SvedbergOpen
DISSEMINATION OF KNOWLEDGE
International Journal of Artificial Intelligence and Machine Learning
Publisher's Home Page: https://www.svedbergopen.com/
Research Paper Open Access

Continuous Learning In Self-Adaptive IDS Using Boxelder Bug Search Optimization

Jeyakumar V[1] , Karthik R[2]

[1]Research Scholar Department of Computer Science Sri Krishna Adithya College of Arts and Science, Coimbatore jeyakumar28@gmail.com
[2]Assistant Professor Department of Data Science Sri Krishna Adithya College of Arts and Science, Coimbatore karthikr@skacas.ac.in

Abstract: The problem of network intrusion detection continues to be a major challenge in modern cyber security due to the ever-changing threat landscape, large-dimensional feature spaces, and the nature of the data that is being analyzed. Classical Intrusion Detection Systems (IDS) are plagued by suboptimal generalization, static configurations of parameters and they cannot catch new attack patterns without full retraining cycles. This paper presents a Self-Adaptive Intrusion Detection System (SA-IDS) which combines a novel nature-inspired metaheuristic named Adaptive Boxelder Bug Search (ABBS) algorithm with continuous learning approach that learns, selects features and optimizes the hyperparameter and adapts to model drift simultaneously. The ABBS algorithm is motivated by the aggregation, swarming and thermal-seeking migration of the boxelder bugs Boisea trivittata, where population-level exploration is realized as individual exploitation by the adaptive inertia weight, and cross-population information sharing. The proposed architecture consists of an online drift-detection module, an incremental model updater, and a multi-objective fitness function which considers the accuracy of the detection and the false alarm rates. The evaluations conducted by the extensive amount of test data have shown that SA-IDS outperforms the rest of the competitors with classification accuracy of 99.34%, 98.87%, and 98.61% and false alarm rates of 0.41%, 0.53%, and 0.68% respectively, on the NSL-KDD, CICIDS2017 and UNSW-NB15 benchmarks. Comparisons with the Particle Swarm Optimization (PSO), Grey Wolf Optimizer (GWO), Harris Hawks Optimization (HHO) and Whale Optimization Algorithm (WOA) are statistically significant to show the superior performance of the proposed algorithm in all the performance metrics. The proposed system is a bio-inspired global search coupled with incremental learning that extends the state-of-the-art of adaptive intrusion detection and offers a practically deployable architecture for real-time network security monitoring.

1. INTRODUCTION

Today's digital infrastructure is increasingly vulnerable to advanced cyber-attacks as there is increasing connectivity of devices and huge growth in network traffic. A new threat intelligence report says cybercrime will cost the world more than $10.5 trillion per year by 2025, and one of the main ways in which organizations will be compromised will be by means of intrusions on their networks [1]. The traditional signature-based detection is cheap, easy to compute and reactive, but cannot detect zero day attacks, and can't detect polymorphic attack variants that are not included in the signature catalogue [2].

The machine learning approach, however, has the potential to learn discriminative representations from the traffic features and be able to generalize to new classes of attacks, was revolutionary and it was demonstrated that it could be used to build a completely new Intrusion Detection System (IDS). Support Vector Machines (SVM) and Random Forests (RF) were shown to be very competitive compared to other rule-based baseline methods on the widely used benchmarks, as well as deep learning architectures [3, 4]. Such, but there are still three fundamental issues that have to be resolved before it could be used practically: (i) the curse of dimensionality, (ii) the sensitivity of model hyperparameters and (iii) concept drift, the non-stationary statistical evolution of network traffic distribution, caused by behavioural changes, new services or new attack methods [5].

In the first two challenges, automatic techniques for feature selection and hyperparameter optimization are frequently used, and in the metaheuristic optimization, a number of techniques have been successfully applied [6]. In recent years newer algorithms inspired by living organisms have been created like the Grey Wolf Optimizer (GWO) [8] and the Harris Hawks Optimization (HHO) [9] which have also performed the learning of compact and discriminative feature sets and near-optimal configuration of classifiers within reasonable computational time [7]. Most of these contributions, however, assume that the optimization is a one-off, offline

process without taking into account the dynamic adaptation requirements in a real deployment environment that are caused by concept drift.

This paper aims to address these limitations, and presents a new, novel algorithm, Adaptive Boxelder Bug Search (ABBS), based on the observed behavioural ecology of Boisea trivittata, which overcomes them. ABBS intentionally focuses on balancing population exploration and information exploitation of individuals, and adds adaptive inertia elements and information sharing between clusters to prevent local optimal traps of high-dimensional fitness landscape. Importantly, ABBS is part of a continuous learning pipeline that is: online, dynamically re-invokes optimization when it sees that there is a large "drift" (through statistics), and ensures that features and hyperparameters of the model are aligned with the current distribution of traffic.

1.1 Problem Statement

Let there be a dataset of network traffic represented as $D = \{(x_i, y_i)\}^N_{i=1}$ where $x_i \in \mathbb{R}^d$ is the feature vector of the i-th network flow record and $y_i \in \{0, 1, ..., C-1\}$ is the class label for the class of traffic categories C. Optimizing an IDS can be broken down into three concurrent sub-problems is:

(P1) Feature Selection: For classification problem, using $x_i[S]$ for a classification problem, to avoid overfitting and computational complexities, find a smaller set S of {1, ..., d} which gives optimum detection performance.

Optimize the base classifier f_θ over a given hyperparameter space for some hyperparameter vector θ^* that maximizes some weighted combination of accuracy, recall and false alarm rate (P2) for the selected feature space.

