. Dynamic Task Scheduling Using Trained Neural Network and Genetic Algorithm

Suhani Kumari
kumarisuhani316@gmail.com
Department of Computer Science & Engineering
RITS, Bhopal

Himanshu Yadav
himanshuyadav86@gmail.com
Department of Computer Science & Engineering
RITS, Bhopal

Chetan Agrawal
chetan.agrawal12@gmail.com
Department of Computer Science & Engineering
RITS, Bhopal

Abstract
This paper focus on allocation of cloud resources where two models were developed for this work. First was TLBO (Teacher Learning Based Optimization) genetic algorithms which find the correct position for the process to execute. Here some information used for analysis are total number of machines, memory, execution time, etc. So, outcome of the selected training process sequence were used as training input to the Convolutional Neural Network for learning. Here training was done in such a way that all set of features were utilized in pair with their process requirement and current position. For increasing the reliability of the work whole experiment was done on real dataset. Result shows that proposed GNNLB model has overcome various evaluation parameters on different scale as compared to previous approaches adopt by researchers.

Keywords:
Cloud Computing, genetic algorithm, Load balancing, neural network, Virtual machines.

1. Introduction

As cloud computing is growing vastly and more services and better results are demanded by the clients, so for the cloud, load balancing has become a very interesting and important research area. The area of Cloud computing is getting more hot, at the same time, a more intensive task waiting to be processed, how to allocate cloud tasks reasonably so that the nodes in the cloud computing environment can have a balanced load become more critical, this task allocation strategy is called load balancing. Load balancing has a significant influence on the performance in cloud computing as load balancing aims to enhance resource consumption, get the most out of throughput, reduce response time, and avoid overload of any single resource.

Better load balancing makes cloud computing more efficient and improves user satisfaction. Therefore, "it is the process of confirming the evenly distribution of work load on the pool of system node or processor so that the running task is accomplished without any disturbance". The objectives of load balancing are to maintain the stability of the system, improves the performance, build the system which is fault tolerance and provide future variation in the system such as security updates, releasing up customers time and resources for further tasks as well. Cloud load balancing is a type of load balancing that is executed in cloud computing which can be completed individually as well as on grouped basis. There are various algorithms designed for balancing the load among different tasks. After completing the literature survey, it can be conclude that most of the load balancing algorithms suggested so far are complex. In Round robin scheduling algorithm method, it considers only current load on each virtual machine. This is static method of load balancing, static load balancing method offer simplest simulation and checking of environment but failed to model heterogeneous nature of cloud.

The rest of this paper is organized as follows: in the second section, the type of load balancing was discussed which was broadly classified as static and dynamic load balancing. Third section list various techniques proposed to handle this problem. While fourth section provide related work of the current approaches achieving load balancing of cloud data centers will be introduced briefly. Research problem is pointed out, and then the proposed problem is formalized in detail.
The conclusion of the whole paper is made in the sixth section.

2. Related Work

Song Ningninget. Al. In [2] Fog computing can improve the resource utilization efficiency of the edge device, and solve the problem about service computing of the delay-sensitive applications. This paper researches on the framework of the fog computing, and adopts Cloud Atomization Technology to turn physical nodes in different levels into virtual machine nodes. On this basis, this paper uses the graph partitioning theory to build the fog computing's load balancing algorithm based on dynamic graph partitioning. The simulation results show that the framework of the fog computing after Cloud Atomization can build the system network flexibly, and dynamic load balancing mechanism can effectively configure system resources as well as reducing the consumption of node migration brought by system changes.

Liang Yu, et. al. in [3] paper, intend to reduce the operational cost of cloud data centers with the help of fog

Manuscript received June 5, 2025
Manuscript revised June 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.6.18

devices, which can avoid the revenue loss due to wide-area network propagation delay and save network bandwidth cost by serving nearby cloud users. Since fog devices may not be owned by a cloud service provider, they should be compensated for serving the requests of cloud users. When taking economical compensation into consideration, the optimal number of requests processed locally by each fog device should be decided. As a result, existing load balancing schemes developed for cloud data centers cannot be applied directly and it is very necessary to redesign a cost-ware load balancing algorithm for the fog cloud system. To achieve the above aim, we first formulate a fog-assisted operational cost minimization problem for the cloud service provider. Then, we design a parallel and distributed load balancing algorithm with low computational complexity based on proximal Jacobin alternating direction method of multipliers.

