Optimal Load Balancing in Cloud using ANFIS and MGWO based Polynomial Neural Network

Uday Chourasia (udaychourasia221@gmail.com)
Rajiv Gandhi Proudyogiki Vishwavidyalaya
Sanjay Silakari
University Institute of Technology RGPV

Research Article

Keywords: Load balancing, ANFIS, PNN, MGWO, nodes, cloud environment, service availability

Posted Date: June 2nd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-529618/v1

Optimal Load Balancing in Cloud using ANFIS and MGWO based Polynomial Neural Network

[1] * Uday chourasia, [2] Dr. Sanjay silakari

[1]Rearch Scholar, Computer science & Engg, Rajiv Gandhi Proudyogiki Vishwavidyalaya Bhopal, India
[2]Professor , Computer science & Engg, University Institute of Technology RGPV,
Bhopal, Madhya Pradesh, India.

*Corresponding Author Email id: udaychourasia221@gmail.com

Abstract: In recent years, cloud computing provides a spectacular platform for numerous users with persistent and alternative varying requirements. Here providing an appropriate service is considered a major challenge in the heterogeneous environment. In the cloud environment, security and service availability are the two most significant factors during the data encryption process. In order to provide optimal service availability, it is necessary to establish a load balancing technique that is capable of balancing the request from diverse nodes present in the cloud. This paper aims in establishing a dynamic load balancing technique using the APMG approach. Here in this paper, we integrated adaptive neuro-fuzzy interference system-polynomial neural network as well as memory-based grey wolf optimization algorithm for optimal load balancing. The memory-based grey wolf optimization algorithm is employed to enhance the precision of ANFIS-PNN and to maximize the locations of the membership functions respectively. In addition to this, two significant factors namely the turnaround time and CPU utilization involved in optimal load balancing scheme are evaluated. In addition to this, the performance evaluation of the proposed MG-ANFIS based dynamic load balancing approach is compared with various other load balancing approaches to determine the system performances.

Keywords: Load balancing, ANFIS, PNN, MGWO, nodes, cloud environment, service availability

1. Introduction

In the past few years, cloud computing technologies are exhibiting spectacular escalation du. to the rapid progression of communication technologies (Gupta et al. 2020). On the other hand, cloud computing is capable of solving various large-scale problems and it permits both software and hardware applications for the cloud consumer through the internet. Cloud computing services offer a new opportunity for both service requesters and providers by employing various service models like software as a service (SaaS), infrastructure as a service (IaaS) as well as platform as a service (PaaS) (Kong et al. 2020). Furthermore, cloud computing technologies are widely utilized by numerous leading IT organizations such as Microsoft, HP, Apple, Amazon, Oracle and IBM. Cloud computing is categorized into hybrid, community, public and private deployment models. By maintaining proper cloud source management, the cloud resource achieves a scalable and most effective feature (Kaur et al. 2020). The virtual form of cloud is considered one of the most significant characteristics of cloud computing technologies. The main intention of the cloud service provider is to offer service to the user based on a rental basis (Jyoti and Shrimali, 2020).

Load balancing is considered one of the major issues in cloud computing technologies. If there occurs any breakdown in one or more than one service component and instead of blocking the entire service, the continuance of service is guaranteed by the cloud computing services (Mukhopadhyay et al. 2020). This can be conducted by two various applications namely the provisioning and de-provisioning applications. By employing the load balancing technique, the dynamic workloads are distributed equally among various nodes that remove inactive nodes and prevent overloading. In addition to this, the user contentment and system performances are enhanced by employing various load balancing approaches. In general, the load balancing in the cloud occurs among virtual machines or physical host (Tiwari et al. 2020). Load balancing in other terms referred to as load balancing as a service (LBaaS) are categorized into dynamic load balancing algorithm (DLBA) and static load balancing algorithm (SLBA). The DLBA enhance the efficiency and adaptability under heterogeneous and homogeneous surroundings whereas the SLBA is fit only under stable heterogeneous condition (Ullah et al. 2020).

A distinctive load-balancing algorithm is composed of various selection, location, information and transfer policies. The pending work which is to be moved from sender to receiver is carried out in the selection policy. The location policies involve selecting an approximate resource to transfer the work (Naresh et al. 2020). The information policies are capable of specifying and updating the information regarding the load. The transfer policies determine the performance of load balancing. In addition to this, depending on the decisions, the load balancing schemes are classified into a distributed scheme and centralized scheme (Arulkumar and Bhalaji, 2020). The entire load balancing decisions are carried out under a distributed load-balancing scheme. The allocation of new jobs and the global view regarding the central node is referred to as the centralized scheme. The integration of both distributed and centralized scheme is known as a hierarchical load balancing scheme. Central node failure and instability are considered as the major drawback of this load balancing scheme (Maswood et al. 2019).

This paper proposes to establish a dynamic load balancing technique using APMG approach. In the cloud

environment, protection of sensitive data and secured communication are considered as the two significant concerns. Here, the enhanced elliptic curve cryptography technique is employed to enhance the confidentiality of the data thereby storing it in the cloud. Here we integrated adaptive neuro-fuzzy interference system-polynomial neural network (ANFIS-PNN) as well as memory-based grey wolf optimization algorithm for optimal dynamic load balancing thus establishing a load balancing technique using MG-ANFIS approach. The memory-based grey wolf optimization algorithm is employed to enhance the precision of ANFIS-PNN and to maximize the locations of the membership functions.

This major emphasis of this paper is obtained as follows

- Proposing MG-ANFIS for optimal dynamic load balancing approach to ensure service availability and security.
- Utilizing the MGWO algorithm to enhance the precision of ANFIS-PNN and to maximize the locations of the membership functions.
- Comparing the proposed MG-ANFIS based dynamic load balancing approach is compared with various other load balancing approaches to determine the system performances.

The remainder of the paper is structured as follows: In section 2, the survey based on load balancing scheme in cloud computing is explained. The designing of a key generation using enhanced elliptic curve cryptography (EECC) technique and the problem formulation is discussed in section 3. In section 4 the proposed MG-ANFIS based dynamic load balancing scheme is illustrated. The performance evaluation and comparative performances are described in section 5. Section 6 concludes the article.

2. Previous Literature Reviews

The field of load balancing based on cloud computing has drawn considerable attention from numerous research scholars. In order to attain better knowledge, numerous significant research works and their related reviews are delineated in the subsequent section.

