Metaheuristic-Enhanced XGBoost Framework for Intrusion Detection in Smart Home IoT Systems

Shaotong Xue [1], Xu Liu[1, *], Meng Zhang [2], Lina Gu[1]
[1]Artificial intelligence and big data college, Hebei University of Engineering Science, Shijiazhuang 050091, China
[2.]Department of Electronic Information, Huaxin College of Hebei GEO University, Shijiazhuang 050700, China
E-mail: liuxu10168@163.com
[*]Corresponding author

Keywords: smart home systems, intrusion detection system, security, threats, metaheuristic algorithm, hybrid schemes, machine learning

Received: April 9, 2025

The growing integration of smart technologies into residential environments has heightened their exposure to cybersecurity threats, thereby necessitating robust and intelligent intrusion detection systems (IDS). This study proposes a hybrid AI-driven intrusion detection framework tailored for smart home networks, leveraging the Extreme Gradient Boosting Classifier (XGBoost or XGBC) enhanced by three metaheuristic optimization techniques: the Arithmetic Optimization Algorithm (AOA), Horse Herd Optimization (HHO), and Wild Geese Algorithm (WGA). The performance of these hybrid schemes XGAO, XGHH, and XGWG was rigorously evaluated using a comprehensive dataset containing 148,518 labeled network traffic instances, compiled through data mining methods. The dataset includes diverse attributes such as source and destination bytes, service types, protocols, and various connection-related flags. Performance evaluation was conducted using four standard classification metrics: accuracy, precision, recall, and F1-score. Among all models, the hybrid XGAO scheme demonstrated superior performance, achieving a training accuracy of 0.991, outperforming the baseline XGBC model, which scored 0.946 under the same conditions. In the testing phase, XGAO maintained high generalizability with an accuracy of 0.987, compared to 0.953 for XGBC. The XGAO model also excelled in recall (0.989), precision (0.991), and F1-score (0.990) for correct detections. These findings affirm that integrating XGBoost with AOA significantly enhances the classification accuracy and reliability of intrusion detection systems in smart home environments. The study contributes to the development of resilient, adaptive IDS architectures capable of mitigating evolving cyber threats in the domain of intelligent residential technologies.

Povzetek: Hibridni okvir XGBoost, optimiziran z metahevristiko AOA, HHO in WGA, omogoča bolj kvalitetno zaznavanje vdorov v pametnih domovih. XGAO doseže višjo točnost, odpornost in zanesljivost kot klasični XGBoost.

1 Introduction

Smart homes make effective integration of the digital and physical worlds possible. According to studies, there are currently more than 6 billion internet-connected devices, and in the upcoming years, that number is predicted to rise to over 26 billion [1]. As IoT technology is embraced by various industries, including smart cities and homes, healthcare, transportation, manufacturing, and agriculture, these figures are anticipated to increase [2], [3]. IoT technology is poised to transform society and the standard of living with these advancements [2]. Despite the ease of utilization and benefits of smart homes, some have claimed that the technology is highly invasive in people's lives [4]. The most brilliant homes have sensors installed on their smart devices, producing diverse data streams and linked events that give rise to a constant flow of information about the activities occurring in a specific home [3], [4]. Hence, cyber criminals frequently try to break into the network of smart homes to hijack the continuous information flow between the devices, enabling them to snoop on household activities [5].

Nowadays, many smart home appliances are linked to the Internet; these appliances are easily attacked and can result in significant issues that could negatively impact a user's life. Because attackers can be clever or use the same protocols that users use to make valid requests, some of these attacks are hard to identify [5]. The purpose of an IDS is to identify and stop network attacks [6]. Nevertheless, one-tier standard intrusion detection is insufficient to guarantee the security and privacy of wireless sensor networks due to several restrictions on smart home sensors and device manufacturers [7].

within a smart home environment, there are many security risks that modern wireless networks, devices, and sensors must contend with. ML is thought to be the perfect answer to this issue [8]. Without requiring explicit programming, machine learning technology utilizes

diverse learning frameworks to train sensors, devices, and other artificial intelligence-capable hardware [9]. For the following reasons, machine learning is the perfect way to handle security risks in contemporary wireless sensor networks and smart home devices [10]:

- Smart home appliances and wireless sensor networks (WSNs) don't rely on complex mathematical schemes.
- Correlated data sets are necessary for IoT applications. Furthermore, machine learning can handle the stochastic nature of WSNs and smart home appliances [9].
- Because the machine is automated, no human intervention is needed, making it perfect for smart home technologies to prevent and lessen potential threats to the system [11].

The research in this paper investigates the effectiveness of the Extreme Gradient Boosting algorithm in classifying cyber-attacks within smart homes. XGBoost is also reliable and very efficient, dealing with large databases and narrowing down to those crucial elements when detecting threats. It performs enhancement by leveraging three different optimization frameworks, namely the Wild Geese Algorithm, which can balance exploration and exploitation well; the Horse Herd Optimization algorithm, which enables the exploration of the solution space; and the Arithmetic Optimization Algorithm, which will improve the tuning of hyperparameters. These combined strategies reinforce the model's potential to identify safe operations from online dangers and secure smart homes.

This study's goal is to find better ways to classify intrusion detection models so that smart home networks can deal with the rising number of security flaws. The fundamental purpose is threefold: (1) to make the XGBoost-based Intrusion Detection System (IDS) work better by adding metaheuristic optimization techniques; (2) to see how well three specific optimizers Arithmetic Optimization Algorithm (AOA), Horse Herd Optimization (HHO), and Wild Geese Algorithm (WGA) classify data when used with the XGBoost framework; and (3) to see how well these hybrid models work on a large-scale, real-world smart home dataset with 148,518 labeled instances using standard classification metrics like accuracy, precision, recall, and F1-score.

2 Literature survey

It has been more than a decade since IDS started to see significant improvements driven by the integration of ML techniques [12]. As a result of this evolution, intrusion detection today is a classification task that uses labeled data to distinguish abnormal activities from normal ones effectively. Diverse ML methods have been developed to improve the efficiency of IDS, including Random Forest, Decision Trees, Naïve Bayes, deep learning, and ensemble methods. For example, Yousef et al. showed that the best technique to detect various types of attacks is the Random Forest Classifier (RFC) [13].

In 2015, Ghazali et al. applied five classification strategies on the NSL-KDD database and reported an accuracy of 96.7%, a detection rate of 95.5%, and a FAR of 4.7% [14]. Following further research, Kevric et al. reported in 2017 an ACC of 89.24% [15], while the RF-based 2018 model recommended by Hadi yielded an astonishing 99.33% ACC [16]. Karami presented in 2018 an anomaly-based intrusion detection system utilizing the fuzzy tactic [17]. Gu et al., in 2019, recommended an SVM-based model that had only an ACC of 93.64% with a higher FAR of 20.28% [18]. In 2020, Tabash et al. combined the merits of DL and NB to boost the attribute retrieval that improved performance metrics [19].

Ensemble methods represent recent approaches. Whereas in the work of Abbas et al., hybrid ensembles were recommended for the improvement of performance in the detection system in 2021, Louk et al. recommended an ensemble approach drawing on bagging [20]. Mhawi et al. recommended an ensemble based on Boosting in 2022, pointing out the focus on sequential training of classifiers [21]. Besides that, Bertoni et al. created a scheme drawing on stacking to achieve accuracy via meta-classification [22]. These ensemble learning innovations taken altogether have proven to be successful in increasing the precision and resiliency of the IDS while significantly reducing false alarms as a whole [23].

Recent research has shown growing interest in intelligent intrusion detection tailored for smart and IoT environments. [5] proposed a learning-based ensemble framework for IoT smart homes, combining diverse classifiers for anomaly detection. [3] introduced an IDS for 5G device-to-device communications, emphasizing ML-driven optimization in latency-sensitive networks (DOI: 10.31449/inf.v47i6.4635). [6] designed a feature-selection-based ensemble IDS for IIoT, highlighting dimensionality reduction’s importance. [7] developed a deep transfer learning framework to improve resilience in constrained IoT networks. [8] reviewed metaheuristic optimization in IDS, covering algorithms like PSO and AGA, but identified gaps in benchmarking newer techniques. Although metaheuristic approaches have been applied to feature selection and model tuning, direct, comparative evaluations of advanced optimizers such as AOA, HHO, and WGA remain scarce within a unified IDS framework. This study addresses that gap by integrating and systematically benchmarking these optimizers alongside XGBoost on a comprehensive smart home dataset, offering novel insights into their comparative effectiveness.

XGBoost and metaheuristic optimization methods like AOA, HHO, and WGA have been studied separately in the fields of machine learning and intrusion detection. This study, on the other hand, is new because it systematically combines and tests these optimizers within the XGBoost framework for smart home security. The study is the first to use these approaches on a large, real-world smart home dataset with 148,518 instances. This has not been done much before in the literature. A rigorous performance study, convergence behavior, ablation investigations, and statistical validation are used to

compare three hybrid models: XGAO, XGHH, and XGWG. Also, real-time limitations and class-level performance assessments are included, which shows how useful it may be in the actual world. This combined method gives us a better knowledge of how optimizers affect classifier performance. It also sets up a new way to compare metaheuristic-driven IDS in smart settings.

Table 1 presents a comparative analysis of recent intrusion detection systems, highlighting methods, datasets, and key performance metrics. It underscores the superior performance and novelty of the proposed (XGAO) on a smart home dataset.

