International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 18 (2018) pp. 14029-14036

Cloud Convolution Model for Dynamic Load Balancing Using Genetic Leaning of Neural Network

Sunita Gond[1], Shailendra Singh[2]

[1]*Barkatullah University Institute of Technology Bhopal, India.*

[2]*Senior member IEEE, National Institute of Technical Teachers' Training & Research, Bhopal, India.*

Abstract

With various services adopt by internet surfers, load on servers was also increases directly. So some smart approach is required to balance these requirements. In this work cloud based services which are dynamic in nature are so balanced that overall system performance get improved. Here use of convolutional model was done to train the neural network. So some of pre-processing step of input data to neural network was processed by convolutional model. This increase the learning of network. While this learned neural network was used to generate population of genetic algorithm, so less number of iteration was required to find a proper sequence of execution. Here because of twice changes strategy of TLBO which is a two phase genetic algorithm was used, as twice crossover operation was perform in one iteration. In order to validate proposed model original dataset from [1] was taken. Various evaluation parameters were used for the comparison of proposed model with other existing model.

Keywords: Crop yield prediction, Data mining, machine learning, Vegetation Index.

INTRODUCTION

Load balancing is the task to be distributed among various PCs, procedures, plate, or different assets keeping in mind the end goal to get ideal asset usage and to reduce the calculation time. Load balancing is an imperative means to accomplish effective asset sharing and use. It has been the hot issue of appropriated computing, network computing and distributed computing research. Load balancing has two implications: first, it puts an huge number of simultaneous access or information activity to numerous node separately to lessen the waiting time of clients for response. Second, it put the computation from a solitary over load to the different node to enhance the asset usage of every node [2].

Load balancing algorithms are utilized to perform the execution of programmers on available free resources or queued to the node as this reduced the execution time as well. These calculations are for the most part used to defeat the circumstance where a node is vigorously loaded, at the same time different node are remain idle (free) this lead to the demand fails [3, 4].

The rest of this paper are organized as follows: in the second section, related work of load balancing was discussed which was done by different researchers. Third section shows proposed working model of convolutional neural network model CNNM with genetic algorithm TLBO (Teacher learning based optimization). While fourth section experimental setup was discuss with dataset and results are compared with other model of SJF-MMBF, PSO, etc. The conclusion of the whole paper is made in the fifth section.

RELATED WORK

In [2], Sau-Ming Lau et al. have coordinated the two techniques of over burden need and light load need. They have proposed a versatile load appropriation calculation to successfully decrease correspondence overhead of the load adjusting process. Using the covetous calculations can take care of the issue of load circulation. Be that as it may, a few calculations above can't meet voracious decision execution and the idea of ideal sub-structure in the meantime. So these load dispersion approaches frequently acquire the nearby ideal arrangements. What's more, the impact of taking care of the issue of load dissemination under certain unique conditions isn't perfect. Cloud server farm can't achieve load balancing of the whole system.

Nguyen Khac Chien et al. (2016) [4] has proposed a load adjusting calculation which is utilized to upgrade the execution of the cloud condition in light of the strategy for assessing the finish of service time. They have prevailing with regards to upgrading the service time and reaction time of the client.

S.Yakhchi et al. (2015) [5] talks about that the vitality utilization has turned into a noteworthy test in cloud computing foundations. They proposed a novel power mindful load balancing strategy, named ICAMMT to oversee control utilization in cloud computing server farms. this work have abused the Imperialism Competitive Algorithm (ICA) for identifying over used has and after that this work relocate one or a few virtual machines of these hosts to alternate hosts to diminish their use. At last, this work consider different has as underutilized have and on the off chance that it is conceivable, move the greater part of their VMs to alternate has and change them to the rest mode.

Surbhi Kapoor et al. (2015) [6] goes for accomplishing high client fulfillment by limiting reaction time of the assignments and enhancing asset use through even and reasonable portion

of cloud assets. The conventional Throttled load balancing calculation is a decent approach for load balancing in cloud computing as it circulates the approaching job or task equitably among the VMs. Yet, the significant disadvantage is that this calculation functions admirably for conditions with homogeneous VMS, does not considers the asset particular requests of the assignments and has extra overhead of examining the whole series of VMs each time an task comes. The issues have been tended to by proposing a calculation Cluster based load balancing which functions admirably in heterogeneous node condition, considers asset particular requests of the assignments and decreases examining overhead by separating the machines into bunches.