Jena et al. (2020) proposed a load balancing on the cloud environment using two various algorithms namely the modified particle swarm optimization algorithm and Q-learning approach thus integrated and formed the QMPSO approach. Standard deviation, Makespan, throughput, energy utilization was the simulation measures employed for evaluation. The throughput rate obtained by this approach is very high but failed to implement the load balancing on reliable tasks.

The concept of load balancing in cloud computing was proposed by Semmoud et al. (2020) that utilized the starvation threshold load balancing algorithm. The performance employed in simulating the proposed approach was Average idle time, total number of tasks, makespan and average response time. The experimental analysis was conducted and the evaluation results revealed that the response time was very less. High communication cost is considered as the major drawback of this approach.

Junaid et al. (2019) developed an optimized approach that utilized Data Files Type Formatting under a cloud environment. Accuracy, precision, f-measure, recall were the performance metrics utilized for evaluation. The efficiency obtained in this approach was very high. But this approach failed to consider task immigrations.

The concept of energy-efficient load balancing in cloud computing using firefly and Improved Multi-Objective Particle Swarm Optimization was proposed by Devaraj et al. (2020). The evaluation measures employed in this approach were Average response time, average load, reliability, makespan. The experimental analysis was conducted and the results revealed that the memory utilization and efficiency were high. But very low complexity rate is considered as the major disadvantage of this approach.

Siddiqui et al. (2019) proposed the queue processing for server load based queuing approach under a cloud computing environment. Response time, throughput, waiting time was the simulation metrics used to evaluate this approach. The service availability and robustness of the proposed approach were very high when compared with other approaches. But this approach failed to consider weighted priority under a cloud environment.

An efficient load balancing system using an adaptive dragonfly algorithm in cloud computing was demonstrated by Neelima et al. (2019). Execution cost, execution time and makespan were the simulation measures employed for evaluation. The execution cost and time obtained in this approach were very low. Less accuracy is considered as the major drawback of this approach.

The mutation-based particle swarm optimization algorithm regarding load balancing was proposed by Agarwalet al. (2020). Average makespan and cloudlets wee the simulation measures employed for evaluation. The experimental analyses were conducted and the results revealed that the life span was very high with less energy rate. In addition, this approach failed to consider various other measures like average time, resource utilization.

Priya et al. (2019) developed the fuzzy-based multi-resource scheduling algorithm for a load balancing scheme. The parameters employed for simulation were Average success rate, efficiency and response time. The experimental analysis was conducted and the analysis revealed that the latency and efficiency of this approach was very high with poor response time and average success ratio

The concept of Dragonfly optimization and constraint measure-based load balancing in cloud computing was proposed by Polepally et al. (2019). Capacity, load, number of the migrated task were the performance metrics utilized for simulation. Here, the performance rate was very high when compared with other approaches. But this approach requires a large time is the major drawback of this approach.

The enhanced maximum-minimum scheduling approach under a cloud environment was proposed by Hung et al. (2019). Bandwidth, length, capacity were the performance measures employed in this approach. The completion time and

quality of search were very high when compared with other approaches but implementation is complex. The summary of the existing literature works is discussed in table 1.

3. System Design And Problem Definition

The following section illustrates the designing of a key generation using enhanced elliptic curve cryptography (EECC) technique. In addition to this, the encryption and decryption process is formulated to enhance data integrity, security, confidentiality and authenticity. Also, the problem definition regarding two significant challenges namely security and service availability are stated in the subsequent section.

3.1. System Design

In the cloud environment, protection of sensitive data and secured communication are considered as the two significant concerns. Here, enhanced elliptic curve cryptography (EECC) technique (Banerjee and Patil, 2018) is employed to enhance the confidentiality of the data thereby storing it in the cloud. In this paper, the EECC generates the cryptographic keys by means of an elliptic curve because the EECC computes power to improve security and less battery source is required to provide power. On the other hand, the EECC offers protection against privacy, confidentiality, and authentication and the efficiency of the EECC is determined by point-multiplications (Rahnama et al. 2016). The following mathematical formulation describes the representation of elliptical curves using binary curve model.

Binary Curve Model

The elliptical curves are characterized by the binary curve. The symmetrical form of elliptical curve is represented by,

$$Y^2 = X^3 + mX + n \quad (1)$$

From equation (1), the standard variables and the elliptical curves are represented as X, Y and m, n respectively. By varying m, n the elliptical curve can be changed accordingly. On the other hand, the EECC utilizes two different kinds of keys namely the public key and the private key to encrypt and verify the signature as well as decrypt and generate signal (Devi et al. 2020). The EECC utilizes doubling operation and point addition and multiplication techniques that are expressed as follows.

Table 1. Summary of existing literature works

References	Techniques or Approaches	Simulation Measures	Merits	Limitations
Jena et al. (2020)	QMPSO	Standard deviation, Makespan, throughput, energy utilization	Enhanced throughput rate	Failed to implement the load balancing on reliable tasks
Semmoud et al. (2020)	STLB	Average idle time, total number of tasks, makespan, average response time	Less response time	High communication cost
Junaid et al. (2019)	DFTF	Accuracy, precision, f-measure, recall	Highly efficient	Failed to consider task immigrations
Devaraj et al. (2020)	FIMPSO	Average response time, average load, reliability, makespan	High memory utilization	High complexity
Siddiqui et al. (2019)	QPSL Queuing approach	Response time, throughput, waiting time	High service availability	Failed to consider weighted priority
Neelima et al. (2019)	ADA	Execution cost, execution time, makespan	Low execution cost and time	Inaccurate
Agarwalet al. (2020)	MPSO	Cloudlets, average makespan	High lifespan	Failed to consider various other measures like average time, resource utilization
Priya et al. (2019)	Fuzzy based MRSA	Average success rate, efficiency, response time	Enhanced latency and efficiency	Poor response time and average success ratio
Polepally et al. (2019)	DO and CMLB	Capacity, load, number of migrated task	Enhanced performance rate	Time required is very high
Hung et al. (2019)	MMSIA	Bandwidth, length, capacity	Enhanced completion time and quality of service	Complexity during implementation

Key Generation Process

During generation, public and private keys play a vital role. Initially, a number is selected randomly with a certain limit. Therefore,

$$P_U = R * P_R \tag{2}$$

From equation (2), P_U *and* P_R signifies the public and private keys. The selected random number is represented by R ; where $R \rightarrow [1 \; to \; n-1]$.

Encryption process

During the encryption process, messages are encoded so that it can be interpreted only by a particular person. The encryption process generates two cipher texts thereby storing them in the cloud storage system. Let us assume, l be the messages that are transmitted by the system. Hence,

$$C(1) = K * E \tag{3}$$

$$C(2) = L + K * F \tag{4}$$

From the above equations, the two generated chipper texts are represented by $C(1)$ *and* $C(2)$ respectively. The selected random number during the encryption process is represented by K ; where $K \rightarrow [1-(n-1)]$.