Huang Peng et. Al. in [4] The description of computational resources and their optimal allocation among tenants with different requirements holds the key to implementing effective software systems for such a paradigm. To address this issue, a systematic framework for monitoring, analyzing and improving system performance is proposed in this research. Specifically, a radial basis function neural network is established to transform simulation tasks with abstract descriptions into specific resource requirements in terms of their quantities and qualities. Additionally, a novel mathematical model is constructed to represent the complex resource allocation process in a multi-tenant computing environment by considering priority-based tenant satisfaction, total computational cost and multi-level load balance. To achieve optimal resource allocation, an improved multi-objective genetic algorithm is proposed based on the elitist archive and the K -means approaches.

Dazhao ChengGongz et. Al. in [5] paper, we observe that the homogeneous configuration of tasks on heterogeneous nodes can be an important source of load imbalance and thus cause poor performance. Tasks should be customized with different configurations to match the capabilities of heterogeneous nodes. To this end, we propose a self-adaptive task tuning approach, Ant that automatically searches the optimal configurations for individual tasks running on different nodes. In a heterogeneous cluster, Ant first divides nodes into a number of homogeneous sub clusters based on their hardware configurations. It then treats each sub cluster as a homogeneous cluster and independently applies the self-tuning algorithm to them. Ant finally configures tasks with randomly selected configurations and gradually improves tasks configurations by reproducing the configurations from best performing tasks and discarding poor performing configurations. To accelerate task tuning and avoid trapping in local optimum, Ant uses genetic algorithm during adaptive task configuration.

Problem Identification

- In previous paper reinforced learning was used which required to calculate feasible solution at first.
- Use of SJF increase the starvation in the system, as some algorithm get high weighting time.
- Dynamic load balancing makespan time increases as each section was treat individually.
- System does not learn previous similar type of sequence pass by the algorithm.

3. 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 environment than training of convolutional neural network model was done. While second is to find correct set of sequence by using CNN and TLBO algorithm. In this work CNN was used as the learning model where input data was processed fig. 1 steps.

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 pattern recognition and classification. As input job matrix is convert into square matrix and passed from the canny algorithm that find edge portion where 1 is represent by edge while non-edge was represent by 0. Fig. 1 represent block diagram of CNN model. Input of matrix and output of matrix at different level of block diagram can be obtained by below formulas:

$$R' = \frac{(R - K + 2 * P)}{S} + 1$$
$$C' = \frac{(C - K + 2 * P)}{S} + 1$$

Where R is number of row for input matrix at any level and R' is number of row of output matrix at that level. In similar way C is number of row for input matrix at any level and C' is number of row of output matrix at that level, k act as filter or kernel, p is padding and s is stride where various block steps are explained below:

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.