Shikha Garg et al. (2015) [7] plans to circulate workload among numerous cloud frameworks or node to show signs of improvement asset use. It is the noticeable intends to accomplish proficient asset sharing and usage. Load balancing has turned into a test issue now in cloud computing frameworks. To meets the client's colossal number of requests, there is a need of appropriated arrangement on the grounds that essentially it isn't generally conceivable or taken a toll proficient to deal with at least one sit services. Servers can't be appointed to specific customers exclusively. Cloud computing contains a substantial system and parts that are available all through a wide region. Thus, there is a need of load balancing on its distinctive servers or virtual machines. They have proposed a calculation that spotlights on load balancing to lessen the circumstance of over-burden or under load on virtual machines that prompts enhance the execution of cloud significantly.

PROPOSED METHODOLOGY

In order to make a general model which work on various available data Indices a whole work is classify into two steps first was to pre-process data as per input environement than training of convolutional neural network model was done. While second is to find correct set of sequence by using CNNM and TLBO algorithm. In this work CNNM is used as the learning model where input data was processed fig. 1 steps. Here each block of final proposed method term as DLBCGM (Dynamic load Balancing by Convolutional and Genetic Model) was developed by fig. 2 diagram was explained by the various steps shown in this section.

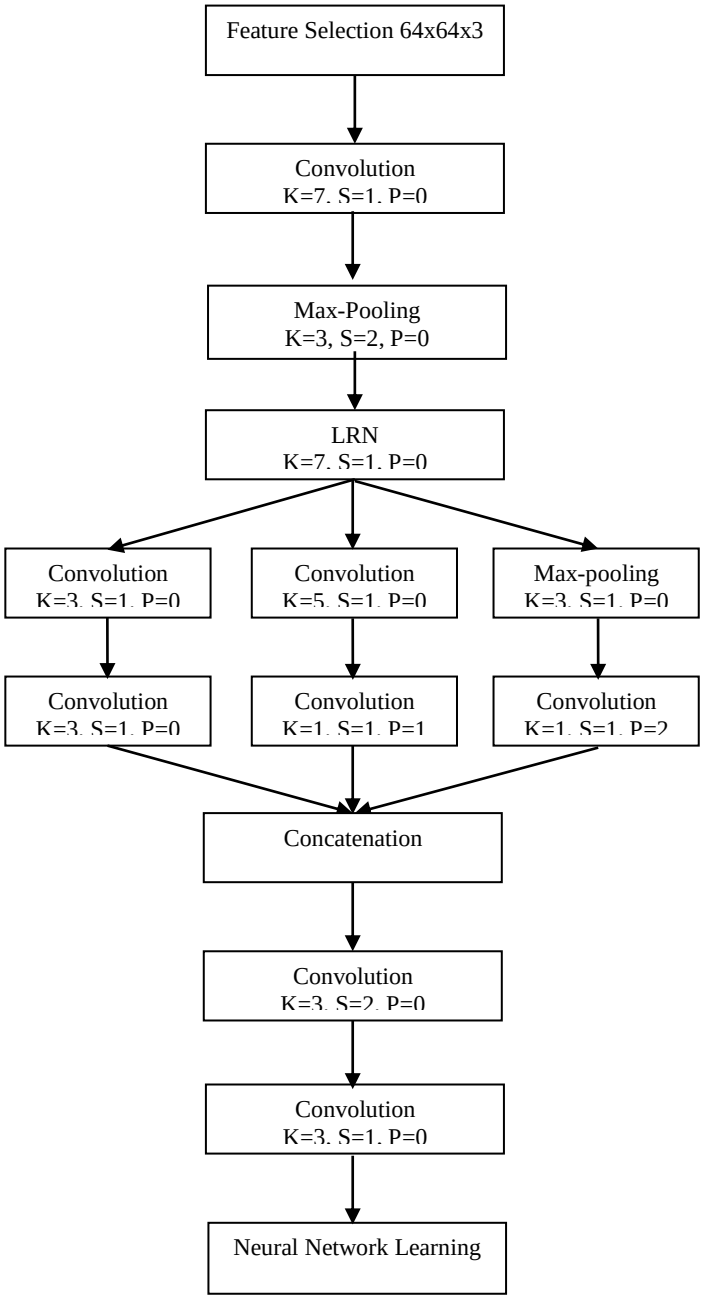

Figure 2. Represent Block Diagram of Proposed Work.