Decryption process

During the encryption process, messages are decoded and it's just the reverse process of encryption. In addition to this, the system retrieves the original messages. Thus,

$$L = C(2) - R * C(1) \tag{5}$$

Point Multiplication

In point multiplication, the two integers namely m *and* n are multiplied by employing the formula represented in equation (6).

$$P = m(0) * n(0) + \sum_{J=1}^{N=1} \sum_{K=0}^{J} m(J) * n(J-K) * 2^{16K} \tag{6}$$

From the above equation, the individual word of a number is denoted by $m(1), m(2), m(3) \ldots m(N-1)$ and $n(1), n(2), n(3) \ldots n(N-1)$ respectively.

3.2. Problem definition

In cloud computing technologies, providing security and service availability is considered a challenging task and it needs to be solved. In order to provide optimal service availability, it is necessary to establish a load balancing technique that is capable of balancing the request from diverse nodes present in the cloud. On the other hand, under-loading and over-loading cause system failure from different aspects namely execution time, machine fault, energy consumption, etc. Therefore in this paper, weintegrated adaptive neuro-fuzzy interference system-polynomial neural network (ANFIS-PNN) as well as memory-based grey wolf optimization algorithm for optimal dynamic load balancing thus establishing a load balancing technique using MG-ANFIS approach.

4. Proposed Mg-Anfis Based Dynamic Load Balancing Approach

In the cloud environment, security and service availability are the two most significant factors during the data encryption process. To optimal service availability, it is necessary to establish a dynamic load balancing technique using the APMG approach. Here in this paper, we utilized adaptive neuro-fuzzy interference system-polynomial neural network (ANFIS-PNN) as well as memory-based grey wolf optimization algorithm for optimal load balancing. In addition to this, turnaround time and CPU utilization are two significant factors involved in the load balancing scheme. The detailed description of the proposed MG-ANFIS based dynamic load balancing approach is stated in the subsequent section. Fig.1 describes the augmented flow diagram of the proposed approach.

4.1. Concept of ANFIS-PNN

In general, ANFIS-PNN structure is a hybrid method that is capable of solving engineering and science-related issues. Initially, the adaptive neuro-fuzzy interference system and polynomial neural network are combined to attain a highly accurate testing and training process. The output obtained from the ANFIS is provided as an input to develop a traditional PNN. The detailed description and the mathematical expression of both approaches are described as follows.

4.1.1 Adaptive Neuro-Fuzzy Integration System (ANFIS)

In general, the fuzzy system and neural network are integrated to form a hybrid technique. ANFIS comprises of two-fold mechanisms: Gradient Descent algorithm and least square approach (Adnan et al. 2015). The steps involved in ANFIS are discussed in the following section.

Layer 1: Fuzzifying layer

Let us assume the output node of the initial layer is $O_U(1,J)$ and the input layer with respect to the membership function is denoted by $A(J)$, $B(J-3)$. Then the mathematical formulations obtained in the Fuzzifying layer are expressed as follows (Ghorbanzadeh et al. 2020).

$$\begin{aligned} O_U(1,J) &= \delta_{A(J)}(A); \quad \forall J = 1,2,3 \\ O_U(1,J) &= \delta_{B(J-3)}(B); \quad \forall J = 4,5,6 \end{aligned} \tag{7}$$

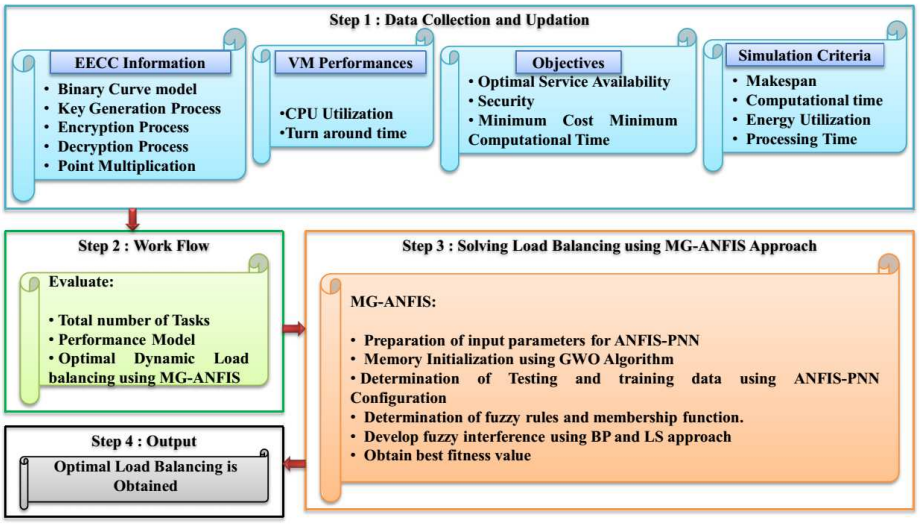

Fig.1. Augmented flow diagram of the proposed approach

Layer 2: Implication layer

The product of given input and the node present in the implication layer is termed as fixed node. Thus,

$$O_U(2,J) = w_J = \delta_{P(J)}(A).\delta_{Q(J)}(B); \quad \forall J = 1,2 \tag{8}$$

From the above equation, w_J signifies the firing weight of the *Jth* node respectively.

Layer 3: Normalizing layer

In the normalizing layer, the fixed nodes are labeled. The fraction of firing weight with respect to the *Jth* node to the sum of both firing weight is termed as the normalizing layer. Hence,

$$O_U(3,J) = \overline{w_U} = \frac{w_U}{w_1 + w_2}; \quad \forall J = 1,2 \tag{9}$$

Layer 4: Defuzzifying layer

In the Defuzzifying layer, the functions along with the linear combination are obtained. Therefore, the output expression obtained in the fourth layer is obtained as follows.

$$O_U(4,J) = \overline{w_J}.F_i = \overline{w_J}[(P_J(A) + Q_J(B) + R_J]; \quad \forall J = 1,2 \tag{10}$$

From equation (10), P_J, Q_J, R_J signifies the set of respective parameters.

