, J. Duggan, E. Howley, and E. Barrett, "An autonomous network aware VM migration strategy in cloud data centres", in 2016 IEEE Int. Conf. Cloud Auton. Comput. (ICCAC), pp. 24-32, 2016.
7. M. Tarighi, S. A. Motamedi, and S. Sharifian, "A new model for virtual machine migration in virtualized cluster server based on fuzzy decision making", arXiv preprint arXiv:1002.3329, 2010.
8. B. Mondal, K. Dasgupta, and P. Dutta, "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach", Procedia Technol., vol. 4, pp. 783-789, 2012.
9. K. Dasgupta, B. Mandal, P. Dutta, J. K. Mandal, and S. Dam, "A genetic algorithm (ga) based load balancing strategy for cloud computing", Procedia Technol., vol. 10, pp. 340-347, 2013.
10. P. V. Krishna and L. D. D. Babu, "Honey bee behavior inspired load balancing of tasks in cloud computing environments", Appl. Soft Comput., vol. 13, no. 5, pp. 2292-2303, 2013.
11. M. Lawanyashri, B. Balusamy, and S. Subha, "Energy-aware hybrid fruitfly optimization for load balancing in cloud environments for EHR applications", Inform. Med. Unlocked, vol. 8, pp. 42-50, 2017.
12. R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms", Softw. Pract. exp., vol. 41, no. 1, pp.23-50, 2011.

AUTHORS PROFILE

V. Radhamani received her M.Sc., & M.Phil degree in Computer Science from Bharathiar University, Coimbatore. Currently, she is pursuing Ph.D in Bharathiar University. She is working as an Assistant Professor in Department of Decision and Computing Sciences at Coimbatore Institute of Technology, Coimbatore. She has published 4 papers in international journals and one paper in UGC journal. She has presented 3 papers in international conferences. She has completed NPTEL Online Certification course on Cloud Computing and attained Elite Topper Grade at Top 5% level out of 2303 candidates. She has also completed SuSE Linux Certification Courses NCLA with 100% of marks and NCLP with 99% of marks. Her area of interest is cloud computing, scheduling algorithms, evolutionary computing, and data analytics.

Dr. G. Dalin received his Ph.D in Computer Science in the year 2013. Currently, he is an Associate Professor of Computer Science at Hindustan College of Arts and Science, Coimbatore. He has a total experience of over 12 years. He has published more than 30 papers in international journals. He has acted as resource person in national and international conferences as well as chaired technical sessions. He is also a reviewer and editorial member in national & international journals. His area of interests include computer networks, routing algorithms and cloud computing.