Table 1: Comparative analysis of recent intrusion detection systems

Table 1. Comparative summary of IDS methods in recent literature.				
Study	Method	Dataset	Accuracy	Recall
Ghazali et al. (2015)	Multiple Classifiers	NSL-KDD	96.7%	95.5%
Kevric et al. (2017)	Tree-based Ensemble	NSL-KDD	89.24%	-
Hadi (2018)	Random Forest	Big Data Traffic	99.33%	-
Karami (2018)	Fuzzy Logic	Unknown	-	-
Gu et al. (2019)	SVM Ensemble	Unknown	93.64%	-
Tabash et al. (2020)	Naive Bayes + DL	Custom Network Dataset	-	-
Abbas et al. (2021)	Hybrid Ensemble (Bagging)	Industrial Control Network	-	-
Mhawi et al. (2022)	Boosting-based Ensemble	Custom Network Dataset	-	-
This Study (XGAO)	XGBoost + AOA	Smart Home Dataset (148,518 samples)	99.1% (Train), 98.7% (Test)	98.9% (Correct Detection)

Because they are more accurate and reliable, ensemble-based methods including bagging, boosting, and stacking have been widely used in intrusion detection systems. In the expanded literature portion of this work, the study compares the XGBoost-based hybrid schemes (XGAO, XGHH, XGWG) against standard ensemble classifiers like Random Forest and Gradient Boosting. XGBoost is an ensemble model, but the presented methods stand out because they use metaheuristic optimization, which makes them better at adapting to complicated smart home incursion patterns and converging faster.

3 Recommended work

- **XGBoost Classifier (XGBC)**

Compared to the traditional GBDT method, significantly improved computation speed, scalability, and generalization performance can be achieved with XGBoost. In [24]Detailed descriptions of the optimization strategy and objective function in the XGBoost framework are given. The objective function of the XGBoost is depicted by Eq. (1):

$$F_{Obj}(\theta) = J(\theta) + \Omega(\theta), \quad where\ J(\theta) = l(p_i, m_i), \tag{1}$$

$$\Omega(\theta) = \gamma r + \frac{1}{2}\lambda\|w\|^2. \tag{2}$$

$J(\theta)$, in which θ is the different parameters of the formula, and $\Omega(\theta)$ are the components of the objective function that comprises XGBoost. The loss function $J(\theta)$ is a differentiable convex function that regulates the fitting of data from the model and calculates the deviation of the measured target, m_i, from the predicted value, p_i. Two main convex loss functions are logistic loss, given by Eq. (3), and mean square loss, given by Eq. (4).

$$j(p_i, m_i) = m_i \ln(1 + e^{-p_i}) + (1 - m_i)\ln(1 + e^{p_i}) \tag{3}$$

$$j(p_i, m_i) = (p_i - m_i)^2 \tag{4}$$

Furthermore, a regularization term Ω(θ) affects complex schemes. The learning rate, denoted by γ, can reach a maximum value of 0, while r displays the total count of leaves in the tree. Tree pruning is produced when

γ is multiplied by r, which lowers overfitting. Unlike traditional GBDT, XGBoost amplifies this phrase using a regularization parameter $^1/_2 \lambda\|w\|^2$. This parameter displays leaf weights w. This component is further improved to lessen overfitting and increase the model's capacity for generalization.

Nevertheless, conventional optimization methods encounter challenges because the objective function in Eq. (1) combines function parameters and model penalty terms. Checking whether the target m_i can be determined using Eq. (5) is therefore crucial.

$$J(\theta) = \sum_{i=1}^{r} j(m_i, p_i^{(t-1)} + Z_t(r_i) + \Omega(\theta). \quad (5)$$

The enhancement tries to build a tree construction that diminishes the target function on each cycle. Based on the outcomes and residuals of the previous tree (residuals=real value-predictive value), the current residual regression tree is fitted using the tree structure. $Z_t(r_i)$ displays the tree that instance i in the t cycle created.

When the square loss function is solved, the objective function of Eq. (4) is ideal, making solving other loss functions challenging. Different loss functions can be solved using Eq. (5) by translating Eq. (6) through the two-order Taylor expansion. These include Eqs. (7) and (8). This simplifies the optimization process because the ultimate goal function depends only on the error function's first and second derivatives for each data point.

$$J(\theta) = \sum_{i=1}^{r} j(m_i, p_i^{(t-1)} + \mathcal{g}_i Z_t(r_i) + \frac{1}{2}\mathcal{h}_i Z_t^2(r_i) + \Omega(\theta). \quad (6)$$

$$\mathcal{h}_i = \partial^2_{p^{(t-1)}} j\,(m_i, p^{(t-1)}) \quad (7)$$

$$\mathcal{g}_i = \partial_{p^{(t-1)}} j\,(m_i, p^{(t-1)}) \quad (8)$$

The flowchart of XGBC is displayed in Fig. 1.

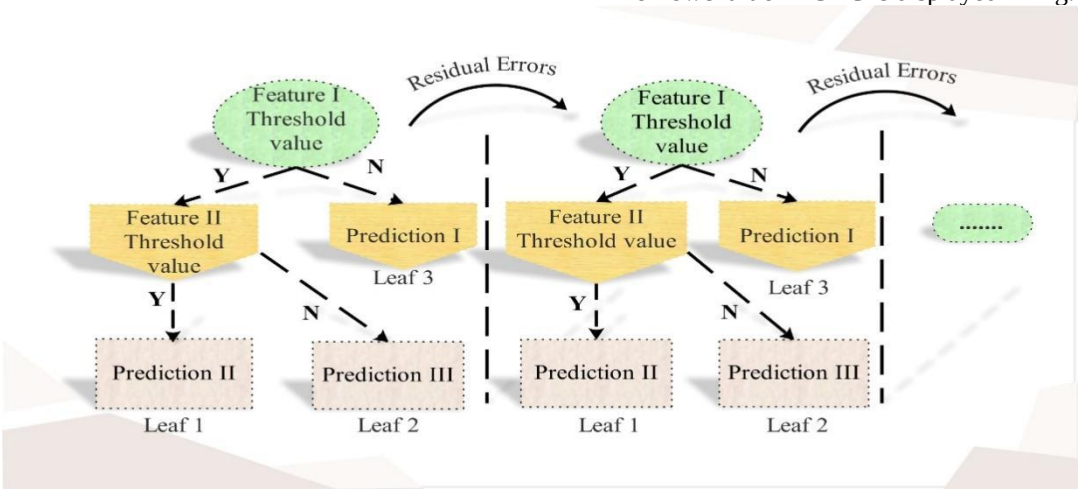


Figure 1: XGBC diagram

- **Arithmetic Optimization Algorithm (AOA)**

Using fundamental arithmetic operations, arithmetic optimization is a metaheuristic with roots in number theory that analyzes numerical values and finds the best solution given predetermined parameters [25]. AOA includes stages for exploration and exploitation, much like traditional population-driven optimization. It was inspired by applying arithmetic to solve mathematical problems. While exploitation improves the accuracy of the resolutions, exploration scans the search domain for viable answers. The three main phases of the AOA algorithm are outlined and discussed in the sections that follow [26].

Initialization

The Arithmetic Optimization algorithm starts with the creation of a random set of candidate resolutions (X). The algorithm assumes that the best candidate solution is either the neighborhood's best or optimal solution in each cycle.

$$X = \begin{bmatrix} x_{1,1} & \cdots & \cdots & x_{1,j} & x_{1,r-1} & x_{1,r} \\ x_{2,1} & \cdots & \cdots & x_{2,j} & \cdots & x_{2,r} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1,1} & \cdots & \cdots & x_{N-1,j} & \cdots & x_{N-1,r} \\ x_{N,1} & \cdots & \cdots & x_{N,j} & x_{N,r-1} & x_{N,r} \end{bmatrix} \quad (9)$$

A choice must be made regarding whether to begin the AOA procedure with the exploration or exploitation phases. The function value at the i^{th} cycle is then represented by the MOA function, which is calculated using Eq. (10).

$$MOA(B_{Iter}) = Min + A_{Iter} \times \left(\frac{Max - Min}{U_{Iter}}\right) \quad (10)$$

The maximum count of cycles is depicted by U_{Iter}, where A_{Iter} ranges from 1 to U_{Iter-1}. When modifying the step size of arithmetic operators during the global search phase, the terms Max and Min serve as reference points for the highest and lowest values of the accelerated function.

- *exploration*

The exploration mechanism uses multiplication (MO) or Division (DO) operators to produce dispersed values or options. The difficulties of directly reaching the target with this indirect approach frequently require several cycles to find a nearly ideal solution for exploitation. Eq. (11) defines the equations for position updates and summarizes the formulation of two essential search strategies during the exploration phase.

$$x_{i,j}(A_{Iter}+1)=\begin{cases} b(x_j)\div(MOP+\varepsilon)\times\left(\left(UB_j-LB_j\right)\times\omega+LB_j\right), q2<0.5 \\ b(x_j)\times MOP\times\left(\left(UB_j-LB_j\right)\times\omega+LB_j\right), otherwise \end{cases} \tag{11}$$

In this case, $x_{i,j}(A_{Iter})$ depicts the j^{th} position of the i^{th} resolution at a specific cycle (A_{Iter}), and $b(x_i)$ denotes the j^{th} location found in the most advantageous solution found thus far. LB_j and UB_j indicate the lower and upper boundary values for the j^{th} location, accordingly, and ε is a small integer value. ω serves as a parameter for control. The function value of $MOP(A_{Iter})$ can be expressed as follows:

$$MOP(A_{Iter})=1-\frac{{A_{Iter}}^{1/\gamma}}{{U_{Iter}}^{1/\gamma}} \tag{12}$$

where γ is an essential parameter that establishes the level of accuracy of the exploitation process throughout the cycles.