Layer 5: Combining layer

The overall output function with respect to the above mentioned four layers is formulated in the combining layer.

$$O_U(5,J) = \sum_J \overline{w_J}.F_J = \overline{w_J}\frac{\sum_J \overline{w_J}.F_i}{\sum_J \overline{w_J}} \tag{11}$$

4.1.2 Polynomial Neural Network (PNN)

Generally, PNN is a self-organizing approach that generates a progressively complex model to evaluate the performances in accordance with the single input-multiple output data pair. PNN is represented by a neuron set in which each layer is correlated by a quadratic polynomial thus a new neuron is created in the upcoming layer (Chrysos et al. 2020). The mathematical formulations involved in PNN are expressed as follows.

$$Y_J = F(X_J(1), X_J(2), ...X_J(N)), \ where \ J = 1,.2,..m \tag{12}$$

From equation (12), the trained actual output value is represented by Y_J. Then from equation (12), the input vector is represented by

$$x = (X_J(1), X_J(2), ...X_J(N) \tag{13}$$

Then the estimated output value $\hat{Y}_J$ with respect to equation (12) and (13) is defined as,

$$\hat{Y}_J = \hat{F}(X_J(1), X_J(2), ...X_J(N)), \ where \ J = 1,.2,..m \tag{14}$$

In order to obtain minimum square error, the estimated and the obtained output value (Abdallah et al. 2020) is determined by,

$$Min\left\{\sum_{J=1}^{m}\left[\hat{F}(X_J(1), X_J(2), ...X_J(N) - \hat{Y}_J\right]^2\right\} \tag{15}$$

In addition to this, to formulate the overall correlation among the output and the input variables using Kolmogorov–Gabor polynomial is stated in equation (16).

$$Y = b_0 + \sum_{J=1}^{m} b_J x_J + \sum_{J=1}^{m}\sum_{K=1}^{m} b_{JK} x_J x_K + ... \tag{16}$$

From equation (16), the polynomials are denoted by b_0, b_J, b_{JK} ……The overall mathematical quadratic polynomial

expression with respect to neurons $g(x_J, x_K)$ is represented in equation (17). Thus,

$$\hat{Y} = g(x_J, x_K) = b_0 + b_1 x_J + b_2 x_K + b_3 x_J x_K \quad (17)$$

By employing equation (17), the coefficient of every quadratic function is obtained by fitting optimally the output from the input-output data pair. Therefore,

$$E = Min\left\{\frac{\sum_{J=1}^{m}\left[Y_J - g_J(x_J, x_K)\right]^2}{m}\right\} \quad (18)$$

Then by employing the least square approach, the optimal quadratic equation is obtained as follows.

$$b = (B^T B)^{-1} B^T y \quad (19)$$

From equation (19), $b = b_0, b_1, b_2, \ldots;$ $y = \left[Y_1, Y_2, Y_2 .. Y_m\right]^T$ where the transpose function is represented by *T*.

4.1.3 ANFIS-PNN Framework

In this section, the adaptive neuro-fuzzy interference system and polynomial neural network are combined to attain a highly accurate testing and training process (Harandizadeh et al. 2020). The steps involved in forming the ANFIS-PNN approach are delineated in the following section.

Step 1: Input data insertion

The initial phase of ANFIS-PNN involves in the insertion of input data; where the data is categorized into two different datasets namely the testing and the training datasets. Here, the testing and training datasets are represented by P *and* Q respectively.

Step 2: Fuzzification of input variables

The divided input variable is fuzzified and then converted into a fuzzy set by means of grid partition. Here the fuzzy operators are employed in the proper selection of number and membership functions. On the other hand, the parameters are selected more precisely since the time of execution is evaluated only by the selected parameters.

Step 3: Fuzzy implication

In the fuzzy implication phase, the respective fuzzy rules and the organization of the ANFIS network are evaluated. Here nine various fuzzy rules are formulated with respect to the obtained testing and training datasets. Therefore,

$$\begin{cases} Rule1: If\ X_1\ is\ P_1\ and\ X_2\ is\ Q_1;\ then\ Y_1 = b_1^0 + b_1^1 X_1 + b_1^2 X_2 \\ Rule2: If\ X_1\ is\ P_2\ and\ X_2\ is\ Q_2;\ then\ Y_2 = b_2^0 + b_2^1 X_1 + b_2^2 X_2 \\ \vdots \\ Rule9: If\ X_1\ is\ P_3\ and\ X_2\ is\ Q_3;\ then\ Y_9 = b_3^0 + b_3^1 X_1 + b_3^2 X_2 \end{cases} \quad (20)$$

Step 4: Identification of ANFIS parameters

In this phase, the parameters of ANFIS are determined on the basis of the ANFIS hybrid learning technique. The training and the testing dataset are evaluated in accordance with two various techniques namely the least square error and back propagation.

Step 5: Identification of PNN parameters

The output of the ANFIS is provided as an input to the PNN architecture for further expansion of the system design. Here, all linear parameters are paired whereas the training and the testing dataset are evaluated in accordance with the least square error technique.

Step 6: Model selection

The sixth phase involves in a selection of an appropriate model in which an optimal candidate solution is chosen with small criteria value.

Step 7: External criteria evaluation

In the final phase, the small criteria value obtained in the previous phase is compared among the previous and current generations. If the value of the present criteria is greater than the previous value an optimal model is achieved. Then the identification of PNN parameters, model selection as well as the external criteria evaluation processes are repeated if it fails to generate theminimum value. In addition to this, the parameters obtained during the implementation of the ANFIS-PNN method are optimized by employing a memory-based grey wolf optimization algorithm that is described briefly in the subsequent section.

4.2. Concept of memory-based GWO algorithm

In standard GWO algorithm (Gupta and Deep, 2020), the searching process indicates the dependencies of search direction only towards the leading wolf. The leading wolf often gets trapped by the local optima that further leads to premature convergence. So to overcome such shortcomings, a memory-based grey wolf optimization is employed in this paper thus to enhance the searching process and to strengthen the group effort among various wolves. The MGWO algorithm involves four main phases that are portrayed as follows.

