Artificial Neural Network Based Load Balancing in Cloud Environment

Sarita Negi, Neelam Panwar, Kunwar Singh Vaisla and Man Mohan Singh Rauthan

Abstract With heavy demand for cloud technology, it is important to balance the cloud load to deliver seamless Quality of Services to the different cloud users. To address such issues, a new hybridized technique Artificial Neural Network based Load Balancing (ANN-LB) is introduced to calculate an optimized Virtual Machine (VM) load in cloud systems. The Particle Swarm Optimization (PSO) technique is used to perform task scheduling. The performance of the proposed ANN-LB approach has been analyzed with the existing CM-eFCFS, Round Robin, MaxMin, and MinMin algorithms based on MakeSpan, Average Resource Utilization, and Transmission Time. Calculated values and plotted graphs illustrate that the presented work is efficient and effective for load balancing. Hybridization of ANN and iK-mean methods obtains a proper load balancing among VMs and results have been remarkable.

Keywords Cloud computing · ANN · iK-mean · Clustering · Load balancing

1 Introduction

In the research computing world, (in 2000) a new trend Computing Technology has arrived to change the working approach of computer and Internet users. From the history of utility computing (or Computer utility, a computing resource package that includes computation, services, storage, and computer resource in rent), researchers

S. Negi (✉)
Uttarakhand Technical University, Dehradun 248007, Uttarakhand, India
e-mail: sarita.negi158@gmail.com

K. S. Vaisla
B.T.KIT, Dwarhat, Uttarakhand, India

S. Negi · N. Panwar · M. M. S. Rauthan
SOET, HNBGU, Srinagar, Garhwal 249161, Uttarakhand, India

M. L. Kolhe et al. (eds.), *Advances in Data and Information Sciences*,
Lecture Notes in Networks and Systems 94,
https://doi.org/10.1007/978-981-15-0694-9_20

have found that cloud is the most sophisticated, reliable, and service-oriented technology. *Abstraction* and *Virtualization* are the two major concepts of the cloud. In *Abstraction*, the cloud hides implementation information from the user as data stored in locations that are unknown and in *Virtualization*, resources are pooled and shared by various users on the metered basis [1].

Uncontrolled growth of DataCenters (DC) may lead to a lack of availability of resources. In such case, a load balancing policy can handle the cloud resources and make resources available to each Host and VM. Scheduling of tasks and load balancing in virtual machines (VMs) is the NP-hard problem hence a suitable scheduling technique is required to solve VM allocation problems [2]. The data that cloud provides to the user must be scheduled and balanced among VM and Hosts [3]. Live migration of VM is a better approach to slow down the processing cost [4] whereas cloud computational cost can be minimized through proper utilization of resources [5]. Utilization of resources may be achieved through proper mapping and allocation of tasks to VM and VM to Host [6]. Scheduling of tasks to single user and multi-user has different aspects of delay bounds for the tasks. Such delay bounds can be deduced by using offloading policy in edge cloud [7]. VM and task scheduling in cloud computing provide flexibility, scalability, load sharing, etc. Proper scheduling of resources improves load balancing such as VM migration [8, 9] and task migration [10, 11]. VM and task migration can have great impact on cloud performance. The various approaches of task migration techniques are non-live migration, post-copy, live migration, pre-copy, and triple TPM etc., [10]. VM live migration and its impact should be low as higher migration increases the processing cost [12].

Management of cloud resources demands the implementation of efficient load balancing techniques. Various categorizations of load balancing have been defined for the traditional computing environment: Static, Dynamic, and Mixed Load Balance [8]. As cloud provides the high mobility of nodes, the spatial distributed nodes need to be balanced using Centralized, Distributed, and Hierarchal Load Balancing methods [13]. This paper focuses on the concept of soft computing based technique, viz, Artificial Neural Network (ANN). The objective of the research is to introduce the process initiated by the VM manager using ANN-based Back Propagation Network (BPN) method. BPN calculates the load of each available VMs and improved K-mean (iK-means) clustering method performs clustering of VMs into underloaded VMs and overloaded VMs. Incoming user tasks that are admitted to cloud at runtime are allocated to underloaded VMs using the PSO algorithm.