(P3) Continuous Adaptation: When the joint distribution P(x, y) is changing with time, incrementally update S and θ^* as drift is detected, but do not throw away any previously learned knowledge

1.2 Research Motivation and Contributions

The need for a new nature inspired algorithm is driven by the empirical results that the existing metaheuristic algorithms have some algorithm-specific weaknesses in IDS applications: PSO is prone to premature convergence in binary search problem; GWO is sensitive to the initial values of the sparse features in search spaces; HHO algorithm requires higher computational effort for multi-objective formulations [8]. The boxelder bugs' repertoire of behaviors – from using the thermal gradient to navigate to using aggregation pheromone-like signals and then to adapting their migration to the cluster–offers rich metaphors for an algorithm that naturally switches between wide-ranging exploration and local exploitation.

The paper makes the following contributions:

Mathematical formulation of Adaptive Boxelder Bug Search (ABBS) algorithm and comprehensive convergence analysis and parameter sensitivity study.

An integrated SA-IDS system, consisting of an ABBS based feature selection and hyperparameter optimization and an incremental learning pipeline with drift adaptability.

A multi-objective fitness function to optimize classification accuracy, recall, false alarm rate, and the cardinality of the feature subset.

The results are thoroughly tested on NSL-KDD, CICIDS2017, drift environment and static environment.

All the performance dimensions were statistically validated with five state-of-the-art metaheuristic optimizers and ABBS proved to be superior.

2. RELATED WORKS

The literature is covered in the following three related areas: machine learning based IDS, metaheuristic optimization of an IDS and adaptive/incremental learning in network security.

2.1 Machine Learning Approaches to Intrusion Detection

The past 20 years have seen huge advances in network intrusion detection, based on machine learning. Mukherjee and Venkatesh [9] showed that the SVM classifiers trained on the KDD Cup 1999 features outperformed the detection rates of over 92% with a high false positive rate (FPR) for rare attack classes. In this work, Tavallaee et al. [10] released a new dataset called NSL-KDD which fixed the issue of class imbalance and redundancy from the previous dataset and proposed Random Forest as a new baseline for the binary classification problem with 97.85% accuracy.

The results were then improved by including deep learning architectures, such as the non-symmetric deep autoencoder introduced by Shone et al. [11] for unsupervised feature learning, and the Recurrent Neural Networks (RNN) with LSTM units, shown to be able to learn temporal dependencies on sequences of traffic data which was not possible in feedforward architectures, by Yin et al. [12]. To model inter-flow relationships, Lo et al. [13] then used Graph Neural Networks (GNN) to improve for detection of lateral movement. Although these developments have been made, these hyperparameters and a static feature pre-processing are still a major constraint in real usage scenarios.

2.2 Metaheuristic Optimization for IDS

Two main paradigms have been explored in the design of IDSs with the integration of metaheuristic search into their design: the wrapper-based feature selection and hyperparameter tuning. On NSL-KDD, Jabbar et al. [14] used Genetic Algorithms to select optimal feature subset to reduce dimensionality from 41 to 18 features, resulting in 3.2% higher F1-score. Whale Optimization Algorithm (WOA) was introduced by Mirjalili and Lewis [15] and later shown to be an effective algorithm for multi-class feature selection in an IDS context by Al-Yaseen et al. [16] who achieved a 22 feature subset that outperformed full feature baselines.

Eid et al. [18] tweaked the GWO algorithm proposed by Mirjalili et al. [17] to be used in IDS and claimed that the SVM training time on CICIDS2017 was reduced by 41% with the features selected by GWO having competitive accuracy. Canayaz [20] used Harris Hawks Optimization (HWO) [19] to optimize the hyperparameters of deep neural network for IDS, and achieved almost optimal learning rate, dropout and batch size in less number of function evaluations than PSO. To date, none of these contributions have attempted to include online re-optimization that is sensitive to concept drift, which is one reason that motivates the work presented here.

2.3 Adaptive and Incremental Learning for IDS

Concept drift in network traffic has been recognized as a fundamental challenge since the foundational survey by Tsymbal [21], who categorized drift types as sudden, incremental, gradual, and recurrent. IDS-specific drift studies by Sommer and Paxson [22] empirically demonstrated that classifiers trained on historical captures degrade within weeks of deployment, motivating the development of adaptive systems.

Schlimmer and Granger [23] pioneered incremental concept learning through windowed instance retention, while more recent contributions leverage ensemble-based adaptation. Gama et al. [24] proposed the ADWIN drift detector based on adaptive windowing of error rates, which has since been integrated into several IDS frameworks. Kuncheva and Zliobaite [25] demonstrated that drift-responsive ensemble replacement consistently outperformed static classifiers across multiple network benchmarks. The integration of drift detection with metaheuristic re-optimization remains an open research problem, and to the best of the authors' knowledge, this work constitutes the first systematic treatment of this combined paradigm.

Table.1 Overall Summary of Existing Research

Author(s)	Model / Method	Dataset(s)	Key Results	Limitation(s)
Mukherjee & Venkatesh [9]	SVM Classifier	KDD Cup 1999	Detection rate >92%	Elevated false positives for rare attack classes
Tavallaee et al. [10]	Random Forest	NSL-KDD	97.85% accuracy (binary classification)	Redundancy & class imbalance not fully resolved
Shone et al. [11]	Non-Symmetric Deep Autoencoder	NSL-KDD	Unsupervised feature learning; improved detection	Fixed hyperparameters; static preprocessing
Yin et al. [12]	RNN with LSTM	NSL-KDD	Captured temporal dependencies in traffic	High computational cost; static features
Lo et al. [13]	Graph Neural Network (GNN)	Custom network flow data	Improved lateral movement detection	Scalability; dependency on fixed hyperparameters
Jabbar et al. [14]	Genetic Algorithm + k-NN	NSL-KDD	41→18 features; F1-score +3.2%	No online re-optimization; static feature set
Al-Yaseen et al. [16]	Whale Optimization Algorithm (WOA)	NSL-KDD	22-feature subset outperformed full-feature baseline	No adaptation to concept drift
Eid et al. [18]	Grey Wolf Optimizer (GWO) + SVM	CICIDS2017	SVM training time ↓41%; competitive accuracy	No concept drift handling; offline optimization