Phase1: Hunting process

The hunting phase involves the collection of personal best data obtained from every individual wolf. Here, by employing the encircling technique, the hunting operation is performed. Therefore, it is necessary to modify the encircling equation initially. Thus,

$$Y_{T+1} = Y_\alpha - B_T \times \left| D_T \times Y_\alpha - Y_B \right| \quad (21)$$

From equation (21), the position of the wolf with respect to the iteration $T+1$ is represented by Y_{T+1}. The coefficient

vector responsiblefor exploitation and exploration is denoted by B_T .The coefficient vector employs in exploration during the failure of B_T is represented by D_T .The personal best value with respect to the memory of the wolf is represented by Y_B . Soon after the formulation of the encircling process, the hunting processes are carried out in accordance with the leading search direction from $\alpha, \beta and\ \Delta$ wolves. For an approximate location of prey, the hunting process employs equation (21). In the memory-based GWO, the individual wolf is capable of sharing its knowledge based on the search space area. Therefore, the updated mathematical expression in terms of the hunting process is explained in equation (22).

$$X_{J,T+1} = \frac{Z_1 + Z_2 + Z_3}{3} \tag{22}$$

From equation (22), the wolves' updated position using the hunting process is denoted by $X_{J,T+1}$.

$$\begin{aligned} Z_1 &= Y_\alpha - B_{\alpha,T} \times \left| D_{\alpha,T} \times Y_\alpha - Y_{JB} \right| \\ Z_2 &= Y_\beta - B_{\beta,T} \times \left| D_{\beta,T} \times Y_\beta - Y_{JB} \right| \\ Z_3 &= Y_\Delta - B_{,T} \times \left| D_{\beta,T} \times Y_\beta - Y_{JB} \right| \end{aligned} \tag{23}$$

The personal best wolf state stored in the wolf memory is denoted by Y_{JB} .

Phase 2: Searching process

During the searching process, the mutual pack strength is enhanced by obtaining a new search equation using random wolves and the personal best data. In order to imitate the characteristic features of an individual wolf as well as to retrace and explore the neighbouring areas, it is necessary to formulate a search equation. Therefore,

$$\hat{Y}_{T+1} = Y_{JB} + S_F \times \left[Y(R_1) - Y(R_2) \right] \tag{24}$$

From equation (24), $\hat{Y}_{T+1}$ and S_F signifies the updated search position and scaling factor where the effect based on the difference control vector is controlled. On the other hand, the scaling factor linearly decreases from 1 to 0. The smaller value of the scaling factor results in exploitation whereas the high value results in exploration. $Y(R_1)\, and\, Y(R_2)$ signifies the random value that ranges from 0 to 1.

Phase 3: Crossover operation

Here, the crossover operation is executed among various positions by employing a modified form of hunting mechanism. Here, the individual wolf and the leading hunter provide the information regarding the prey followed by merging. Therefore the mathematical expression in terms of cross over operator is determined by,

$$Y^I_{J,T+1} = \begin{cases} X^I_{J,T+1};\ if\ R_3 < C_P;\ \ Here\, C_P = 0.5 \\ Y^I_{J,T+1};\ otherwise \end{cases} \tag{25}$$

From equation (25), the crossover probability rate is denoted by C_P. R_3 signifies the random number that ranges from (0,1).

Phase 4: Restoring process

In the final phase, the information is restored by employing a greedy selection techniquefrom the search space area. Also, the greedy search technique is employed in selecting the optimal wolf among two successive iterations.

4.3. MGANFIS based dynamic load balancing framework

Generally, every input in ANFIS comprises numerous membership functions and at someplace the membership functions are utmost. But the rapid variation in membership functions affects the estimation accuracy. Here the memory-based grey wolf optimization algorithm is employed to enhance the precision of ANFIS-PNN and to maximize the locations of the membership functions respectively.The ANFIS-PNN and MGWO approaches are employed in identifying the load balancing issues. In addition to this, the MGW is capable of updating the parameters and updation of the ANFIS-PNN design. The steps involved in the proposed MG-ANFIS based dynamic load balancing approach are described as follows and the flow diagram is represented in fig. 2

Step 1: The initial phase involves the preparation of input parameters for the ANFIS-PNN approach to obtain an appropriate output.

Step 2: In this phase, the initialization of memory-based grey wolf optimization algorithm is initiated. Here the parameters of the MGWO algorithm and population initialization process are configured.

Step 3: In the ANFIS-PNN configuration, the testing data and the training data are determined. Here, 70% of the data are involved in the training process and the rest 30% is employed for testing the model.

Step 4: This phase involves the determination of membership function and fuzzy rules as mentioned in equation (20). Here nine various fuzzy rules are formulated with respect to the obtained testing and training datasets.

Step 5: The back propagation and the least square technique is employed in developing the fuzzy interference system. Then accordingly the datasets are simulated.

Step 6: In this step, the fitness function is evaluated and if the criteria are not met, the hunting, searching and cross-over operations for optimal load balancing are repeated till the stopping criteria are satisfied.

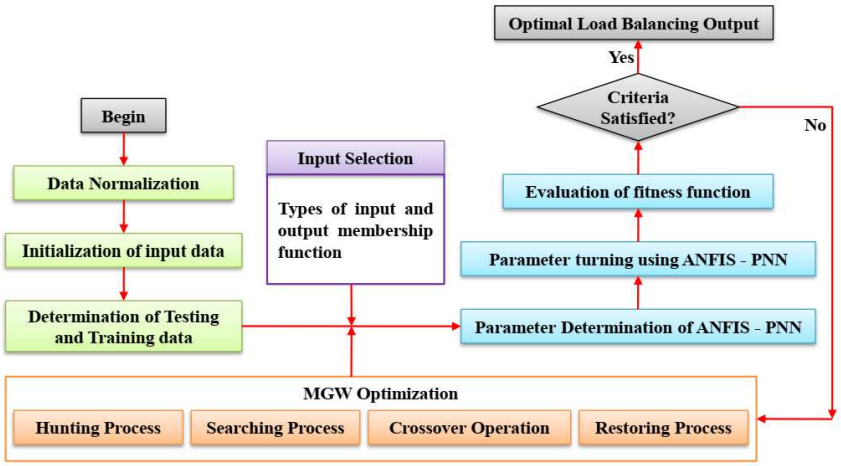

Fig.2. MG-ANFIS based dynamic load balancing approach

Table 2. Parameter Settings

Methods	Parameter Metrics	Values
ANFIS	Number of layers	4 input, 1 output
	Learning rules	Hybrid learning
	Number of epochs	225
	Type of output membership function	Constant
	Type of input membership function	Gaussian
	Type of inference	Linear Sugeno
	Training algorithm	Back propagation
	Transfer function	Linear
PNN	Number of layers	4
	Training technique	Radial basis
	Scaling approach	Normalization
	Total number of input variables	25
MGWO	Population size	30
	Total number of iterations	100
	Crossover probability	0.5
	Scaling factor	1,0

5. Evaluation Results

This section describes the experimental analysis for the proposed MG-ANFIS based dynamic load balancing in the cloud. The performances for various simulation metrics namely cost energy utilization, Makespan and processing time. The performance evaluation and comparison performances of the proposed approach are described in the following section.