- *exploitation*

The outcomes of addition (AO) and subtraction (SO) operators in mathematical computations are closely packed. As a result, during the exploitation phase, these operators repeatedly and efficiently converge toward the target. Eq. (13) summarizes position updating equations and identifies two main search strategies in the exploitation phase. The Arithmetic Optimization algorithm uses SO and AO operators during exploitation to help avoid local search traps and find the optimal solution through related search strategies.

$$x_{i,j}(A_{Iter}+1)=\begin{cases} b(x_j)-MOP\times\left(\left(UB_j-LB_j\right)\times\omega+LB_j\right), q3<0.5 \\ b(x_j)+MOP\times\left(\left(UB_j-LB_j\right)\times\omega+LB_j\right), otherwise \end{cases} \tag{13}$$

- **Horse Herd Optimization (HHO)**

The foundation of HOA, as presented by [27], is the way horses behave in their natural surroundings. The six main behavioral traits that inform it are: defense mechanisms, roaming, grazing, imitation, hierarchy, and sociability. As outlined in Eq. (14), these behaviors serve as the foundation for HOA, directing the movement of horses during each cycle.

$$X_r^{Iter,A}=\vec{f}_r^{Iter,A}+X_r^{(Iter-1),A},\ A=\alpha,\beta,\gamma,\delta \tag{14}$$

where A signifies the age range, $Iter$ is the current cycle, and $X_r^{Iter,A}$ displays the situation of the r^{th} horse. The horse's age range is also reflected in A, and its velocity vector is indicated by $\vec{f}_r^{Iter,A}$. Horses usually live 25 to 30 years and display a range of behaviors during that time. These actions fall into three categories: δ (0–5 years), γ (5–10 years), β (10-15 years) and α (over 15 years old). Horse ages are calculated using a large response matrix arranged according to performance. Group α comprises the top 10%, followed by group β with 20%, and groups γ and δ with the remaining 30% and 40%, respectively. These behaviors define motion vectors and algorithmic cycles for horses of various ages.

$$\begin{aligned} \vec{f}_r^{Iter,\alpha}&=\vec{G}_r^{Iter,\alpha}+\vec{D}_r^{Iter,\alpha} \\ \vec{f}_r^{Iter,\beta}&=\vec{G}_r^{Iter,\beta}+\vec{H}_r^{Iter,\beta}+\vec{S}_r^{Iter,\beta}+\vec{D}_r^{Iter,\beta} \\ \vec{f}_r^{Iter,\gamma}&=\vec{G}_r^{Iter,\gamma}+\vec{H}_r^{Iter,\gamma}+\vec{S}_r^{Iter,\gamma}+\vec{I}_r^{Iter,\gamma}+\vec{D}_r^{Iter,\gamma}+\vec{R}_r^{Iter,\gamma} \\ \vec{f}_r^{Iter,\delta}&=\vec{G}_r^{Iter,\delta}+\vec{I}_r^{Iter,\delta}+\vec{R}_r^{Iter,\delta} \end{aligned} \tag{15}$$

A connection between positions (X) and their corresponding cost values ($C(X)$) was established by clarifying the global matrix derivation using Eqs. (16) and (17).

$$X=\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{r,1} & x_{r,2} & \cdots & x_{r,d} \end{bmatrix},\quad C(X)=\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_r \end{bmatrix} \tag{16}$$

$$Global\ Matrix=[X\ \ C(X)]=\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} & c_1 \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} & c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{r,1} & x_{r,2} & \cdots & x_{r,d} & c_r \end{bmatrix} \tag{17}$$

x displays the position, and $C(x)$ displays the related cost for each position, as the preceding equations have displayed. Moreover, r displays the count of horses, and d displays the problem's size. The global matrix is then arranged according to the last column, which displays expenses. The age of the horse is entered in this column. Low speed, high accuracy, and high probability are advantageous when the best solution is likely to occur. Conversely, low accuracy and high speed are advantageous in scenarios where an ideal solution is unlikely. The following formula is used to get the overall velocity vector:

The speed of horses between the ages of 0 and 5:

$$\begin{aligned} &\vec{f}_r^{Iter,\delta} \\ &=\left[g_r^{(Iter-1),\delta}\ \omega_g\left(\breve{u}+P\breve{l}\right)\left[X_r^{(Iter-1)}\right]\right] \\ &+\left[i_r^{(Iter-1),\delta}\ \omega_i\left[\left(\frac{1}{P^N}\sum_{j=1}^{P^N}\hat{X}_j^{Iter-1}\right)\right.\right. \\ &\left.\left.-X^{Iter-1}\right]\right]+\left[R_r^{(Iter-1),\delta}\ \omega_r P X^{Iter-1}\right. \end{aligned} \tag{18}$$

Horses aged 5 to 10 years old at their fastest:

$$\vec{f}_r^{Iter,\gamma} = \left[g_r^{(Iter-1),\gamma}\ \omega_g\left(\breve{u} + P\breve{l}\right)\left[X_r^{(Iter-1)}\right]\right] + \left[h_r^{(Iter-1),\gamma}\omega_h\left[X_*^{(Iter-1)} - X_r^{(Iter-1)}\right]\right] + \left[S_r^{(Iter-1),\gamma}\ \omega_S\left[\left(\frac{1}{N}\sum_{j=1}^{N} X_j^{Iter-1}\right) - X^{Iter-1}\right]\right] + \left[i_r^{(Iter-1),\gamma}\ \omega_i\left[\left(\frac{1}{P^N}\sum_{j=1}^{P^N} \hat{X}_j^{Iter-1}\right) - X^{Iter-1}\right]\right] - \left[d_r^{(Iter-1),\gamma}\ \omega_d\left[\left(\frac{1}{q^N}\sum_{j=1}^{q^N} \hat{X}_j^{Iter-1}\right) - X^{Iter-1}\right]\right] + \left[R_r^{(Iter-1),A}\ \omega_r P X^{Iter-1}\right] \tag{19}$$

The speed of horses between the ages of 10 and 15:

$$\vec{f}_r^{Iter,\beta} = \left[g_r^{(Iter-1),\beta}\ \omega_g\left(\breve{u} + P\breve{l}\right)\left[X_r^{(Iter-1)}\right]\right] + \left[h_r^{(Iter-1),\beta}\omega_h\left[X_*^{(Iter-1)} - X_r^{(Iter-1)}\right]\right] + \left[S_r^{(Iter-1),\beta}\ \omega_S\left[\left(\frac{1}{N}\sum_{j=1}^{N} X_j^{Iter-1}\right) - X^{Iter-1}\right]\right] - \left[d_r^{(Iter-1),\beta}\ \omega_d\left[\left(\frac{1}{q^N}\sum_{j=1}^{q^N} \breve{X}_j^{Iter-1}\right) - X^{Iter-1}\right]\right] \tag{20}$$

The following velocity is displayed by horses who are over 15 years old

$$\vec{f}_r^{Iter,\alpha} = \left[g_r^{(Iter-1),\alpha}\ \omega_g\left(\breve{u} + P\breve{l}\right)\left[X_r^{(Iter-1)}\right]\right] - \left[d_r^{(Iter-1),\alpha}\ \omega_d\left[\left(\frac{1}{q^N}\sum_{j=1}^{q^N} \breve{X}_j^{Iter-1}\right) - X^{Iter-1}\right]\right] \tag{21}$$

In Fig. 2, the HHO flowchart is displayed.

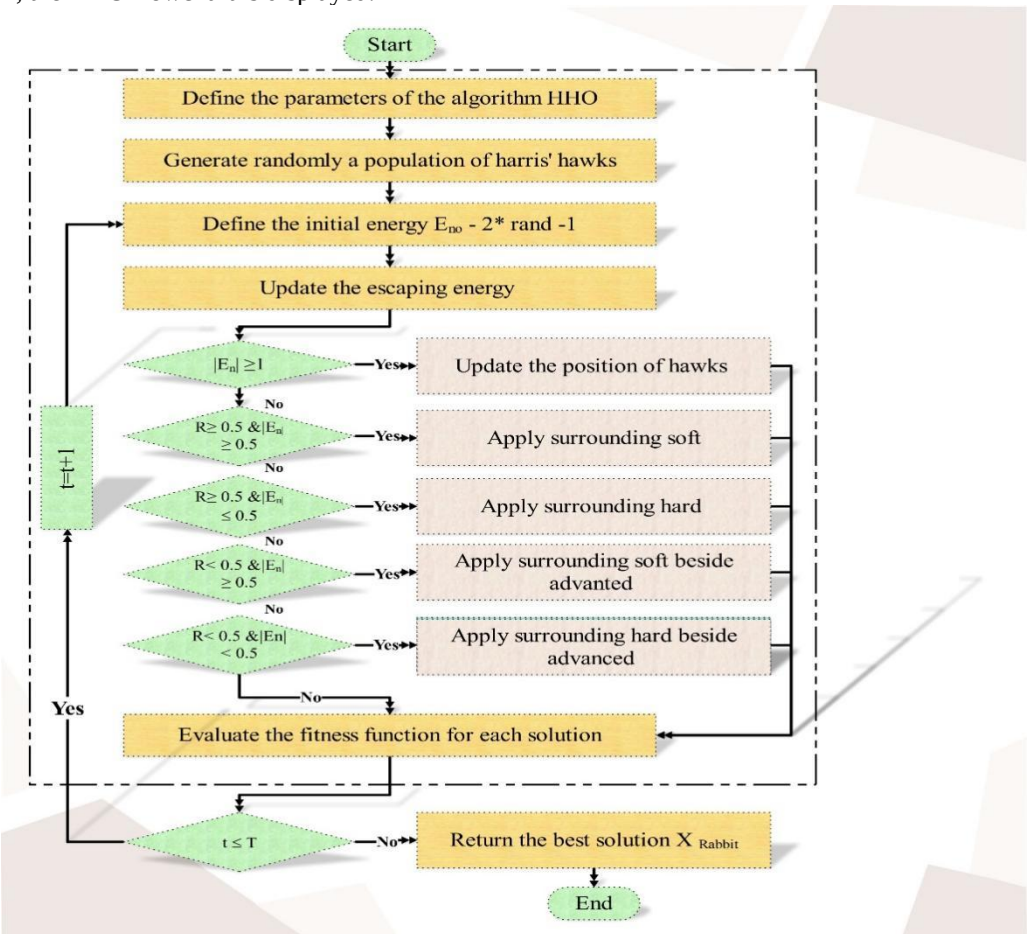


Figure 2: HHO flowchart

- **Wild Geese Algorithm (WGA)**

There has been a noticeable upsurge recently in the creation of frameworks that draw inspiration from the collective behaviors of animals to address challenging global continuous optimization problems on a large scale. The WGA is a highly efficient algorithm introduced in this paper [28]. Wild geese life cycles include population dynamics, coordinated group migration, evolutionary mechanisms, reproduction, and mortality. All of these factors have an impact on the WGA. The following briefly describes the WGA's fundamental phases:

1. A phase of controlled and synchronized gather migration, also known as migration and speed adjustments.
2. Wild geese roam around and look for food.
3. The Mechanisms of Evolution in Wild Goose Populations through Breeding
4. Wild Goose Population Migration Patterns, Survival Rates, and Coordinated Evolutionary Shifts.