Convolutional Neural Network Model

Convolutional Neural Networks (**ConvNets** or **CNNs**) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. LeNet was one of the very first convolutional neural networks which helped propel the field of Deep Learning. This pioneering work by Yann LeCun was named LeNet5 after many previous successful iterations since the year 1988 [8]. There have been several new architectures proposed in the recent years which are improvements over the LeNet.

Convolution: ConvNets derive their name from the "convolution" operator. The primary purpose of Convolution in case of a ConvNet is to extract features from the input image. Convolution preserves the input matrix of job and its requirement. This work will go into the mathematical details of Convolution here, but will try to understand how it works over images. As discussed above, every job requirement can be considered as a matrix of values. Consider a 5 x 5 block B values are only 0 and 1. As input feature vector is passed from the canny algorithm that find edges in the images, so all edge portion is represent by 1 while non edge was represent by 0.

$$B=\begin{vmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{vmatrix}$$

$$F=\begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{vmatrix}$$

slide the F matrix over original block image by 1 pixel called 'stride' represent as s and for every position, compute element wise multiplication and add the multiplication outputs to get the final integer which forms a single element of the output matrix. Note that the 3×3 matrix F act as filter or kernel, where size of filter depends on k. In this case padding value p=0 is consider.

$$\begin{vmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{vmatrix} X \begin{vmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 4 & 3 & 4 \\ 2 & 4 & 3 \\ 2 & 3 & 4 \end{vmatrix}$$

Max-pooling: Pooling (also called subsampling or downsampling) reduces the dimensionality of each feature map but retains the most important information. Spatial Pooling can be of different types: Max, Average, Sum etc.

In case of Max Pooling, define a spatial filter kxk window and take the largest element from the rectified feature map within that window. In practice, Max Pooling has been shown to work better. Here shifting was done as per stride value s and padding will be done as per p value.

$$\begin{vmatrix} 1 & 1 & 2 & 4 \\ 5 & 6 & 7 & 8 \\ 3 & 2 & 1 & 0 \\ 1 & 2 & 3 & 4 \end{vmatrix} \rightarrow \begin{vmatrix} 6 & 8 \\ 3 & 4 \end{vmatrix}$$

In above matrix let k=2, so window of 2x2 move around the image block. While s=1 and p=0.

ReLu: ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

Steps of MCNN: Here whole model is divide into ten layes where first nine are various combination of convolution, ReLu and Max-pooling steps in each step fix set of stride, padding and window size fig.1 represent all working steps. Out of the last ninth layer of MCNN was pass in the final or tenth layer which adjust the weight value as per softmax function.

Fully Connected Layer

The Fully Connected layer is a traditional Multi Layer Perceptron that uses a softmax activation function in the output layer (other classifiers like SVM can also be used, but will stick to softmax in this post). The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer. I recommend reading this post if you are unfamiliar with Multi Layer Perceptrons.

The output from the convolutional and pooling layers represent high-level features of the input image. The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset.

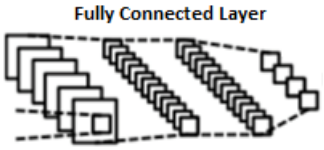

Fig. 3 Fully connected layer of modified convolutional network.

In order to understand above steps let us consider an example where W_{ij} have some weight values.

$$W_{ij} = \begin{vmatrix} W_{11} \\ W_{21} \\ W_{31} \end{vmatrix}$$

Let input vector of three value which include values from each input process for each machine. These values get multiply by above weight values.

