[1]Deepthi Kothapeta

[2] Dr.M.Jagadeeshwar

[3] V. Shobha Rani

[4] Dr. MVS Prasad

Optimized Deep Neural Network Based Load Balancing in Fog Computing with Robust Dynamic Scheduling Algorithm

Journal of Electrical Systems

Abstract: The load balancing (LB) in FOG architecture is a formidable task with the availability of minimal resources. This field encounters enormous studies that portray the routing of task between the fog physical devices and cloud nodes. The presence of different types of heterogeneity devices are at helm of difficult scheduling process. To allocate LB based on the available resource we propose a novel Deep Neural Network (DNN) based Tuna swarm strategy based Bacterial Foraging optimization algorithm (TBFO) which employs three stages such as monitoring the fog resources, Classification based on deep learning technique and dynamic scheduler that are optimized. With the dynamic scheduling algorithm this work aims to provide LB in real time application. The first stage is to monitor the resources of server and amassed in fog resource table. The next step is to identify the unerring server and is effectuated with the proposed DNN technique. The last stage is to allocate the process to the selected server with the TBFO. It certifies the robust continuous services in the fog LB. Simulations are conducted and compared the outcomes with the existing works and ensures effective load balancing, make span and resource utilization.

Keywords: Load balancing, Fog computing, deep Neural Network, Bacterial Foraging algorithm, and server allocation

I. INTRODUCTION

In a fog framework [1], load balancing helps to evenly distribute the strain on bandwidth to continue to offer amenities if a portion of service malfunctions. This is achieved by properly and allocating bandwidth when deploying and de-provisioning usage processes. The process of dividing up communication over the network evenly among an assortment of assets for assisting a site is known as load balancing [2]. Many of clients must be processed concurrently by contemporary apps, and every client must receive accurate material, footage, pictures, and other content quickly and reliably. For service suppliers, the loud infrastructure offers an integrated framework for processing-intensive operations, this can be due to the broadband backbone's plentiful capabilities. Here, removing regional analysis reduces the cost of computing and energy usage at connected interfaces [3].

Nevertheless, as a result of the longer route dissemination wavelengths between endpoints and physically sedentary obscure points, the use of clouds causes a surge in connection latency. An upon-request outstanding durability platform that can manage both the highest and lowest communication volumes [4] is made possible by distributing the load. The ideal period of operation would be attained if all users completed the final task simultaneously, so there could be no waiting around. It is to guarantee that, to satisfy demand, every process in the line is operating at the same speed.

Developing the perfect combination of goods to guarantee minimal stockpile and growing adaptability by cutting the conversion time are the main goals of load leveling. It increases reaction time and decreases latency among networks to enhance application productivity. Gradually divide the task among hosts to enhance the efficiency of the application, and minimize delays by rerouting customer inquiries to a computer that is nearby to the user's location. Such as making concurrent applications more complicated and overhead-prone, causing task completion and customer interaction to be unpredictable and inconsistent, relying on the accessibility and correctness of knowledge, and interfering with additional maximizing the efficiency. To achieve effective load balancing we propose an innovative approach known as TBFO based DNN. Some of the main points are,

- The load balancing is achieved by balancing the requirements of server and to determine the incoming processes. It also considerate the requirements of CPUs per server.

[1] [1] Research Scholar, Department of Computer Science, Chaitanya Deemed to be University, Telangana, India, deepthivaishu18@gmail.com

[2] Professor, Department of Computer Science, Chaitanya Deemed to be University, Telangana, India,jagadeeshwar07@gmail.com

[3] Assistant Professor, Department of Computer Science & Artificial Intelligence, SR University, Telangana, India.

[4] Professor, Department of Computer Science & Engineering, Malla Reddy Engineering College

- The requirements of RAM, storage requirements and if physical server is used then it is necessary to determine the overall size and traffic of the network. The computing power of the system also determined.

The roadmap of the work is: in section 2 the relevant works are explained with their demerits. The proposed work is explained in section 3 in a wider manner. In section 4 the results and performance are analyzed. The summary is attached in section 5.

II. RELATED WORKS

Karthik et al. [5] have described a fog computing-based whale optimization algorithm (FGWHO) employed to calculate the structure's generator location, energy production, and electrical consumption. The chosen controllers focused on expanding the connection's connectivity to achieve the ideal capacity. The infrastructure was built to determine interactions, vitality, and separation between networked devices. Network reliability was enhanced by the suggested method in the areas of efficiency, retention time, and residue capacity. However, it is inadequate to transfer the information between nodes.

Liao et al. [6] have presented a cognition-centric fog computing resource balancing (CFCRB) scheme an intellectual balancing infrastructure with a perception platform that stores information concerning intellectual fog materials, processes policies, and monitors the consumption of services. The implementation of networked procedures involves the use of a multiagent integrity concept. These distributed processes can improve Internet of Things flexibility and resilience because they only require regional processing as well as interaction. Initially continue to anticipate that more effective protocols will enable the implementation of the system. Thus, it is impossible to carry out resource forecasting on hardware devices.