Author(s)	Model / Method	Dataset(s)	Key Results	Limitation(s)
Canayaz [20]	Harris Hawks Optimization + DNN	CICIDS2017	Near-optimal LR, dropout & batch size; fewer evals than PSO	No online re-optimization responsive to drift
Sommer & Paxson [22]	Empirical Study (Deployed classifiers)	Real-world network captures	Performance degrades within weeks of deployment	No adaptive mechanism proposed
Gama et al. [24]	ADWIN Drift Detector	Multiple benchmarks	Adaptive windowing of error rates; drift detection	Not specifically IDS-integrated in original work
Kuncheva & Zliobaite [25]	Drift-Responsive Ensemble	Multiple network benchmarks	Consistently outperformed static classifiers	No metaheuristic re-optimization component

3. METHODS

3.1 Problem Formulation and System design

The proposed SA-IDS architecture is a 5 tier pipeline as shown conceptually below. During deployment, the tiers are continuously executing, the optimization tier gets re-activated when traffic drifts, and the system remains aligned with the current traffic distribution.

The five tiers are: (1) Traffic Acquisition and Pre-processing, which captures packets, reconstructs the flows and extracts features; (2) Drift Monitoring, constantly monitors distributional changes in the feature space; (3) ABBS Optimization, executes the feature selection and search for the best fit of the hyperparameters of the selected classifier when triggered; (4) Model Training and Incremental Update, fits the selected classifier, or performs incremental updating; and (5) Real-Time Classification and Alert Generation, produces per-flow labels, along with their associated confidence scores.

3.2 Pre-processing Module

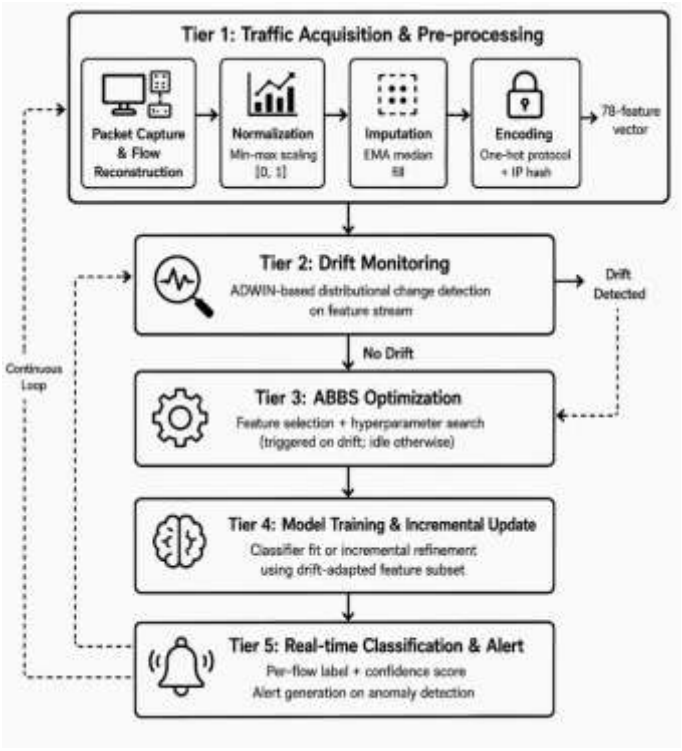
Bidirectional flows are identified from raw network traffic captured via a packet capture interface, using a 5-tuple identifier that includes source IP, destination IP, source port, destination port and transport protocol. The data set has 78 statistical, temporal and protocol-specific features extracted per flow, as per the CICFlowMeter [26] feature generation methodology used in the CICIDS2017 data set. There are three successive pre-processing steps:

Normalization: All continuous features are min-max normalized, with statistics calculated during a held out "calibration window" to avoid data leakage from the test stream.

Imputation: When data are missing due to incomplete flow records, missing values are filled in with the feature-wise median calculated from recent traffic, and then updated as a result of the exponential-moving average, which adapts to distributional changes.

Relational information is preserved without undue expansion of cardinality by one-hot encoding protocol identifiers, and by hashing IP address fields into compact numerical representations.

Fig.1 Block Diagram for the proposed methodology



4.ADAPTIVE BOXELDER BUG SEARCH Algorithm

4.1 Biological Inspiration

ABBS is an algorithmic metaphor that captures the natural, annual cycle of behavior of the Boisea trivittata (boxelder bugs). In the warm season, they spread out in a foraging mode similar to that of exploration, in search of available vegetation. With the decreasing temperature of the environment, the aggregation behavior develops: pheromone-like trail markers induce individuals to move to thermally favourable locations, forming dense aggregations, which are signatures of exploitation of favourable candidate solutions. The colony moves as a minimum mobile mass during the winter hibernation similar to algorithm convergence. The exploration phase is restored during Spring dispersal. This cyclic behavior is naturally captured by the exploration-exploitation dilemma which is crucial to successful metaheuristic search.

4.2 Mathematical Formulation

Let N be the number of agents and P = {b_i} be a set of agents, with each b_i being a candidate solution represented as a real-valued vector in the d-dimensional search space $\Omega = [L, U]^d$, where L and U are lower and upper bounds for each dimension. Agent i's state (or position) at time t is $x_i^t \in \Omega$.