Now this act as input $H1_{input}$ to next layer of hidden neurons. In this some biasing is also possible which was neglect in this example. So weight values of the neuron for next level is assumed as shown in below matrix.

$$W_{jk} = \begin{vmatrix} W`_{12} \\ W`_{22} \\ W`_{32} \end{vmatrix}$$

Where each value obtained from the previous weight matrix multiplication is passed through the soft-max function. Therefore small variation in the output value was done by this function.

$$SoftMax = e^{O_{ij}} \sum_{k=1}^{j} e^{O_k} ----eq(1)$$

So difference between the expected with obtained is consider as the error. This error need to be correct by adjusting the weight values of each layer. So here forward movement of the neural network is over and error back propagation starts by equation 2 and 3.

$$\frac{\partial E_i}{\partial O_i} = \frac{\partial(-1 * ((y_i * \log(O_i) + (1 - y_i) * \log(1 - O_i)}{\partial O_i} \quad Eq.(2)$$

$$\frac{\partial E_i}{\partial O_i} = (-1 * ((y_i * \log(O_i) + (1 - y_i) * \log(1 - O_i) \quad Eq.(3)$$

In similar fashion other values can be calculate to find other set of derivatives for sigmoid of equation 4. Here as per output derivative value may vary.

$$\frac{\partial O_i}{\partial H_i} = \frac{\partial(\frac{1}{1 + e^{-x}})}{\partial x}$$

$$= \left((1/(1 + e^{-x}) \times (1 - (1/(1 + e^{-x}))\right) --Eq.(4)$$

For each input to neuron let us calculate the derivative with respect to each weight. Now let us look at the final derivative by equation 5.

$$\sum_{i=1:n} \frac{\partial H_i}{\partial W_{i(j,k)}} = \frac{\partial(h_{i(ouput)} * W_{i(j,k)})}{\partial W_{i(j,k)}} --Eq.(5)$$

Now by using chain rule final derivate were calculated for the same. Here multiplication of output obtained from equation 3, 4 and 5 was done in following way:

$$\frac{\partial E_i}{\partial W_i} = \frac{\partial E_i}{\partial O_i} * \frac{\partial O_i}{\partial H_i} * \frac{\partial H_i}{\partial W_i} --Eq.(6)$$

So overall ∂W_i can be obtained by getting value of weight from above equation, here all set of weight which need to be update are change by below matrix values.

$$\partial W_i = \begin{bmatrix} \frac{\partial E_1}{\partial W_{1,2}} \\ \frac{\partial E_1}{\partial W_{2,2}} \\ \frac{\partial E_1}{\partial W_{3,2}} \end{bmatrix}$$

- So error corresponds to the input data was estimate by equation 8 differencing desired output obtain from output layer.

$$e_k(n) = d_k(n) - y_k(n) ---- Eq.(8)$$

- The ebpnn weight updation was done by equation 9 and ∂W_i was obtained by using equation 7.

$$w_{ij} = w_{ij} + \Delta w_{ij} ----Eq.(9)$$

- So end of above iteration steps over when error obtained from the output layer get nearer to zero or some constant such as 0.001.