Maswood et al. [7] have developed a Mixed-Integer Linear Programming (MILP) based optimization model a unified fog cloud framework for assisting applications that move quickly at a decreased running price by minimizing energy expenses and reducing delays. The dual goals of this system's combined ultimate function are load distribution and transmission price reduction. The findings of virtual reality demonstrate that this system can efficiently use a collaborative setting to reduce the price of transmission and regulate the burden. To determine the degree of preference, one gives strength elements to the various aims in the entire mission statement. This effort can assist fog firms in efficiently allocating the lack of funds. However, it is insufficient to examine the system on a broad scale.

Siasi et al. [8] suggested deep learning network is made up of minimal and high-capacity cloud nodes that are situated close to the station. Predicting the subsequent inbound procedure and prefetching it onto the associated node is the intent. As a result, the identical task can be retained immediately by the server for additional requests, saving on goods, computational expense, and power usage. Additionally, this results in greater demands being fulfilled and more bandwidth. Furthermore, implementation expenses are significantly raised by the enlarged fog levels framework.

Asghar et al. [9] highlighted a fog-based health monitoring system architectureto reduce network utilization and delay. Devices are used to send packets of the client's metabolic attribute statistics that are created by detectors to the cloud cluster. Receiving messages of information are processed by nodes located in fog to determine the extent to which an individual is critically ill. The number of new inquiries received by each mist cluster in immediate systems may differ, thereby causing an imbalance in the cloud stations' capacity. Thus, there is a lack of using varied data sets.

2.1 Problem Statement

From the literature review we have observed some of the shortcomings like not considering the priority and number of tasks, other factors such as response time and waiting time also not considered. To overcome these issues, we introduced a standardizing the resource attributes and the resources are split for the mitigation of scale of the resources. It also reduces the latency and therein accomplished the load balancing.

III. PROPOSED METHOD

Load balancing is predominant task when there is restricted resources available and especially in the

Fog computing. The proposed load balancing employs three stages and are: (i) Fog Resource Monitor, (ii) DNN based classifier and (iii) TBFO based dynamic scheduler and is framed in figure 1.

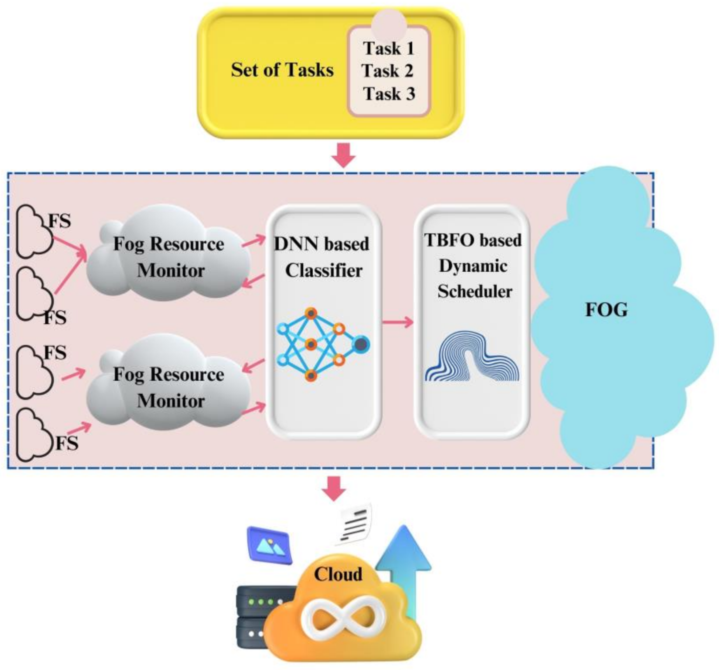

Figure 1: Stages employed in proposed load balancing approach in Fog computing

3.1 Fog Resource Monitor

It is a main stage for monitoring the server and its respective resources and stored it in Fog Resource table (FRT) for further processing. The FRT is usually placed in the server of the FRM and a sample FRT is displayed in table 1 [10]. The classification of fog server as suitable or not can be effectuated based on the features such as Storage (ST), Computing (CM), and the RAM. Below are procedures that follow for the allocation of status for server and is listed below.

Mean storage estimation (M_ST)

Mean computing estimation (M_CM)

Mean RAM size estimation (M_RAM)

Application of rules, example (i) if $ST \geq M\_ST$ then set $ST = 1$, otherwise set $ST = 0$,(ii) if $CM \geq M\_CM$, set $CM = 1$, otherwise set $CM = 0$, (iii) if $RAM \geq M\_RAM$, set $RAM = 1$, otherwise set $RAM = 0$. These are features indulged for the input of DNN.

