Advances in Nonlinear Variational Inequalities
ISSN: 1092-910X
Vol 28 No. 7s (2025)

A Blockchain-Based Trust Evaluation Mechanism for Intrusion Detection in WSNs Using Improved Dove Swarm Optimization (IDSO) and Hybrid improved whale optimization algorithm with enhanced genetic Algorithm (IWOA-IGA)

[1*]P. Vijayalakshmi, [2]Dr. P. M. Gomathi

[1*]Assistant professor, Department of Computer Science, P.K.R. Arts College for Women, Gobichettipalayam -638452

Mail Id: vijiperumalsas@gmail.com

[2]Dean & Associate professor, Department of Computer Science, P.K.R. Arts College for Women, Gobichettipalayam

Mail Id: gomathipm@pkrarts.org

Article History:

Received:21-12-2024

Revised:03-02-2025

Accepted:18-2-2025

Abstract: Block chain technology can provide an efficient security mechanism for creating a trusted network in Wireless Sensor Networks (WSNs). By utilizing distributed ledger technology, network nodes can securely exchange and verify information. This study further explores the potential of block chain technology in enhancing the security and trustworthiness of WSNs. An efficient security mechanism using block chain technology can be implemented in WSNs to establish a trusted network. Block chain ensures secure communication by providing decentralized consensus, data integrity, and tamper-proof verification. In this work initially all the nodes in the wireless sensor network will form clusters based on the communication range to reduce the energy consumption and to increase the network lifetime. And then cluster heads will be selected using Improved Dove Swarm Optimization (IDSO). Once the nodes sensed the information it will send the gathered data to its cluster heads and cluster heads will transfer it to the base station. And then to detect attacks accurately significant features will be selected using hybrid improved whale optimization algorithm with enhanced genetic properties (IWOA-IGA). Finally blockchain-based trust evaluation mechanisms based intrusion detection will be performed based on Integrating Modified Elman Neural Network (MENN) with Graph Neural Networks (GNNs) called MENN-GNN to establish secure communication in wireless sensor network. Also a Trust-Based Intrusion Detection System (IDS) in WSNs using the NSL-KDD'99 dataset combines trust management mechanisms and IDS techniques to detect and mitigate security threats in WSNs. This approach leverages the notion of trust to monitor and evaluate the behaviour of sensor nodes and identify potential malicious activities. The system leverages the NSL-KDD'99 dataset for robust detection of security threats, extending network lifetime and reducing energy consumption.

Keywords: Wireless Sensor Networks (WSNs); Intrusion Detection System (IDS); Security attacks; Types of Intrusion Detection, Block chain technology, Trust factor, Dove Swarm Optimization, Modified Elman Neural Network (MENN), Graph Neural Networks (GNNs).

I.INTRODUCTION

The Wireless Sensor Networks (WSNs) play a significant role in many areas, such as medical, military, surveillance, industrial, etc., [1]–[3]. A WSN is a self-organizing network where Sensor Nodes (SNs) are randomly deployed and have limited storage, energy and computational power [4]–[6]. They monitor the parameters like temperature, humidity, etc. and transmit the data towards the Base Stations (BSs) [7]. However due to the limited resources, the attackers can easily attack the SNs [8].

A blockchain is an emerging technology that consists of nodes. The nodes maintain the state of a distributed ledger. It efficiently keeps a record of transactions among multiple parties [9], [10]. No one can easily perform data tampering as blockchain is immutable. The data in blockchain is secure because the blocks are connected via hashes. Moreover, each block also consists of the block header and the block body [11]. The root hash of the Merkle Tree is present in the block header. While the block body comprises of the transactions. If any data tampering is performed, it can be easily identified by comparing the hash of the data with the root hash [12]. In WSNs, the security threats are becoming more serious day by day [13], [14]. There are two types of attacks namely internal and external in WSN. In internal attack, nodes behave selfishly while in external attack; attackers force the nodes to perform malicious activities. Therefore, it is very crucial to identify malicious nodes and remove them from the network. In WSNs, the detection of malicious nodes is widely studied and divided into WSNs protocols and trust model.

In a network without authentication, any node can enter in the network and forge the identity of benign nodes. The adversary nodes broadcast wrong route information on the behalf of the compromised nodes. The existing malicious node detection methods do not guarantee the authentication of nodes. Moreover, the malicious node detection is performed on the basis of trust value only. Different metrics are used to check the node's fairness; however, detection is performed at a later stage which consumes more resources of the network [8], [15], [16]. The presence of a large number of malicious nodes in the network has negative impact on the routing [17], [18]. The malicious nodes modify the data and broadcast wrong route information which affects the performance of the network. The attacks in WSNs occur in two ways one of them is data and other is routing. In data attack, the adversary nodes perform data tampering. While in routing attack, adversary nodes choose a least energetic route that depletes the energy of SNs. The efficiency of the network is affected in terms of high delay and low packet delivery ratio [19], [20]. Moreover, the SNs are resource constrained devices that are vulnerable to different attacks like spoofing and impersonation, etc. To overcome the challenges mentioned above, blockchain based authentication and trust evaluation mechanism for secure routing in the WSNs is proposed. In [19], authors propose a trust aware localized routing scheme for WSNs. However, the authentication and malicious node detection are not considered. Due to this, unregistered nodes can access the network resources and data [15]. The malicious nodes can enter in the network and forge the real identity of the nodes. Moreover, they drop the data packets, which increase the network delay and decrease the packet delivery ratio [20]. The contributions of this paper are summarized as follows

1. An Improved Dove Swarm Optimization (IDSO) approach for selecting energy-efficient cluster heads.
2. A hybrid Improved Whale Optimization Algorithm with Genetic properties (IWOA-IGA) for selecting significant features.
3. An intrusion detection system integrating Modified Elman Neural Network (MENN) with Graph Neural Networks (GNNs) for high detection accuracy.
4. A blockchain-based trust evaluation mechanism to securely maintain and update node trust scores, leveraging the NSL-KDD'99 dataset for IDS training and performance validation.

This is how the remainder of the paper is structured. Brief introductions to relevant works are provided in Section II. The proposed system model is explained in Section III. Section IV displays the outcomes of simulation tests used to assess performance. This work is finally summarized in Section V.

II.RELATED WORK

Almomani et al. [21] developed a specialized dataset called WSN-DS, tailored for WSN intrusion detection. They employed an artificial neural network (ANN) on this dataset, which resulted in enhanced accuracy in the detection and classification of DoS attacks. Subsequently, in 2023, Dener et al. [22] introduced a novel dataset, WSN-BFSF, specifically designed to detect DoS attacks in WSNs. Their investigation involved the evaluation of four ML and eight deep-learning models, yielding notable outcomes.

Vinayakumar et al. [23] devised a hybrid IDS for detecting and classifying attacks. They introduced a scalable deep neural network (DNN) framework called scale-hybrid-IDS-AlertNet, which was designed to combat network attacks. The proposed methodology assessed diverse datasets, including CICIDS2017, WSN-DS, UNSW-NB15, and NSL-KDD, and exhibited commendable accuracy across various network traffic types. However, their study lacked a comprehensive discussion regarding computational time efficiency [24].

Salmi and Oughdir [25] developed and implemented various models, including DNN, convolutional neural networks (CNNs), recurrent neural networks (RNNs), and hybrid RNN–CNN architectures, for the detection of DoS attacks in WSNs. The models were trained using the WSN-DS dataset, with the CNN exhibiting superior performance with an accuracy rate of 98.79%. Notably, compared to ML models, deep learning models introduce a heightened computational overhead. Given the inherent constraints of WSNs, lightweight security solutions are imperative for addressing these network limitations.

Hussain et al. [26] proposed an IDS designed specifically for WSNs, utilizing a CNN architecture for classification and a hybrid WOA–artificial bee colony (ABC) algorithm for FS. When combined with the recommended CNN architecture, this method outperformed PSO and achieved an overall accuracy of 98%, according to the experimental evaluation on the NSL-KDD dataset. However, it is important to mention that the hybrid WOA-ABC algorithm, which combines two large optimization algorithms, might increase the computing complexity and resource requirement.

Mohiuddin et al. [27] presented an effective IDS that combines a whale optimization technique, which is a modified wrapper-based algorithm with a sine-cosine FS algorithm and a Weighted Extreme Gradient Boosting (XgBoost) classifier. The objective of this method is to enhance the prediction quality and prevent local optima by broadening the search and effectively selecting an ideal solution using the sine–cosine function. Using the UNSW-NB15 and CICIDS datasets, the proposed model successfully classified binary and multi-class attacks with notable accuracy, precision, recall, and F1-score metrics. Nevertheless, significant constraints exist concerning the scalability, computational efficiency, and applicability of the proposed model to various network contexts that have not been discussed.

Blockchain-based secure routing and trust management (BSRTM) has been demonstrated in WSNs by Saba Awan et al. [28]. In this article, a concept for encryption and trust assessment was put out using a blockchain to hold the identities of ANs and SNs. ANs and SNs were authenticated in private and public blockchains, correspondingly. To remove malicious nodes from the system, the trust levels of SNs were determined. Consideration was given to the SNs' trust values and residual energy when performing secure routing within the system. Additionally, RSA, an asymmetric key encryption algorithm, was employed to secure the communication protocol. This suggested paradigm used BSs, which have powerful computational capabilities, to identify and validate ANs.

A blockchain technology method has been demonstrated by Sung-Jung Hsiao and Wen-Tsai Sung [29] to improve the data security of WSNs. This study builds a secure WSN framework using data transfer and blockchain-based technology. The current wireless technology uses a blockchain-based approach to strengthen transmission dependability and is built on IoT architecture. This research methodology transforms the blockchain-based transaction ledger into a database of sensor data. As a result, the suggested system collects and evaluates sensor data to increase the dependability of the wireless sensing network. Additionally, this cutting-edge blockchain-based system has the ability to handle a private cloud end. However, as the amount of data grows, the amount of time required for the computation process grows, decreasing the quality of the data transfer.

A trusted distributed routing strategy for ESN combining blockchain and the Deep Convolutional Neural Network (DCNN) algorithm has been presented by M Revanesh and Venugopalachar Sridhar [30]. The distributed routing information in the WSN is managed using a blockchain technique, and the Salp Swarm Optimization algorithm is used to achieve the best routing. Using DCNN, the best routing choices were made while taking into account the variances in routing information between the nodes. In order to create a reliable routing method for WSN, this article intercorrelates the blockchain notion with DCNN. In the PoA blockchain, every routing node was represented as a minion with limited privileges and rights but a distinct address.

In order to improve the security and effectiveness of WSNs' routing, Ibrahim A. et al. [31] have developed a trusted routing system that integrates deep blockchain and Markov decision processes (MDPs). The suggested method makes use of a PoA technique within the blockchain network to authenticate the transmission of the node. The correct next hop was then selected

using MDPs as a forwarding node capable of swiftly and securely delivering messages. The MDPs' design was adopted to help routing nodes choose the most trustworthy and effective route connections and to make more informed routing decisions. Backward error propagation requires an additional layer to accomplish the operation.

In the system, Sana Amjad et al. [32] showed how to use DDR-LEACH, a low-energy adaptive clustering hierarchy for blockchain-based authentication and CH selection. Based on maximum residual energy, degree, and minimum distance from BS, the DDR-LEACH protocol was utilized to replace CHs with regular nodes. Additionally, using the blockchain to store a significant amount of data is highly expensive. The inter-planetary file system (IPFS), which operates in a superior fashion to the current encryption systems, was employed to ensure data security. Furthermore, employing a consensus protocol approach to validate transactions required a significant computing investment. The suggested manner in this case was using the PoA consensus technique. However, PoA has a high processing cost and blockchain data storage is quite expensive.