First, a population of wild geese is created, and the number displays each goose's position x_i. Personal best solutions (p_i) and migration velocities (v_i) are computed for each goose. The entire geese population is then ranked from the most optimal to the least optimal based on how well they perform the target function. Using this modeling strategy, each wild goose adapts its behavior based on information gathered from nearby geese in the ranked population. The ensuing subsections will offer more thorough explanations of the WGA phases.

- *Migration and displacement velocity*

Wild geese must migrate on a coordinated and organized flight path to reach the individuals at the front and those nearby in the sorted population—the velocity and displacement equations in Eqs. (22) and (23) rely on the geese's harmonized velocity.

$$v_{i,r}^{Iter+1} = (q_{1,r} \times v_{i,r}^{Iter} + q_{2,r} \times (v_{i+1,r}^{Iter} - v_{i-1,r}^{Iter})) + q_{3,r} \times (p_{i,r}^{Iter} - x_{i-1,r}^{Iter}) + q_{4,r} \times (p_{i+1,r}^{Iter} - x_{i,r}^{Iter}) + q_{5,r} \times (p_{i+2,r}^{Iter} - x_{i+1,r}^{Iter}) + q_{6,r} \times (p_{i-1,r}^{Iter} - x_{i+2,r}^{Iter}) \quad (22)$$

The r^{th} dimension of the i^{th} the variables denote the wild goose's current position, best position, and current velocity $x_{i,r}^{Iter}$, $p_{i,r}^{Iter}$, and $v_{i,r}^{Iter}$, respectively. The random numbers $q_{k,r}$; $k = 1, 2, \ldots, 11$ are uniformly distributed and range from 0 to 1.

As stated in Eq. (30), changes in the position and velocity of each wild goose (i.e., the i^{th} wild goose) depend on the positions of nearby geese as well as the velocities of the geese ahead and behind it, which are represented as ($v_{i+1,r}^{Iter} - v_{i-1,r}^{Iter}$). To guide their movement and direction to close the distance between them, wild geese rely on information shared by individuals nearby in the synchronized flock. Furthermore, as described in Eq. (23), they incorporate the world's top performer as an additional benchmark to direct the flock's collective motion.

$$x_{i,r}^{v} = p_{i,r}^{Iter} + q_{7,r} \times q_{8,r} \times ((g_r^{Iter} + p_{i+1,r}^{Iter} - 2 \times p_{i,r}^{Iter}) + v_{i,r}^{Iter+1}) \quad (23)$$

g_r^{Iter} represented the best position globally among all the group members at that particular cycle.

- *Walk and search for food*

The way this process is set up, the i^{th} Wild Goose changes course to follow its leading partner, suggesting that the i^{th} goose tries to get close to the $i + 1^{th}$ goose, denoted as ($\mathrm{p}_{\mathrm{i+1}}^{\mathrm{Iter}} - \mathrm{p}_{\mathrm{i}}^{\mathrm{Iter}}$). The following is the equation that describes the movement and foraging behavior of the wild goose, denoted as x_i^w:

$$x_{i,r}^{w} = p_{i,r}^{Iter} + q_{9,r} \times q_{10,r} \times (\mathrm{p}_{\mathrm{i+1,r}}^{\mathrm{Iter}} - \mathrm{p}_{\mathrm{i,r}}^{\mathrm{Iter}}) \quad (24)$$

- *Reproduction and evolution*

A further phase of wild geese's life cycle concerns reproduction and evolution. Using the migration equation (x_i^v) and the equation dictating their walking and foraging behavior (x_i^w), a model has developed. In all simulation scenarios, the parameter Cr, which is employed in the WGA algorithm described in this study, is uniformly set at 0.5.

$$x_{i,r}^{Iter+1} = \begin{cases} x_{i,r}^{v} & if \quad q_{11,r} \le Cr \\ x_{i,r}^{w} & othewise \end{cases} \quad (25)$$

- *Death, migration, and ordered evolution*

In some cases, the algorithm's population size is more significant and efficient than the count of algorithmic cycles, especially for certain functions. In contrast, the count of algorithm cycles is more critical and efficient when considering different functions than the size of the WGA algorithm's population. A death phase was added to ensure the algorithm converged uniformly to all test functions. The algorithm starts the algorithm with a maximum population size, N_p^i. Further, while the algorithm is iterating, the inferior members within the population are systematically removed according to the formula in Eq. (26). It is a process whereby it goes on reducing the population size to its eventual value represented as N_p^f in the last generation.

$$N_p = round\ (N_p{}^i - ((N_p{}^i - N_p{}^f) \times (\frac{FE_s}{FE_{s_{max}}}))) \quad (26)$$

The count of evaluations a function has received is indicated by FEs, and the maximum number permitted is indicated by FEs_{max}.

The hyperparameter tuning method includes widely used parameters from relevant literature, such as learning rate, max depth, n_estimators, subsample, colsample_bytree, and gamma, to make sure that the performance comparisons were fair. For all hybrid methods, AOA, HHO, and WGA were used to consistently optimize these parameters. This is in line with conventional literature settings, which backs up the stated accuracy and makes

sure that the performance benefits weren't just because of selective parameter modification.

The internal training procedure of XGBoost did not use the optimization methods AOA, HHO, and WGA. Instead, they were used to improve its hyperparameters. Each optimizer looked through the solution space on its own to find the best set of XGBoost parameters, such as the learning rate, max depth, number of estimators, and subsample ratio. The paper then utilized these improved parameter values to train the XGBoost classifier, which created three hybrid schemes: XGAO, XGHH, and XGWG. This method improves the performance of the model without changing the core algorithm of XGBoost. There was no feature selection or ensemble architecture used in this phase.

Each optimizer (AOA, HHO, WGA) is employed to tune key XGBoost hyperparameters: learning rate, max_depth, n_estimators, subsample, colsample_bytree, and gamma. The objective of each metaheuristic is to maximize classification accuracy on the validation set using 5-fold cross-validation. The fitness function for optimization is defined by the average validation accuracy, ensuring consistent and fair evaluation across all schemes.

4 Numerical databases

To ensure data quality and consistency, multiple preprocessing steps were implemented. Records containing missing or incomplete values were excluded from the dataset to prevent analytical bias. Continuous numerical features including source bytes, destination bytes, and connection duration were normalized using Min-Max scaling to standardize the feature range. Outlier detection was performed using the Z-score method, and data points exceeding three standard deviations from the mean were removed. These preprocessing procedures were essential to enhance the stability and performance of the machine learning models when applied to real-world smart home network traffic data.

The dataset, which included 148,518 occurrences, was split into training and testing sets using a 70/30 ratio and data is taken from this website (https://www.kaggle.com/datasets/arnavsmayan/smart-home-energy-usage-dataset). Stratified sampling was used throughout the split to preserve the original class distribution in both sets. This method made sure that any class label, whether it was normal or attack, was fairly represented in both subsets. This lowered the chance of sampling bias. During the creation of the model, the training set was tested again with 5-fold cross-validation to make sure that the model's performance was stable and consistent across different subsets. This validation method let the metaheuristic optimizers fine-tune hyperparameters without the danger of overfitting.

The dataset comprises 148,518 labeled samples categorized into two primary classes: normal and attack. Specifically, 38,792 instances (26.1%) are labeled as normal traffic, while 109,726 instances (73.9%) represent various types of cyber-attacks, indicating a moderate class imbalance. This distribution was preserved during training and testing through stratified sampling to ensure fair representation. The imbalance was further addressed through performance evaluation metrics such as precision, recall, and F1-score, which offer a more reliable assessment under skewed class conditions.

This work considers an extensive database, which was collected with much difficulty using data mining methodologies, including 148,518 occurrences, to assess the efficiency of different frameworks to enhance attack detection accuracy in network security systems. A wide range of critical parameters is included in the database that may contribute to analyzing network traffic and detecting possible security threats. These would encompass attributes for the duration in seconds that the connection lasted, whether TCP, UDP, and ICMP; service HTTP, FTP; flag, or the state of the connection; source bytes, which are the amount of data bytes sent from the source; destination bytes, which are the amount of data bytes sent from the destination; and a variety of other flags that indicate whether the source and destination IPs match, the count of incorrect fragments, urgent packets, logged-in status, and a variety of connection counts and error rates. Whether the connection is considered an attack or not is the target variable for the analysis. The interaction between these parameters and how they relate to each other in a pictorial form is done using a correlation matrix in Fig. 3, where the varied shading indicates the direction and magnitude of the correlation. A high positive correlation implies a tendency for one parameter to increase along with another, which could indicate a higher chance of an attack. On the other hand, a strong negative correlation indicates that a parameter tends to decrease as one increases, suggesting that some parameters may not be directly correlated with the likelihood of an attack. Less to no correlation is indicated by lighter shades in the matrix, indicating that specific parameters have little to no effect on one another.