Table 1: Sample Fog Resource Table

S_i	ST (GB)	CM (MHz)	RAM (GB)	Characteristics	Status
S_1	250	3500	4	Proper, in adequate	Not suitable
S_2	200	6000	8	Proper, Adequate	Suitable
S_3	40	2000	5	Inadequate, Improper	Unsuitable
	….	…..	…..	…..	…..
S_n	….	…..	…..	…..	….

3.2 DNN based Classifier

The classification of fog server as suitable or not, has been effectuated with the proposed DNN based classifier and the details are elucidated in the following section.

3.2.1 Deep Neural Network:

For architecture construction, compose and extract the features. One fully connected output layer w4ith the hidden layers, one layer of input interprets the fully connected DNN [13]. Figure 1 depicts the DNN model. Figure 2 explains the structure of DNN. Optimize the features to provide feature vectors. Based on the input layers, receive the hidden layers depicts as follows;

$$G_{1(i)} = \sum_i M_{1(i,j)} w_i + c_{1(i)} \tag{1}$$

Where, $c_{1(i)}$ and $M_{1(i,j)}$ are the DNN bias and weights and the hidden layer outcome is expressed by,

$$G_{O(i)} = AF(G_{1(i)}) \tag{2}$$

Where, $AF(.)$ denotes the DNN activation function. Mark and embed the data design with DNN. The feature vectors are optimized by requiring the optimizing DNN.

Swiftly effectuate the process of training to perform predominant tuning of DNN. The extracting features are efficacy and enhance speed. Minimize or maximize the required optimization to deem the fitness value. The below expression defines the cost function.

$$Z(M,c) = \frac{1}{L} \sum_{i=1}^{L} l\left(q^{ti}, q^{i}\right) \tag{3}$$

Where, q, q^i and Z are original cost, evaluated cost and cost function mean value with L-loss of l. From this, predict the difference among predictions models as the loss function. The gradient descent utilized by acquiring biases and weights with the cost function Z is minimized. Below formula minimizes the cost function utilizing gradient descent.

$$M = M - \beta \frac{\partial}{\partial M} Z(M) \tag{4}$$

$$c = c - \beta \frac{\partial}{\partial c} Z(c) \tag{5}$$

Mitigates the cost function as well as update the bias and weights. The following formula calculates the $P(M)$ values.

$$P(M) = \sum_L \left\| M_L \right\|_E^2 \tag{6}$$

From this, $\|\cdot\|_E^2$ and L represent the Frobenius and controlling coefficient. The hidden layer output penalty sparsity is $P(M)$.

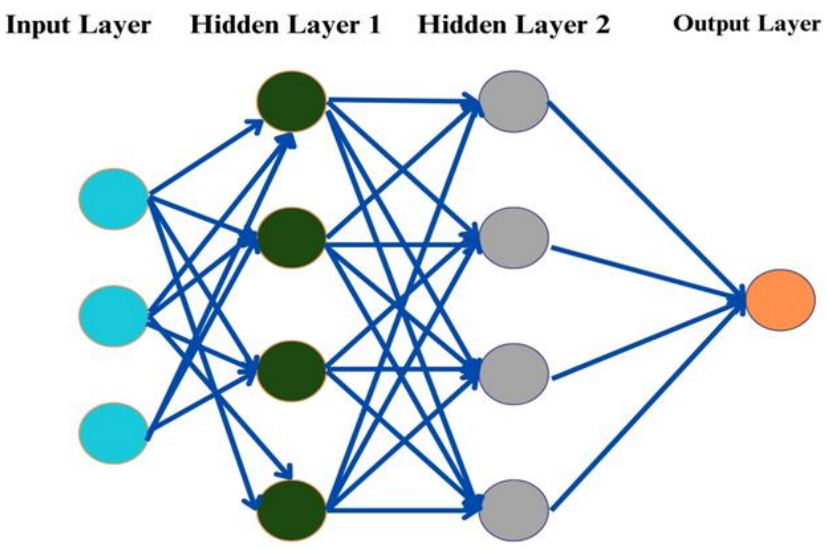

Figure2: The DNN structure

3.2.2 Problem Definition

This step is for the detection of features of the fog servers in order to label as suitable or unsuitable. This has been achieved with the features that are hoarded in the FRT. The basic steps involved in this classifier stage are (i) collection and cleaning of data, (ii) converting the labels that are read by the CNN i.e., with 1 and 0. Henceforth the estimation of M_ST, M_CM, and M_RAM are made. (iii) the training and testing of data are effectuated for the real time application. Based on the classification result the Fog suitable table (FST) is updated and is displayed in table 2.

Table 2: Updated FST

S_i	ST	CM	RAM	Fitness Estimation(FE_i)
S_2	4	7	7	5
S_4	3	5	6	6
S_7	7	7	5	7
	….	….	….	….
S_m	….	….	…..	….

The algorithm used for the classification of server using the proposed DNN is depicted in algorithm 1.