The blockchain method has been used by Rekha Goyat et al. [33] to demonstrate a secure localization strategy for WSNs based on trust evaluation. This study focuses on the development of blockchain technology and the trust-evaluation algorithm. Each beacon node's trust value was calculated during the localization system based on several trust measures, and the relevant weights were then continuously changed. Higher trustworthy beacon nodes were then solely chosen for the mining process. Beacon nodes with the greatest blockchain trust levels carry out the mining operation. Proof of work (PoW) consensus is primarily utilized as a mathematical problem for certifying newly created blocks, although PoW demands intensive processing and substantial resources to carry out the activity of mining, with high energy consumption.

A routing technique that offers a shared memory across the network's nodes by utilizing blockchain technology has been demonstrated by Hilmi Lazrag et al. [34]. The technology is quite young; it originally gained popularity in the world of cryptocurrencies, and recently, it has been used in numerous industries where it has demonstrated its effectiveness. Presented is a methodology that takes advantage of the potential of Blockchains in order to obtain a better resolution to the imbalanced traffic load, the high levels of disturbance, and the security problems. The strategy entails using blockchain to store all network activity as a shared memory across nodes. Despite the positive outcomes, using a cost function to decide on a route and make decisions is only presented as a proof of concept.

III.PROPOSED METHODOLOGY

The suggested IDS in WSNs are covered in detail in this section. In this work initially all the nodes in the wireless sensor network will form clusters based on the communication range to reduce the energy consumption and to increase the network lifetime. And then cluster heads will be selected using Improved Dove Swarm Optimization (IDSO). And then to detect attacks accurately significant features will be selected using hybrid improved whale optimization algorithm with enhanced genetic properties (IWOA-IGA). Finally blockchain-based trust

evaluation mechanisms based intrusion detection will be performed based on Integrating MENN with Graph Neural Networks (GNNs) to establish secure communication in WSN. Figure 1 depicts the overall architecture of the suggested paradigm.

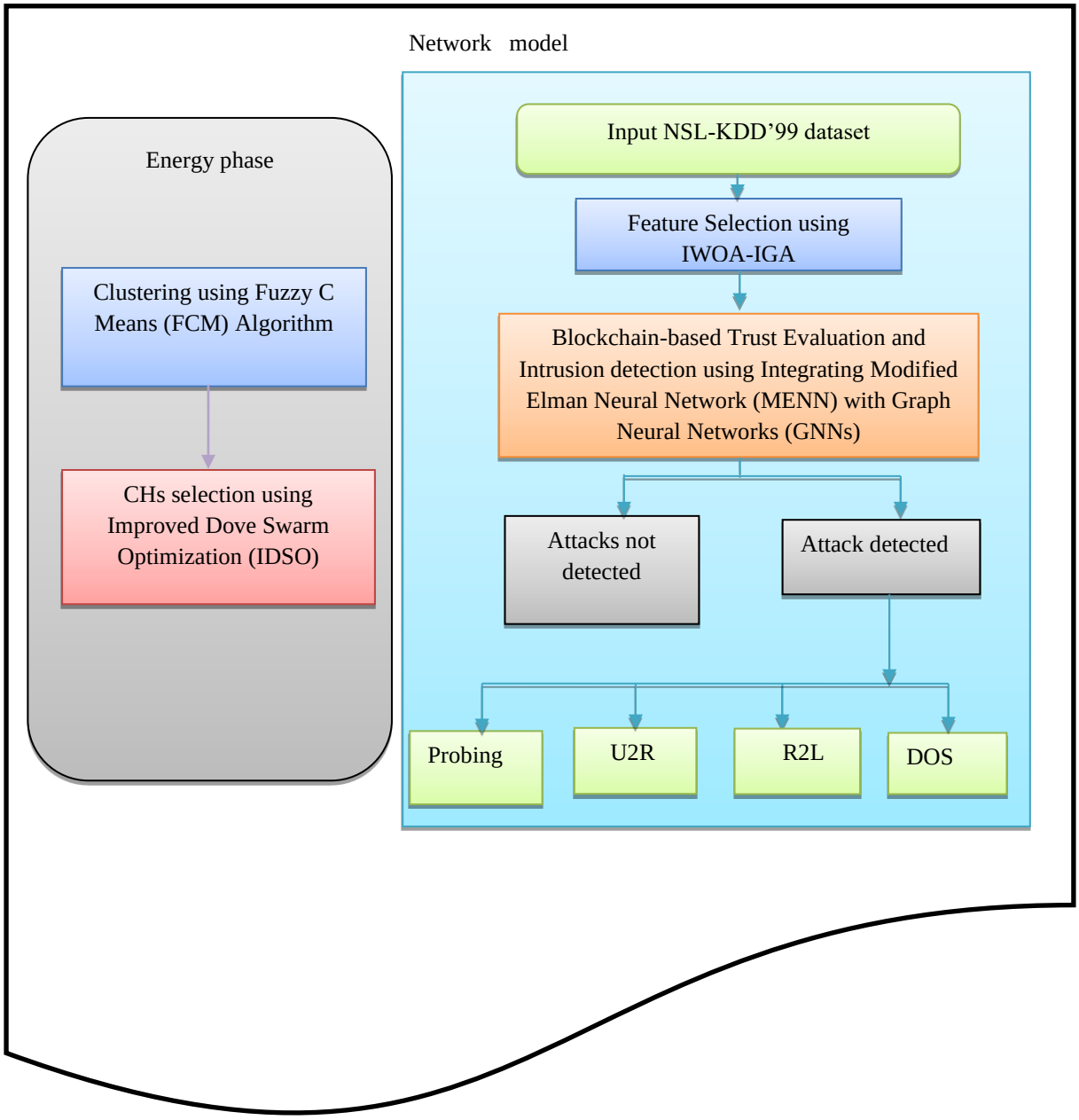

Figure: 1.Overall architecture of the proposed model

3.1. Node clustering using Fuzzy C-Means (FCM) clustering

Node clustering in WSNs is primarily used for improving energy efficiency, scalability, and network management. In WSNs, sensor nodes typically have limited power supplies, and their energy consumption must be minimized to prolong the network's lifespan. Clustering helps achieve this by organizing the network into smaller groups, or clusters, where designated CHs handle communications between clustered nodes and central BS.

Clustering is the most prevalent topology management strategy in WSNs, since it clusters nodes for administration and/or distributed task execution, including resource management. The energy consumption in this work is regulated using clustering methods. The network coverage region will determine how these work nodes are grouped. A sensor node should join the Cluster closest to its communication range in order to use less energy. Energy usage will decrease with decreasing distance. KMC is used in our previous work to cluster the nodes. The Weighted KMC can still be sensitive to the initial choice of centroids, affecting convergence to a global optimum and also Overfitting with High Weights. Hence to solve the issues Fuzzy C-Means (FCM) is used to cluster the nodes here. Fuzzy C-Means (FCM) clustering is a soft clustering algorithm where each data point can belong to multiple clusters with varying degrees of membership. In the context of Wireless Sensor Networks (WSNs), FCM is used to effectively group nodes into clusters based on their attributes, optimizing energy consumption and communication efficiency [35].

Steps in Node Clustering Using FCM

1. **Initialization**:

- Define the number of clusters (**C**) and initialize the cluster centers (**V**).
- Assign a membership matrix (**U**) with random values between 0 and 1, representing the degree to which each node belongs to each cluster.

2. **Membership Calculation**:

- For each sensor node i and each cluster j, calculate the degree of membership u_{ij}using the formula:

$$u_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{\|x_i - v_j\|}{\|x_i - v_k\|}\right)^{\frac{2}{(m-1)}}} \quad (1)$$

Where:

- x_i: The data point representing the node's features (e.g., location, residual energy).
- v_j: The cluster center of cluster j.
- m: Fuzzification parameter controlling the level of cluster fuzziness (typically $m > 1$).
- $\|\cdot\|$: Distance measure (e.g., Euclidean distance).

3. Cluster Center Update:

- Recalculate the cluster centers v_j based on the membership degrees

$$v_j = \frac{\sum_{i=1}^{N} u_{ij}^{m} . x_i}{\sum_{i=1}^{N} u_{ij}^{m}} \quad (2)$$

Where:

- N: Total number of nodes.

4. Iteration:

- Repeat steps 2 and 3 until convergence criteria are met (e.g., minimal change in cluster centers or membership matrix).

5. Final Clustering:

- Assign each node to a cluster based on its highest membership value.

3.2. Clustering and Cluster Head Selection with Improved Dove Swarm Optimization (IDSO)

Form clusters to minimize energy consumption and optimize cluster head selection. The Nodes are grouped based on communication range. Then the Cluster heads are selected using Dove Swarm Optimization (DSO), considering energy levels, distance, and connectivity.

- A metaheuristic optimization technique inspired by the swarm intelligence of doves (or other birds).
- Does not rely on gradient information directly.
- Uses a population (swarm) of candidate solutions (sets of parameters) to explore the parameter space.
- Balances exploration and exploitation by updating positions based on swarm behavior, aiming to minimize a given loss function.

The doves' foraging behavior has motivated ones to propose a novel optimal algorithm. In this method, the optimization objective function is f (W). Each data pattern, W, in a data set is regarded as a position with crumbs and the amount of crumbs in these place, W, has f (W) crumbs. The best solution means where it is the place with the most crumbs. Figure 2 demonstrates the flowchart for the DSO algorithm.

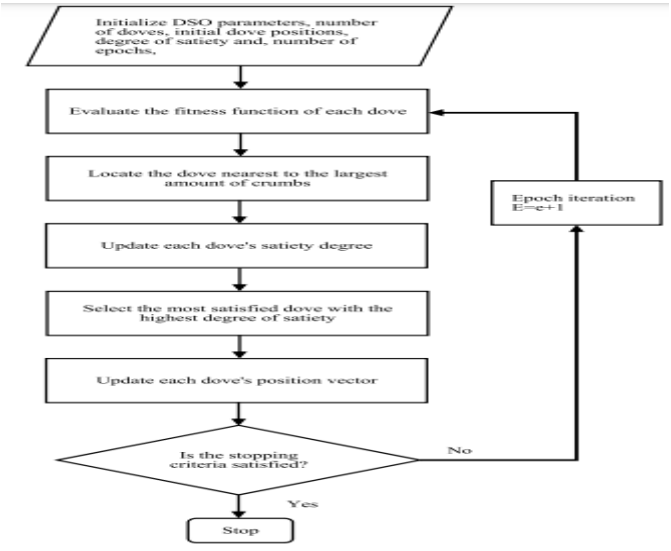

Figure 2: Flowchart of DSO

Step1: Decide the number of doves and then deploy them on the solution space. Assume that the number of doves is pre-specified to be N. These doves can be randomly distributed on the space; however, we suggest deploying them uniformly on a rectangular region.