The organization of the paper is as follows: Previous work on load balancing in the cloud environment has been discussed in Sect. 2. Section 3 highlights the proposed system model. Section 4 elaborates the implementation of the proposed model while Sect. 5 explains the evaluated outcomes of the proposed work. Finally, the conclusion and future scope are covered in Sect. 6.

2 Literature Review

Many researchers worked in load balancing to improve Quality of Service (QoS) of cloud computing. The researches have suggested different methods for load balancing. In this section, various load balancing algorithms are discussed in detail.

Hamsinezhad et al. [10] add up task and VM migration schemes to achieve efficient load balancing in a cloud environment. The work has shown the migration methods on task by combining Yu-Router and Post-Copy migration methods. The algorithm decreases the migration time, overhead, and transmitted data rate. The authors have explained migration in a mesh network that partitions the network into the subnetworks (P_{sub}). The migration of P_{sub} is given in Eq. (1). Number of stages (S) for the migration of subtasks to the D.M is calculated using Eq. (2).

$$p_{sub} = (d/p \times w/q \times h/r) \tag{1}$$

$$S = \max(d/p \times w/q \times h/r) \tag{2}$$

where the size of the network is represented by $(p \times q \times r)$ and $(d \times w \times h)$ is the number of nodes distributed on the network. The research work of [10] reduces task transmission time and data overhead but delay overhead is still a drawback of the algorithm. These drawbacks motivate to introduce a new approach that enhances the transmission time of tasks.

To minimize the workload between servers, an application live migration method has been introduced for large-scale cloud networks [14]. The application live migration takes place by three events, i.e., workload arrival, workload departure, and workload resizing (varying resource size). Li et al. [14] introduced a concept of workload in an encapsulation of application and the underlying operating system of VM. The server node that is running VM is referred to as open box and the server node lacking of VMs is referred to as close box. The arrival of workload is further assigned to an open box. The work shows "*how application (task) migration can perform remapping of workloads to the resource node*". This migration reduces the number of open boxes. The size of workload has been divided into subintervals 2 M-2 and is represented in levels. The approach seems to be energy efficient but large number of migrations can lead to high processing time. The use of three different algorithms, i.e., workload arrival, workload departure, and workload recycling may increase complexity of the network. The proposed approach reduces complexity by introducing supervised learning approaches for load balancing.

Devi et al. [15] introduced an Improved Weighted Round Robin (IWRR) load balancer where all the tasks are assigned to the VMs according to the IWRR scheduler. After completion of each task, the IWRR load balancer checks if there is a need for load balancing. If the number of tasks assigned to VM is higher, then the IWRR load balancer identifies VMs load. IWRR estimates the possible completion time of all

tasks assigned to that VM. The number of task migrations is significantly reduced in IWRR load balancer due to widespread identifying of the most suitable VM for each task. When overloaded VM drops below its threshold value, the task can be migrated from overloaded to underloaded VM. In order to identify the VMs having the highest and lowest load, load imbalance factor is calculated using the sum of loads of all VM, load per unit capacity (LPC), and threshold (T_i), which are defined in Eqs. (3), (4), and (5), respectively.

$$L = \sum_{i=1}^{k} l_i \tag{3}$$

$$LPC = \frac{L}{\sum_{i=1}^{m} c_i} \tag{4}$$

$$T_i = LPC \times C_i \tag{5}$$

where the number of VMs in a DataCenter (DC) is represented by i and C_i represents the node capacity. VM load imbalance factor is defined by

$$\text{VM load} \begin{cases} < T_i - \sum_{v=1}^{k} l_i, & \text{Underloaded} \\ > T_i - \sum_{v=1}^{k} l_i, & \text{Overloaded} \\ = T_i - \sum_{v=1}^{k} l_i, & \text{Balanced} \end{cases} \tag{6}$$

The drawbacks of the reviewed literatures motivate to introduce a new method of finding VM load using intelligence artificial neural network method. The obtained load is further clustered into underloaded and overloaded VMs; thus the tasks are assigned to underloaded VMs. The introduced model focuses to enhance resource utilization, transmission time, and makespan.