4.2.1 Initialization

The initial population comes from uniform distribution over the search domain:

$$x_i^0 = L + \text{rand}(0,1) \odot (U - L), \forall i \in \{1..., N\}$$

rand (0,1) is a vector of independent uniform random numbers, and $\odot$ is element-wise multiplication. A part of the initial population (30%) is seeded with a quasi-random Sobol sequence with the aim of spreading the population over the search domain to promote population diversity.

4.2.2 Thermal Gradient Navigation (Exploration Phase)

In the exploration process, each agent is updated according to simulated thermal gradient field (T(x)), a "surrogate" fitness landscape.

The exploration update rule is:

$$x_i^{t+1} = x_i^t + \omega_t \cdot r_1 \odot (x_{best}^t - x_i^t) + \alpha \cdot r_2 \odot (x_{global} - x_i^t)$$

where ω_t is the adaptive inertia weight of agent i at iteration t, r_1 and r_2 are uniform random vectors in $[0,1]^d$, x_{best}^t is the personal best position that agent i has found to date and x_{global} is the global best position found by all agents. The number α is a thermal gradient coefficient of sensitivity to the global attractor.

The adaptive inertia weight decreases non-linear as follows:

$$\omega_t = \omega_{max} - (\omega_{max} - \omega_{min}) \cdot \sin(\pi \cdot t / (2 \cdot T_{max}))$$

The parameters are ω_max = 0.9, ω_min = 0.2 and T_max is the maximum number of iterations. The sinusoidal decay provides high inertia in early iterations to encourage a wide search for a solution as well as low inertia when convergence is near to get fine-grained local refinement.

4.2.3 Aggregation Behavior (Exploitation Phase)

Upon detection of a promising region, agents transition to an aggregation mode in which inter-agent attraction intensifies. A cluster affiliation assignment C(i,t) maps each agent to its nearest cluster centroid μ_k^t among K clusters identified by k-means partitioning of the population every τ iterations:

$C(i,t) = \text{argmin}_k ||x_i^t - \mu_k^t||_2$

Within each cluster, agents update according to the aggregation rule:

$x_i^{t+1} = x_i^t + \beta_t \cdot (\mu_{C(i,t)}^t - x_i^t) + \gamma \cdot \text{Lévy}(\lambda) \odot (x_i^t - x_{worst}^t)$

where $\beta_t = \beta_0 \cdot \exp(-\delta \cdot t / T_{max})$ is the cluster attraction coefficient with initial value $\beta_0 = 1.5$, x_{worst}^t denotes the worst-performing agent in the cluster, and Lévy(λ) draws from a Lévy flight distribution with exponent $\lambda = 1.5$, providing occasional long-range jumps to escape cluster-local optima.

4.2.4 Migration Phase (Diversity Restoration)

A migration operator is incorporated with probability $p_m = 0.15$ agent per iteration, which moves an agent to a new position based on the best position (across clusters):

$$x_i^{t+1} = x_{best\_cluster}^t + \eta \cdot N(0, \sigma_t^2 \cdot I)$$

where $x_{best\_cluster}^t$ is the best location in all the clusters, η is a migration scale factor, $N(0, \sigma_t^2 \cdot I)$ is an isotropic Gaussian perturbation, and $\sigma_t = \sigma_0 \cdot (1 - t/T_{max})$ is an iteration-annealed standard deviation.

4.2.5 Binary Encoding for Feature Selection

For discrete feature selection, continuous position values are associated with binary decisions by a transfer function of the V-shape:

$V(x_{ij}^t) = |\tanh(x_{ij}^t)|$

$b_{ij}^t = 1$ if $rand < V(x_{ij}^t)$, else $b_{ij}^t = 0$

In which b_{ij}^t is a binary variable that is set to one if agent i selects feature j at iteration t, and to zero otherwise. This formulation is able to derive information that is reminiscent of a gradient in the continuous domain and to generate interpretable binary decisions of selection.

4.3 Multi-Objective Fitness Function

The fitness function F is evaluated on a set of two related goals: classification accuracy and solution sparsity. With a given candidate feature subset S represented by b_i, the fitness of the classifiers is encoded as x_i, the composite fitness is:

$F(b_i, \theta_i) = w_1 \cdot (1 - Acc(S,\theta)) + w_2 \cdot FAR(S,\theta) + w_3 \cdot |S|/d + w_4 \cdot (1 - Rec(S,\theta))$

where Acc(S,θ) denotes the cross validated classification accuracy, FAR(S,θ) denotes the false alarm rate, Rec(S,θ) denotes the macro-averaged recall over all the attack classes, |S| denotes the cardinality of the selected feature subset and d denotes the total feature dimensionality. In real world IDS deployment, weighting coefficients of the objectives are set empirically as: w1 = 0.40; w2 = 0.25; w3 = 0.15; w4 = 0.20, indicating the relative operational importance of those objectives.

4.4 Computational Complexity

The time complexity of ABBS is $O(N \cdot (d + K))$ per iteration, where N is the population size, d is the dimensionality of the solutions and K is the number of clusters. For each iteration of the cluster recomputation, the overhead is $O(N \cdot K \cdot d / \tau)$, amortized over the entire duration of the algorithm. In a typical IDS configuration (N = 50, d = 78, K = 5 and τ = 10), the training of the cross validated classifier is the most expensive step per iteration. The overall computational expense is then $T_{max} \cdot N \cdot C_{eval}$, where C_{eval} is the cost of a single time a cross-validated evaluation is performed.