Step2: Set the number of epochs, e = 0 and set the degree of satiety, s_d^e for dove d, $d = 1, \ldots, N$. The initialization of the position vector $W_d \subset R^M$ of dove d can be done in two ways. The simplest way is to randomly initialize W_d around the solution space. The other way is to initialize lattice initialized method. The steps are shown as follows:

Two efficient weight initialization schemes were proposed to initialize the weight vectors to accelerate the training process for constructing a topologically ordered feature map. Based on the initialization scheme, we propose an initialization method especially suitable for the algorithm. Let the smallest hyper-rectangle for the parameter space, which contains the valid values of all the parameters, be denoted as $[l_1, u_1], \ldots, [l_M, u_M]$ where l_a and u_a represent the low bound and the up bound of the a-dimension in the solution space. The basic idea of the proposed initialization method is to squeeze the n-dimensional hyper-rectangle into a 2-dimensional plane so a two-dimensional net can effectively cover the solution space. For the clarity purpose, we use i and j to index the rectangular cells from 1 to $A \times B$. The detail steps are as follows:

Step 1: Initialization of the cells on the four corners: The weight vectors of the four neurons on the corners of the network are initialized as (3).

$$w_{1,1} = (l_1, l_2, \ldots, l_M)^T \tag{3}$$

$$w_{A,B} = (u_1, u_2, \ldots, u_M)^T$$

$$w_{1,B} = \left(l_1, l_2, \ldots, l_{\lfloor \frac{M}{2} \rfloor}, u_{\lfloor \frac{M}{2} \rfloor + 1}, \ldots u_M\right)^T$$

$$w_{A,1} = \left(u_1, u_2, \ldots, u_{\lfloor \frac{M}{2} \rfloor}, l_{\lfloor \frac{M}{2} \rfloor + 1}, \ldots l_M\right)^T$$

Step 2: Initialization of the cells on the four edges: We initialize the cells' value on the four edges according to (4):

$$w_{1,j} = \frac{w_{1,B} - w_{1,1}}{B-1}(j-1) + w_{1,1} \tag{4}$$

$$= \frac{j-1}{B-1} w_{1,B} + \frac{B-j}{B-1} w_{1,1} \quad \text{for j=2,…B-1}$$

$$w_{A,j} = \frac{w_{A,B} - w_{A,1}}{B-1}(j-1) + w_{A,1}$$

$$= \frac{j-1}{B-1} w_{A,B} + \frac{B-j}{B-1} w_{A,1} \quad \text{for j=2,…B-1}$$

$$w_{i,1} = \frac{w_{A,1} - w_{1,1}}{A-1}(i-1) + w_{1,1}$$

$$= \frac{i-1}{A-1} w_{A,1} + \frac{A-i}{A-1} w_{1,1} \quad \text{for i=2,…A-1}$$

$$w_{i,B} = \frac{w_{A,B} - w_{1,B}}{A-1}(i-1) + w_{1,B}$$

$= \frac{i-1}{A-1} w_{A,B} + \frac{B-i}{B-1} w_{1,B}$ for i=2,…A-1

Step 3: Initialization of the remaining cells: The weight vectors of the four neurons on the corners of the network are initialized.

We initialize the remaining neurons from top to bottom and from left to right. The pseudo-code description of the initialization method for the remaining neurons is given as follows.

Begin

For j from 2 to B-1

Begin

For i from 2 to A-1

Begin (3), as shown at the bottom of the next page

End;

End;

End;

$$w_{i,j} = \frac{w_{A,j} - w_{1,j}}{A-1}(i-1) + w_{1,j} = \frac{i-1}{A-1} w_{A,j} + \frac{A-i}{A-1} w_{1,j} \quad (5)$$

$$= \frac{i-1}{A-1}\left(\frac{j-1}{B-1} w_{A,B} + \frac{B-j}{B-1} w_{A,1}\right) + \frac{A-i}{A-1}\left(\frac{j-1}{B-1} w_{1,B} + \frac{B-j}{B-1} w_{1,1}\right)$$

$$= \frac{((j-1)(i-1)_{w_{A,B}} + (j-1)(A-i)_{w_{1,B}} + (B-j)(i-1)_{w_{A,1}} + (B-j)(A-i)_{w_{1,1}})}{(B-1)(A-1)}$$

We choice different sizes and evaluate the result. In order to evaluate the training result, we count winner numbers of all neurons and the difference value. The initial value of learning rate is set to 0.1, and the decrease rate of the learning rate is as follows.

$$\eta(n) = \eta_0 \times \left(1 - \frac{t}{T}\right) = 0.1(1 - \frac{t}{100}) \quad (6)$$

where t is the iterative number, and η_0 is the initial learning rate.

Step 3: Compute the all doves' fitness function $f(w_j^e), j = 1, \ldots, N$ at epoch as a total number of the crumbs in the position of the dth dove.

Step 4: Locate the dove d_j^e nearest to the largest amount of crumbs using the maximum criterion at epoch e:

$$d_j^e = \arg\max\{f(w_j^e)\}, \quad for\ j = 1, \ldots N \quad (7)$$

Step 5: Update each dove's satiety degree by using the following equation:

$$S_j^e = \lambda S_j^{e-1} + e^{(f(w_j) - f(w_{d_f}))}, \quad for\ j = 1, \ldots N \quad (8)$$

Step 6: Select the most satisfied dove, d_s^e, with the highest degree of satiety using following maximum criterion:

$$d_s^e = \arg \max_{1 \le j \le N} \{S_j^e\}, \quad for\ j = 1, \dots, N \qquad (9)$$

The dove, ds, selected by (7) is the dove which presents the best foraging performance and it is the one which deserves to be imitated by other doves in the flock.

Step7: Update each dove's position vector using following maximum criterion:

$$w_j^{e+1} = w_j^e + \eta \beta_j^e (w_{d_s}^e - w_j^e) \qquad (10)$$

Where

$$\beta_j^e = \left(\frac{S_{b_s}^e - S_j^e}{S_{b_s}^e}\right)\left(1 - \frac{\|w_j^e - w_{d_s}^e\|}{\max Distance}\right)$$

$$\max Distance: \max_{1 \le j \le N} \|w_j - w_i\|$$

The parameters, η, is the learning rate for updating the dove position vector, respectively. Detailed descriptions of the updating Equations (8)-(10) are given in next step.

Step 8: Go to step 3 and increase the number of epochs by one (i.e., e = e + 1) until the terminate condition is met. The terminate condition is as follows

$$\left|f_{d_s}^e - T(e)\right| \le \varepsilon ore \le the\ set \max epoch \qquad (11)$$

The dove swarm optimization algorithm has the order of complexity, $O(NN_d e)$ where N_d is the number of data points in the data set, N is the number of doves, and e is the number of epochs. If the optimization is the minimum criterion that it's the best solution to find the minimum (w_j^e), then (5) and (6) can be respectively change to (12) and (13).

$$d_j^e = \arg\min\{f(w_j^e)\}, \quad for\ j = 1, \dots N \qquad (12)$$

$$S_j^e = \begin{cases} \lambda\, S_j^{e-1} + e^{\left(f(w_j) - f\left(w_{d_f}\right)\right),} & if\ f(w_{d_f}) \ne 0 \\ \lambda\, S_j^{e-1} + 1, & if\ f\left(w_{d_f}\right) = 0 \end{cases} \qquad (13)$$

For j=1,…N

For easier understanding, we interpret the updating rules given in Equations (8)-(10) as follows:

1. An individual is influenced by the success of the best individual in the flock and tries to imitate the behavior of the best individual. That is, doves move toward the dove with the highest degree of satiety to find more food. This social learning is simulated by updating the position vector w_j^e, to be more like the position vector of the dove with the highest degree of satiety $w_{d_s}^e$. (i.e., $w_j^{e+1} = w_j^e + \eta \beta_j^e (w_{d_s}^e - w_j^e)$)
2. When a dove is with a higher degree of satiety, it is prone to become conservative and would hesitate to change its present foraging policy. On the contrary, when a dove is with a

lower degree of satiety it would probably have a strong desire to change its present foraging policy and be more willing to imitate the behavior of the best individual. This social influence is simulated by making the adjustment proportional to the value of the first term on the right - hand-side of (9). (i.e., $\left(\frac{S_{b_s}^e - S_j^e}{S_{b_s}^e}\right)$).

3. Basically, social impact gradually decays as it spreads out; therefore, the degree of impact is inversely proportional to the distance between the dove and the best dove in the flock. This kind of social impact is simulated by making the adjustment amount proportional to the value of the third term on the right-hand-side of (9) (i.e $\left(\left(1 - \frac{\left\|w_j^e - w_{a_s}^e\right\|}{\max Distance}\right)\right)$

To improve the performance of Dove Swarm Optimization (DSO), balancing exploration (global search) and exploitation (local search) is crucial. Exploration ensures the algorithm can effectively survey the solution space to locate promising regions, while exploitation focuses on refining solutions within those regions to identify the optimal point. Below are some modifications and strategies that can enhance this balance in DSO:

Adaptive Step Sizes

Adaptive step sizes dynamically adjust the movement of doves during the optimization process based on the search stage.

- **Mechanism:**

  - **Early Stage (Exploration):** Larger step sizes allow doves to explore the global search space more effectively.

  - **Later Stage (Exploitation):** Smaller step sizes ensure finer adjustments around promising regions.

  - Use a **decaying function** for step size adjustment, such as:

$$Step\ size = \alpha . e^{-\beta . t} \qquad (14)$$

where α is the initial step size, β is the decay rate, and t is the current iteration.

By incorporating adaptive step sizes, the Dove Swarm Optimization algorithm becomes more efficient, reliable, and effective across a wide range of optimization scenarios, particularly those requiring a fine balance between exploration and exploitation. Hence the Sensor nodes send data to their respective cluster heads. Then Cluster heads aggregate the data and forward it to the base station which reduces direct transmissions and conserves energy.

3.3. Data Sensing and Transmission

1. **Sensing**: Each node periodically senses environmental parameters (e.g., temperature, humidity, or network traffic).

2. **Data Forwarding**: Sensor nodes forward the sensed data to their respective cluster heads.

3. **Aggregation**: Cluster heads aggregate data to remove redundancy, then transmit to the base station.

3.4. Feature Selection with Hybrid Improved Whale Optimization Algorithm with Enhanced Genetic Algorithm (Hybrid IWOA-IGA)

The next critical step in the methodology is Feature Selection using Hybrid IWOA-IGA. After clustering and selecting CHs, the focus shifts to reducing the computational complexity and energy consumption involved in detecting intrusions in WSNs, Feature selection is essential because large datasets, like the NSL-KDD'99 dataset, contain numerous features, many of which may be redundant or irrelevant. Processing all these features increases time complexity, energy consumption, and can lead to less accurate intrusion detection results.

Input NSL-KDD'99 database might have more features and consume more time to get classification results. This work uses feature selections based on Improved Whale Optimization (IWOA). Recently, the revolutionary stochastic population-based optimization approach called WOA which draws inspiration from nature, finds optimal solutions utilizing groups of search agents for optimizations. By using bubble-net hunting technique, WOA mimics actions of humpback whales as they pursue their preys. The three general processes of WOAs are to surround preys, assault using bubble nets, and hunt for best preys [35, 36, and 37].

Whales enclose their prey including fishes while updating their positions to find optimal solutions. Mathematically Eqs. (12) and (13) depict WOA.

$$X(t+1) = X^*(t) - A.|C.X^*(t) - X(t)| if p < 0.5 \quad (15)$$

$$X(t+1)|C.X^*(t) - X(t)|.e^{bl}\cos(2\pi t) + X^*(t) if p \geq 0.5 \quad (16)$$

Where X implies vectors of whales' positions; t represents times or iteration indices; X* implies best solutions found so far; A=2a. (r-a);C=2.r; a are coefficient vectors that linearly decrease from 2 to 0 in iterations; r stands for random vectors between 0 and 1; b implies constant values that define shapes of logarithmic spirals based on paths and is 1 for this work; l stands for random numbers between - 1 and 1; p implies random numbers between 0 and 1 and used to switch between (15) and (16) while updating whales' positions; Eqs. (15) and (16) have 50% probabilities implying during optimizations whales select paths randomly with equal chances. During bubble-net phases, A's random values are in the range [- 1, 1], however in searching phases, these values may be larger than or less than 1. Search processes are illustrated as Eq. (17).

$$X(t+1) = X_{rand} - A\,.|C.X_{rand} - X(t)\,| \quad (17)$$

Random searches with values of |A| > 1 emphasize searches and require WOA algorithm to do global searches. WOA searches begin with generations of random solutions. The responses are then updated in iterations and searches continue until present max. iterations are reached.