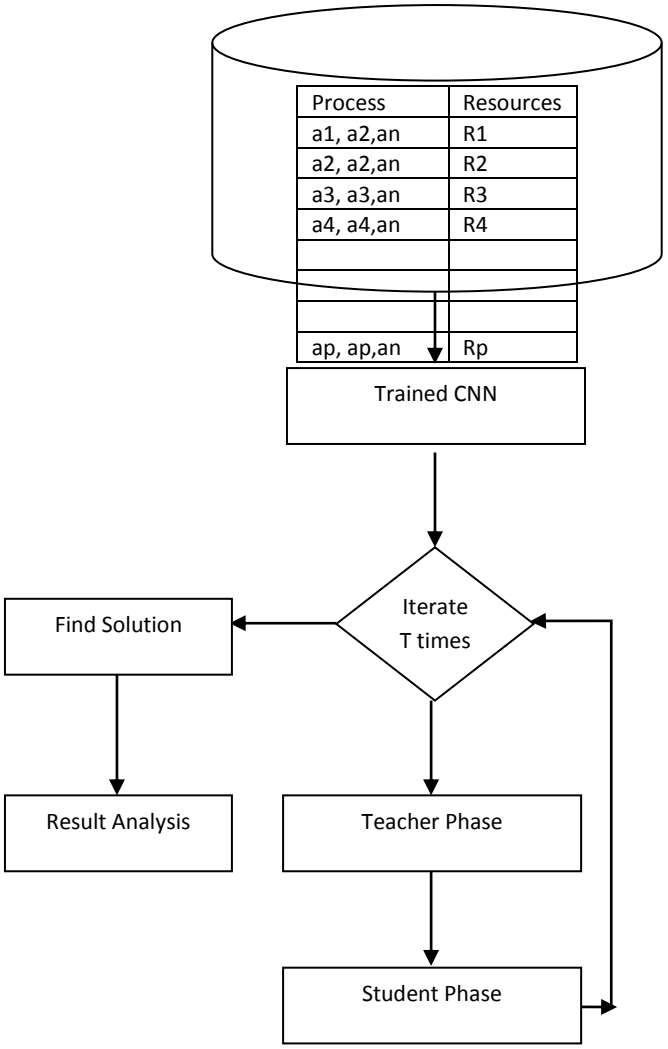

Figure 2. Block diagram of proposed model.

Pre-Processing

As the dataset available for processing is present in different file format so, some pre-processing steps are required for the conversion of data into experimental environment. In this work data is inform of vectors of the job timing for different machine. So reading of vectors in string form and conversion of those string in proper numeric value is done in pre-processing. Collection of all vector is done in a single matrix is also done here.

TLBO (Teacher Learning Based Optimization)

In this model TLBO (Teachers Learning Based Optimization Algorithm was used for assigning the incoming process to respected machine as per requirement. In this work genetic algorithm TLBO is use because this takes two phase learning. Main motive of this model is to reduce the dataset size and increase the leaning accuracy of the neural network. Here iteration of teacher and student phase was done while two similar solutions were not obtained.

Generate Population: Here assume some possible solution set that are the combination of all the job which represent there execution sequence. This is generate by the random function shown in eq. 10. This can be understand as let the number of job be n and number of initial population is IP, then one of the possible solution is Cc ={J_1, J_5, J_7…..J_m} this can be assume as the solution set. While P = [Cc1, Cc2,……Ccn] is an population obtain randomly, where m is number of jobs and n is number of chromosomes.

$$P \leftarrow Rand(m, n) \text{---------eq (10)}$$

Fitness Function: In order to obtain good chromosome from the bunch of available set fitness value of each probable solution set is passed in this function. So fitness value was returned. So makeSpan time is estimate as shown in equation (11) was used for finding the fitness value. This can be understand as let solution set Cc fitness value need to calculate. Than time taken by all machine to execute the each job in the batch is total MakeSpan time. So sum of all job execution time is term as the probable solution fitness value shown in equation (11).

$$J_{max}=Max\_Execution\_Time \{ J_1, J_2, J_3,\ldots\ldots\ldots J_n\} \text{---(11)}$$

Teacher Phase: This phase was used for the crossover of the chromosomes by the single best solution from the population. Here best solution act as a teacher and its selection is based on the minimum fitness value. In order to do crossover operation random position probable solution value is copied from the teacher chromosome and it was replaced to the non teacher chromosome. This improves the population quality. This can be understood as let best solution is Cc_b than crossover operation done by equation 12.

$$Cc[m, r] \leftarrow Cc_b[b, r] \text{ where } r=[1\ldots n] - - - - - eq(12)$$

Student Phase: In this phase some random group of chromosome were made automatically and then each group was used for the crossover of the chromosomes by the single best solution in that group. Here best solution act as a teacher among other chromosomes and its selection is based on the minimum fitness value. In order to do crossover operation random position probable solution value is copied from the teacher chromosome and it was replaced to the non teacher chromosome using equation 12. Here each new chromosome was cross verified that either its fitness value improved then previous, if fitness improves than new chromosome is included in the population and older one get removed. Vice versa if fitness value not improves.

Training Vector

Here output of genetic algorithm was used as the outpur desired position of input process requirement vector. Here this combination of random input requirement and output is learn by neural network. So input and output parameters are combine to generate a training data for the Error Back Propagation Neural Network.