Algorithm 1: Pseudocode for the DNN based classifier for the classification of fog servers as suitable or not

```
Input of the classifier: Three learned attributes such as ST, CM, and RAM
Output of the classifier: FST which is the updated version of FRT
    Data collection using FRT
For each server do
    Estimate the mean storage M_ST
    Estimate the mean computing M_CM
    Estimate the mean RAM (M_RAM)
    If $ST \geq M\_ST$
            Set ST=1;
    Else
            Set ST=0;
    If $CM \geq M\_CM$
            Set CM=1;
    Else
            Set CM=0;
    If $RAM \geq M\_RAM$
        Set RAM=1;
    Else
        Set RAM=0;
Next
    Train the learned attributes of DNN
    Testing the DNN
For each server
    If (ST==1&&CM==1&&RAM==1) then:
        Set status=1
    Else
        Set status=0
    End if
Next
    FRT up-gradation
```

3.3 TBFO based Dynamic scheduler

This section is to present the dynamic scheduler that is used to allocate the input to the respective server. For this process we proposed a novel TBFO algorithm and are elucidated in the next section.

3.3.1 Tuna swarm bacterial foraging optimization

Various kinds of operations included in the E-coli foraging behavior named as bacterial foraging optimization (BFO) algorithm. Different stages of BFO [11] is explained as;

- **Chemotaxis**

Prior to changing the direction, bacteria stayed with longest time and this phase effectuates BF strategy. The step takes direction changing. At given $L^i(J, M, N)$, *N* is the neglect dispersal with *M* reproduction and *J* chemotaxis when the bacterium location is *L*. Following formula computes the bacteria tumbling.

$$\delta(i) = \frac{\gamma(i)}{\sqrt{\lambda(i)^t \gamma(i)}} \tag{7}$$

The vector *I* falls under 1 to P with $\gamma(i)$ is the random vector. From this, $\gamma_k(i), k = 1,2,,.......,y$ each element by P is an entire number of bacteria falls under the interval of [0, 1]. Below expression frames the bacteria up-gradation location.

$$L^i(J+1, M, N) = L^i(J, M, N) + F(i)\delta(i) \tag{8}$$

From this, $F(i)$ is the swim moving step size.

- **Parabolic Foraging**

While attaining higher nutrients, chemical substances are released with bacterial to other bacteria. Repel when the situation is daunting. The swimming behavior of BFO is enhanced with tuna optimization with its foraging of parabolic. Reference point as the food with parabolic representation. 505 of probability selection simultaneously performed. Below expression describes its mathematical model.

$$CC(L,L^i(J,M,N)) = \sum_{i=1}^{P}\left[-g_{AT}\exp\left(-U_{AT}\sum_{h=1}^{R}\left(L_h - L_h^i\right)^2\right)\right] + \sum_{i=1}^{p}\left[f_{RE}\exp\left(-V_{RE}\sum_{h=1}^{r}\left(L_h - L_h^i\right)^2\right)\right] \quad (9)$$

Where, $L = [L_1,....,L_p]^t$ takes the bacteriumoptimization domainand L_h^i is the i^{th} bacterium with h^{th}componentlocation. Here, $CC(L,L^i(J,M,N))$ is the chemotaxis stage based on the communication of cell to cell. Below formula computes the swarming characteristics with tuna foraging of parabolic.

$$Q(i,j,m,n) = Q(i,j,m,n) + CC(l,l^i(j,m,n)) + Y_j^{t+1} \quad (11)$$

$$Y_j^{t+1} = \begin{cases} Y_{best}^T + random\cdot\left(Y_{best}^T - Y_j^T\right) + FT\cdot R^2\left(Y_{best}^T - Y_j^T\right), & if\ \ random < 0.5 \\ FT\cdot R^2\cdot Y_j^t, & if\ \ random \geq 0.5 \end{cases} \quad (10)$$

The random number is FT .

- **Reproduction**

Take the step of reproduction next to completing the phase of chemotactic. The positive number as *P* is considered. Without mutations, offsprings created and P_s is the satisfactory nutrients with the bacteria population [12].

$$P_s = \frac{P}{2} \quad (12)$$

A bacterium health is detected by using accumulated cost and unhealthy is lower nutrients that represented with higher value and it not fit for reproduction. Reversely organize bacteria depending upon the status of health. In similar position, place and divide health bacteria into two new bacteria with unhealthy bacteria loss their life.

- **Elimination-Dispersal**

The bacteria with kill wide range increase in temperature. Where, R_{EL-DE} perform possibilities. From the other living bacteria, replace to produce new bacteria. The pseudocode for the proposed TBFO based dynamic scheduler is illustrated in algorithm 2.