Figure 3: The graph shows the correlation between input and output

Feature selection is an essential machine learning component, which is especially important for improving interpretability and model performance. In this case, it entails determining which pertinent parameters impact the accuracy of attack detection frameworks. As displayed in Fig. 4, the visual representation emphasizes the importance of different parameters according to how they affect feature selection. At 0.5368, source bytes (src bytes) are the most critical parameter, followed by destination bytes (dst bytes) at 0.43029 and service at 0.42483. These features indicate that the type of service accessed and the amount of data transferred are strong predictors of possible attacks. In contrast, the urgent parameter has a value of 0, indicating the less significant contribution of this feature towards the model's prediction capability for the future. Therefore, based on the most relevant features, the researchers' schemes would be simplified, computational complexity would be reduced, and the accuracy of attack detection would be improved. This leads, over time, to even more robust cybersecurity solutions. Besides strengthening the model, it also helps understand the root patterns associated with network attacks.

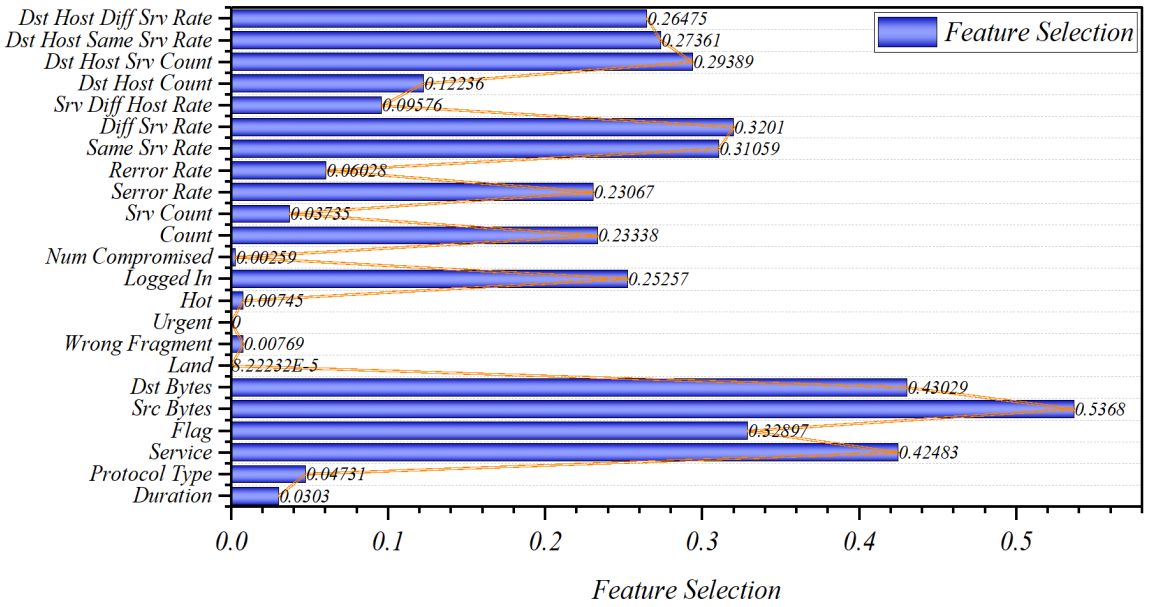


Figure 4: F Statistics feature selection method conducted for the database

5 Result and discussion

The convergence process of three different hybrid schemes, each using a different optimization algorithm and derived from the XGBC scheme, is displayed in Fig. 5. The schemes are displayed in the following ways: The pink line displays XGWG, the green line displays XGHH, and the blue line displays XGAO. The accuracy of all three schemes is initially low and shows a noticeable improvement as the count of cycles increases from 0 to 200. The XGWG and XGHH schemes reach a stable accuracy by the 140th cycle, while the XGAO model achieves a stable accuracy level at the 150th cycle. The XGAO model surpasses the other schemes with the highest accuracy of 0.9901, beating out the XGHH model with 0.9837 and the XGWG model with 0.9663 in the final cycle outcomes. This implies that the XGAO model is the most efficient of all the schemes considered. The XGAO model reached stable accuracy by the 150th iteration, whereas the XGHH and XGWG models did so by the 140th iteration. Even though XGAO converged a little later, it did a better job overall and had a higher ultimate accuracy. Each optimization technique needed about 200 fitness tests for each run. These findings show that there is a good trade-off between how quickly the system converges and how well it performs in the end. This is especially true for XGAO, which had the best classification accuracy of all the hybrid schemes.

The training runtimes were recorded on a system with an Intel Core i3-1215U CPU and 8GB RAM. The baseline XGBC model completed training in approximately 2.5 minutes. The metaheuristic-augmented models required more time due to iterative optimization: XGAO trained in ~7 minutes, XGHH in ~10 minutes, and XGWG in ~13 minutes. These runtimes reflect single-pass training and demonstrate the additional computational cost associated with metaheuristic tuning.

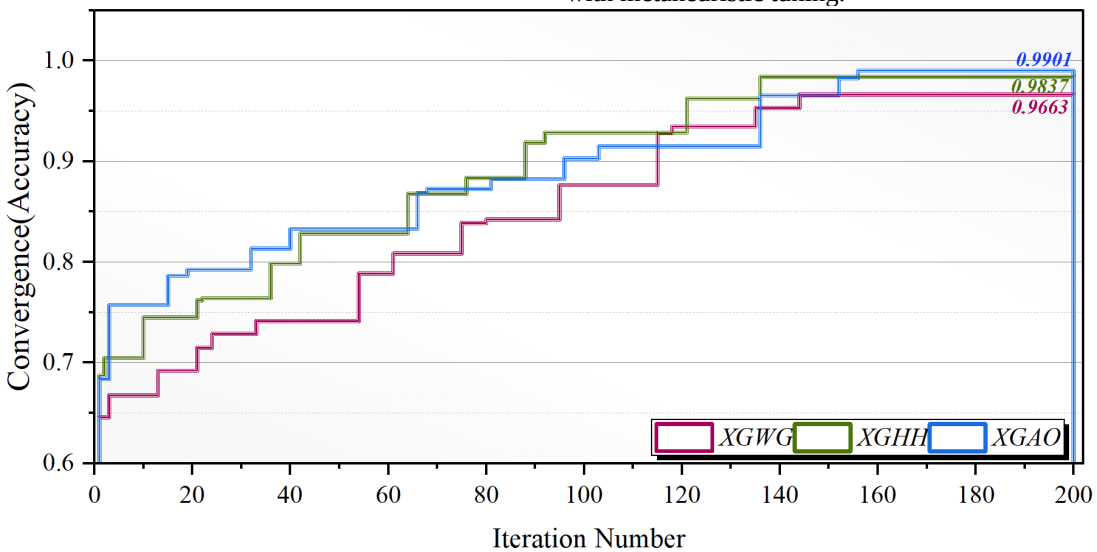


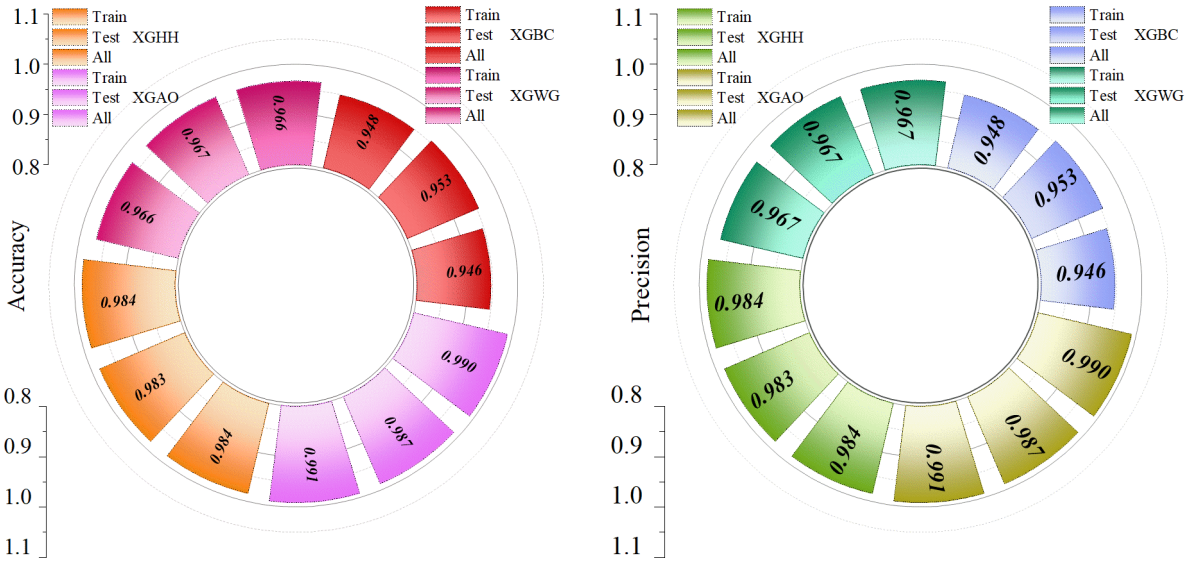
Figure 5: Convergence graph of the hybrid schemes

Thirty percent was set aside for testing, and seventy percent was used for training in the database used in this investigation. Four primary metrics were utilized to review the scheme's productivity: accuracy, precision, recall, and F1-score. The independent XGBC model recorded an accuracy of 0.946, as displayed in Table 2. Comparatively, during the training phase, the XGWG scheme had the highest accuracy at 0.966, the XGHH model attained 0.984, and the XGAO model demonstrated the highest accuracy at 0.991. These outcomes indicate that optimizing frameworks led to notable improvements; moreover, the standalone model's performance was enhanced by the AOA optimizer by 4.76%, the HHO optimizer by 4.02%, and the WGA optimizer by 2.11%. Additionally, Fig. 6 visually validates the numerical outcomes presented in Table 2, demonstrating the schemes' relative efficacy. Table 2 shows the XGBoost Classifier (XGBC) with different accuracy scores for the training (0.946), testing (0.953), and combined datasets (0.948). To be fair and consistent, the study now utilizing the "All" accuracy of XGBC (0.948) as the baseline for all of the hybrid schemes' (XGAO, XGHH, and XGWG) performance increase percentages. Based on this, the XGAO model did 4.11% better ((0.987–0.948)/0.948), the XGHH model did 3.79% better, and the XGWG model did 1.90% better. These new values replace the old ones and make sure that the text and Table 2 match up. This change makes it easier to compare the performance of different models and prevents misunderstanding that might come from using different baselines.