3 System Model and Proposed Work

3.1 Artificial Neural Network Based Load Balancing (ANN-LB)

In this system model, it is assumed that there is a set of a physical machines $PM = (PM_1, PM_2 \ldots, PM_M)$ where each PM holds the set of virtual machines $VM = (VM_1, VM_2 \ldots, VM_j)$. For the execution, a number of tasks $(t_1, t_2 \ldots, t_i)$ are assigned to VMs, respectively. VMs use their resources and run parallelly and independently. Load balancing has always been necessary to remove imbalance execution of a task.

The heavy load on the current VMs leads to unbalanced DCs and resource underutilization. Such issues can be resolved by introducing the clustering process on VMs in each PM based on current load of VMs. Based on the load, VMs are grouped.

3.1.1 Back Propagation Network (BPN)

The Back Propagation Network (BPN) is used to support several VMs all together for load calculation with the aim to reduce clustering time. A supervised learning Artificial Neural Network based BPN is one of the best neural approaches. Rumelhart, Hinton, and Williams introduced the BPN in 1986. It has the facility to propagate errors toward the back from the output layer units to the hidden layer units.

The role of BPN is to calculate the load of VMs which is further realized by improved K-means (i-Kmean) for VM clustering. In the first step of the algorithm, all VMs with their information are fed into the BPN to evaluate their current processing load on VMs. BPN algorithm performs weight calculation during the learning period of the network. It works on different phases: input A_i feed-forward, error back-propagation, and weight updation (v_{ij} and w_{jk}). The feed-forward phase is the testing phase of BPN in which a number of hidden layers are used in the network to achieve the desired output. It is important to train BPN for calculation of the VMs load. There are various learning factors and activation functions that are responsible to train BPN. The use of large number of weights in BPN may slow down the convergence of the network. Hence, a Momentum Factor (η) is used to save the previous information of weights for weight adjustment and for better solution. It enhances the weight updation stage and makes fast convergence. Equations (7) and (8) are the weight update expressions of the output layer units and the hidden layer units, respectively.

$$w_{jk}(t_k + 1) = w_{jk}(t_k) + \alpha(\delta_k)b_k + \eta\left[w_{jk}(t_k) - w_{jk}(t_k - 1)\right] \tag{7}$$

$$v_{ij}(t_k + 1) = v_{ij}(t_k) + \alpha\left(\delta_j\right)a_i + \eta\left[v_{ij}(t_k) - v_{ij}(t_k - 1)\right] \tag{8}$$

where w_{jk} is the output weight between jth hidden layer unit and kth output layer unit and, v_{ij} is the hidden weight between ith input layer unit and jth hidden layer unit.t_k is the targeted value of the network. To get trained output from the BPN, an activation function is used which increases monotonically. BPN mostly uses binary sigmoid function (or unipolar) that reduces computational burden during the learning process as defined in Eq. (9),

$$f(x) = 1/(1 + e^{-\lambda x}) \tag{9}$$

where λ is the steepness parameter.

The capacity of VM includes a number of processors, Million Instruction Per Second (MIPS) and bandwidth of that VM. The capacity and target (expected output) of BPN is calculated using the following expression:

$$C_{vmj} = \left(vm_{pj} \times vm_{mipsj} + vm_{bwj}\right)/100 \tag{10}$$

C_{vmj} is the capacity, $vm_{pj,}$ $vm_{mipsj,}$ and vm_{bwj} are the number of processors, MIPS, and bandwidth on jth VM, respectively. C_{vmj} is used as an initial weight (v_{ij}) on hidden layer units and calculated using Eq. (10). The initial load Il_{ij} on each VM denoted as $VML = VML_1, VML_2 \ldots VML_j$ is obtained by the summation of the total length of all tasks (task length as $TL = \{TL_1, TL_2 \ldots, TL_i\}$) on jth VM expressed in Eq. (11),

$$Il_{ij} = \sum_{1}^{i} TL_{ij} j \in VM,\ i \in task \tag{11}$$

$$Eo_k = \frac{\left(TL_{ij}/vm_{mipsj}\right)}{\left(\left(Il_{ij}/vm_{mipsj}\right)/\text{Total Number of VMs}\right)} \tag{12}$$

where TL_{ij} is the ith task length on jth VM. Eo_k is the expected (target) load of jth VM to update the network weights through errors expressed in Eq. (12). Algorithm 1 illustrates each step of BPN that calculates the optimized load of VMs.