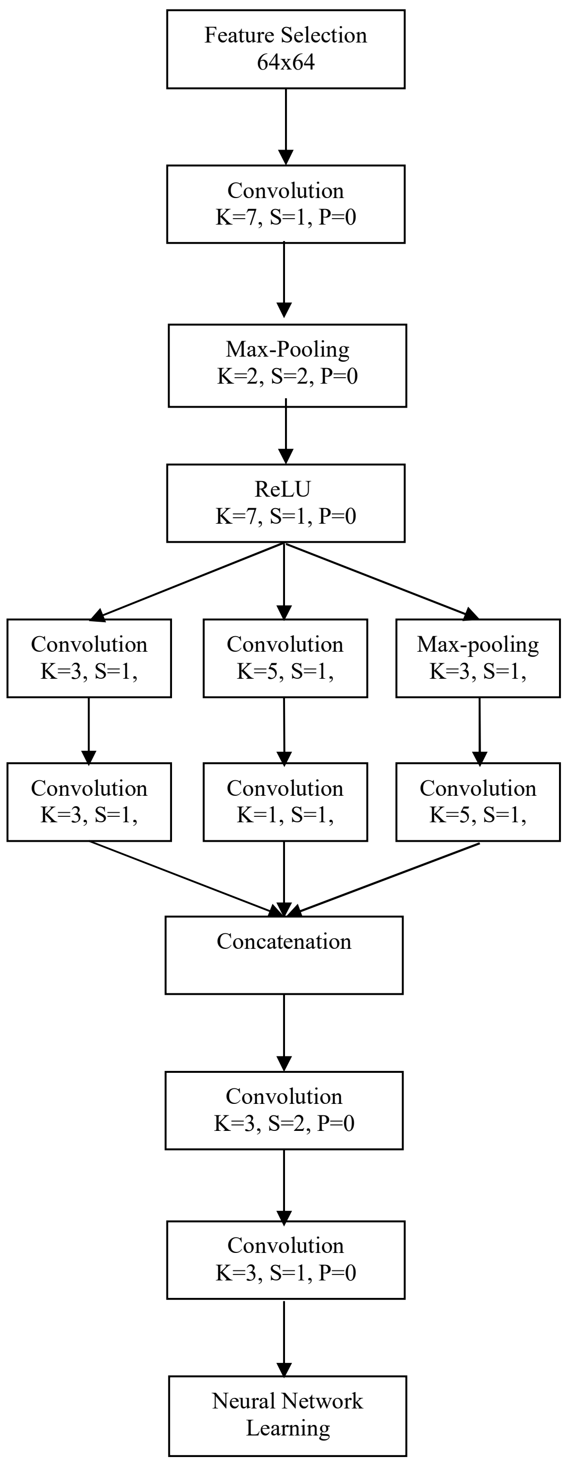

Fig. 1 Represent Block Diagram of GNNLB.

Stride: 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.

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

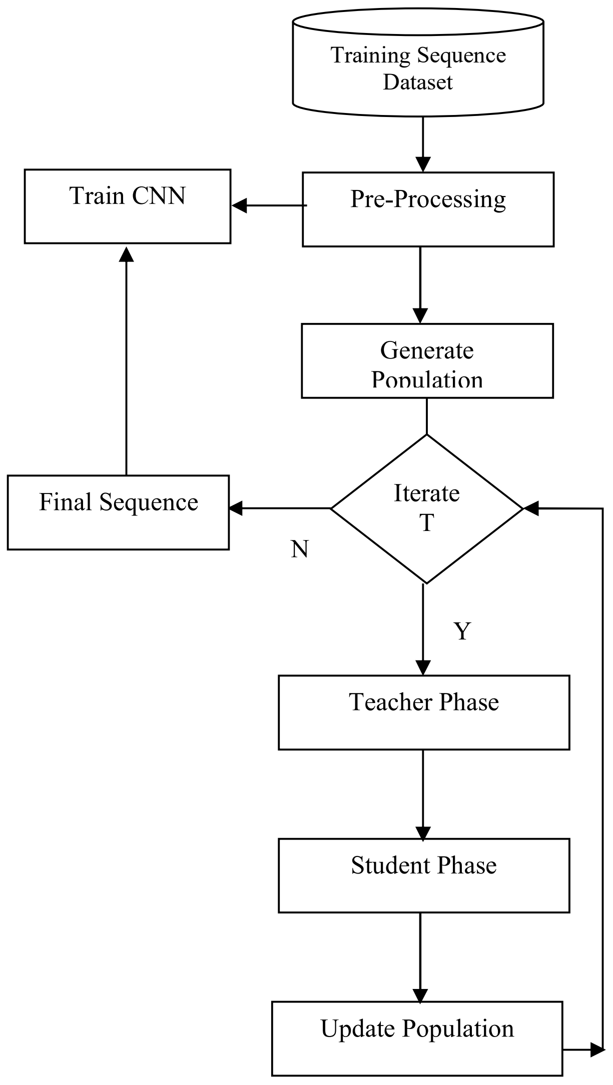

Fig. 2 Flow Chart of proposed model.

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.

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 layers 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.

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 strings 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 the entire job which represent there execution sequence. This is generating by the random function.

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

Where m is number of jobs and n is number of chromosomes.

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 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.

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

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.

$$Cc\,[m, r] \leftarrow Cc_b\,[b, r] \text{ where } r=[1\ldots n]$$

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. 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 output desired position of input process requirement vector. Here this combination of random input requirement and output is learnby neural network. So input and output parameters are combine to generate a training data for the Error Back Propagation Neural Network.

Testing of (GNNLB)

Above trained neural network was used in GNNLB where input process with their 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

$$P \leftarrow \text{Tained\_CNNM}(m, n, D)$$

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.

Training of GNNLB
Input: JSM // Input Job Sequence Matrix
Output: TNN // TNN: Trained Neural Network

```
1.  P←Generate_Population()
2.  While two solutions are not same
3.  P←Teacher_phase(P, JSM)
4.  P←Student_phase(P, JSM)
5.  Endloop
6.  FS←Fitness(P,JSM) // FS: Final Sequence
7.  PJSM←Pre-Processing(JSM) // PJSM: Processed
    JSM(64x64)
8.  BJSM←Binary(PJSM) //BJSM: Binary JSM
9.  BJSM←Convolution(BJSM)
10. BJSM←Max_Pooling(BJSM)
11. BJSM←LRN(BJSM)
12. BJSM_L←Level_4(BJSM)
13. BJSM_L←Level_5(BJSM_L)
14. BJSM←Concatenation(BJSM_L)
15. BJSM←Convolution(BJSM)
16. BJSM←Convolution(BJSM)
17. TNN←EBPNN(BJSM, FS)
```

4. 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 a 2.40 GHz Intel Core i3 machine, equipped with 8 GB of RAM, and running under Windows 10 Pro.