Algorithm 2: Pseudocode for the TBFO based dynamic scheduling

```
Input: FST with the details of suitable servers, input task to the server
Output: Allocating to the input task to the suitable server
Initialize the parameters of TBFO, $Y_{best}$ and Best server (BS)
Data collection from the FST
FOR each server from the FST do
    Estimate the $FE_i$ using the values of ST, CM, and RAM
    If ( $FE_i > Y_{best}$ ) then
        $Y_{best} = FE_i$ and $BS = S_i$
    End if
FST up-gradation
Allocation of task to BS
```

IV. RESULTS AND DISCUSSIONS

This section is to extend the analysis of proposed analysis in the field of load balancing in the fog computing. the simulation tool, number of tasks used and server details are added in the section.

4.1 Simulation details

For demonstration we have indulged with NS2 simulator which is installed in system with the specification of with the specification of 2.83 GHz processor and RAM of 8 GB. For the experimental purpose we have taken 200 fog servers and the incoming processes or tasks are around 180. The performance analyzes are explained below in a detailed manner.

4.2 Performance Investigation

Figure 3 represent the comparative figure of number of tasks with average resource utilization. The state-of-art methods such as Grasshopper optimization algorithm (GHO) [14], Gray Wolf Optimization (GWO) [15], Particle Swarm Optimization (PSO) [16] and Genetic Algorithm (GA) [17] are compared with proposed method to estimating the comparison with respect to the average utilization of resource. The number of task increase to increasing the average resource utilization. The GA outperforms minimum resource utilization. But, the resource utilization of proposed method is higher than that of GHO, GWO, PSO and GA by varying the number of task.

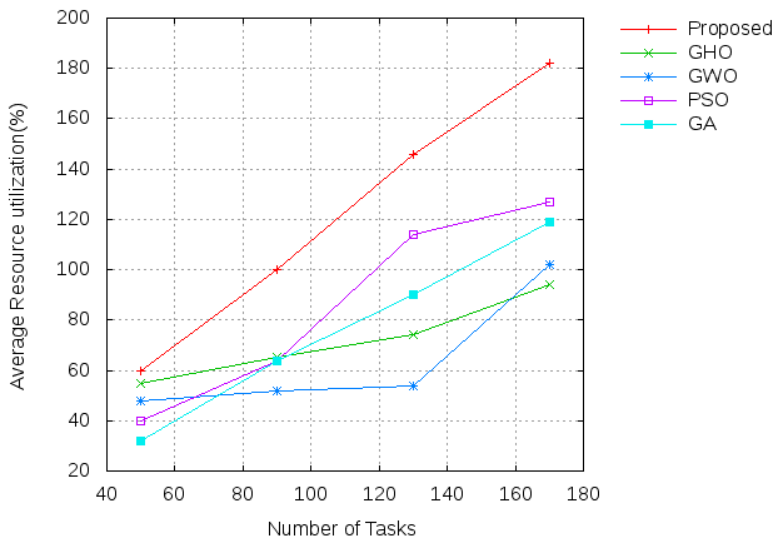

Figure 3: Performance of number of tasks Vs average resource utilization

Performance of number of tasks with average response time is plotted in Figure 4. The comparative models of GHO, GWO, PSO and GA with proposed approach to calculating the comparison with respect to the average utilization of resource. The number of task increase to minimizing the average response time and it is represented in milliseconds. The GA and PSO outperforms minimum resource utilization. However, an average response time of proposed approach is minimal that of GHO, GWO, PSO and GA with the variation of number of task.

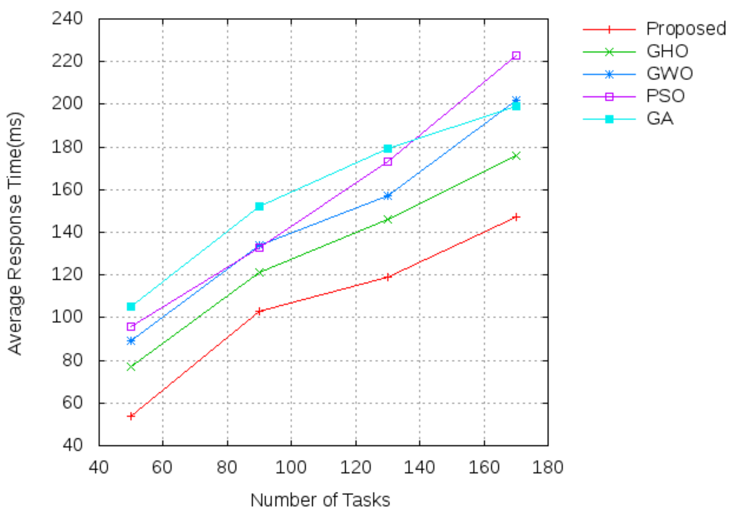

Figure4: Performance of number of tasks Vs average response time

The performance of number of tasks based on delay is plotted in Figure 5. The existing GHO, GWO, PSO, and GA comparative models with proposed approach to computing the comparison with respect to delay results. The number of tasks increases in order to reduce the delay performance, which is measured in seconds. Maximum delay is outperformed by the GWO and PSO. Though, with a variable number of tasks, the proposed approach's delay is shorter than that of GWO, GHO, PSO, and GA