ALGORITHM 1: Back Propagation Network

Step1: **Start** For each VM ($VM = VM_1, VM_2 \ldots VM_j$), receive vm_{pj}, vm_{mipsj}, $vm_{bwj,}$ and TL information.

Step2: Initialize input dimension, number of hidden units, number of output units, maximum epoch, learning rate (α), weight $W_{i,}$ and T_k.

Step3: Calculate v_{ij} using Eq. (10). Set $v_{oj} = 1$ and $w_{ok} = 1$. The weights on output layer w_{jk} are set as random values between 0.0 to 0.1.

Step4: Calculate hidden input from input layer units (A_i),

$$h_{inj} = V_{0j} + \sum_{i=1}^{n} \left(a_i v_{ij}\right) \tag{13}$$

$$h_j = F\left(h_{inj}\right) \tag{14}$$

where h_{inj} refers to the hidden input signal,v_{0j} is the bias weight to hidden layer, a_i is the ith input unit, v_{ij} is the input weight from ith input unit to jth hidden unit, and h_i is the output of the ith hidden unit using Eq. (9).

Step5: Calculate output from the hidden layer,

$$b_{ink} = w_{0k} + \sum_{i=1}^{p} \left(h_j w_{jk}\right) \tag{15}$$

$$b_k = F(b_{ink}) \tag{16}$$

where b_{ink} is the ouput layer input signal,w_{0k} is the bias weight to output layer, w_{jk} is the input weight from jth hidden unit to kth output unit, and b_k is the output of the kth output unit calculated using Eq. (9).

Step6: With the target pair E_{ok} as T_k from Eq. (12), compute error-correcting factor (δ_k) between output layer units and hidden layer units.

$$\delta_k = (t_k - (b_k))F(b_{ink}) \tag{17}$$

Binary sigmoid activation function from Eq. (9) is used to reduce the computational burden.

Step7: Calculate delta output weight Δw_{jk} and bias correcting Δw_{0k} terms,

$$\Delta w_{jk} = \alpha(\delta_k h_j) \tag{18}$$

$$\Delta w_{0k} = \alpha(\delta_k) \tag{19}$$

Step8: Calculate error terms δ_j between the hidden and input layer,

$$\delta_{inj} = \sum_{k=1}^{m} \left(\delta_k w_{jk}\right) \tag{20}$$

$$\delta_j = \delta_{inj} F\left(h_{inj}\right) \tag{21}$$

Step9: Calculate delta hidden weights Δv_{ij} and bias Δv_{0j} based on δ_j,

$$\Delta v_{ij} = \alpha\, \delta_j t_k \tag{22}$$

$$\Delta v_{0j} = \alpha\, \delta_j \tag{23}$$

Step10: Update output weight and bias unit using Eq. (7).
Step11: Update hidden weight and bias unit using Eq. (8).
Step12: If the specified number of epochs are reached or $B_k = T_k$, goto **Step13**, else goto **Step3**.
Step13: Calculated load of VM_j.
Step14: **End**

3.1.2 Improved K-Mean (iK-Mean)

The calculated load obtained from BPN forms the input to the iK-mean cluster algorithm. The algorithm uses minimum distance data points of the cluster center.

The procedure of the algorithm starts with an initial K cluster centers. The algorithm calculates the minimum distance to classify the nearest cluster center of all data points with each K selected number of centers. Next, the mean value of data to the center value is modified. The process repeats until new center value becomes equal to the previous center value. Finally, C_1 (underloaded cluster) and C_2 (overloaded cluster) are formed where $C_1 = VM_U = \{VM1_U, VM2_U \ldots VMi_U\}$ and $C_2 = VM_o = \{VM1_o, VM2_o \ldots VMi_o\}$.