5.1. Experimental Configuration

The experiments for the proposed MG-ANFIS based dynamic load balancing approach is implemented under the JAVA platform, 500GB, 4GB RAM and Windows 7 Operating System. Here, processors are employed in the execution of various cloudlets.

5.2. Parameter specifications

The parameter specification of ANFIS-PNN and memory-based gray wolf optimization algorithm are set accordingly (see Table 2).

5.3. Simulation measures

The performance metrics employed in this proposed approach for dynamic load balancing in the cloud are discussed in the consequent section. As mentioned above, turnaround time and CPU utilization are two significant factors involved in the load balancing scheme.

Turnaround time

The term turnaround time is defined as the time difference among completion time and arrival time. The mathematical expression in terms of both times is represented in equation (26)

$$T_t = C_t - A_t \tag{26}$$

From the above equation, the turnaround time, completion time and arrival time is represented as T_t, C_t *and* A_t.

CPU Utilization

The mathematical formulation for CPU capacity percentage is expressed in equation (27).

$$CPU(U) = P_{ts}\ \% \tag{27}$$

From the above equation, the CPU utilization rate is denoted by $CPU(U)$. P_{ts} signifies the total time spent in the idle tasks.

Makespan time

Makespan time is the time occupied among the commencing of implementation till the end of the program.

Service Availability

The service (cryptography and security services) provided during authentication and data transferring for responding to the request of the user.

Energy Utilization

Energy utilization in other words referred to as energy efficiency that specifies the entire infrastructure to minimize the cost functions thereby providing an effective, efficient network at various nodal points.

5.4. Evaluation performances

This section provides the performances of the proposed approach by employing the following parameters: cost, energy utilization, makespan, processing time. The graphical representation with respect to the above-mentioned parameters is described in the subsequent section.

Cost

In general, the cost based on the cloud service provider relies on CPU utilization of active resources. Here, fig.3 describes the graphical analysis for cost and the total number of tasks. The x-axis represents the cost function whereas the y-axis represents the total number of tasks. From the analysis, it is noted that the cost function enhances with respect to total number of tasks. This describes that the proposed MG-ANFIS approach achieves higher performances.

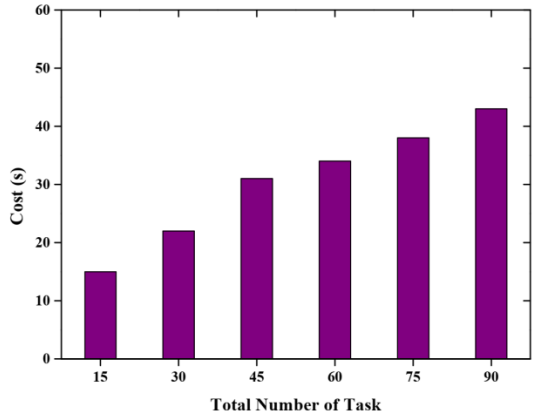

Fig.3. Cost analysis with respect to total number of task

Energy Utilization

The resource pool is served for multi-users using multi-tenant approach. Here, the virtual resources are dynamically allocated and re-allocated in accordance with the requirement of the user. The energy utilization graph is plotted in fig.4; where x-axis represents the total number of tasks performed and y-axis signifies the utilization of resources. Here, by employing our proposed MG-ANFIS approach, optimal load balancing is conducted with respect to two major factors namely the turnaround time and CPU utilization. On the other hand, energy utilization is enhanced by selecting the virtual machine optimally.

Makespan

Fig.5 describes the graphical analysis for the makespan time and the total number of respective tasks. Here the total numbers of tasks (30, 60, 90, 120, 150 and 180) are represented on the x-axis and the y-axis denotes the makespan time (ms). Here, from the figure, it is well noted that the makespan time increases with an increase in task numbers. Thus, the makespan time provides an effective result by employing MG-ANFIS based dynamic load balancing approach.

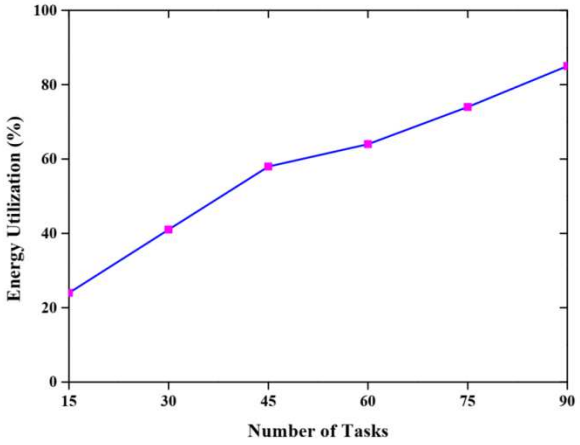

Fig.4. Analysis based on Energy Utilization

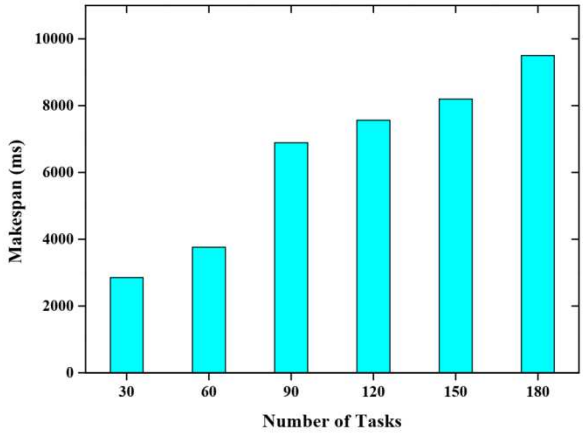

Fig.5. Evaluation of makespan with respect to total number of tasks

Processing time

This paper utilizes an enhanced elliptical curve cryptography approach is employed to enhance security and to minimize the processing time. Fig.6 provides the graphical representation for processing time measured in ms and the total number of tasks. The x-axis represents the total task number and the processing time is represented on y-axis. From the graphical analysis, it is noted that the processing time obtained by the proposed MG-ANFIS based dynamic load balancing approach is very low.

5.5. Performance Comparisons

This section describes the comparative performances of various load balancing approaches namely Particle swarm optimization based load balancing approach (Ramezani et al. 2014), Fuzzy based Hybrid firefly optimization algorithm (Shri et al. 2018), Modified adaptive neuro-fuzzy optimization algorithm (Devi et al. 2020) as well as MG-ANFIS.