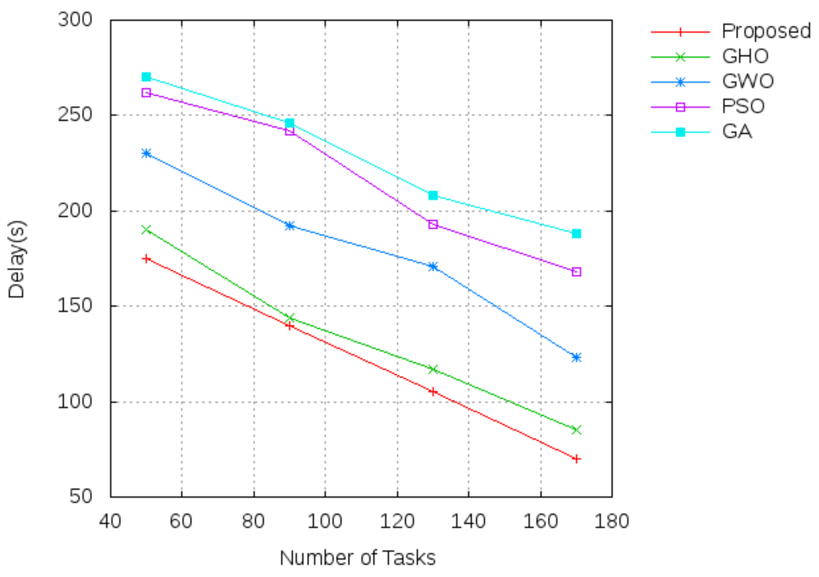

Figure5: Performance of number of tasks Vs delay

Figure 6 depicts the performance of a number of tasks depending upon energy consumption. The previous methodologies of GHO, GWO, PSO, and GA comparative models, as well as the proposed approach to compute the results of energy consumption and it is measured in terms of joules. The number of tasks is increased to reduce the en energy consumption model. The PSO and GHO consumes maximum energy. With a variable number of tasks, the proposed approach has a shorter consumption of energy than GWO, GHO, PSO, and GA.

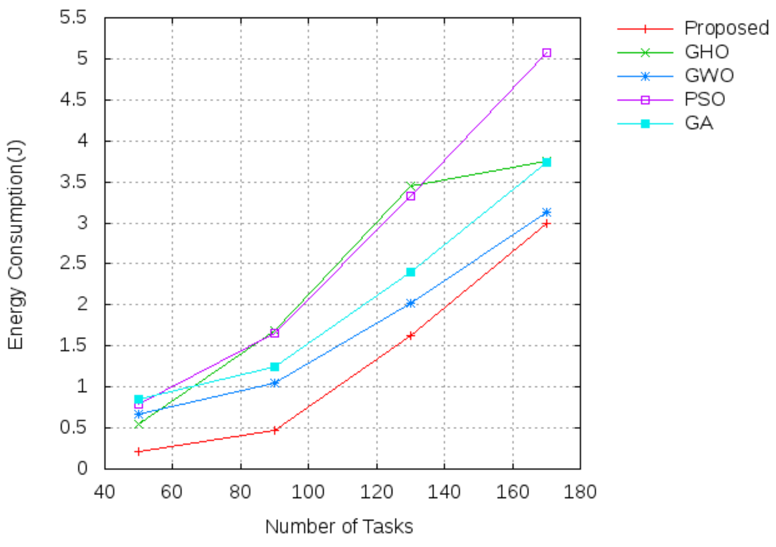

Figure6: Performance of number of tasks Vs energy consumption

Figure 7 displays the performance of several tasks based on the level of load balancing. Previous approaches of GHO, GWO, PSO, and GA comparison models with proposed approach are used to compute the outcomes of load balancing levels, which are quantified in percentages. To higher the level of load balancing, the number of tasks have been rose. The PSO and GA use the minimal level of load balancing. The proposed approach performs higher level of load balancing than GWO, GHO, PSO, and GA with a variable number of tasks.