3.1.3 Particle Swarm Optimization (PSO)

Incoming user tasks are allocated to underloaded VMs to maintain load balancing. Task scheduling is performed by the PSO algorithm. PSO is a biological concept based algorithm that is inspired by flocking of birds. A swarm-based intelligence algorithm approach uses self-additive global search technique to achieve optimized results. It initializes the particle (P). These are the potential solutions that move into the problem space by subsequent current optimum particles. In PSO, each P is represented by velocity and position which are obtained using Eqs. (24) and (25), respectively. Each P adjusts its velocity and position according to its best position and the position of the best particle (global best) in the entire population at each *k* iteration. Fitness value (FV) is the problem specific and used to measure the performance of a particle.

$$V_P[k+1] = w{*}V_P[k] + Y1 * rand_1 * (pbest - X_P[k]) + Y2 * rand_2 * (gbest - X_P[k]) \tag{24}$$

$$X_P[k+1] = X_P[k] + V_P[k+1] \tag{25}$$

where $V_P[k+1]$ and $V_P[k]$ is the present velocity and earlier velocity of P, respectively. $X_P[k+1]$ and $X_P[k]$ are existing and earlier locations of P. Two cognitive and social acceleration coefficients values are Y_1 and Y_2 respectively and $rand_1$, $rand_2$ (between 0 and 1) are used as self-regulating random numbers in the velocity computation. The population shows the particles in the search space. Best particle position in the population is denoted by pbest and gbest. The pbest is the best particle that has reached best outcome whereas best particle in global search space is denoted as gbest. w represents inertia weights that are used to balance best local and global search of particles. The value of w is a positive linear or nonlinear of time or a positive constant. Large w supports global exploration whereas small w supports local exploration. Particles are initialized randomly. The velocities synchronize P movement. At any point of time, the position of each P is influenced by its pbest in the search space. The whole working of the PSO algorithm is listed in ALGORITHM 2.

ALGORITHM 2: Particle Swam Optimization

Input: **Initialize required parameters**
User tasks T= {t_1, t_2, t_3……t_i},k=0 and Under loaded VM_U = {VM_{1U},VM_{2U}…..,VM_{iU}}
Output: Optimal mapping between T and VM_U.
begin
Set particle dimension ←number of tasks in T
Initialize particles randomly (velocity V_i, location X_i) with pbest and gbest
for each t_i∈ T
while (k<epoches)
Calculate FV for each particle and update X_i
if current FV<pbest
Assign pbest← X_i
else
Keep pbest
endif
for each pbest
Assign gbest← highest pbest
if present gbest< fitness value
Allot gbest← X_i
else
Keep previous gbest
endif
Assign VM_U to particle with highest gbest
Assign t_i to VM_U
t_i++
end while
end for
end

4 Implementation

The proposed Artificial Neural Network based Load Balancing (ANN-LB) algorithms have been implemented for improved cloud load distribution among underloaded VMs. CloudSim tool has been used to simulate algorithms. The implementation is performed on 2.20 GHz processor with 16 GB memory.

BPN algorithm is used for load calculation of each VM. To describe the theoretical functioning of the BPN, we include five different tasks that are allocated to five different VMs. The calculation is done for one epoch set to check the leniency of the BPN algorithm. Even if VMs and tasks may increase, the working procedure will remain the same. After performing BPN algorithm, the final calculated loads of VM_0, VM_1, VM_2, VM_3, and VM_4 are 0.199, 0.198, 0.203, 0.199, and 0.201, respectively. VM_i load obtained from BPN will form the input to iK-Mean algorithm to perform clustering between VM_1, VM2…,VM_i into clusters C_1 (Underloaded VMs) and C_2 (Overloaded VMs) $C_1 = \{0.199, 0.198, 0.199\}$ $C_2 = \{0.203, 0.201\}$. Once the clustered VMs are obtained, the PSO algorithm schedules the runtime tasks to the underloaded VMs. In the above-given case, initially we have five VMs and five tasks. The tasks in the dynamic time will be assigned to VMs that belong to C_2. The

PSO technique initializes particles. These particles are nothing but the tasks taken from the task list that are further assigned to the VMs. Using Eqs. (24) and (25), each particle is initialized randomly with their velocity and position, respectively. A particle represents allocation of user tasks to the VMs that have available resources. This method manages load among VMs and gives load sharing facility to the cloud environment.