DatasetBenchmarks for basic scheduling problems e. Taillard [10]. Following table show detail structure of experimental dataset, where we use two job sequences.

Description of dataset with attributes [10, 11].		
Attributes	Set1	Set2
Number of Machines	5	10
Number of Jobs	20	20

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

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

Relative Percent Deviation(RPD)

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

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

Trail Index It is defined as the average arrival rate of jobs coming in system to execute in one sequence.

$$\boldsymbol{TI=\frac{1}{n}\sum_{t=1}^{n} A_t}$$

A is arrival time of tth job of sequence set.

Results

Table 1 Makespan of the GNNLB.

Techniques	Set 1	Set 2
SJF-RL [1]	1420	1740
GNNLB	1131	1575

From table 1 is obtained that GNNLB has reduced the makespan 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 2 Comparison of proposed and previous work Total flow Time.

Techniques	Set 1	Set 2
SJF-RL [1]	23382	28187
GNNLB	13246	20235

From table 2 is obtained that GNNLB has reduced the total flow time of the dynamic load balancing of input testing dataset. This was done because of CNN 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 [1] shortest job sequences engage one machine when some part of job is time taken to complete the job.

Table 3 Comparison of RPD for MakesSpan.

Techniques	Set 1	Set 2
SJF-RL [1]	23.4783	9.98736
GNNLB	1.65217	0.442478

From table 3was obtained that GNNLB has reduce the RPD Makespanof 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.

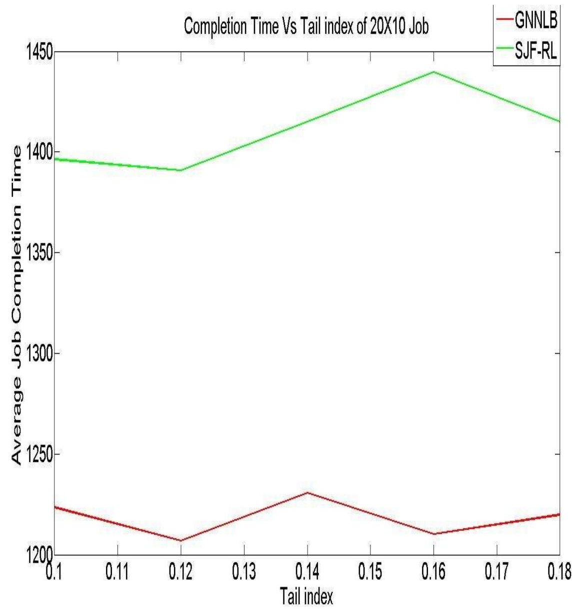

Fig. 3 Comparison of GNNLB and SJF-RL on the basis of arrival time tail index.

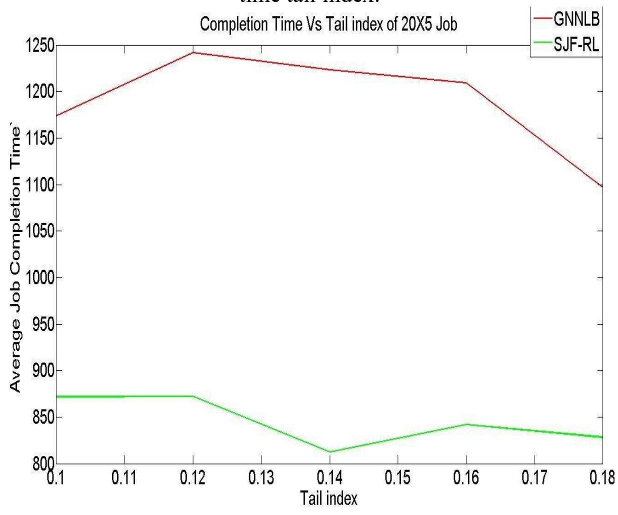

Fig. 3 Comparison of GNNLB and SJF-RL on the basis of arrival time tail index.

Above fig. 3 and 4 shows that proposed algorithm GNNLB has low average completion time on different tail index as compared to previous job scheduling algorithm. Here it was also obtained that various job set used in experiment shows low completion time of proposed work as job scheduling for different machine was good.

Table 4 Comparison of RPD for Total flow Time.

Techniques	Set 1	Set 2
SJF-RL [1]	63.5105	76.1688
GNNLB	7.37063	26.4688

From table 4were obtained that GNNLB has reduce the total flow time RPD of the dynamic load balancing of input testing dataset. This was done because of CNN 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 [1] shortest job sequences engage one machine when some part of job is time taken to complete the job.