3.4.1. IWOA

The good trade-off between exploration and exploitation, two critical components of an optimization algorithm, allows for a precise solution to be obtained by escaping the local optima. In WOA, a search agent's step size decreases linearly as iteration counts increases. This step size is determined by a parameter known as A. Nonetheless, it has been demonstrated that insufficient divergence restricts WOA's capacity to capture a local optimum in later rounds.

This paper employs an updated whale optimization approach to get around such problems. This changes the value A by introducing the levy flying function. Levy flight is a mathematical concept used to describe a type of random walk in which step lengths have a probability distribution that is heavy-tailed (e.g., follows a Levy distribution). In optimization algorithms, such as the Improved Chicken Swarm Algorithm (ICSO), Levy flight is applied as a search strategy to improve efficiency and avoid premature convergence to local optima. It improves WOA's capacity for simultaneous exploration and exploitation.

The Levy probability distribution function, a power-law function, is utilized in Levy flight to determine jump sizes where Levy distributions can be mathematically formulated as:

$$\mathrm{L}(s,\gamma,\mu) = \begin{cases} \sqrt{\frac{\gamma}{2\pi}} \exp\left[-\frac{\gamma}{2(s-\mu)}\right] \frac{1}{(s-\mu)^{3/2}} if\ 0 < \mu < \infty \\ 0 \qquad\qquad if\ s \le 0 \end{cases} \qquad (18)$$

Where μ, γ, and s are positions and scales which control scale distributions and samples in distributions respectively.

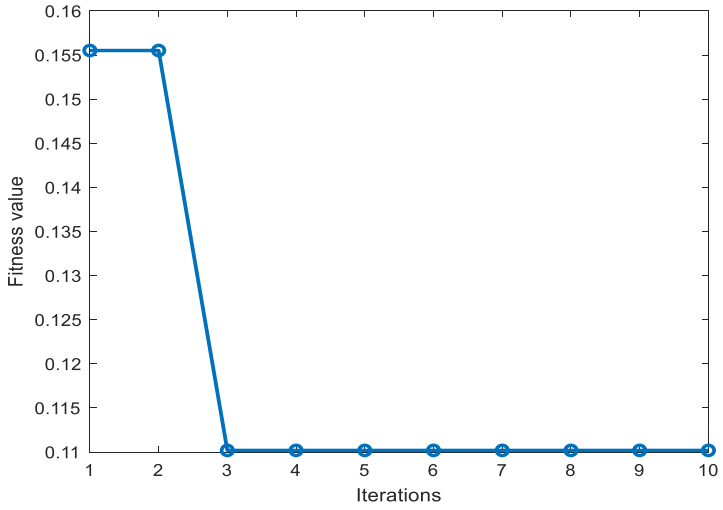

Figure 3: Convergence curve of IWO

The figure 3 depicts a line graph showing the relationship between iterations and fitness values in an optimization algorithm. The graph reveals a significant drop in the fitness value from approximately 0.155 at iteration 1 to about 0.11 at iteration 2. After the second iteration, the

fitness value stabilizes and remains constant at approximately 0.11 through iterations 3 to 10, indicating that the optimization process has likely converged. Using Mean Squared Error (MSE) as the fitness function in the Improved Whale Optimization Algorithm not only aids in effectively assessing the quality of solutions but also demonstrates the algorithm's capability to minimize errors in predictive modelling.

In the IWOA, Levy flights play critical roles in enhancing explorations and exploitations of the algorithm, particularly to avoid issues of getting trapped in local optimums during later iterations.

Role of Levy Flight in IWOA:

1. Exploration and Exploitation Balance:

- Explorations refer to algorithm's abilities to search broadly across solution spaces, while exploitations refer to focusing on refining current best solutions.
- In standard Whale Optimization Algorithm (WOA), the parameter A controls the step size of the whales' movement. However,
-
- as iterations increase, A decreases linearly, reducing the step size and thus restricting the ability to explore new regions of the search space.
- To enhance both exploration and exploitation simultaneously, Levy flight is used to introduce random long-distance jumps, which allows for more diverse movements even in later stages of the search. This nonlinear random walk based on the Levy distribution helps escape local optima by allowing whales to make larger, random steps.

Algorithm for IWO

START

1. import data
2. Set initial locations of whales **X**
3. Compute whales fitness
4. Set initial values of **a** and **r**, calculate **A** and **C**
5. Set initial **X*** as best hunters' whale locations
6. initialize $t = 1$
7. **while** $t \leq$ max iterations **do**
8. **for** each hunting whale **do**
9. **if** $p < 0.5$
10. **if** $|\mathbf{A}| < 1$
11. update existing locations of hunting whales with (12)
12. **else if** $|\mathbf{A}| \geq 1$
13. another search agent randomly
14. update existing locations of hunting whales with (12)
15. **end if**

16. **else if** $p \geq 0.5$
17. update existing locations of hunting whales with (13)
18. **end if**
19. **end for**
20. update **X*** on better solutions
21. $t = t + 1$
22. **end while**
23. output **X*** Best Features
24. **END**

In this paper, a modified and improved WOA (IWOA) is presented to solve the application mapping problem in NoC. The incorporated IWOA algorithm has better global and local search ability; however, for certain scenarios, such as large-scale complex problems, it may undergo lower convergence accuracy and can fall into local optimum. To avoid such problems and scenarios, an improved GA is executed along with the IWOA to achieve optimal solutions. The GA uses a biological evolutionary mechanism, and, as a global-based search optimizer, it searches for the optimum solution in the search space for complex problems. GAs have proven to be robust and effective in exploring complex spaces. Hence, they can effectively solve a wide range of pattern recognition, artificial intelligence, resource allocation, and similar complex challenges. To achieve better mapping solutions, the proposed modified IWOA algorithm is integrated with an improved genetic algorithm incorporating improved crossover and mutation abilities. This further enhances the ability to find the optimum solution within the search space with better convergence towards the final mapping solution.

The genetic flowchart depicted in Figure 4 resembles biological evolution. A traditional GA begins with an initial random population comprised of randomly chosen chromosomes that produce offspring via crossover and mutation. It continues to work iteratively until a predetermined count of iterations is completed or a termination criterion is satisfied.

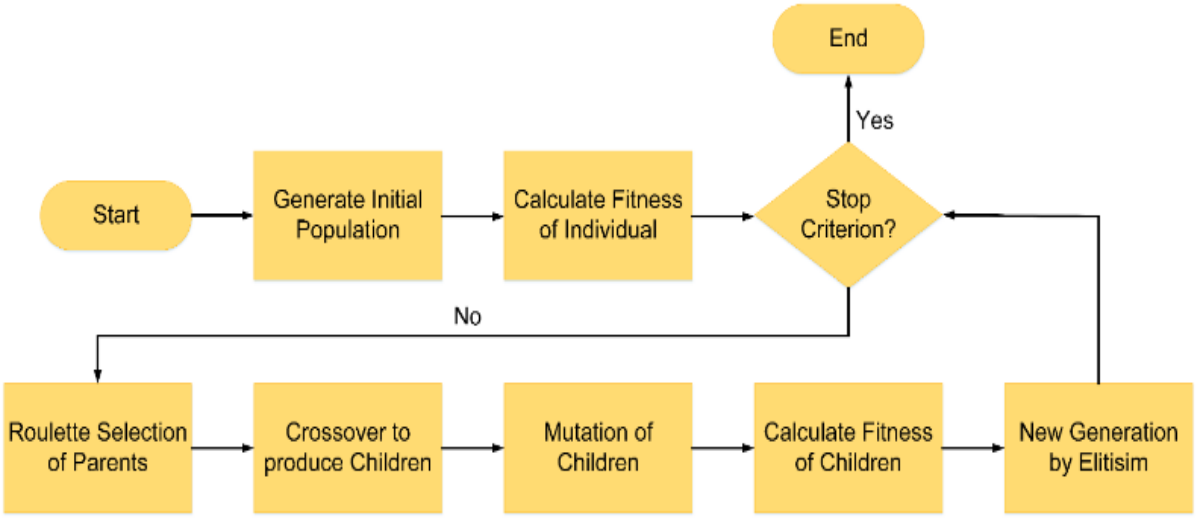

Figure 4: Genetic algorithm flowchart

3.4.2. IGA Framework

This article focuses solely on a single objective, namely, use of an improved GA integrated with an improved WOA for solving application mapping through better search ability and population diversity. The population obtained from the IWOA is fed into the modified GA, which uses direction-based crossover features [38] and mutation ability to generate high-quality mapping solutions. The subtasks imply the genes in the chromosomes. The various individual mapping solutions undergo through the genetic-based process for some number of iterations and generations to produce high-quality mapping solutions.

Improved Genetic Algorithm Formulation

The genetic operators exert different levels of impact on the algorithm, with selection identifying the most promising chromosomes for crossover to enhance the solutions. Crossover combines genetic data to improve population characteristics, while mutation introduces new genes to address the weaknesses of crossover.

The selection operator selects the individuals that participate in crossover and mutation; hence, their selection substantially impacts the entire GA process [39]. The proposed modified algorithm uses expert criteria-based selection to choose the initial mapping solutions to undergo crossover and mutation. This is contrary to the random selection used in the conventional GA. Initially, the mapping solutions achieved by the IWOA are ordered according to the objective function $F(X)$ and treated as an initial population for the IGA, represented as

$$X = \{X_1, X_2, \dots, X_n\} \quad (19)$$

while the sorted population is provided by

$$X^s = \{X_1^s, X_2^s, \dots., X_n^s\} \quad (20)$$

which satisfies $F(X_1^s) \geq F(X_2^s) \geq, \dots \dots F(X_n^s)$. Each element in X and X^s shows individual mapping solutions. The population obtained by the IWOA is sorted and divided into four groups (X^1, X^2, X^3, X^4), which are paired with each other $((X^1, X^2), (X^1, X^3), (X^1, X^4), (X^2, X^3), (X^2, X^4), (X^3, X^4))$ to form $X^{(map_a)}$ and $X^{(map_b)}$. Crossover is performed using the elements of $X^{(map_a)}$ and $X^{(map_b)}$.

The GA's central process significantly affects the algorithm's ability to seek improved and optimal solutions [40]. The more optimal the objective function is for any individual solution, the closer the mapping combination will be to the optimal region. Hence, the proposed IWOA-IGA uses a direction-based crossover operator. The modified crossover operation has good ability for searching the overall search space, and can produce mapping solutions with a larger probability of enhancing the objective function, thereby speeding up the algorithm's convergence. As in the initial step of the GA, the population of mapping solutions is divided into two groups, $X^{(map_a)}$ and $X^{(map_b)}$. The communication cost of the individual solutions in $X^{(map_a)}$ is superior to those of the solutions in $X^{(map_b)}$; thus, $X^{(map_a)}$is the leader in the direction of crossovers for producing high quality solutions. The mathematical equation that generates the mapping solutions with directional-based crossover is provided by

$$\begin{cases} X_i^* = X_i^{map_a} + r_{ij}.\vec{D}_{ij} \\ \vec{D}_{ij} = X_i^{map_a} - X_i^{map_b} \qquad i = 1, \dots \frac{3n}{2}, j = 1, \dots, m' \end{cases} \tag{21}$$

where $\vec{D}_{ij}$ represents the directional vector. The parameter r_{ij} is a uniformly distributed random number ranging from −1 to 1. Thus, in directional crossover based on grouping-wise solutions, each mapping pair develops into one mapping solution, eventually generating new individuals. Finally, it sorts the high-quality mapping solutions with the best communication cost, developing n individuals as offspring of the crossover.