5 Result and Discussion

In the result section, experimental observations are analyzed to examine the results of ANN-LB algorithm. The proposed methods have been compared with CM-eFCFS, RR (Round Robin), MaxMin, and MinMin algorithms. The illustration of cloud metrics and achieved results is analyzed in the following section:

MakeSpan (M) is the completion time of VM that can be calculated from the initial scheduling procedure time up to the final task completed. In this work, M is calculated using Eq. (26).

$$M = \max\left(CT_j\right) \tag{26}$$

This metric obtains the task completion time. The objective is to minimize the M to achieve faster execution. Figure 1 illustrates the total MakeSpan for CM-eFCFS, RR, MaxMin, and MinMin. Obtained results assured that introduced ANN-LB algorithm has achieved 16%, 28%, 28%, and 52% less MakeSpan than CM-eFCFS, RR, MaxMin, and MinMin, respectively.

Transmission Time (TX_T) is the time utilized to transfer the ith task (t_i) on VM_j. It is the ratio of ith task size and jth VM bandwidth. TX_T is calculated using Eq. (27).

$$TX_T = \frac{Size_i}{BW_j} \tag{27}$$

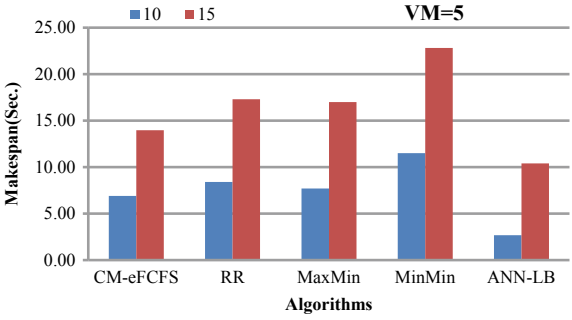

Fig. 1 Calculated MakeSpan for different algorithms

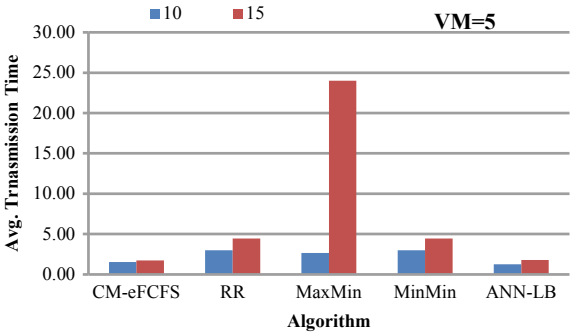

Fig. 2 Analysis on transmission time

where $Size_i$ refers to the ith task size and BW_j is the bandwidth of jth VM. Figure 2 clearly shows the graph representation of TX_T for CM-eFCFS, RR, MaxMin, and MinMin algorithms. The obtained result shows that the proposed algorithm maintains 1.16, 7.04, 1.41, and 7.04% less TX_T than CM-eFCFS, RR, MaxMin, and MinMin algorithms respectively which show better result.

Average Resources Utilization (AU) shows the efficiency of the system to utilize allocated cloud resources. Resource utilization should always be as high as possible to reduce wastage of resources. AU of an algorithm is evaluated using Eq. (28),

$$\mathrm{AU} = \sum\nolimits_{j \in \mathrm{VMs}} \mathrm{CT_j}/\mathrm{MakeSpan} \times \mathrm{NumberofVMs} \tag{28}$$

Figure 3 depicts the average resource utilization (AU). The obtained results clearly depict that ANN-LB algorithm attains higher utilization which is approximately 93% and 97% for 10 and 15 number of tasks, respectively. The obtained simulated results are better than CM-eFCFS, RR, MaxMin, and MinMin algorithms.