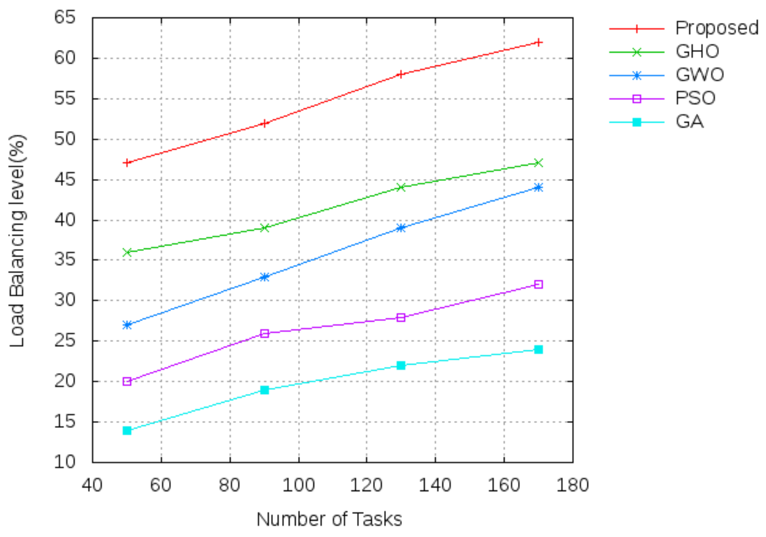

Figure7: Performance of number of tasks Vs load balancing level

Figure 8 depicts the execution of various tasks based on the time plot. Previous methodologies of GHO, GWO, PSO, and GAcomparison models are employed in conjunction with the proposed approach to compute the results of time in milliseconds. The number of tasks is increased to reducing the time performance. The PSO and GA utilizing minimum of time performance. With a variable number of tasks, the proposed approach achieves a minimum level of time than GWO, GHO, PSO, and GA approaches

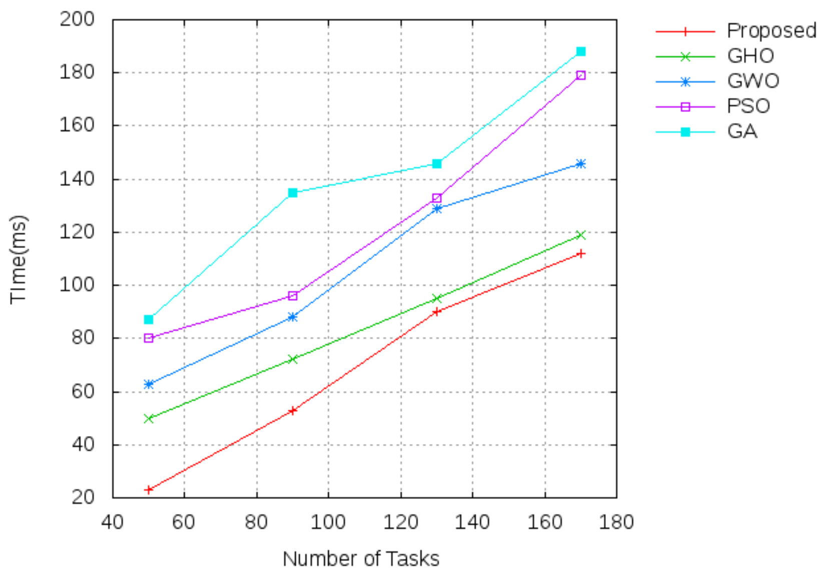

Figure 8: Performance of number of tasks Vs time in milliseconds

Figure 9 illustrates the total cost based execution of numerous tasks. To compute the results of total cost, the prior methods of GHO, GWO, PSO, and GA comparison models are used in conjunction with the proposed approach. The quantity of tasks is raised in order to reduce total cost performance. The PSO and GA are utilizing the more cost performance. Considering a variable number of tasks, the proposed approach takes less cost than the GWO, GHO, PSO, and GA strategies.

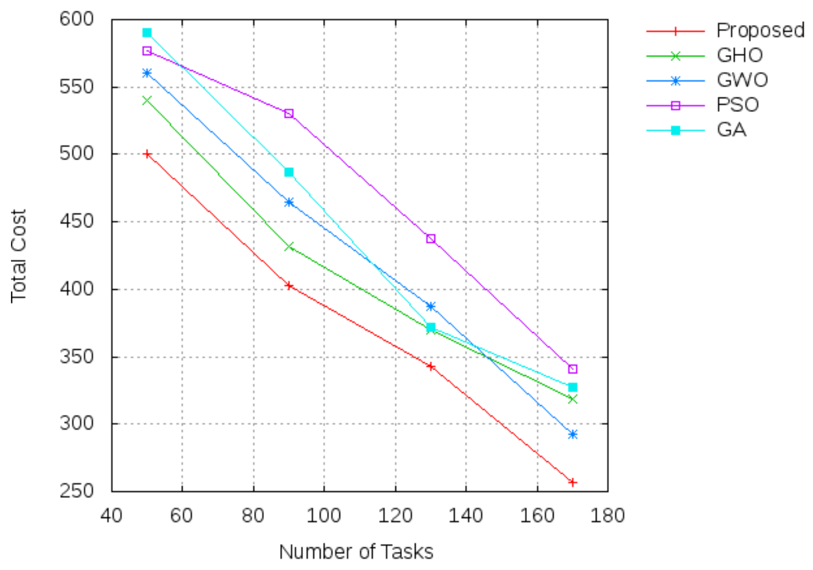

Figure 9: Performance of number of tasks Vs total cost

V. CONCLUSION

For the allocation of LB with the available resources the work in this article proposed an innovative DNN approach. This work involved three stages; the first stage was monitoring the resources of the server and filed it in the FRT. The next stage is to classify the servers as suitable or not suitable for the tasks to be performed and followed by the appropriate selection of server with the proposed TBFO algorithm. This step also allocates the exact server for the application based on the requirement and available resources. The dynamic scheduling algorithm was used for the optimization purpose. Further, the simulation was fulfilled with the NS2 simulator and achieves continuous performance for the LB in fog computing. Finally we have analyzed the performance in terms of total cost, make span, and resource utilization. Our proposed work assured the robust and continuous performance of LB

REFERENCES

[1] Samy, A., Yu, H. and Zhang, H., 2020. Fog-based attack detection framework for internet of things using deep learning. *IEEE Access*, *8*, pp.74571-74585.