To see the effect of the direction-based crossover operation on two of the paired mapping solutions, paired individuals are shown to occupy a certain space in the appropriate region. As r_{ij} takes on boundaries between 1 and −1, the paired mapping solutions will take various directions to reach the optimal communication cost value. The mutation operation is applied to offspring to achieve more optimal results. Mutation allows for altering the placement of one or more mapped tasks of individuals in a population with a specified mutation probability, which can help to promote population diversity and prevent premature events. The communication cost of all the resulting mappings are computed after mutation, and the mutation is accepted if the communication cost of the mutated mapping is less than that of the original ones. To increase population diversity and prevent the GA version from falling into local optima, the mechanism of replacement operation is incorporated. The basic goal of the replacement operation is to set aside the most elite K individuals from the population for every K generation. In traditional approaches, certain elite individual solutions are retained in parents. Nevertheless, the high-quality solutions created as a result of crossover may be affected by the mutation operation, and their quality may degrade to some extent. Thus, in this work we adopt a better approach to retain the most elite mapping solutions produced by the genetic operation. In this way, the number of elite mapping solutions can be maximized to enhance the population attributes.

The proposed hybrid method necessitates the specification of a few fundamental parameters to determine the efficacy of group searching. Numerous simulations and satisfying results were used to pick the parameter values for the proposed algorithm in order to produce an effective solution. Well balanced exploration and exploitation ability is derived using adaptive parameter adjustment, which results in updating of the vector $\vec{A}$. The algorithm uses a few other parameters, such as r (a random vector between [0, 1]), and a linear decline from 2 to 0 is performed throughout iteration during both the exploration and exploitation phases. For the shrinking encircling mechanism, A takes a random value between $-a$ and a, where a gradually diminishes from 2 to 0 throughout iteration. Configuring A within [−1, 1] can result in a new position for the mapping solution between the original position and the current best position. For the spiral mapping position, the parameter b is kept at a constant value to maintain the shape of the spiral, while l is varied between [−1, 1] to help the algorithm achieve better mapping solutions.

Overall Process:

1. **Initialization**: Randomly generate feature subsets.

2. **Fitness Evaluation**: For each subset, evaluate using an objective function (e.g., intrusion detection accuracy, false positives).

3. **Update via IWOA**: Apply whale swarm strategies to improve feature subset.

4. **Crossover/Mutation via IGA**: Introduce genetic variations to explore new feature subsets.

5. **Convergence**: Output an **optimal subset of features** for intrusion detection.

The Hybrid IWOA-IGA is a key component of the proposed methodology for feature selection optimization. By reducing the dataset's dimensionality, Hybrid IWOA-IGA identifies the most relevant features, enhancing the classification model's overall performance while simultaneously reducing time complexity and energy consumption. The inclusion of the Levy flight strategy within IWOA ensures a thorough exploration of the solution space, avoiding local optima and leading to the discovery of the optimal set of features. Out of the 41 initial features, the algorithm selects the 15 most effective for the detection task: Hot, Num_root, Num_shells, Num_access_files, Num_outbound_cmds, Is_host_login, Count, Serror_rate, Rerror_rate, Same_srv_rate, Dst_host_srv_count, Dst_host_same_srv_rate, Dst_host_srv_diff_host_rate, Dst_host_serror_rate, and Dst_host_srv_rerror_rate.The selection of these 15 specific features by the IWOA is based on their strong relevance to distinguishing between normal and malicious network behavior. Each feature plays a unique and crucial role in identifying patterns linked to network intrusions:

Hot, Num_root, Num_shells, Num_access_files, and Num_outbound_cmds: These features are directly tied to unauthorized access attempts, such as gaining root privileges or accessing sensitive files, which are critical indicators of potential system compromise (e.g., Root or User to Root attacks).

Is_host_login: This feature helps detect abnormal login patterns, indicating unauthorized access or probing attempts, which are important in identifying login-related vulnerabilities.

Count, Serror_rate, and Rerror_rate: These features measure the frequency of connections and errors during those connections, helping to identify denial-of-service (DoS) or scanning attacks where a high volume of error-prone connections is typical.

Same_srv_rate, Dst_host_srv_count, and Dst_host_same_srv_rate: These features monitor the frequency and consistency of connections to a particular service or destination, allowing the detection of probing or flooding attacks, which target specific services repeatedly.

Dst_host_srv_diff_host_rate, Dst_host_serror_rate, and Dst_host_srv_rerror_rate: These features track the variance in service errors across different hosts, which are key indicators of network-level anomalies, such as Distributed Denial of Service (DDoS) attacks where multiple hosts are targeted.

Hence the Combine Improved Whale Optimization Algorithm (IWOA) with enhanced genetic properties (e.g., crossover, mutation) are proposed for feature selection. The most significant

features from the NSL-KDD'99 dataset are selected, improving detection accuracy and reducing computational load.

3.5. Blockchain-Based Trust Evaluation

To continuously monitor each node's behavior in the WSN, compute trust scores, and detect anomalous (potentially malicious) nodes. Trust values are recorded on a blockchain, ensuring immutability (they cannot be tampered with) and transparency (all authorized nodes can verify trust scores). Establish trust among nodes for secure communication is discussed below

Trust-Based IDS Module:

The Trust-Based IDS continuously monitors node behaviour in WSNs. It consists of:

1. **Trust Calculation**: Evaluates each node's trustworthiness based on metrics like packet forwarding, energy consumption, and data integrity.

2. **Anomaly Detection**: Detects malicious behavior by comparing a node's current actions with expected normal behavior.

Trust Calculation:

1. **Packet Forwarding**: Measures how reliably a node forwards packets.

$$T_f = \frac{\text{Packets Forwarded by Node}}{\text{Packets Sent to Node}} \quad (22)$$

- A higher T_f indicates the node forwards incoming packets reliably.
- Nodes that drop packets (intentionally or due to faulty behavior) will have a lower T_f.

Higher values indicate more reliable forwarding behavior.

2. **Energy Consumption**: Evaluates how efficiently a node uses energy

$$T_e = \frac{(\text{Initial Energy} - \text{Current Energy})}{\text{Intial Energy}} \quad (23)$$

- A lower T_e reflects more efficient energy usage (i.e., the node has consumed less energy relative to its initial state).
- Malicious or compromised nodes may consume excessive energy (e.g., due to flooding attacks), resulting in a higher T_e.

A lower value reflects more efficient energy consumption.

3. **Data Integrity**: Assesses the accuracy of data sent by a node

$$T_d = \frac{\text{Accurate Data Packets Received}}{\text{Total data packets received}} \quad (24)$$

- Measures how **accurately** the node handles data packets (not injecting or forwarding corrupted data).

This metric ensures that nodes are not injecting false or corrupted data into the network.

Overall Trust Value:

After each node's three metric values (T_f, T_e, T_d) are calculated, they are combined into a **single trust score**: The overall trust T_{total} is a weighted sum of the individual metrics:

$$T_{total} = w_f.T_f + w_e.T_e + w_d.T_d \quad (25)$$

where $w_f, w_e \; and \; w_d$ are the respective weights assigned to each trust metric based on the application's requirements.

Nodes with abnormal trust values are flagged as suspicious. Modified Elman Neural Network (MENN) can be used to detect anomalies based on deviations in trust metrics.

Threshold Checking & Anomaly Flagging

1. **Set Threshold**: Define a **minimum** acceptable T_{total} value or range.
2. **Compare:**

- If $T_{total}(i)$ **falls below** the threshold, the node is flagged as **suspicious**.
- Further action may include isolating the node or subjecting it to **detailed analysis**.

Blockchain Integration

1. Record Trust Values:

- Each node's trust metrics (T_f, T_e, T_d) or the combined T_{total} are written to the blockchain as transactions.

2. Consensus Mechanism:

- The network validates trust updates (e.g., via Proof of Stake).
- Once validated, these updates form a block that is appended to the distributed ledger.

3. Immutability & Transparency:

- Trust scores become tamper-proof once on-chain.
- Other nodes can verify the trust history for any given node.

3.6. Intrusion Detection with MENN-GNN

MENN captures temporal patterns in sequential data, enhancing the detection of behavioral anomalies. Graph Neural Network (GNN) models relational data in the network, detecting spatial and contextual irregularities. Combined MENN-GNN architecture improves detection accuracy by leveraging temporal and relational insights. The MENN model is trained to classify node behaviors based on the features selected by IWO. The Modified Elman Neural Network with self-feedback gain tends to be more efficient than the previous used approach of Enhanced Fuzzy Neural Network (EFNN) is due to its simpler architecture, easier parameter tuning, reduced computational complexity, better integration with existing optimization frameworks. The Modified Elman Neural Network (MENN) enhances traditional Elman Neural Networks by adapting their structure for more robust handling of time-sequenced data, making it effective for classifying nodes in trust-based Intrusion Detection Systems (IDS). In Elman Neural Networks (ENN), the context layer acts as a feedback mechanism by storing the previous hidden layer's outputs and feeding them back into the network. This feedback, often referred to as self-feedback gain, allows the network to capture temporal dependencies and learn sequential patterns effectively, hence it is called as MENN.

Trust-Based IDS: Nodes are evaluated based on trust scores (packet forwarding, data integrity, energy consumption). MENN uses this trust data to classify nodes as either trustworthy or malicious. In Figure 5, the Elman neural network's fundamental structure is shown. The input layer, hidden layer, context layer, and output layer make up the majority of the Elman network, which can be easily identified as having four layers in total. Each pair of nearby layers is connected by adjustable weights [41]. In most contexts, it is understood to be a subtype of feed-forward neural network that also contains local feedback and extra memory neurons. Below is a list of the symbols used in this section:

$w1_{ij}$: The weight between input layer node i and hidden layer node j.

$w2_{ij}$: The hidden layer-output layer weight.

$w3_{ij}$: The hidden layer weight that links context node i to node j.

m, n, r : Input, output, and buried layer nodes, respectively.

$u_i(k), y_i(k)$: Elman neural network inputs and outputs, where i=1,2,.....m and j=1,2,.....n.

$x_i(k)$: Hidden node i output, i=l, 2,.....r.

$c_i(k)$: the result of context node i, or the result of hidden node i from the previous execution. z^{-1}**:** A unit delay.

A second unit called a context unit is created for each unit in the concealed layer. With regard to all hidden units, the context unit is directly related to them all. Accordingly, each context unit has a weight that is applied to each concealed unit. In addition, the context units and hidden units are connected repeatedly. However, each hidden unit is only linked to the context unit with which it is related is provided in figure 5.

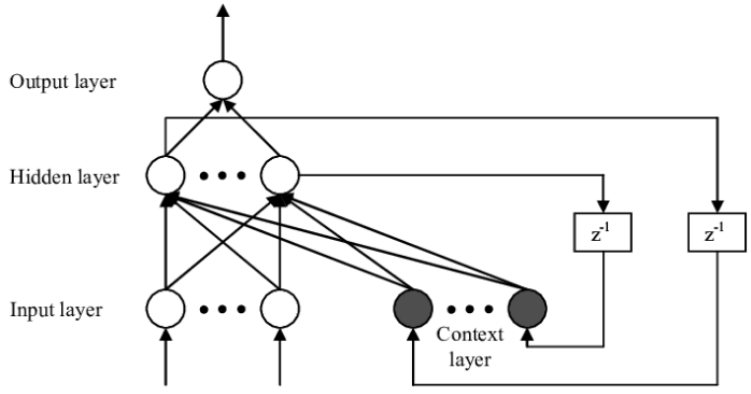

Figure 5: Structure of the Elman neural network model

While the weights of the recurrent connections are fixed, the forward weights are learned by back-propagation. During the forward phase, the context units' function similarly to input units. The hidden unit and output unit values are calculated similarly to how feedforward networks are built. The current values are replicated into the associated context units through the recurrent connections once the hidden units have calculated their outputs.