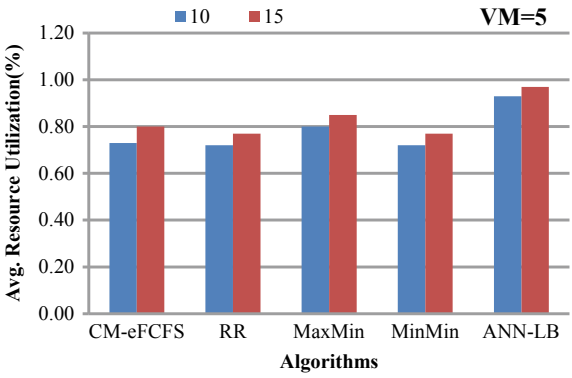

Fig. 3 Comparative analysis on average resource utilization

6 Conclusion

The well responsive load-balancing technique plays the key role for a cloud computing environment that brings improvement for dynamic cloud. This paper, introduced a hybrid approach of Artificial Neural Networking and Improved K-mean clustering-based technique to achieve load balancing (ANN-LB). The role of BPN is to train the system and to get an optimized load of VMs. These optimized loads of VMs are further clustered into underloaded and overloaded. Dynamic tasks are assigned to underload VMs using PSO-based task scheduling approach. We performed the implementation in CloudSim tool and found that the effort has shown improvement of cloud metric, i.e., Resource Utilization, Transmission Time and MakeSpan. The ANN-based BPN approach has been remarkable for dynamic cloud environment. In future, we are planning to expand the proposed work for the dynamic load balancing by taking other performance parameters.

References

1. Sosinsky, B. (2011). *Cloud computing Bible*. Wiley.
2. Shabeera, T. P., Madu Kumar, S. D., Salam, M. S., & Krishnan, K. M. (2016). Optimizing VM allocation and data placement for data-intensive applications in cloud using ACO metaheuristic algorithm. *Engineering Science and Technology, an International Journal*, *20*(2), 616–628.
3. Guo, L. (2012). Task scheduling optimization in cloud computing based on heuristic algorithm. *Journal of Networks*, *7*(3), 547–553.
4. Choudhary, A., Govil, M. C., Shingh, G., Aawasthi, L. K., & Pilli, E. S. (2017). A critical survey of live virtual machine migration techniques. *Journal of Cloud Computing: Advances, Systems and Application*, *6*(23), 1–41.
5. Li, T.,& Zhang, X. (2014). On the scheduling for adapting to dynamic changes of user task in cloud computing environment. *International Journal of Grid Distribution Computing*, *7*(3), 31–40.
6. Sharkh, M. A., Shami, & Ouda, A. (2017). Optimal and suboptimal resource allocation techniques in cloud computing data centers. *Journal of Cloud Computing: Advances, Systems and Applications*, Springer Open, *6*, 1–17.
7. Zhao, T., Zhou, S., Guo, X., & Niu, Z. (2017). Task scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing. In *SAC Symposium Cloud Communications and Networking Track IEEE ICC*.
8. Singh, A., Juneja, D., & Malhotra, M. (2015). Autonomous agent based load balancing algorithm in cloud computing. *International Conference on Advanced Technologies and applications (ICACTA), 45,* 823–841.
9. Mollamotalebi, M., & Hajireza, S. (2017). Multi-objective dynamic management of virtual machines in cloud environments. *Journal of Cloud Computing: Advances, Systems and Applications*, *6*(16), 1–13.
10. Hamsinezhad, E., Shahbahrami, A., Hedayati, A., Zadeh, A. K., & Banirostam, H. (2013). Presentation methods for task migration in cloud computing by combination of yu router and post-copy. *International Journal of Computer Science Issues (IJCSI)*, *10*(1), 98–102.
11. Pop, F., Dobre, C., Cristea, V., & Besis, N. (2013). Scheduling of sporadic tasks with deadline constrains in cloud environment. In *3rd IEEE International Conference on Advanced Information Networking and Application* (ICAINA), pp. 764–771.

12. Xiao, Z., Song, W., & Chen, Q. (2013). Dynamic resource allocation using virtual machines for cloud computing environment. *IEEE Transaction on Parallel and Distributed Systems, 24*(6), 1107–1117.
13. Katyal, M., & Mishra, A. (2013). A comparative study of load balancing algorithms in cloud computing environment. *International Journal of Distributed and Cloud Computing*, *1*, 5–14.
14. Li, B., Li, J., Huai, J., Wo, T., Li, Q. & Zhong, L. (2009). EnaCloud: An energy-saving application live placement approach for cloud computing environments. *International Conference on Cloud Computing. IEEE*, pp. 17–24.
15. Devi, D. C. & Rhymend Uthariaraj, V. (2016). Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks. In *Hindawi Publishing Corporation The Scientific World Journal*, pp. 1–14.