Testing of (DLBGNA)

Above trained neural network was used in DLBGNA where input process with there requirement of different machine is pass in the trained neural network which generate the population. For the Genetic algorithm as this population was depend on the previous experience of genetic algorithm so result to find the best sequence for process execution get high. As instead of eq. 13 used in generate population section work utilized

$$P \leftarrow Tained\_CNMN(m, n, D) \text{----------eq.(13)}$$

So difference between the expected with obtained is consider as the error. This error need to be correct by adjusting the weight values of each layer. So here forward movement of the neural network is over and error back propagation starts.

EXPERIMENT AND RESULTS

In order to conduct experiment and measure evaluation results MATLAB 2012a version software is use. This section of paper show experimental setup and results. The tests were performed on an 2.27 GHz Intel Core i3 machine, equipped with 4 GB of RAM, and running under Windows 7 Professional. Here comparison of proposed model DLBCGM was done with exsisting PSO [9], SJF-MMBF [10] and one more method which was term as DLBGNA (Dynamic Load Balancing by Genetic and Neural Apporach), its an combination of EBPNN and TLBO method. Here major difference between DLBCGM and DLBGNA was that in DLBCGM convolutional steps were added.

Dataset

Table 1. Description of dataset with attributes [5].

Attributes	Set1	Set2
Number of Jobs	20	20
Number of Machines	10	5
Sample Size	20x10	20x5

Sample of number of jobs 15, number of machines 5, upper bound 1278 and lower bound 1232, processing times:

27 92 75 94 18 41 37 58 56 20 2 39 91 81 33 14 88 22 36 65

79 23 66 5 15 51 2 81 12 40 59 32 16 87 78 41 43 94 1 93

22 93 62 53 30 34 27 30 54 77 24 47 39 66 41 46 24 23 68 50

93 22 64 81 94 97 54 82 11 91 23 32 26 22 12 23 34 87 59 2

38 84 62 10 11 93 57 81 10 40 62 49 90 34 11 81 51 21 39 27

Evaluation Parameter

Makespan is defined as the time required for processing all thejobs or the maximum time required for completing a given set of jobs. Minimization of makespan ensures better utilization of the machines and leads to a high throughput [7].

$$J_{max}=\text{Max}\ \{ J_1, J_2, J_3,\ldots\ldots\ldots J_n\}$$

Total flowtime is defined as the sum of completion time of every job or total time taken by all the jobs. Totalflowtime of the schedule is computed using equation [8]:

$$F = \sum_{i}^{n} J_i$$

Completion Time variance is defined as the variance about the mean flowtime and iscomputed using equation [9]:

$$V = \frac{1}{n}\sum_{1}^{n}(J_i - \bar{F})^2$$

Where $\bar{F}$ is the mean flowtime.

Relative Percent Deviation (RPD)

$$RPD = \left[\frac{G - C^*}{C^*}\right] \times 100$$

where, G represents the global best solution obtained by the proposed algorithm for a given problem and C^*represents the upper bound value.

RESULTS

Table 2. Total Flow time Value

Techniques	Set 1	Set 2
PSO [9]	24581	17384
SJF-MMBF[10]	28434	24775
DLBGNA	17243	16598
DLBCGM	15506	13870

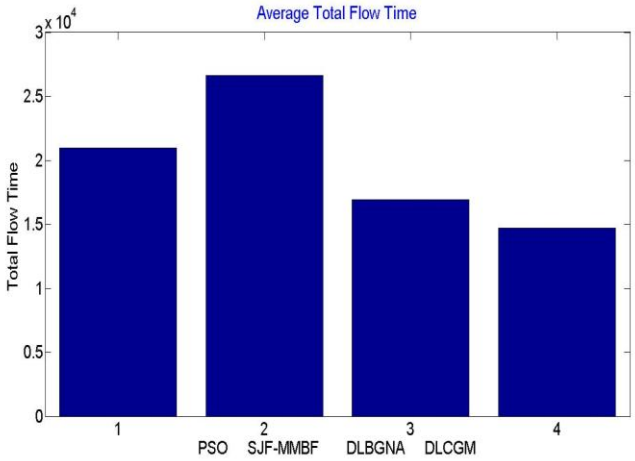

Figure 3. Average Total Flow Time comparison.