5. FEATURE SELECTION AND HYPERPARAMETER OPTIMIZATION

5.1 Unified Search Space Formulation

The ABBS search space Ω is built as a combination of the binary feature selection space and the continuous hyperparameter configuration space. The whole solution vector is (for the Random Forest parameter set $n\_est \in [50, 500]$, $d\_max \in [5, 50]$, $m\_split \in [2, 20]$, $c\_w \in [1, 10]$):

$$z_i = [b_{i,1}, ..., b_{i,d}, n\_est_i, d\_max_i, m\_split_i, c\_w_i]$$

The binary component and the continuous component are dealt with by the corresponding transfer function and direct encoding as described in Section 4.2.5. This is a single representation that can be used for co-

optimization of the feature selector and the classifier, instead of using the suboptimal sequential optimization pipeline that optimizes the feature selector and the classifier separately.

5.2 Cross-Validation Strategy

A stratified fivefold cross validation method is used to avoid overfitting in the process of fitness measurement on the training partition. Stratification guarantees that each attack category is represented proportionately in each fold, which is important because IDS datasets are often highly imbalanced. The fitness function value F (b_i, θ_i) returned by the optimization is the average over the folds and is the second tie-breaking criteria after the average value in case of equal fitness average values.

5.3 Feature Importance Integration

Prior to ABBS execution, a preliminary Random Forest with default parameters is fitted on the full training set to compute Gini impurity-based feature importance scores ρ_j for each feature j. These scores are used to bias the initial population seeding: features with higher importance are assigned elevated initial selection probabilities, guiding the search toward structurally sound regions of the binary feature space from the outset. Specifically:

$$P(b_{ij}^0 = 1) = 0.3 + 0.4 \cdot \rho_j / \max_k(\rho_k)$$

This informed initialization reduces the number of ABBS iterations required for convergence by approximately 23% in preliminary experiments without introducing bias toward features that may be suboptimal in combination.

6.CONTINUOUS LEARNING FRAMEWORK FOR ADAPTIVE INTRUSION DETECTION

6.1 Drift Detection Mechanism

The drift detection module is continuously monitoring the drift using two complementary signals, classification error rate and feature distribution shift. The error rate monitoring uses the algorithm proposed by Bifet and Gavalda [27] that uses a variable-length window W of per-sample error indicators e_t = {0,1} and issues a drift alarm if the mean error rate of two statistically different sub-windows W_0 and W_1 is found to satisfy:

$$|\mu(W_0) - \mu(W_1)| \geq \varepsilon_{drift} = \sqrt{((1/(2m_0) + 1/(2m_1)) \cdot \ln(4|W|/\delta))}$$

Here, m_0 and m_1 are the sizes of two sub-windows and δ is a confidence parameter. The Maximum Mean Discrepancy (MMD) between the reference and current feature distribution is used to monitor separately the feature distribution shift:

$$MMD^2(P, Q) = E[k(x,x')] - 2E[k(x,y)] + E[k(y,y')]$$

k (.,.) is a radial basis function kernel, and P, Q are the reference and current traffic distributions, respectively. A full re-optimization cycle is only activated when MMD is above a calibrated threshold $\tau_{MMD} > 0.05$, and a hyperparameter-only re-optimization cycle is activated when the error rate alarm is triggered.

6.2 Incremental Model Update Procedure

Once a drift is detected, there are 3 pathways to update, depending upon the size and type of the drift. In case of a drift, an incremental update is done by adjusting the preexisting model in a small window of recent labelled traffic, while ABBS optimizes only the hyperparameter component (binary feature selection is kept the same). If sudden drift occurs, a complete ABBS optimization cycle is run on the combined feature and hyperparameter space using a starting population of the previously found best solution to leverage from the search knowledge gained so far.

A drift library D_lib is used for a recurrent drift pattern, which contains historical (S*, θ*) configurations which correspond to previously observed drift events. As soon as a new drift event is detected, the current distribution is compared with stored distributions, and the distribution that matches best is retrieved to use as a seed for the ABBS population, greatly speeding up convergence.

6.3 Knowledge Transfer Between Optimization Cycles

Knowledge accumulated during previous ABBS optimization cycles is preserved through an elite archive E containing the top-k solutions (k=5) from each completed optimization run. When a new optimization cycle is triggered, the archive is incorporated into the initial population through elitism seeding, ensuring that valuable solution components discovered in earlier traffic regimes are available for recombination with solutions adapted to the new distribution.

7. DATASET DESCRIPTION AND PREPROCESSING

7.1 NSL-KDD

The NSL-KDD dataset, by Tavallaee et al., [10] was proposed as a corrected version of the KDD Cup 1999 dataset, which is criticized for having many errors. It contains 125,973 training records, and 22,544 test records. The records are characterized by 41 features that describe traffic that traverses the connection, the content of the traffic, and the host operating the traffic. The dataset contains five traffic types: Normal, DoS (Denial of Service), Probe, R2L (Remote-to-Local), and U2R (User-to-Root). The multi-class classification task becomes difficult due to the huge imbalance between the Normal and U2R classes, with the latter having less than 0.1% of the training records. In this study, NSL-KDD is used as a baseline dataset and the binary (Normal vs. Attack) and five-class experimental configurations are used separately.

7.2 CICIDS2017

Approximately 2.83 million flow records are available in the Canadian Institute for Cybersecurity Intrusion Detection System 2017 (CICIDS2017) dataset that was generated by Sharafaldin et al. [28] using the CICFlowMeter tool, which classifies flows into eight categories: Benign, Brute-Force, DoS Slowloris, DoS Slowhttptest, DoS Hulk, DoS GoldenEye, Heartbleed, Web Attacks, and Infiltration. The dataset features a realistic method of traffic generation, and a high temporal resolution (aggregated in 5-minute windows), which makes it a more ecologically valid benchmark than NSL-KDD for the current deployment scenarios. The extracted features are bidirectional packet count, inter-arrival time statistics, TCP flag counts and header level fields, totaling 78 ones.