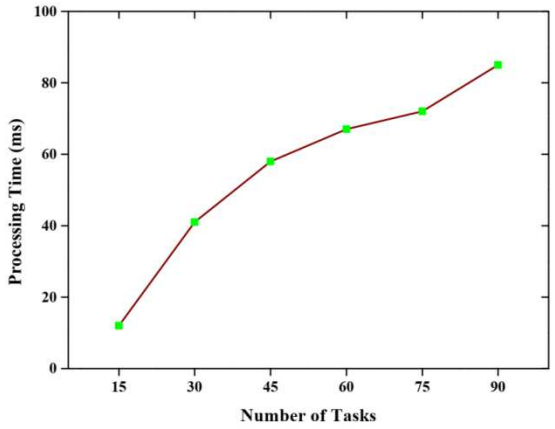

Fig.6. Processing time evaluation

Fig.7 describes the comparative analysis for makespan time with respect to number of approaches. Here, approaches are represented in the x-axis and the y-axis denotes the makespan time (ms). The comparative analysis is carried out for various existing approaches namely the PSO, FHFO, MANFIS and the proposed MG-ANFIS approach. The analysis reveals that the proposed approach provides greater makespan when compared with all other existing approaches.

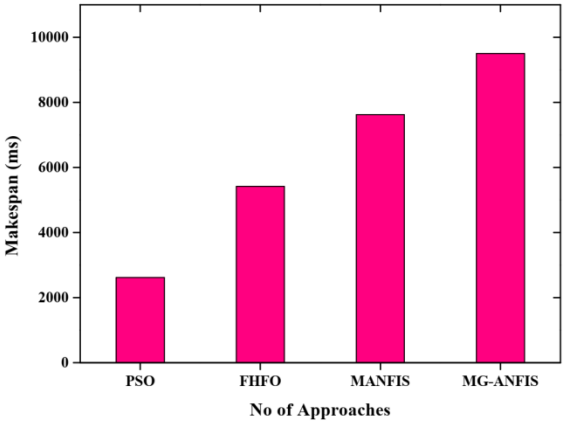

Fig.7. Comparative evaluation in terms of makespan

The comparative graphical representation in terms of service availability and various load balancing approaches are employed in fig.8. Here, the x-axis represents various existing approaches and y-axis signifies service availability rate (%).The comparative analysis is carried out for various existing approaches namely the PSO, FHFO, MANFIS and the proposed MG-ANFIS approach. The evaluation results reveal the proposed MG-ANFIS based load balancing approach provides a service availability of 97%. This proves that the proposed approach outperforms various other existing approaches.

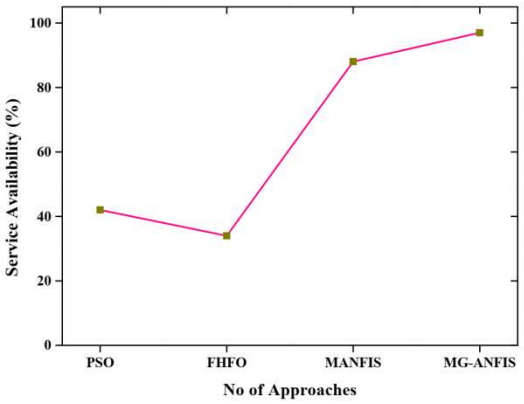

Fig.8. Comparative evaluation in terms of service availability

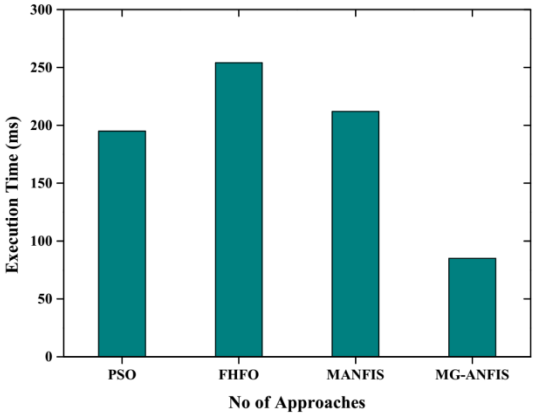

Fig.9. Execution time analysis for various approaches

Fig.9. provides the graphical analysis based on the execution time for various approaches namely the PSO, FHFO, MANFIS, and the proposed MG-ANFIS approach. Here, the x-axis represents various existing approaches and the y-axis signifies execution time (ms).The experimental analysis is conducted and the results reveal that the proposed approach provides an execution time of about 85ms. But the execution time of other existing approaches seems a bit higher when compared with the proposed approach.

6. Conclusion

Load balancing is considered one of the major issues in cloud computing technologies. By employing the load balancing technique, the dynamic workloads are distributed equally among various nodes that remove inactive nodes and prevent overloading. In order to provide optimal service availability, it is necessary to establish a load balancing technique that is capable of balancing the request from diverse nodes present in the cloud. This paper proposed a dynamic load balancing technique using the APMG approach. Here, an adaptive neuro-fuzzy interference system-polynomial neural network (ANFIS-PNN), as well as memory based grey wolf optimization algorithm, is integrated for optimal dynamic load balancing. The MGWO algorithm is employed to enhance the precision of ANFIS-PNN and to maximize the locations of the membership functions. The experimental analyses are carried out for the proposed MG-ANFIS based dynamic load

balancing in the cloud. The performances for various simulation metrics namely cost, energy utilization, Makespan and processing time and the proposed MG-ANFIS approach achieves higher performances for all simulation measures. Furthermore, the comparative analysis is carried out for various existing approaches namely the PSO, FHFO, MANFIS and the proposed MG-ANFIS approach. The analysis reveals that the proposed approach provides greater Makespan when compared with all other existing approaches. In our futurework, the optimization model is extended with new objective functions to attain better load balancing performances and service availability.

Compliance with Ethical Standards

Conflict of interest

The authors declare that they have no conflict of interest.

Human and Animal Rights

This article does not contain any studies with human or animal subjects performed by any of the authors.

Informed Consent

Informed consent was obtained from all individual participants included in the study.

References

Abdallah, Habib Ben, Christopher J. Henry, and Sheela Ramanna. (2020) 1-Dimensional polynomial neural networks for audio signal related problems. arXiv preprint arXiv:2009.04077.

Adnan, Sarkheyli, Zain and Haron, (2015) Fuzzy logic for modeling
machining process, Artif.Intell. Rev., 43(3): pp. 345–379.

Agarwal, Ronak, Neeraj Baghel, and Mohd Aamir Khan. (2020) Load Balancing in Cloud Computing using Mutation Based Particle Swarm Optimization. In 2020 International Conference on Contemporary Computing and Applications (IC3A), pp. 191-195. IEEE, 2020.