From table 2 and fig. 3 it was obtained that proposed work has reduce the total flow time of the dynamic load balancing of input testing dataset. This was done because of use of two tier population updating. Here in single iteration has improved work probable solution twice. In PSO learning or population updating was done once in a iteration. While in [10] SJF-MMBF shortest job sequence engages one machine when some part of job is time taken to complete the job. While SJF-MMBF is better as compared to approaches in [11-14].

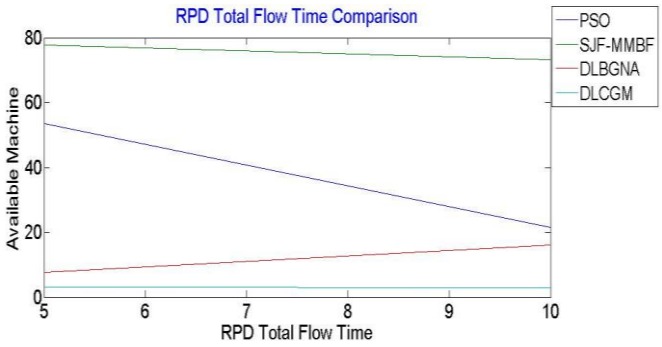

Figure 4. Comparison of RPD Total Flow Time for different methods.

Table 3. RPD comparison of Total Flow time Value.

Techniques	Set 1	Set 2
PSO [9]	53.63	21.5664
SJF-MMBF [10]	77.7125	73.2517
DLBGNA	7.768	16.0699
DLBCGM	3.087	3.0069

From Fig. 4 and table 2 it was obtained that proposed work has reduce the RPD total flow time of the dynamic load balancing of input testing dataset. This was done because of use of two tier population updating. Here in single iteration has improved work probable solution twice.

Table 3. Comparison of Completion Makespan time

Techniques	Set 1	Set 2
PSO [9]	1949	1493
SJF-MMBF [10]	1672	1523
DLBGNA	1684	1242
DLBCGM	1541	1154

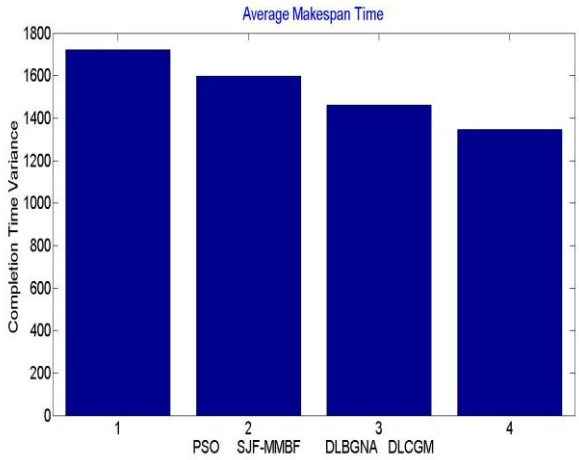

Figure 5. Average Completion Time Variance comparison.

From table 3 and fig. 5 it was obtained that proposed work has reduce the competition variance time of the dynamic load balancing of input testing dataset. This was done because of EBPNN for generating initial population as this generate relevant probable solution as per prior experience. As same kind of jobs may get execute on the machine as a trending way. While in [10] SJF-MMBF shortest job sequence engage one machine when some part of job is time taken to complete the job. In PSO learning or population updating was done once in a iteration.

Table 4. Comparison of RPD for MakesSpan

Techniques	Set 1	Set 2
PSO [9]	23.1985	29.826
SJF-MMBF [10]	5.689	32.434
DLBGNA	6.4475	8
DLBCGM	2.5916	0.3478

From table 4 it was obtained that proposed work has reduce the total flow time of the dynamic load balancing of input testing dataset. This was done because of use of two tier population updating. Here in single iteration has improved work probable solution twice. In PSO learning or population updating was done once in a iteration. While in [10] SJF-MMBF shortest job sequence engages one machine when some part of job is time taken to complete the job.

Table 5. Comparison of RPD for Total completion Time

Techniques	Set 1	Set 2
PSO [9]	30.838	37.629
SJF-MMBF [10]	79.6415	37.8325
DLBGNA	31.9973	32.4574
DLBCGM	16.9245	2.5399

From table 5 it was obtained that proposed work has reduce the competition variance time of the dynamic load balancing of input testing dataset. This was done because of EBPNN for generating initial population as this generates relevant probable solution as per prior experience. As same kind of jobs may get execute on the machine as a trending way. While in [10] SJF-MMBF shortest job sequence engages one machine when some part of job is time taken to complete the job. In PSO learning or population updating was done once in a iteration.