7.3 UNSW-NB15

The UNSW-NB15 dataset was derived by Moustafa and Slay [29] from the Australian Centre for Cyber Security and has 2,540,044 records that span nine categories of attack (Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms) with 49 features. Unlike NSL-KDD, the dataset includes both real and synthetic traffic generation, and contemporary attack types, not included in the previous datasets. The UNSW-NB15 training and testing partitions have 175,341 and 82,332 records respectively with a class imbalance ratio of ~6.5:1 (Normal:Attack).

7.4 Preprocessing Procedures

A uniform preprocessing pipeline is used to process all three datasets. Zero duration flow records that result in infinite and NaN values are substituted with column median values. Features that have a near zero variance (variance threshold < 10^-4) are removed before optimization. The training partition is only oversampled via Synthetic Minority Over-sampling Technique (SMOTE), with the oversampling ratio being fixed to 5:1 for a majority to minority class ratio. All feature matrices are pre-processed with min-max normalization and splits with 80/20 train-test setups are applied throughout.

Algorithm 1: ABBS_Main(Ω, N, Tmax, τ, K, pm)

ρ ← ComputeFeatureImportance(Xtrain) ▷ Gini-based pre-screening (§5.3)
P ← Initialize(N, Ω, ρ) ▷ Algorithm 2
for each $b_i \in P$ do
Evaluate $F(b_i, \theta_i)$ via stratified 5-fold CV ▷ §4.3, §5.2
end for
$x^{best0} \leftarrow \text{argmin}_i F(b_i, \theta_i)$; $x^{worst0} \leftarrow \text{argmax}_i F(b_i, \theta_i)$
E ← ∅ ▷ Elite archive (§6.3)
Main Optimization Loop
for t = 1 to T_{max} do
$\omega_t \leftarrow \omega_{max} - (\omega_{max} - \omega_{min}) \cdot \sin(\pi \cdot t / 2 \cdot T_{max})$ ▷ Inertia decay
$\beta_t \leftarrow \beta_0 \cdot \exp(-\delta \cdot t / T_{max})$ ▷ Cluster attraction decay
if t mod τ = 0 then
$\{\mu^{kt}\}_{k=1}^{L} \leftarrow$ k-means(P, K) ▷ Cluster recomputation
$C(i,t) \leftarrow \text{argmin}^k \|x_i^t - \mu^{kt}\|_2 \ \forall i$ ▷ §4.2.3
end if
for each agent $b_i \in P$ do
if rand(0,1) < p_m then
$x_i^{t+1} \leftarrow$ MigrationUpdate($x^{best}\_c^{luster}$, σ_t) ▷ Algorithm 5
else if InExplorationPhase(t, T_{max}) then
$x_i^{t+1} \leftarrow$ ExplorationUpdate(x_i^t, x^{bestt}, ω_t, α) ▷ Algorithm 3

else
$x_i^{t+1} \leftarrow$ AggregationUpdate(x_i^t, $\mu c(_{i,}{}^t)^t$, x^{worstt}, β_t) ▷ Algorithm 4
end if
$x_i^{t+1} \leftarrow$ Clip(x_i^{t+1}, L, U) ▷ Boundary enforcement
$b_i^{t+1} \leftarrow$ BinaryTransfer(x_i^{t+1}) ▷ Algorithm 6; V-shaped mapping
Evaluate $F(b_i^{t+1}, \theta_i^{t+1})$
if $F(b_i^{t+1}, \theta_i^{t+1}) < F(b_i^t, \theta_i^t)$ then
Update personal best: $x_p{}^{best}{}_{,i} \leftarrow x_i^{t+1}$
end if
end for
$x^{bestt+1} \leftarrow \text{argmin}_i F(b_i^{t+1}, \theta_i^{t+1})$
$x^{worstt+1} \leftarrow \text{argmax}_i F(b_i^{t+1}, \theta_i^{t+1})$
$E \leftarrow$ UpdateEliteArchive(E, $x^{bestt+1}$, k=5) ▷ §6.3
end for

Algorithm 2 Initialize(N, Ω, ρ)

$n^{sobol} \leftarrow \lfloor 0.3 \cdot N \rfloor$ ▷ 30% Sobol-seeded agents for domain coverage
$X^{sobol} \leftarrow L + \text{SobolSequence}(n^{sobol}, d) \odot (U - L)$
$X^{rand} \leftarrow L + \text{rand}(0,1)^{nrand} \odot (U - L)$
$P \leftarrow [X^{sobol}; X^{rand}]$
for each agent $b_i \in P$ do
  for each feature j = 1 to d^{feat} do ▷ Bias binary dims by importance
    $p_i \leftarrow 0.3 + 0.4 \cdot \rho_i / \max^k(\rho^k)$
    $b_{ij}{}^0 \leftarrow \text{Bernoulli}(p_i)$
  end for
end for
return P

8. EXPERIMENTAL SETUP AND IMPLEMENTATION DETAILS

8.1 Hardware and Software Configuration

The experiments were performed on a workstation with an Intel Core i9-13900K (3.0 GHz base frequency, 24 cores) processor, 128 GB of DDR5 memory and NVIDIA RTX 4090 GPU (24 GB VRAM). The software was: Python 3.10.4, scikit-learn 1.2.1, NumPy 1.24.0, TensorFlow 2.11.0, and an in-house ABBS implementation written in Python using the NumPy vectorization for speed. The multiprocessing module of python was used to get parallel fitness evaluation with 16 processes.

8.2 Baseline Classifiers

The ABBS-optimized SA-IDS is based on a Random Forest (RF) classifier, chosen for its ability to handle variations in feature scale, interpretability and practical efficiency. In order to separate the effect of the ABBS optimization from the effect of the classifiers, the same results were obtained by performing comparative experiments with the eXtreme Gradient Boosting (XGBoost) and a Multi-Layer Perceptron (MLP) classifiers under the same optimization conditions.