These parameters must be set to some starting values before to using them in the subsequent time step. Target values are utilized for the outputs during the backward training phase, and back-propagation is employed to change the forward weights. The network's inputs are: $(k) \in R^m\ y(k) \in R^n, x(k) \in R^r$, further, each layer's outputs may be provided via

$$x_j(k) = f(\sum_{i=1}^{m} w2_{i,j}u_i(k) + \sum_{i=1}^{r} w1_{i,j}c_i(k)) \quad (26)$$

$$c_i(k) = x_i(k-1) \quad (27)$$

$$y_j(k) = g(\sum_{i=1}^{r} w3_{i,j}x_i(k) \quad (28)$$

where the corresponding output functions of the hidden layer and output layer. Formally, if we denote the activation functions in the hidden and output layers by $f(.)$and $g(.)$ respectively, they can be chosen to be linear or nonlinear. Regardless of this choice, the key benefit of an Elman network is that it does not require explicit external state input or a training signal for its internal state. All the necessary dynamics are managed through the recurrent context connections. This makes Elman networks particularly flexible and powerful for tasks involving sequential or time-varying data, as compared to standard static feed-forward networks.

- **Self-feedback gain (SFG)**

The functions utilized here are linear activation functions. Since the weight of the feedback connections is set at 1, they are not used in the training process. This results in a network that, despite the fact that it contains recurrent connections, can be trained more like a straightforward feedforward network. The network's memory or dynamics are provided by the context layer, which serves as a storage layer for the hidden layer's state. The hidden and context layers each contain the same amount of neurons. Even while an Elman network can theoretically mimic any nth order system, where n is the number of context neurons, with all feedback connections from the hidden layer set to 1, it cannot be taught to do so using the conventional backpropagation approach. To solve this shortcoming, a modified Elman network that still uses regular backpropagation training has been developed and shown to be capable of modeling higher order systems. In order to prolong the memory, the change involves adding self-feedback connections to the context neurons.

$$x_j^c(k) = \alpha x_j^c(k-1) + x_j(k-1) \quad (29)$$

where $x_j^c(k)$ is produced by the context layer neuron j at time k, and $x_j(k-1)$ at time k-1, is the hidden layer neuron j's output. Before training, a single value between 0 and 1 is selected to represent all the self-feedback connection weights in this equation. A greater value will often be needed α to lengthen the memory when modeling a higher-level system. If, on the other hand, is raised to an excessive degree, training will become unstable, necessitating a compensatory decrease in the learning rate; nevertheless, this will result in an increase in the total amount of time spent training.

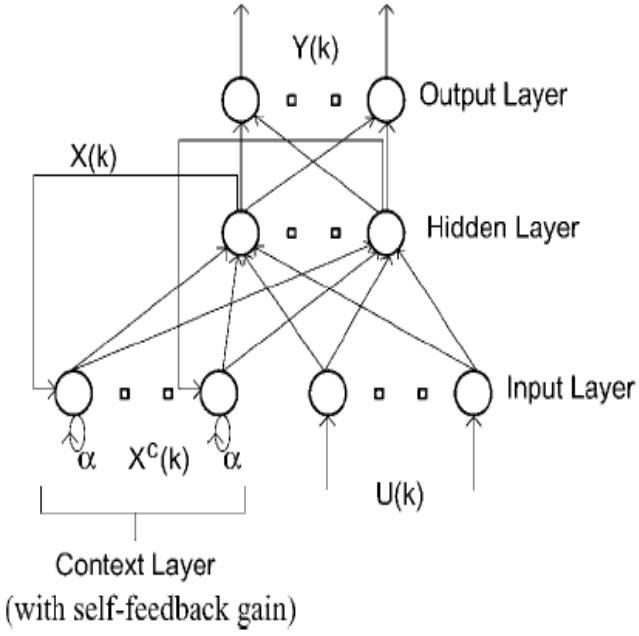

Figure 6: Elman neural network modified

Choosing a range (0 to α max) allows the value of α for context neuron n to be determined by Equation (22). This is an alternate approach for setting that is suggested here:

$$\alpha_n = \frac{n-1}{N-1}\alpha_{max}, \qquad n = 1, \ldots\ldots, N \qquad (30)$$

N is the context layer's total number of neurons. The network now has both short-term and long-term memory. The second suggestion is to include the values into the backpropagation training process since they play a significant role in deciding how well the modified Elman network performs. Each n is then updated separately as if it were a typical weighted connection as the error is then back propagated across the self-feedback connections.

Graph Neural Network (GNN)

Nowadays have seen rising attention gained by graph neural networks [42] which are graph-structured data oriented, capable of learning the network topology and relationships among nodes. GNNs can be applied to different-level tasks such as node-level prediction, edge-level prediction, and graph-level prediction. The intrusion detection problem studied in this paper is narrowed down to a binary classification task, i.e., predicting whether a node *v* is under attack (class 1) or not (class 0), belonging to the node-level prediction:

For a node *v* in graph *G*, GNN learns a hidden representation, $\boldsymbol{h}_v$, by taking both the node features $\boldsymbol{x}_v$ and graph structure (edges) *E* as input. Most of the advanced GNNs follow a message-passing and aggregation strategy, in which the information of one node is iteratively passed to the others along edges, and then the node's representation is updated by aggregating and learning the information of itself and its neighbors. When conducting the message propagation with *l*-layered GNN, one node could gain the information of neighboring nodes within *l* hops. Without loss of generality, the latent representation at the *l*-th layer of GNN can be formulated as:

$$h_v^l = f^l(h_v^{l-1}, AGG^l(\{h_u^{l-1}, \forall u \in \mathcal{N}_v\})) \quad (31)$$

where $\mathcal{N}_v$ refers to the neighboring nodes of node *v*. AGG^lrefers to aggregations that could be rather simple ones (e.g., mean, max, sum), to advanced ones (e.g., median, var, std), or even learnable and exotic ones. f^l is a function with learnable parameters, e.g., multi-layer perceptron (MLP). The combination of $AGG^l$and f^l determines the types of graph operators, e.g., graph convolutional operator and graph attentional operator.

For node-level prediction, the output of the last hidden layer could be directly passed to the output layer Φ, e.g., a fully connected layer. For graph-level prediction, it is passed to a readout function R with learnable parameters. Formally, the representations of the output layer can be expressed as:

$$h_v = \Phi(h_v^l) \quad (32)$$

$$h_G = R(h_v^l | v \in V) \quad (33)$$

Graph convolutional operator. Graph convolution, essentially the same as the convolution for image classification task, is a generalization of convolution in Convolutional Neural Network (CNN) in which the information of neighboring nodes (corresponding to adjacent pixels) is convoluted with that of central node (corresponding to central pixel) to learn a hidden representation. The graph convolution operation in [43] could be formulated as:

$$h_i' = \sigma(W^T \sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{w_{ji}}{\sqrt{\hat{d}_j \hat{d}_i}} h_j) \quad (34)$$

where $\hat{d}_i = 1 + \sum_{j \in \mathcal{N}_i} w_{ji}$. $w_{ji} \in \mathbb{R}$ denotes the weight of edge from source node *j* to target node *i*. **W** is the filter weight to be optimized. σ is the activation function. *T* refers to transposition. By default, all edge weights are fixed as 1, meaning neighboring nodes are of the same importance to central node. Edge weights can also be optimized to learn the importance of neighboring nodes in an implicit manner.

Algorithm MENN_GNN_IDS:

Input: Graph G = (V, E), adjacency matrix A (or edge list), node features X

Time-series trust data {u_v(k)} for each node v

Labels y_v for training nodes

```
# Step 1: Preprocess and Build Graph
X_processed, A_processed = preprocess(X, A)
# Step 2: Train MENN
Initialize MENN parameters (w1, w2, w3, alpha)
for epoch in range(num_epochs_MENN):
    for each node v in training set:
        # Time-series forward pass
```

```
        x_hidden, x_output = MENN_forward(u_v(1..K))
        # Compute loss and backprop
        MENN_loss = loss_function(x_output, y_v)
        update(MENN parameters via backprop)
  # Extract MENN embeddings e_v for each node v
  # Step 3: Combine MENN Embeddings with Original Features
  for each node v:
    X_final[v] = concat(X_processed[v], e_v)
  # Step 4: Train GNN
  Initialize GNN parameters (W, possibly attention parameters)
  for epoch in range(num_epochs_GNN):
    H = GNN_forward(X_final, A_processed)
    GNN_loss = loss_function(H, y_v)
    update(GNN parameters via backprop)
  # Step 5: Inference
  for each node v in test set:
    e_v_test = MENN_forward(u_v_test(1..K))  # temporal features
    x_v_test = concat(X[v], e_v_test)        # combine
    y_hat_v = GNN_inference(x_v_test, A_processed)
    intrusion_decision(v) = (y_hat_v > threshold) ? Malicious : Benign
  return intrusion_decision
```

Hence A trained GNN that leverages both node-level temporal insights (from MENN embeddings) and graph relational structure to classify each node as malicious or benign.

ALGORITHM: Secure WSN Framework using IDSO, IWOA-IGA, MENN-GNN, & Blockchain

INPUTS:

1. WSN with N nodes, each having initial energy E_i^init and position (x_i, y_i).
2. Communication range R, Base Station (BS).

3. NSL-KDD dataset (or real-time WSN traffic data).
4. Hyperparameters for IDSO, IWOA-IGA, MENN, and GNN.

OUTPUT:

- Clusters & Cluster Heads (CHs)
- Optimal Feature Subset
- Intrusion Alerts & Blockchain-based Trust Scores

STEP 1: NETWORK FORMATION & CLUSTERING

1. For each node $i \in \{1..N\}$:
   - Deploy at (x_i, y_i).
2. Assign nodes to clusters based on distance $\leq R$.

STEP 2: CLUSTER HEAD SELECTION USING IDSO

1. Initialize population of candidate CH nodes.
2. Evaluate fitness F_CH for each candidate (residual energy, dist to BS, etc.).
3. Update candidate solutions via IDSO operators (social, adaptive learning, etc.).
4. Repeat until termination criterion is met.
5. Select top-ranked solutions as final CHs.

STEP 3: DATA TRANSMISSION

For each cluster:

1. Member nodes sense data => send to CH.
2. CH aggregates & transmits aggregated data to BS.
3. Update node energies according to E_tx, E_rx formulas.