Arulkumar, V., and N. Bhalaji. (2020) Performance analysis of nature inspired load balancing algorithm in cloud environment." Journal of Ambient Intelligence and Humanized Computing, pp: 1-8.

Banerjee S, Patil A (2018) ECC based encryption algorithm for lightweight cryptography. In: International conference on intelligent
systems design and applications, pp 600–609

Chrysos, G.G., Moschoglou, S., Bouritsas, G., Panagakis, Y., Deng, J. and Zafeiriou, S., (2020). P-nets: Deep Polynomial Neural Networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 7325-7335).

Devi, TJB Durga, A. Subramani, and P. Anitha. (2020) Modified adaptive neuro fuzzy inference system based load balancing for virtual machine with security in cloud computing environment. Journal of Ambient Intelligence and Humanized Computing, pp: 1-8.

Devaraj, A.F.S., Elhoseny, M., Dhanasekaran, S., Lydia, E.L. and Shankar, K., (2020). Hybridization of firefly and Improved Multi-Objective Particle Swarm Optimization algorithm for energy efficient load balancing in Cloud Computing environments. Journal of Parallel and Distributed Computing.

Ghorbanzadeh, Omid, Thomas Blaschke, Jagannath Aryal, and Khalil Gholaminia. (2020) A new GIS-based technique using an adaptive neuro-fuzzy inference system for land subsidence susceptibility mapping. Journal of Spatial Science 65(3): 401-418.

Gupta, S., & Deep, K. (2020). A memory-based Grey Wolf Optimizer for global optimization tasks. Applied Soft Computing, 106367.

Gupta, Abhishek, H. S. Bhadauria, and Annapurna Singh. (2020) SLA-aware load balancing using risk management framework in cloud. Journal of Ambient Intelligence and Humanized Computing pp: 1-10.

Harandizadeh, Hooman, and Danial Jahed Armaghani. (2020) Prediction of air-overpressure induced by blasting using an ANFIS-PNN model optimized by GA. Applied Soft Computing, pp: 106904.

Hung, Tran Cong, Le Ngoc Hieu, Phan Thanh Hy, and Nguyen Xuan Phi. (2019) MMSIA: improved max-min scheduling algorithm for load balancing on cloud computing. In Proceedings of the 3rd International Conference on Machine Learning and Soft Computing, pp. 60-64.

Jena, U. K., P. K. Das, and M. R. Kabat. (2020) Hybridization of meta-heuristic algorithm for load balancing in cloud computing environment. Journal of King Saud University-Computer and Information Sciences.

Junaid, Muhammad, Adnan Sohail, Rao Naveed Bin Rais, Adeel Ahmed, Osman Khalid, Imran Ali Khan, Syed Sajid Hussain, and Naveed Ejaz. (2019) Modeling an Optimized Approach for Load Balancing in Cloud. IEEE Access 8: 173208-173226.

Jyoti, A. and Shrimali, M., (2020). Dynamic provisioning of resources based on load balancing and service broker policy in cloud computing. Cluster Computing, 23(1), pp.377-395.

Kaur, Navleen, Jaspreet Singh, Shanky Goyal, and Bharti Duhan. (2020) LOAD BALANCING IN CLOUD COMPUTING: THE ONLINE TRAFFIC MANAGEMENT. Journal of Natural Remedies 21(2): 202-209.

Kong, Lingfu, Jean Pepe Buanga Mapetu, and Zhen Chen. (2020) Heuristic load balancing based zero imbalance mechanism in cloud computing. Journal of Grid Computing 18(1): 123-148.Ghh

Maswood, Mirza Mohd Shahriar, MD Rahinur Rahman, Abdullah G. Alharbi, and Deep Medhi. (2019) A Novel Strategy to Achieve Bandwidth Cost Reduction and Load Balancing in a Cooperative Three-Layer Fog-Cloud Computing Environment. IEEE Access 8: 113737-113750.

Mukhopadhyay, B., Bose, R. and Roy, S., (2020). A Novel Approach to Load Balancing and Cloud Computing Security using SSL in IaaS Environment. International Journal, 9(2).

Naresh, A., V. Pavani, M. Meghana Chowdary, and V. Lakshman Narayana. (2020) Energy consumption reduction in cloud environment by balancing cloud user load. Journal of Critical Reviews 7(7): 1003-1010.

Neelima, P., and A. Rama Mohan Reddy. (2019) An efficient load balancing system using adaptive dragonfly algorithm in cloud computing. Cluster Computing (2020): 1-9.

Polepally, Vijayakumar, and K. Shahu Chatrapati. (2019) Dragonfly optimization and constraint measure-based load balancing in cloud computing. Cluster Computing, pp: 1-13.

Priya, V., C. Sathiya Kumar, and Ramani Kannan. (2019) Resource scheduling algorithm with load balancing for cloud service provisioning. Applied Soft Computing 76: 416-424.

Rahnama B, Sari A, Ghafour MY (2016) Countering RSA vulnerabilities and its replacement by ECC: elliptic curve cryptographic
scheme for key generation. In: Network security attacks and countermeasures, pp 270–312

Ramezani, Fahimeh, Jie Lu, and Farookh Khadeer Hussain. (2014) Task-based system load balancing in cloud computing using particle swarm optimization. International journal of parallel programming 42(5): 739-754.

Semmoud, Abderraziq, Mourad Hakem, Badr Benmammar, and Jean-Claude Charr. (2020) Load balancing in cloud computing environments based on adaptive starvation threshold. Concurrency and Computation: Practice and Experience 32(11): e5652.

Siddiqui, Shadab, Manuj Darbari, and Diwakar Yagyasen. (2019) An QPSL Queuing Model for Load Balancing in Cloud Computing. International Journal of e-Collaboration (IJeC) 16(3): 33-48.

Shri, M. Lawanya, et al. (2018) A Fuzzy Based Hybrid Firefly Optimization Technique for Load Balancing in Cloud Data centers. International Conference on Innovations in Bio-Inspired Computing and Applications. Springer, Cham.

Tiwari, P.K., Rani, G., Jain, T., Mundra, A. and Gupta, R.K., (2020). Load Balancing in Cloud Computing: Challenges and Management Techniques. In Critical Approaches to Information Retrieval Research (pp. 294-316). IGI Global.

Ullah, Arif, Nazri Mohd Nawi, and Mubashir Hayat Khan. (2020) BAT algorithm used for load balancing purpose in cloud computing: an overview. International Journal of High Performance Computing and Networking 16(1): 43-54.