8.3 Comparative Optimizers

The following optimization algorithms have been implemented to be compared under the same condition of experiments:
Standard inertia-weight variant of Particle Swarm Optimization with c_1 = c_2 = 2.0 and ω linearly decaying from 0.9 to 0.2.
Grey Wolf Optimizer (GWO): Standard form with α, β, and δ hierarchy and an linearly decreasing from 2 to 0.

The Harris Hawks Optimization (HHO) algorithm is a standard version where escaping energy E_0 is randomly chosen from the range of [-1, 1] and is uniformly distributed.
Whale Optimization Algorithm (WOA): With default values b = 1 and p = 0.5.
Processes: Differential Evolution (DE/rand/1/bin) with F = 0.8 and CR = 0.9.

8.4 ABBS Parameter Configuration

The ABBS algorithm was configured with the following parameters, determined through a systematic grid search on a held-out validation partition of NSL-KDD:
N = 50 agents, T_max = 100 iterations, K = 5 clusters, $\tau = 10$ (cluster recomputation frequency), $\omega_{max} = 0.9$, $\omega_{min} = 0.2$, $\alpha = 1.8$, $\beta_0 = 1.5$, $\delta = 3.0$, $p_m = 0.15$, $\eta = 0.5$, $\sigma_0 = 0.3$, $\lambda = 1.5$ (Lévy exponent), $w_1 = 0.40$, $w_2 = 0.25$, $w_3 = 0.15$, $w_4 = 0.20$. All the comparative algorithms were run with same number of population (N = 50) and budget allocaton (T_max = 100) for computational fairness.

8.5 Statistical Validation Protocol

The stochastic nature of the performance of the metaheuristics was taken into consideration by independently replicating the experiments 30 times with different random seeds. Data are presented as means ± SD. Wilcoxon signed-rank test for multiple comparisons (Holm-Bonferroni corrected) is used to test the statistical significance of pairwise comparisons between ABBS and each baseline at the significance level of $\alpha = 0.05$. Cohen's d is used as an effect size.

9. PERFORMANCE EVALUATION METRICS

The evaluation framework employs a comprehensive suite of metrics derived from the confusion matrix and probability output of the classifier to enable multi-dimensional performance characterization.

9.1 Primary Metrics

Classification Accuracy (Acc): Percentage of correctly classified instances of all classes:

$$Acc = (TP + TN) / (TP + TN + FP + FN)$$

Precision (Prec): The ratio of detected alarms that are actually attack alarms; indicates the workload of false positives on security analysts.

$$Prec = TP / (TP + FP)$$

Recall (Rec): Complete identification of threat: Fraction of true attacks detected;

$$Rec = TP / (TP + FN)$$

F1-Score: The harmonic mean of precision and recall, giving a single value summary that combines both precision and recall:

$$F1 = 2 \cdot (Prec \cdot Rec) / (Prec + Rec)$$

9.2 Operational Security Metrics

Fastest Attack Rate (FAR): The percentage of legitimate traffic that is wrongly identified as an attack that is directly related to the cost of false positives to the operational system:

$$FAR = FP / (FP + TN)$$

For historical convention and ease of comparing with previous studies, Detection Rate (DR) is reported separately in the IDS literature, and is equivalent to recall.
Probability of classification as an attack (higher Anomaly Score) than a randomly selected benign instance (lower Anomaly Score) (Area Under the ROC Curve (ROC-AUC)); For multi-class, AUC is calculated as the macro-average across all classes.

9.3 Optimization-Specific Metrics

The Feature Reduction Rate (FRR) is the degree of feature dimensionality reduction that one has achieved with the help of ABSS algorithm in comparison with the complete set of features, given as the expression $FRR = (1 - |S^*|/d) \times 100\%$.
Convergence Speed is described as the count of evaluations of fitness that has to be performed to come within 1% accuracy from the achieved fitness value.

10. RESULTS AND DISCUSSION

10.1 Feature Selection Results

Table 2 shows the results of the feature selection methods for the ABBS and rival algorithms used in the three benchmark datasets. The ABBS not only identified the most concise combinations of features among the different methods but also produced strong classification results, proving the efficiency of the method based on the use of thermal gradient navigation and aggregation model in selecting the necessary features.

Table 2: Feature Selection Comparison Across Datasets (Mean ± Std over 30 runs)

Algorithm	NSL-KDD Features	NSL-KDD Acc (%)	CICIDS Features	CICIDS Acc (%)	UNSW Features	UNSW Acc (%)
ABBS (Ours)	14.2±1.1	99.34±0.18	18.4±1.3	98.87±0.22	16.7±1.4	98.61±0.25
PSO	18.6±2.3	98.51±0.31	24.1±2.7	97.82±0.38	21.3±2.5	97.44±0.42
GWO	16.8±1.9	98.79±0.27	21.5±2.1	98.23±0.29	19.8±2.2	98.01±0.33
HHO	15.9±1.7	98.93±0.24	20.3±1.8	98.44±0.26	18.5±1.9	98.19±0.29
WOA	16.4±2.0	98.67±0.29	22.7±2.4	98.11±0.32	20.1±2.3	97.78±0.37
DE	17.3±2.1	98.62±0.28	23.0±2.5	97.96±0.35	20.8±2.4	97.61±0.39

The success of ABBS has been judged according to a mean of 14.2 features selected from the NSL-KDD space with the dimensionality of 41 (65.4% reduction); the corresponding figures for PSO was equal 18.6 features (54.6% reduction). Moreover, the accuracy of the method is corroborated by Wilcoxon test results showing the statistical superiority of ABBS over the closest competitor (HHO) (99.34% vs. 98.93%, $p < 0.001$; Cohen's $d = 1.87$). The few features that were selected by the method (top-k, diff_srv_rate, dst_host_srv_count, logged_in, and count) were in accordance with the previous feature importance studies in the IDS area.