[2] Junaid, M., Sohail, A., Rais, R.N.B., Ahmed, A., Khalid, O., Khan, I.A., Hussain, S.S. and Ejaz, N., 2020. Modeling an optimized approach for load balancing in cloud. *IEEE Access*, *8*, pp.173208-173226.

[3] Ponce, P., Meier, A., Méndez, J.I., Peffer, T., Molina, A. and Mata, O., 2020. Tailored gamification and serious game framework based on fuzzy logic for saving energy in connected thermostats. *Journal of Cleaner Production*, *262*, p.121167.

[4] Ashokkumar, N., C. Kanmani Pappa, Siva Kumar, Dhamodharan Srinivasan, and M. M. Vijay. "Certain Investigation of Optimization Methods of Sensor Nodes in Biomedical Recording Systems." In *2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS)*, vol. 1, pp. 1175-1181. IEEE, 2023.

[5] Karthik, S.S. and Kavithamani, A., 2021. Fog computing-based deep learning model for optimization of microgrid-connected WSN with load balancing. *Wireless Networks*, *27*, pp.2719-2727.

[6] Liao, S., Wu, J., Mumtaz, S., Li, J., Morello, R. and Guizani, M., 2020. Cognitive balance for fog computing resource in Internet of Things: An edge learning approach. *IEEE Transactions on Mobile Computing*, *21*(5), pp.1596-1608.

[7] Maswood, M.M.S., Rahman, M.R., Alharbi, A.G. and Medhi, D., 2020. A novel strategy to achieve bandwidth cost reduction and load balancing in a cooperative three-layer fog-cloud computing environment. *IEEE Access*, *8*, pp.113737-113750.

[8] Siasi, N., Jasim, M., Aldalbahi, A. and Ghani, N., 2020. Deep learning for service function chain provisioning in fog computing. *IEEE Access*, *8*, pp.167665-167683.

[9] Asghar, A., Abbas, A., Khattak, H.A. and Khan, S.U., 2021. Fog based architecture and load balancing methodology for health monitoring systems. *IEEE Access*, *9*, pp.96189-96200.

[10] Talaat, Fatma M., Hesham A. Ali, Mohamed S. Saraya, and Ahmed I. Saleh. "Effective scheduling algorithm for load balancing in fog environment using CNN and MPSO." *Knowledge and Information Systems* 64, no. 3 (2022): 773-797.

[11] Long, Y., Liu, S., Qiu, D., Li, C., Guo, X., Shi, B. and AbouOmar, M.S., 2023. Local Path Planning with Multiple Constraints for USV Based on Improved Bacterial Foraging Optimization Algorithm. *Journal of Marine Science and Engineering*, *11*(3), p.489.

[12] Yan, Zheping, Jinyu Yan, Yifan Wu, Sijia Cai, and Hongxing Wang. "A novel reinforcement learning based tuna swarm optimization algorithm for autonomous underwater vehicle path planning." *Mathematics and Computers in Simulation* 209 (2023): 55-86.

[13] Malhotra, Priyanka, Sheifali Gupta, Deepika Koundal, Atef Zaguia, and Wegayehu Enbeyle. "Deep neural networks for medical image segmentation." *Journal of Healthcare Engineering* 2022 (2022).

[14] Meraihi, Yassine, Asma Benmessaoud Gabis, Seyedali Mirjalili, and Amar Ramdane-Cherif. "Grasshopper optimization algorithm: theory, variants, and applications." *Ieee Access* 9 (2021): 50001-50024.

[15] Li, Yu, Xiaoxiao Lin, and Jingsen Liu. "An improved gray wolf optimization algorithm to solve engineering problems." *Sustainability* 13, no. 6 (2021): 3208.

[16] Zhao, Guojun, Du Jiang, Xin Liu, Xiliang Tong, Ying Sun, Bo Tao, Jianyi Kong, Juntong Yun, Ying Liu, and Zifan Fang. "A tandem robotic arm inverse kinematic solution based on an improved particle swarm algorithm." *Frontiers in Bioengineering and Biotechnology* 10 (2022): 832829.

[17] Squires, M., Tao, X., Elangovan, S., Gururajan, R., Zhou, X. and Acharya, U.R., 2022. A novel genetic algorithm based system for the scheduling of medical treatments. *Expert Systems with Applications*, *195*, p.116464.