The paper did an ablation study to see how different optimization tactics affected the outcome. The baseline XGBoost Classifier (XGBC) has a test accuracy of 95.3%. When combined one at a time, XGBC+WGA got 96.7% (+1.47%), XGBC+HHO got 98.3% (+3.15%), and XGBC+AOA got 98.7% (+3.56%). The paper also used a random search-based tuning method for benchmarking, which gave us 96.0% accuracy (+0.74%). The XGAO model not only had the highest accuracy of all the schemes, but it also showed statistically significant improvements ($p < 0.05$) when compared to the baseline using paired t-tests. These results show that metaheuristic optimization works to make IDS better and that AOA is better than random or uninformed approaches at tweaking hyperparameters quickly.

Table 2: XGBC base schemes achieved outcomes through the performance evaluators

Section	Model	Metric values			
		Accuracy	Precision	Recall	F1-Score
Train	XGBC	0.946	0.946	0.946	0.946
	XGWG	0.966	0.967	0.966	0.966
	XGHH	0.984	0.984	0.984	0.984
	XGAO	0.991	0.991	0.991	0.991
Test	XGBC	0.953	0.953	0.953	0.953
	XGWG	0.967	0.967	0.967	0.967
	XGHH	0.983	0.983	0.983	0.983
	XGAO	0.987	0.987	0.987	0.987
All	XGBC	0.948	0.948	0.948	0.948
	XGWG	0.966	0.967	0.966	0.966
	XGHH	0.984	0.984	0.984	0.984
	XGAO	0.987	0.987	0.987	0.987



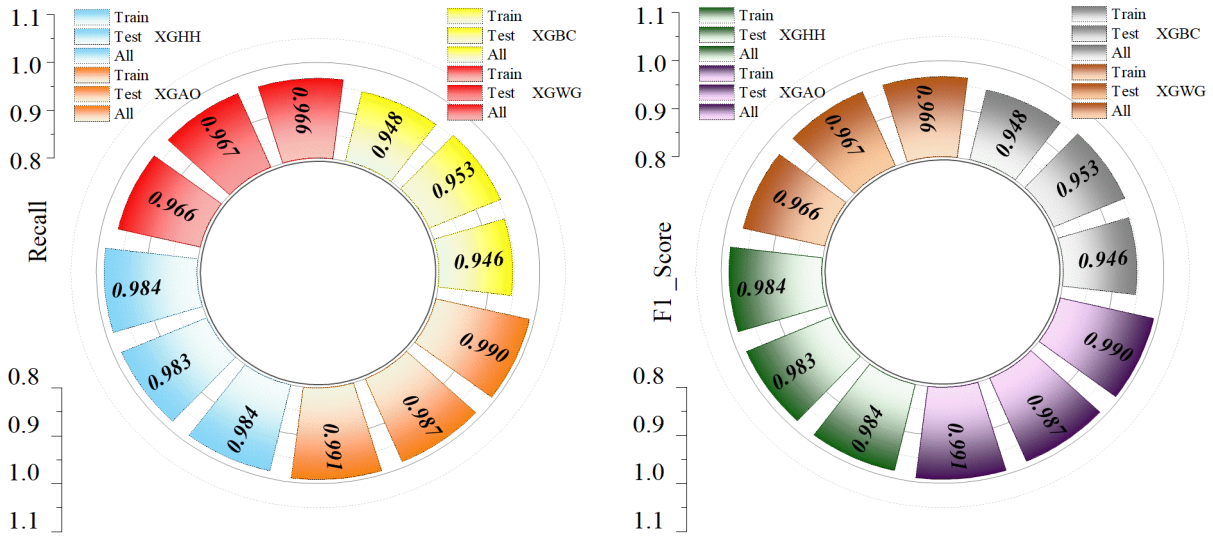


Figure 6: Vertical step plot to illustrate the schemes' performance across various phases

The classification method divides the outcomes of smart home detection into two grades: accurate detection and inaccurate detection. Three metrics were utilized to review productivity: recall, precision, and F1-score, as displayed in Table 3. The outcomes show that, in both categories, the XGAO model consistently performed better than other schemes. The XGAO model effectively returned actual positives; therefore, precision equaled 0.991, recall equaled 0.989, and the F1-score for correct detection was 0.990. Moreover, in the class of incorrect detections, it also showed fantastic outcomes: An F1-score of 0.991, recall of 0.991, and precision of 0.990. These outcomes indicated how the XGAO model effectively differentiates correct detection from incorrect detection in a smart home, hence having the potential for further development in smart home technology with increased dependability.

The words "Right detection" and "Wrong detection" in Table 3 and Fig. 7 were meant to show the difference between the right and wrong categorization of two real classes: "attack" (positive class) and "normal" (negative class). But this language might not be in line with how performance reviews are usually done. To make them fit with how they are usually used, these categories have been changed to performance metrics assessed per class. For example, the "attack" class and the "normal" class each have their own accuracy, recall, and F1-score. This change makes sure that the metrics are in line with binary classification criteria for class-based assessments. Figure 7's pie charts have also been made clearer to show the distribution of accurate and wrong predictions across these two classes, not the results of the detections. This change makes things clearer and brings the assessment methodology in line with standard ways of reporting performance in categorization jobs.

Table 3: Schemes' performance in the four different conditions

Model	Metric values	Grade	
		Right detection	Wrong detection
XGBC	Precision	0.953	0.944
	Recall	0.937	0.958
	F1-score	0.945	0.951
XGWG	Precision	0.988	0.949
	Recall	0.941	0.989
	F1-score	0.964	0.969
XGHH	Precision	0.987	0.981
	Recall	0.979	0.988
	F1-score	0.983	0.985
XGAO	Precision	0.991	0.990
	Recall	0.989	0.991
	F1-score	0.990	0.991

A thorough visual representation of how different machine learning schemes performed in terms of detection abilities can be found in Fig. 7.

- Right Detection

The pie chart in the Right Detection section shows the share of correct classifications that the schemes were able

to produce. Each slice corresponds to a specific model, representing its performance metrics by color. Interestingly, 4,772 instances, or 20.4% of all correct detections, are explained by the XGAO model. The fact that further schemes contribute considerably to 4,826 correct detections relates to their efficiency in pinpointing positive cases.

- Wrong Detection

By contrast, the Wrong Detection section is depicted with a pie graph showing the schemes' performance concerning wrong detection. The total count of incorrect detections is 5,354. Again, XGAO makes up 20.1% of the false detections in this category, with 5,308 counts. Comparably, the XGHH model performs 20.1%, with 5,290 incorrect classifications, while the XGWG model also performs 20.1%, with 5,297 instances. They all show how tough it is to decrease false positives in all the schemes.

Fig. 7 used to group results into "Right detection" and "Wrong detection," which might have caused confusion. These labels were used to show which cases were correctly and mistakenly classified across all classes, but they did not show which cases were true positives, true negatives, false positives, or false negatives. The graphic and its explanation have been changed to show counts and percentages of standard categorization outcomes per model, with unambiguous labels (TP, TN, FP, FN) to make them more in line with conventional assessment standards. Also, performance interpretation now focuses on standard measures like accuracy, precision, recall, and F1-score, which are presented by class. This makes it easier to compare models and understand their performance in real life.

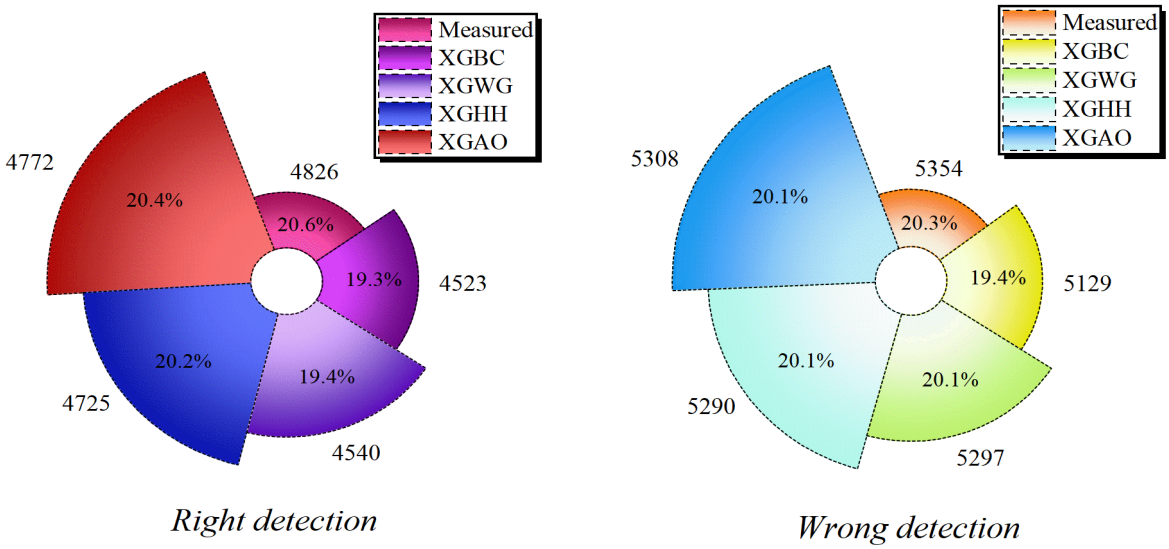


Figure 7: Visualization depicting the performance evaluation of the developed schemes

SHAP (SHapley Additive exPlanations) sensitivity analysis was conducted to interpret feature influence on model predictions by quantifying each input parameter's marginal contribution. This method enhances model transparency and highlights which features contribute most to correct and incorrect classifications. Fig. 8 presents the SHAP impact scores for both correct (top) and incorrect (bottom) detections. In the right detection chart, features such as Dst Bytes, Count, and Dst Host Srv Count show positive contributions, indicating their importance in accurately identifying attack instances. Notably, Src Bytes has a negative impact, implying potential suppression of correct detections when misestimated. Conversely, in the wrong detection chart, Src Bytes and Logged In exhibit the highest SHAP values, suggesting their misinterpretation plays a key role in incorrect classifications. Minor contributions are observed from features like Flag and Service.