STEP 4: FEATURE SELECTION WITH IWOA-IGA (using NSL-KDD or real-time data)

1. Initialize population of feature subsets.
2. Evaluate F_FS (accuracy vs. subset size).

3. Apply Whale Optimization steps (encircling, spiral).

4. Apply Genetic operators (crossover, mutation) to enhance diversity.

5. Repeat until best feature subset is found.

6. Output optimal subset S_opt.

STEP 5: MENN-GNN BASED INTRUSION DETECTION (BLOCKCHAIN-INTEGRATED)

1. For each node i, compute trust metrics T_f, T_e, T_d => T_total.

2. Create transaction (NodeID, T_total, Timestamp) => broadcast to blockchain.

3. Consensus mechanism validates & appends to ledger => trust is immutable.

4. Use S_opt features + trust metrics as input to the GNN.

5. GNN => produce node embeddings capturing network topology + node attributes.

6. Feed embeddings (and sequential trust data) to MENN => produce classification (normal vs. malicious).

7. Flag malicious nodes; store final results (alerts) on blockchain for transparency.

STEP 6: ACTIONS & UPDATES

1. If node is flagged malicious => isolate or restrict from routing paths.

2. If node recovers or changes behavior => node's trust can be re-evaluated over time.

3. Periodically repeat Steps 1–5 to maintain updated clusters, feature sets, and intrusion detection.

END ALGORITHM

IV.EXPERIMENTAL RESULTS

The experiments conducted on the suggested model are analyzed in this section. The use of MATLAB facilitates the implementation of this concept. To assess the suggested method, a comparison is made between the already available KMC, SVM algorithms and EFNN in terms of accuracy, specificity, time complexity, energy consumption, and error rate using the network security laboratory dataset (NSL-KDD'99). Here, dividing the dataset into 70% training and

30% testing is a standard method for assessing the effectiveness of the model. One may obtain the open-source dataset from the internet. Features from [16] in the following table were implemented in this work 1.

Table.1. Features of NSL-KDD dataset

NO	FEATURES	TYPES	NO	FEATURES	TYPES
1	duration	Continuous	22	is_guest_login	Symbolic
2	Protocol_type	Symbolic	23	count	Continuous
3	service	Symbolic	24	srv_count	Continuous
4	flag	Symbolic	25	serror_rate	Continuous
5	src_bytes	Continuous	26	Srv_serror_rate	Continuous
6	dst_bytes	Continuous	27	rerror_rate	Continuous
7	land	Symbolic	28	srv_rerror_rate	Continuous
8	'wrong_fragment'	Continuous	29	same_srv_rate	Continuous
9	urgent	Continuous	30	diff_srv_rate	Continuous
10	hot	Continuous	31	srv_diff_host_rate	Continuous
11	num_failed_logins	Continuous	32	dst_host_count	Continuous
12	logged_in	Symbolic	33	dst_host_srv_count	Continuous
13	num_compromised	Continuous	34	dst_host_same_srv_rate	Continuous
14	root_shell	Continuous	35	dst_host_diff_srv_rate	Continuous
15	su_attempted	Continuous	36	dst_host_same_src_port_rate	Continuous

16	num_root	Continuous	37	dst_host_srv_diff_host_rate	Continuous
17	num_file_creations	Continuous	38	dst_host_serror_rate	Continuous
18	num_shells	Continuous	39	dst_host_srv_serror_rate	Continuous
19	num_access_files	Continuous	40	dst_host_rerror_rate	Continuous
20	num_outbound_cmds	Continuous	41	dst_host_srv_rerror_rate	Continuous
21	is_host_login	Symbolic			

The confusion matrix presented below summarizes the performance of a classification model on a four-category experiment using the intrusion detection dataset. It allows us to analyse how well the model distinguishes between different types of network attacks: Denial-of-Service (DoS), Probing, Remote-to-User (R2L), and User-to-Root (U2R). Table 2 describes the confusion matrix of the proposed work.The diagonal entries of the matrix represent the true positives for each class, showcasing the model's ability to accurately identify attacks. For instance, the model correctly classified a substantial 17,225 instances as DoS attacks, demonstrating its effectiveness in recognizing this prevalent attack type. Similarly, it identified 3,897 instances of Probing correctly, along with 399 instances of R2L attacks and an impressive 18,898 instances of U2R attacks.

Table.2. Confusion Matrix for the Four-Category Experiments on Dataset

Actual Class / **Predicted Class**	**DOS**	**Probing**	**R2L**	**U2R**
DOS	**17843**	**4**	**1**	**17**
Probing	**15**	**4040**	**0**	**23**
R2L	**25**	**6**	**412**	**31**
U2R	**10**	**3**	**0**	**19560**

The **False Positive Rate (FPR)** in intrusion detection refers to the proportion of normal nodes that are incorrectly classified as malicious. It measures the IDS's tendency to raise unnecessary alerts, impacting network performance and response strategies.

$$FPR = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

Minimizing FPR is critical to enhancing IDS reliability, as a high FPR could lead to wasted resources and decreased trust in the system's detection.

Table.3.Performance Comparison Results

Metrics	Methods							
	NB	**RF**	**MLP**	**SVM**	**GWOSVM**	**EFNN**	**MENN**	**GNN + MENN**
Accuracy (%)	**78.92**	**80.25**	**85.46**	**87.82**	**98.06**	**98.31**	**99.01**	**99.51**
Error rate (%)	**21.07**	**19.75**	**14.54**	**12.18**	**1.94**	**1.69**	**1.21**	**0.96**
Specificity (%)	**87.66**	**89.89**	**91.02**	**93.86**	**94.57**	**98.83**	**99.90**	**99.95**
False Positive Rate (%)	**20**	**18**	**15**	**12.5**	**8.2**	**6.3**	**4.7**	**3.9**
Time complexity(sec)	**35.67**	**30.08**	**28.27**	**25.52**	**20.12**	**3.85**	**3.41**	**3.01**

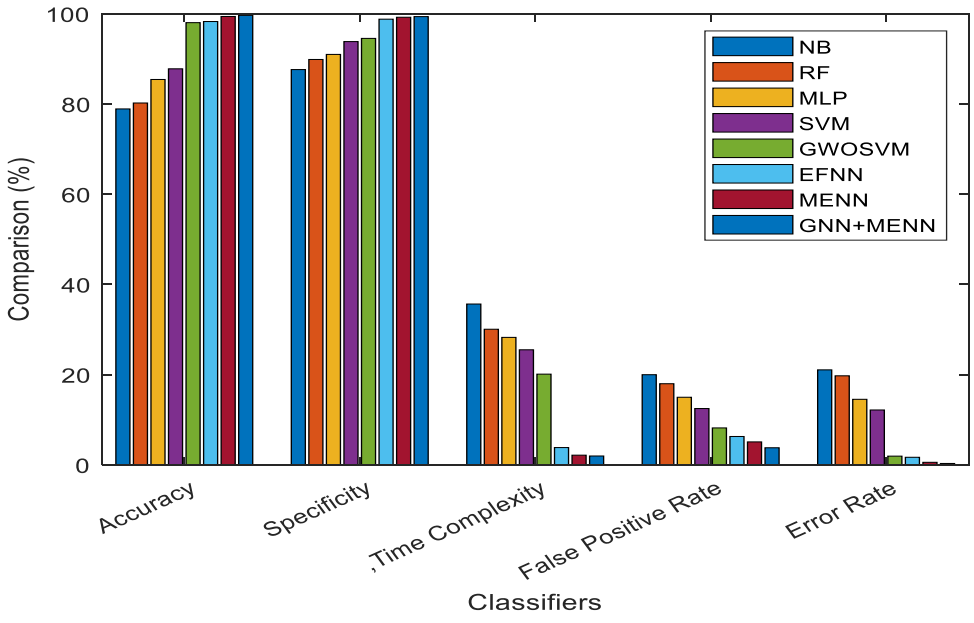

Figure 8: Accuracy, Error rate and specificity results Comparison of Various Classifiers

The figure 8 above compares the performance of the classifiers NB, RF, MLP, SVM, GWOSVM, EFNN and MENN the suggested MENN+GNN schemes in terms of accuracy, error rate, and specificity. In the graph above, several approaches are displayed on the X-axis, while accuracy, error rate, and specificity numbers are represented on the Y-axis. The findings show that the newly presented MENN+GNN model outperformed the other existing models. For example in the above figure proposed MENN+GNN produces higher accuracy which is 99.5124(%) while available, NB, RF, MLP, SVM, GWOSVM, EFNN and MENN technique yields only 78.9272(%),80.2451(%),85.4561(%),87.8216(%), 98.0565 (%),98.3108 (%) and 99.0121 (%) respectively.

Table.4.Comparison of the Energy Consumption with the other methods

No. of nodes	Energy consumption (J)							
	NB	RF	MLP	SVM	GWOSVM	EFNN	MENN	GNN + MENN
30	18	16	15	14	13	8	7	6
60	38	27	21	17	15	10	8	7
90	41	37	29	25	20	15	12	10
120	52	48	41	35	28	22	18	15
150	54	50	44	39	35	28	24	20

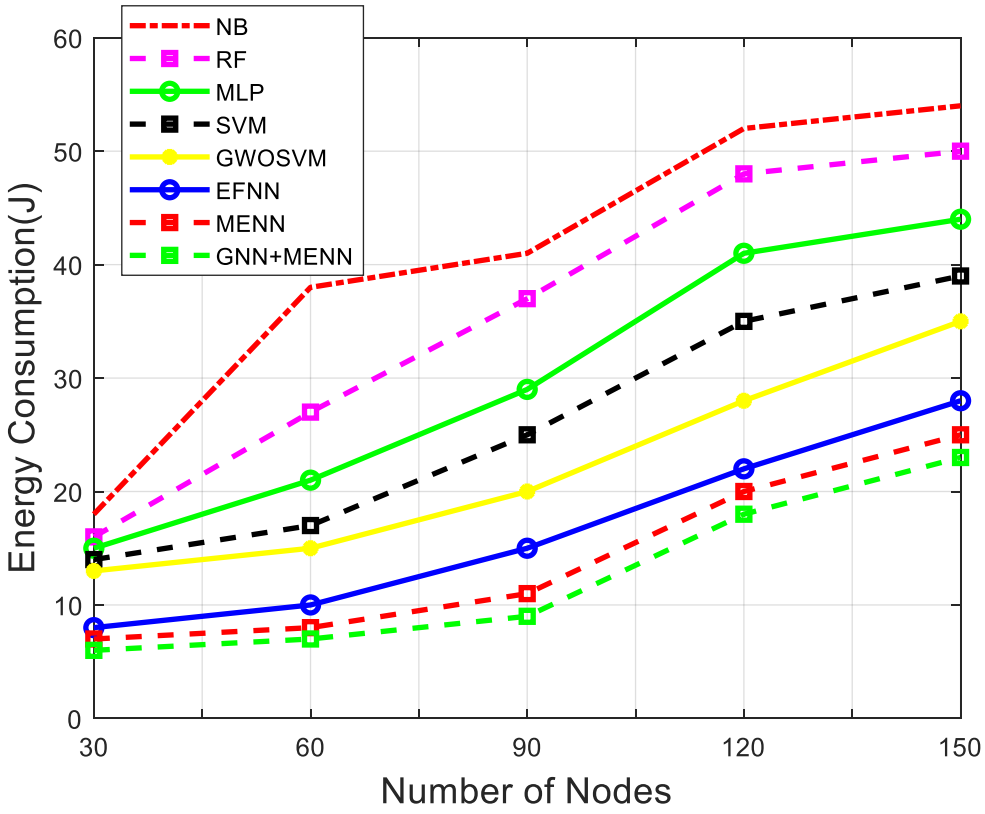

Figure 9: Energy consumption results

The figure 9 above compares the energy consumption results of the proposed MENN+GNN to current classifier methods such as NB, RF, MLP, SVM, GWOSVM, EFNN and MENN. The X and Y axes in the graph above reflect the number of **nodes**. Energy usage figures are displayed. As seen in the data, the newly adopted MENN+GNN model yielded reduced energy usage results 20(J) while available NB, RF, MLP, SVM, GWOSVM, and EFNN technique consumes 54(J), 50(J), 44(J), 39(J), 35(J) and 28(J) respectively.