5. Conclusions

Dynamic Load balancing help has help clouds to handle multiple requests from various dimensions. Towards this end many techniques came into existence for this work. Here many researchers have already done lot of work based on neural network classification. In this model TLBO (Teachers Learning Based Optimization Algorithm was used for assigning the input jobs. In this work genetic algorithm TLBO is use because this takes two phase learning. Main motive of this model is to find good solution in any dynamic condition. Genetic algorithm output is used for learning in ANN so this trained neural network generates probable solutions which are closer to desired one. Here result shows that GNNLB has improve the accuracy by reducing the RPD. Here use of proper training and rich input vector resultant neural network is less time consuming. It was obtained that GNNLB has reduced the Total Flow time. Here overall accuracy of the propose work was also improved.

References

[1]. MianGuo, Quansheng Guan andWendeKe. "Optimal Scheduling of VMs in Queuing Cloud Computing Systems With a Heterogeneous Workload". *Digital Object Identifier 10.1109/ACCESS.2018.2801319.*

[2]. Song Ningning, Gong Chao, AnXingshuo. "Fog Computing Dynamic Load Balancing Mechanism Based On Graph Repartitioning" China Communication, IEEE Volume 13 ISSUE 3, 2017.

[3]. Liang Yu, Tao Jiang, andYulongZou. “Fog-Assisted Operational Cost Reduction For Cloud Data Centers”. Date Of Current Version August 8, 2017. Digital Object Identifier 10.1109/Access.2017.2728624.

[4]. Gongzhuang Peng, Hongwei Wang, Jietao Dong, Heming Zhang. “Knowledge-Based Resource Allocation For Collaborative Simulation Development In A Multi-Tenant Cloud Computing Environment”. Ieee Transactions On Services Computing (Volume: 11, Issue: 2, March-April 1 2018)

[5]. Dazhao Cheng ; JiaRao ; YanfeiGuo ; Changjun Jiang ; Xiaobo Zhou . “Improving Performance Of Heterogeneous Mapreduce Clusters With Adaptive Task Tuning”. Ieee Transactions On Parallel And Distributed Systems (Volume: 28, Issue: 3, March 1 2017)

[6]. ShengjunXue, Wenling Shi and XiaolongXu. A Heuristic Scheduling Algorithm based on PSO in the Cloud Computing Environment. International Journal of u- and e- Service, Science and Technology Vol.9, No. 1 (2016), pp.349-362.

[7]. Framinan, J.M. &Leisten, R. (2003). An efficient constructive heuristic for flowtimeminimization in permutation flow shops, *Omega,* Vol.31, 311-317.

[8]. Chandrasekaran, S.; Ponnambalam, S.G.; Suresh, R.K. &Vijayakumar N. (2006). An application of Particle Swarm Optimization Algorithm to Permutation FlowshopScheduling Problems to Minimize Makespan, Total Flowtime and CompletionTime Variance, *Proceedings of the IEEE International Conference on Automation Scienceand Engineering*, *2006 (CASE '06.)*, pp-513-518, ISBN: 1-4244-0311-1, Shanghai,China,

[9]. Gowrishankar, K.; Rajendran, C. &Srinivasan, G. (2001). Flowshop scheduling algorithmsfor minimizing the completion time variance and the sum of squares of completiontime deviation from the common due date, *European Journal of Operational Research*, vol.132, No: 31, 643-665.

[10]. Taillard, E. (1993). Benchmarks for basic scheduling problem, *European Journal of Operational Research*, Vol.64, 278-285.

[11]. Li Chunlin, Zhou Min andLuoYoulong**. “**Efficient Load-Balancing Aware Cloud Resource Scheduling for Mobile User”. Computer And Communications Networks And Systems The Computer Journal, 2017

[12]. Jia Zhao, Kun Yang, Xiaohui Wei*,* Yan Ding, Liang Hu, GaochaoXu. “A Heuristic Clustering-based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment”. IEEE Transactions On Parallel And Distributed Systems, June 2014.

[13]. S. M. Lau, Q. Lu, and K. S. Leung, “Adaptive Load Distribution Algorithms for Heterogeneous Distributed Systems with Multiple Task Classes,” Journal of Parallel and Distributed Computing, vol. 66, no. 2, pp. 163-180, 2006.

[14]. 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.

[15]. ByungChulTak, Youngjin Kwon, and BhuvanUrgaonkar. “Resource Accounting of Shared IT Resources in Multi-Tenant Clouds”. 10.1109/TSC.2015.2453980, IEEE Transactions on Services Computing

[16]. 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.