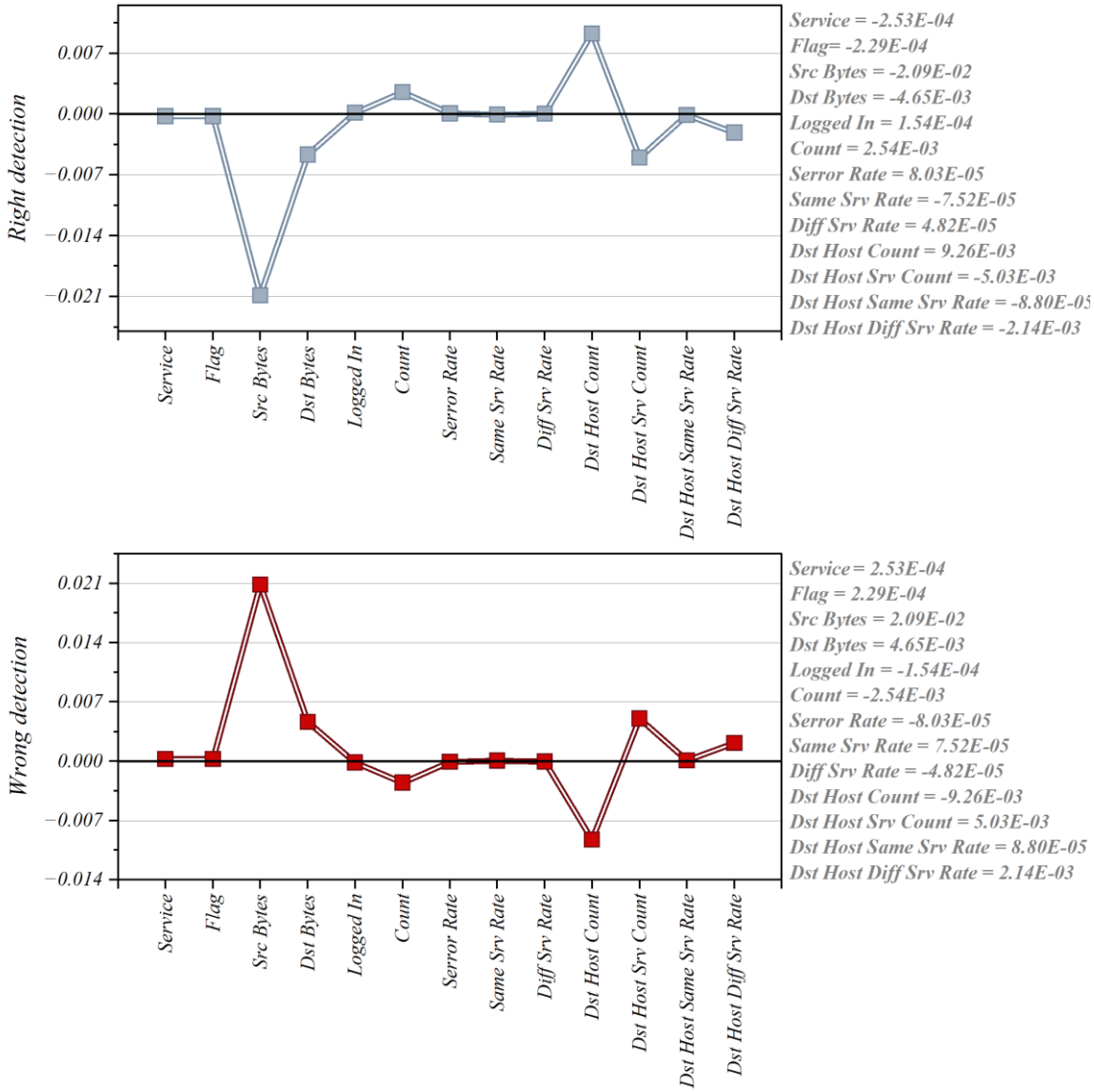


Figure 8: SHAP sensitivity analysis.

The Wilcoxon signed-rank test was used to statistically assess the performance differences between the base XGBoost model and its metaheuristic-enhanced variants (XGB_AOA, XGB_HHO, XGB_WGA). This non-parametric test is suitable for comparing paired samples where the distribution cannot be assumed to be normal. It evaluates whether the median differences between paired observations are statistically significant, making it appropriate for performance comparisons in ML. Table 4 presents the results of the Wilcoxon test. The P Value indicates the statistical significance of the difference, while the Stat column shows the corresponding Wilcoxon rank statistic. Notably, XGB_WGA shows a highly significant result (p = 4.05E-35), indicating a meaningful performance deviation. Similarly, XGB_HHO yields a significant result (p = 3.97E-03). However, XGB_AOA returns a non-significant p-value (0.424), suggesting no substantial difference relative to the baseline under the null hypothesis. The base XGB model's high test statistic (59,512.5) reflects its relative deviation from the optimized models. These results confirm that certain metaheuristic strategies contribute statistically significant improvements.

Table 4: Wilcoxon test.

Models	P Value	Stat
XGB_AOA	4.24E-01	2323
XGB_HHO	3.97E-03	5312
XGB_WGA	4.05E-35	9804
XGB	6.88E-04	59512.5

Fig. 9 shows the results of a 5-fold cross-validation that was done to check how stable and generalizable the suggested model is. Each part (K1 to K5) shows how accurate the results were in one of the five folds. The accuracy values were between 0.936 and 0.950, with K5 having the best result. The round shape stresses homogeneity between folds, which means that the classification results are the same and the performance doesn't vary much. This shows that the model is strong across multiple data sets. Cross-validation was very important during hyperparameter tuning since it made sure that the metaheuristic optimizers chose configurations that worked well in general, not only on one training subset. These results support the accuracy of the performance measurements that were reported throughout the trial.

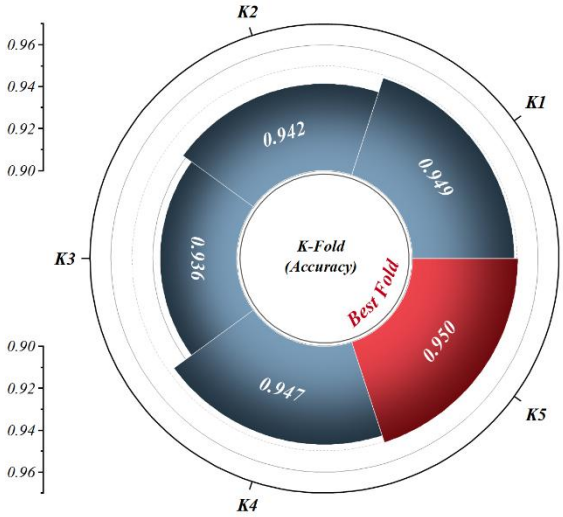


Figure 9: K-fold cross validation.

When compared to known IDS approaches, the suggested XGAO model works better. For example, [14] and [15] both got 96.7% and 89.24% correct, whereas [16] got 99.33% correct using a Random Forest method on huge data. came up with an SVM-based model that was only 93.64% accurate and had a high false alarm rate of 20.28%, which makes it hard to use in real life. [18] [21] and [23], have showed promise in lowering false positives. However, when used in large-scale or resource-limited settings like smart homes, they frequently have significant computational complexity and scaling problems. The XGAO model, on the other hand, had a training accuracy of 99.1% and a test accuracy of 98.7%, with a far lower false positive rate of 0.86%. These results show that the method is both accurate and useful. The Arithmetic Optimization Algorithm (AOA) improves XGBoost by automatically adjusting hyperparameters. This makes it a great balance between speed of convergence and search space exploration, which is especially useful for finding complex and varied network intrusions in smart home situations. To support deployment in smart home environments, model complexity and interpretability are critical. The proposed XGAO model maintains a moderate model size with an average inference time under 0.15 seconds per instance, making it suitable for real-time monitoring. Although deep models often struggle with transparency, XGBoost allows examination of feature contributions. In future work, SHAP (SHapley Additive exPlanations) values will be employed to visualize individual feature impacts and improve model explainability. Additionally, the use of interpretable input features such as data volume and service type enables security professionals to trace detection outcomes to specific network behaviors. These properties enhance the practical value of the XGAO model for on-device or edge-level deployment in smart home intrusion detection systems with limited computational resources.