V.CONCLUSION AND FUTURE WORK

This paper presents a the proposed approach demonstrates that blockchain technology, when integrated with optimized clustering, advanced feature selection, and a robust intrusion detection framework, can significantly enhance the security, trustworthiness, and energy efficiency of wireless sensor networks. By forming clusters using Improved Dove Swarm Optimization (IDSO) and selecting significant features with the hybrid IWOA-IGA algorithm, the system effectively reduces energy consumption and prolongs network lifetime. Furthermore, the integration of a blockchain-based trust evaluation mechanism with the Modified Elman Neural Network and Graph Neural Networks (MENN-GNN) provides a decentralized, tamper-proof, and consensus-driven security model that accurately detects and mitigates threats. Leveraging the NSL-KDD'99 dataset for training and evaluation strengthens the reliability and robustness of the proposed Trust-Based Intrusion Detection System (IDS). By continuously monitoring node behavior and identifying malicious activities, the solution ensures secure data transmission while preserving the overall integrity of the WSN. Future research can explore real-world implementations, scalability to large-scale networks, and adaptive learning mechanisms to further solidify blockchain's role in creating trusted, energy-efficient WSN environments. The proposed MENN+GNN produces higher accuracy which is 99.5124(%) while available, NB, RF, MLP, SVM, GWOSVM, EFNN and MENN technique yields only 78.9272(%),80.2451(%),85.4561(%),87.8216(%), 98.0565 (%),98.3108 (%) and 99.0121 (%) respectively.

REFERENCES

1. Abd, E.; Ibrahim, A.; Darwish, S.M. Towards designing a trusted routing scheme in wireless sensor networks: A new deep blockchain approach. IEEE Access 2021, 9, 103822–103834. [Google Scholar]
2. Almomani I., Al-Kasasbeh B., Al-Akhras M. WSN-DS: A dataset for intrusion detection systems in wireless sensor networks. J. Sens. 2016;2016:4731953. doi: 10.1155/2016/4731953.
3. Amjad, S.; Abbas, S.; Abubaker, Z.; Alsharif, M.H.; Jahid, A.; Javaid, N. Blockchain Based Authentication and Cluster Head Selection Using DDR-LEACH in Internet of Sensor Things. Sensors 2022, 22, 1972. [Google Scholar] [CrossRef] [PubMed]
4. Alghamdi, W., Rezvani, M., Wu, H., & Kanhere, S. S. (2019). Routingaware and malicious node detection in a concealed data aggregation for WSNs. ACM Transactions on Sensor Networks (TOSN), 15(2), 1-20.

5. Awan, S.; Javaid, N.; Ullah, S.; Ullah Khan, A.; Qamar, A.M.; Choi, J.G. Blockchain Based Secure Routing and Trust Management in Wireless Sensor Networks. Sensors 2020, 22, 411. [Google Scholar]
6. Azarhava, H., & Niya, J. M. (2020). Energy efficient resource allocation in wireless energy harvesting sensor networks. IEEE Wireless Communications Letters, 9(7), 1000-1003.
7. Bao, Z., Wang, Q., Shi, W., Wang, L., Lei, H., & Chen, B. (2020). When Blockchain Meets SGX: An Overview, Challenges and Open Issues. IEEE Access.
8. Dener M., Okur C., Al S., Orman A. WSN-BFSF: A New Dataset for Attacks Detection in Wireless Sensor Networks. IEEE Internet Things J. 2023;11:2109–2125. doi: 10.1109/JIOT.2023.3292209. [DOI] [Google Scholar]
9. Elhoseny, M.; Tharwat, A.; Hassanien, A.E. Bezier curve based path planning in a dynamic field using modified genetic algorithm. *J. Comput. Sci.* 2018, *25*, 339–350.
10. Goyat, R.; Kumar, G.; Alazab, M.; Saha, R.; Thomas, R.; Rai, M.K. A secure localization scheme based on trust assessment for WSNs using blockchain technology. Future Gener. Comput. Syst. 2021, 125, 221–231. [Google Scholar] [CrossRef]
11. Gourisetti, S. N. G., Mylrea, M., & Patangia, H. (2019). Evaluation and demonstration of blockchain applicability framework. IEEE Transactions on Engineering Management, 67(4), 1142-1156.
12. Goyat, R., Kumar, G., Rai, M. K., Saha, R., Thomas, R., & Kim, T. H. (2020). Blockchain Powered Secure Range-Free Localization in Wireless Sensor Networks. Arabian Journal for Science and Engineering, 45(8), 6139-6155.
13. Hsiao, S.; Sung, W.T. Employing blockchain technology to strengthen security of wireless sensor networks. IEEE Access 2021, 9, 72326–72341. [Google Scholar] [CrossRef]
14. Hussain K., Xia Y., Onaizah A.N., Manzoor T., Jalil K. Hybrid of WOA-ABC and proposed CNN for intrusion detection system in wireless sensor networks. Optik. 2022;271:170145. doi: 10.1016/j.ijleo.2022.170145. [DOI] [Google Scholar]
15. Ismkhan, H. Black box optimization using evolutionary algorithm with novel selection and replacement strategies based on similarity between solutions. *Appl. Soft Comput.* 2018, *64*, 260–271.
16. Jiang, Q., Zeadally, S., Ma, J., & He, D. (2017). Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks. IEEE Access, 5, 3376-3392.
17. Kandris, D., Nakas, C., Vomvas, D., & Koulouras, G. (2020). Applications of wireless sensor networks: an up-to-date survey. Applied System Innovation, 3(1), 14.
18. Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks?*arXiv preprint arXiv:*1810.00826 (2018).
19. Khan, Z. A., Latif, G., Sher, A., Usman, I., Ashraf, M., Ilahi, M., & Javaid, N. (2019). Efficient routing for corona based underwater wireless sensor networks. Computing, 101(7), 831-856.

20. Kim, T. H., Goyat, R., Rai, M. K., Kumar, G., Buchanan, W. J., Saha, R., & Thomas, R. (2019). A novel trust evaluation process for secure localization using a decentralized blockchain in wireless sensor networks. IEEE Access, 7, 184133-184144.
21. Kumar, M. H., Mohanraj, V., Suresh, Y., Senthilkumar, J., & Nagalalli, G. (2020). Trust aware localized routing and class based dynamic block chain encryption scheme for improved security in WSN. Journal of Ambient Intelligence and Humanized Computing, 1-9.
22. Latif, K., Javaid, N., Ullah, I., Kaleem, Z., Abbas, Z.,& Nguyen, L. D. (2020). DIEER: Delay-Intolerant Energy-Efficient Routing with Sink Mobility in Underwater Wireless Sensor Networks. Sensors, 20(12), 3467
23. Li, M., Zhou, W., Liu, J., Zhang, X., Pan, F., Yang, H., ... & Luo, D. (2021). Vehicle interior noise prediction based on Elman neural network. Applied Sciences, 11(17), 8029.
24. Ling, Y., Zhou, Y. and Luo, Q., 2017. Lévy flight trajectory-based whale optimization algorithm for global optimization. IEEE access, 5, pp.6168-6186.
25. Lee, H. C., & Ke, K. H. (2018). Monitoring of large-area IoT sensors using a LoRa wireless mesh network system: Design and evaluation. IEEE Transactions on Instrumentation and Measurement, 67(9), 2177-2187.
26. Lazrag, H.; Chehri, A.; Saadane, R.; Rahmani, M.D. Efficient and secure routing protocol based on Blockchain approach for wireless sensor networks. Concurr. Comput. Pract. Exp. 2021, 33, e6144.
27. Mohiuddin G., Lin Z., Zheng J., Wu J., Li W., Fang Y., Wang S., Chen J., Zeng X. Intrusion detection using hybridized meta-heuristic techniques with Weighted XGBoost Classifier. Expert Syst. Appl. 2023;232:120596. doi: 10.1016/j.eswa.2023.120596.
28. Moinet, A., Darties, B., & Baril, J. L. (2017). Blockchain based trust & authentication for decentralized sensor networks. arXiv preprint arXiv:1706.01730.
29. Noel, A. B., Abdaoui, A., Elfouly, T., Ahmed, M. H., Badawy, A., & Shehata, M. S. (2017). Structural health monitoring using wireless sensor networks: A comprehensive survey. IEEE Communications Surveys & Tutorials, 19(3), 1403-1423.
30. Revanesh, M.; Sridhar, V. A trusted distributed routing scheme for wireless sensor networks using blockchain and meta-heuristics-based deep learning technique. Trans. Emerg. Telecommun. Technol. 2021, 32, e4259. [Google Scholar] [CrossRef]
31. Salmi S., Oughdir L. Performance evaluation of deep learning techniques for DoS attacks detection in wireless sensor network. J. Big Data. 2023;10:1–25.
32. Shin, S., & Kwon, T. (2019). A lightweight three-factor authentication and key agreement scheme in wireless sensor networks for smart homes. Sensors, 19(9), 2012.
33. She, W., Liu, Q., Tian, Z., Chen, J. S., Wang, B., & Liu, W. (2019). Blockchain trust model for malicious node detection in wireless sensor networks. IEEE Access, 7, 38947-38956.
34. Sharawi, M., Zawbaa, H.M. and Emary, E., 2017, February. Feature selection approach based on whale optimization algorithm. In 2017 Ninth international conference on advanced computational intelligence (ICACI) (pp. 163-168). IEEE.

35. Song, Y.; Wang, F.; Chen, X. An improved genetic algorithm for numerical function optimization. *Appl. Intell.* 2019, *49,* 1880–1902.
36. Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:*1609.02907 (2016).
37. Vinayakumar R., Alazab M., Soman K., Poornachandran P., Al-Nemrat A., Venkatraman S. Deep learning approach for intelligent intrusion detection system. IEEE Access. 2019;7:41525–41550. doi: 10.1109/ACCESS.2019.2895334. [DOI] [Google Scholar]
38. P.Vijayalashmi,P.M.Gomathi,Trust-Based Intrusion Detection System (IDS) with Energy Efficient Secure Communication in Wireless Sensor Network Using Modified Chicken Swarm Optimization and Modified Elman Neural Network (MENN), Telematique, ISSN: 1856-4194, Vol 24 Issue 1, 237– 258(2025)
39. P.Vijayalashmi,P.M.Gomathi, Intrusion Detection System in Wireless Sensor Network using Improved Whale Optimization and Enhanced Fuzzy Neural Network, Communications on Applied Nonlinear Analysis,ISSN: 1074-133X Vol 32 No. 9s(2025)
40. Wang, J., Gao, Y., Liu, W., Sangaiah, A. K., & Kim, H. J. (2019). Energy efficient routing algorithm with mobile sink support for wireless sensor networks. Sensors, 19(7), 1494.
41. Wazirali R., Ahmad R. Machine Learning Approaches to Detect DoS and Their Effect on WSNs Lifetime. Comput. Mater. Contin. 2022;70:4922–4946. doi: 10.32604/cmc.2022.020044.
42. Wu, X., Zhang, S., Xiao, W. and Yin, Y., 2019. The exploration/exploitation tradeoff in whale optimization algorithm. IEEE Access, 7, pp.125919-125928.
43. Xu, Y., & Huang, Y. (2020). Segment blockchain: A size reduced storage mechanism for blockchain. IEEE Access, 8, 17434-17441.
44. Yadav, M., Fathi, B., & Sheta, A. (2019). Selection of WSNs inter-cluster boundary nodes using PSO algorithm. Journal of Computing Sciences in Colleges, 34(5), 47-53.
45. Yetgin, H., Cheung, K. T. K., El-Hajjar, M., & Hanzo, L. H. (2017). A survey of network lifetime maximization techniques in wireless sensor networks. IEEE Communications Surveys & Tutorials, 19(2), 828-854.