CONCLUSIONS

Dynamic Load balancing improved the cloud efficiency to manage number of jobs or services at same instant. So researcher proposed various approaches to handle this issue by soft computing technique like PSO, TLBO, etc. In this work dynamic load was balance by inserting the convolutional learning model which help TLBO algorithm to generate good set of population. Here convolutional step improved the cloud performace on different measures like makesan, completion time, etc. Experiment was done on real dataset with upper bound values from [1]. Here result shows that proposed work has improve the makespan accuracy by by 1.469%. Here use of proper training and rich input vector resultant neural network is less time consuming model. It was obtained that proposed work has reduced the Total Flow time by 3.046% as compared to other existing work, while completion time variance RPD was also get reduced by 9.732%. Here overall accuracy of the propose work was also improved.

REFERENCES

[1]. Taillard, E.. Benchmarks for basic scheduling problem, European Journal of Operational Research, Vol.64, 278-285.1993

[2]. V. Shrivastava, P. Zerfos, K. W. Lee, H. Jamjoom, Y. H. Liu, and S. Banerjee, ”Application-aware Virtual Machine Migration in Data Centers,” Proc. IEEE INFOCOM, pp. 66-70, 2011.

[3]. Byung Chul Tak, Youngjin Kwon, and Bhuvan Urgaonkar. “Resource Accounting of Shared IT Resources in Multi-Tenant Clouds”. 10.1109/TSC.2015.2453980, IEEE Transactions on Services Computing

[4]. N. K. Chien, N. H. Son and H. D. Loc, "Load Balancing Algorithm Based on Estimating Finish

Time of Services in Cloud Computing," ICACT, pp. 228-233, 2016.

[5]. S. Yakhchi, S. Ghafari, M. Yakhchi, M. Fazeli and A. Patooghy, "ICA-MMT: A Load Balancing Method in Cloud Computing Environment," IEEE, 2015.

[6]. S. Kapoor and D. C. Dabas, "Cluster Based Load Balancing in Cloud Computing," IEEE, 2015.

[7]. S. Garg, R. Kumar and H. Chauhan, "Efficient Utilization of Virtual Machines in Cloud Computing using Synchronized Throttled Load Balancing," 1st International Conference on Next Generation Computing Technologies (NGCT-2015), pp. 77-80, 2015.

[8]. Seyedeh Leili Mirtaheri, Seyed Arman Fatemi, Lucio Grandinetti.. "Adaptive Load Balancing Dashboard in Dynamic Distributed Systems". The Authors 2017. This paper is published with open access at SuperFri.org.

[9]. LI CHUNLIN, ZHOU MIN AND LUO YOULONG. "Efficient Load-Balancing Aware Cloud Resource Scheduling for Mobile User". Computer And Communications Networks And Systems The Computer Journal, 2017

[10]. Ryoichi Kawahara, "An adaptive load balancing method for multiple paths using flow statistics", Teletraffic Science and Engineering, Volume 5, Issue null, 2003, Pages 301-310.

[11]. Dhinesh Babu L.D, P. VenkataKrishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments", Applied Soft Computing 13 (2013) 2292–2303.

[12]. Ruchika Aggarwal, Latika Gupta . "Load Balancing In Cloud Computing". ISSN 2320–088X IMPACT FACTOR: 6.017 IJCSMC, Vol. 6, Issue. 6, June 2017, pg.180 – 186.

[13]. B. Godfrey, K. Lakshminarayanan, S. Surana, R. Karp, I. Stoica, "Load balancing in dynamic structured P2P systems", in: INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, vol. 4, IEEE, 2004, pp. 2253–2262.

[14]. Lei Yu, Liuhua Chen, Zhipeng Cai, Haiying Shen, Yi Liang, Yi Pan. "Stochastic Load Balancing For Virtual Resource Management In Datacenters". IEEE Transactions On Cloud Computing, Vol. , No. , November 2014.