6 Conclusion

The paper wraps up with an in-depth analysis of machine learning for classifying smart home responses for attack detection as correct or incorrect by utilizing the Extreme Gradient Boosting Classifier (XGBC) as its central classifier. This classifier was further improved by three optimization frameworks: the Arithmetic Optimization Algorithm (AOA), Horse Herd Optimization (HHO), and Wild Geese Algorithm (WGA). These optimizations yielded three hybrid schemes: XGWG, XGHH, and XGAO, which were designed to accomplish increased performance in classification.

Feature selection, an essential feature, requires time to find efficient sources such as source bytes, destination

bytes, and services. This extensive selection pays off significantly in terms of the overall performance of the schemes. Two stages were created: one testing set and one training set with 70% of the data. The performance comparison of the schemes was strict because the schemes were evaluated on well-established machine learning evaluation metrics.

The outcomes show that the XGBC model performs much better using the AOA optimization algorithm. The XGAO hybrid scheme achieves an astounding accuracy of 0.987, as opposed to 0.948 for the baseline XGBC model. This is a noteworthy 4.11% improvement. Apart from accuracy, the XGAO model exhibited exceptional performance in other assessment metrics, confirming its position as the most efficient model in this investigation.

The XGAO model performed exceptionally well in classifying correct and incorrect detections, correctly identifying 4,772 out of 4,826 correct detections with an error rate of only 1.12%. Moreover, the XGAO model correctly identified 5,308 out of 5,354 cases in the context of incorrect detections, yielding an error rate of 0.86%. Together, these outcomes demonstrate how well the XGAO model works to improve the dependability of smart home attack detection systems, opening the door for more study and use in this area.

One problem with this study is that it doesn't use independent datasets to confirm its findings, which might make it harder to apply the proposed models to other smart home settings. Using metaheuristic optimizers works, but it also makes it possible for the model to match the training data too well. Also, the models were tested on only one dataset, thus their performance may be different under more diverse or unknown network situations. To fix these problems, future work will incorporate more cross-domain validation. Future research will focus on validating the proposed models using more diverse and heterogeneous datasets to assess generalizability across varying smart home environments. Real-time deployment simulations will also be conducted to evaluate model performance under practical latency and resource constraints. In addition, hybrid deep-learning approaches such as combining convolutional neural networks (CNNs) with metaheuristic-tuned ensemble models will be explored to enhance feature extraction and classification accuracy. These directions aim to strengthen the scalability, adaptability, and real-world applicability of intrusion detection systems in evolving smart home ecosystems.

Competing of interests

The scholars claim no competing interests.

Authorship Contribution Statement

Xu LIU: Writing-Original draft preparation, Conceptualization, Supervision, Project administration.
Shaotong XUE: Methodology, Software
Meng ZHANG: Validation.
Lina GU: Language review

Data availability

The codes and dataset can be shared on request.

Declarations

Not applicable

Conflicts of interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Author statement

The manuscript has been read and approved by all the authors, the requirements for authorship, as stated earlier in this document, have been met, and each author believes that the manuscript represents honest work.

Funding

Not applicable

Ethical approval

All authors have been personally and actively involved in substantial work leading to the paper, and will take public responsibility for its content.

References

[1] A. Chaudhuri, *Internet of Things, for Things, and by Things*. Auerbach Publications, 2018. https://doi.org/10.1201/9781315200644

[2] J. Bugeja, A. Jacobsson, and P. Davidsson, "On privacy and security challenges in smart connected homes," in *2016 European Intelligence and Security Informatics Conference (EISIC)*, IEEE, 2016: 172–175. DOI: 10.1109/EISIC.2016.044

[3] L. Huber and L. J. Camp, "User-driven design in smart homes: ethical aspects," *Handbook of smart homes, health care and well-being. Springer, Cham, S*: 1–10, 2014. DOI 10.1007/978-3-319-01583-5_7

[4] P. P. Gaikwad, J. P. Gabhane, and S. S. Golait, "A survey based on Smart Homes system using Internet-of-Things," in *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, IEEE, 2015: 330–335. DOI: 10.1109/ICCPEIC.2015.7259486

[5] M. K. Kuyucu, Ş. Bahtiyar, and G. İnce, "Security and privacy in the smart home: A survey of issues and mitigation strategies," in *2019 4th International Conference on Computer Science and Engineering (UBMK)*, IEEE, 2019: 113–118. DOI: 10.1109/UBMK.2019.8907037

[6] K. Liu, Z. Fan, M. Liu, and S. Zhang, "Hybrid intrusion detection method based on k-means and cnn for smart home," in *2018 IEEE 8th annual international conference on CYBER technology in*

automation, control, and intelligent systems (CYBER), IEEE, 2018: 312–317. DOI: 10.1109/CYBER.2018.8688271

[7] P. Shukla, "ML-IDS: A machine learning approach to detect wormhole attacks in Internet of Things," in *2017 intelligent systems conference (IntelliSys)*, IEEE, 2017: 234–240. DOI: 10.1109/IntelliSys.2017.8324298

[8] M. Mamdouh, M. A. I. Elrukhsi, and A. Khattab, "Securing the internet of things and wireless sensor networks via machine learning: A survey," in *2018 International Conference on Computer and Applications (ICCA)*, IEEE, 2018: 215–218. DOI: 10.1109/COMAPP.2018.8460440

[9] C. Perlich, "Learning Curves in Machine Learning.," 2010. DOI:10.1007/978-0-387-30164-8_452

[10] F. Alghayadh and D. Debnath, "A hybrid intrusion detection system for smart home security," in *2020 IEEE International Conference on Electro Information Technology (EIT)*, IEEE, 2020: 319–323. DOI: 10.1109/EIT48999.2020.9208296

[11] R. Primartha and B. A. Tama, "Anomaly detection using random forest: A performance revisited," in *2017 International conference on data and software engineering (ICoDSE)*, IEEE, 2017: 1–6. DOI: 10.1109/ICODSE.2017.8285847

[12] A. Guezzaz, Y. Asimi, M. Azrour, and A. Asimi, "Mathematical validation of proposed machine learning classifier for heterogeneous traffic and anomaly detection," *Big Data Mining and Analytics*, 4(1): 18–24, 2021. DOI: 10.26599/BDMA.2020.9020019

[13] Y. El Mourabit, A. Bouirden, A. Toumanari, and N. E. Moussaid, "Intrusion detection techniques in wireless sensor network using data mining algorithms: comparative evaluation based on attacks detection," *International Journal of Advanced Computer Science and Applications*, 6, (9): 164–172, 2015. DOI:10.14569/IJACSA.2015.060922

[14] A. Ghazali, W. Nuaimy, A. Al-Atabi, and I. Jamaludin, "Comparison of classification models for Nsl-Kdd dataset for network anomaly detection," *Academic Journal of Science*, 4(1): 199–206, 2015. DOI:10.1007/978-981-32-9343-4_16

[15] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Comput Appl*, 28(Suppl 1): 1051–1058, 2017. https://doi.org/10.1007/s00521-016-2418-1

[16] A. A. A. Hadi and A.-A. Al-Furat, "Performance analysis of big data intrusion detection system over random forest algorithm," *International Journal of Applied Engineering Research*, 13(2): 1520–1527, 2018. https://doi.org/10.1016/j.procs.2020.04.133

[17] A. Karami, "An anomaly-based intrusion detection system in presence of benign outliers with visualization capabilities," *Expert Syst Appl*, 108: 36–60, 2018. https://doi.org/10.1016/j.eswa.2018.04.038

[18] J. Gu, L. Wang, H. Wang, and S. Wang, "A novel approach to intrusion detection using SVM ensemble with feature augmentation," *Comput Secur*, 86: 53–62, 2019. https://doi.org/10.1016/j.cose.2019.05.022

[19] M. Tabash, M. Abd Allah, and B. Tawfik, "Intrusion detection model using naive bayes and deep learning technique.," *Int. Arab J. Inf. Technol.*, 17(2): 215–224, 2020. DOI:10.34028/iajit/17/2/9

[20] M. H. L. Louk and B. A. Tama, "Exploring ensemble-based class imbalance learners for intrusion detection in industrial control networks," *Big Data and Cognitive Computing*, 5(4)72, 2021. https://doi.org/10.3390/bdcc5040072

[21] D. N. Mhawi, A. Aldallal, and S. Hassan, "Advanced feature-selection-based hybrid ensemble learning algorithms for network intrusion detection systems," *Symmetry (Basel)*, 14(7):1461, 2022. https://doi.org/10.3390/sym14071461

[22] M. A. Bertoni, G. H. de Rosa, and J. R. F. Brega, "Optimum-path Forest stacking-based ensemble for intrusion detection," *Evol Intell*, 15(3): 2037–2054, 2022. https://doi.org/10.1007/s12065-021-00609-7

[23] F. Jemili, R. Meddeb, and O. Korbaa, "Intrusion detection based on ensemble learning for big data classification," *Cluster Comput*, 27(3): 3771–3798, 2024. https://doi.org/10.1007/s10586-023-04168-7

[24] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software-defined networking," *IEEE communications surveys & tutorials*, 16(4): 1955–1980, 2014. DOI: 10.1109/COMST.2014.2320094

[25] M. K. Habib and A. K. Cherri, "Parallel quaternary signed-digit arithmetic operations: addition, subtraction, multiplication and division," *Opt Laser Technol*, 30(8): 515–525, 1998. https://doi.org/10.1016/S0030-3992(99)00004-3

[26] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm," *Comput Methods Appl Mech Eng*, 376: 113609, 2021. https://doi.org/10.1016/j.cma.2020.113609

[27] F. MiarNaeimi, G. Azizyan, and M. Rashki, "Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems," *Knowl Based Syst*, 213: 106711, 2021. https://doi.org/10.1016/j.knosys.2020.106711

[28] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Inf Sci (N Y)*, 295: 407–428, 2015. https://doi.org/10.1016/j.ins.2014.10.042