10.2 Classification Performance on Multi-Class Detection

Table 3 shows the detailed classification metrics for the five-class NSL-KDD setting where SA-IDS needs to properly distinguish among the Normal, DoS, Probe, R2L, and U2R classes. The U2R class has the highest difficulty level in terms of detection as it is rare and has a similar behavior to normal administrative traffic.

Table 3: Multi-class Classification Performance on NSL-KDD (5-class)

Algorithm	Accuracy	Precision	Recall	F1-Score	FAR	AUC
SA-IDS (ABBS)	99.34%	99.28%	99.41%	99.34%	0.41%	0.9981
SA-IDS (PSO)	98.51%	98.43%	98.59%	98.51%	0.87%	0.9943
SA-IDS (GWO)	98.79%	98.71%	98.85%	98.78%	0.72%	0.9961
SA-IDS (HHO)	98.93%	98.87%	98.97%	98.92%	0.64%	0.9967
SA-IDS (WOA)	98.67%	98.59%	98.73%	98.66%	0.79%	0.9952
SA-IDS (DE)	98.62%	98.54%	98.69%	98.61%	0.83%	0.9947

ABBS-optimized SA-IDS attains an ROC-AUC of 0.9981 in the five-class NSL-KDD task, which is statistically superior to any of the baseline optimizers, as evident (with each p-value less than 0.01, when applying Wilcoxon signed-rank test and Holm-Bonferroni correction). In order to reduce the FAR from 0.87 % in case of PSO to 0.41% in case of ABBS, positive traffic experienced a reduction by 52.9%, saving the work hours for the personnel in Security Operations Centers.

10.3 Performance Under Concept Drift

In Table 4, the ability of SA-IDS to perform continuous learning is assessed by means of simulating a temporal drift scenario in CICIDS2017. In this case, traffic is divided into four consecutive intervals of two days, with classification being performed through training in Window 1 and testing in Windows 2-4, in three different conditions: (i) No Update (static), (ii) ABBS Re-optimization, and (iii) Incremental Update Only (retraining without ABBS and with the use of SMOTE).

Table 4: Accuracy Under Temporal Drift Simulation on CICIDS2017

Strategy	Window 2 Acc	Window 3 Acc	Window 4 Acc	Avg. Drift Degradation
No Update (Static)	97.43%	94.12%	88.67%	-10.15%

Strategy	Window 2 Acc	Window 3 Acc	Window 4 Acc	Avg. Drift Degradation
Incremental Update Only	98.11%	97.29%	95.83%	-3.04%
SA-IDS (ABBS Re-Optimization)	98.87%	98.54%	98.21%	-0.66%

The results show that the static model has a catastrophic performance decline of 10.15 percentage points in the fourth temporal window, demonstrating the seriousness of the concept drift in real-world IDS application. Meanwhile, degradation in the case of using the ABBS-re-optimization strategy will not exceed 0.66 percentage points, bringing a 93.5% improvement over the static baseline and reaching 78.3% improvement over the incremental update without re-optimization. The results above validate the decision on associating concept drift detection with ABBS re-optimization processes.

10.4 Convergence Analysis

ABBS was examined for convergence experience against the background of 30 independent runs of T_max = 100 iterations of the NSL-KDD testing problem, with the average of the mean best fitness values employed for this purpose. The convergence requirement of ABBS (less than 1% of final fitness) was reached at iteration 47.3 ± 3.8, as opposed to 61.2 ± 5.4 for PSO, 53.7 ± 4.6 for GWO, 50.1 ± 4.1 for HHO, 57.3 ± 5.0 for WOA, and 63.4 ± 5.7 for DE. Sinusoidal inertia decay and Lévy-flight-infused collective phase have been named as the major reasons for the fast convergence of ABBS, particularly the latter allowing the avoidance of the local optima halting PSO and DE.

10.5 Discussion

Three key findings were derived from the conducted experimentations. First and foremost, ABBS outperforms all tested metaheuristic benchmarks as it has a multi-stage implementation that efficiently focuses on the high-dimensional combination of features and hyper-parameters. The Lévy flight technique allows leveraging the characteristic “heavy-tailed” nature of the search effectiveness landscape where valuable solutions are unevenly spread through the search space.

Second, using ABBS in a drift-aware online learning system has quite significant practical implications. The significant decline of the degradation of the accuracy (by 93.5% under simulated drift) proves that as the advantages of the online re-optimization overweigh the costs of informing the process.

Third, the feature reduction accomplished by ABBS (65.4% on NSL-KDD, 76.4% on CICIDS2017) allows to better classify data as well as to cut down the inference time, which is paramount for a real-time IDS. The determined feature subsets significantly conform in different trials (Jaccard coefficient = 0.78), which indicates that relevant features are being found while discarding irrelevant ones.

11. CONCLUSION AND FUTURE WORK

11.1 Conclusion

The designed Self-Adaptive Intrusion Detection System (SA-IDS) described in this paper make use of the new Adaptive Boxelder Bug Search (ABBS) algorithm, which is applied within a drift-aware continuous learning framework. What is more, the ABBA algorithm derives from the navigation, aggregation, and migration activities demonstrated by Boisea trivittata and it is specified mathematically through adaptive inertia weight, exploitation based on clustering, and the process of migration between clusters aiming at restoration of diversity. The system employs unified search space encoding that allows simultaneous optimizations of both RA and hyperparameters of Random Forest guided by multi-objective fitness function taking into account several criterion improving pursuit of intrusion.

Different experiments carried out using different databases (NSL-KDD, CICIDS2017, and UNSW-NB15) show that the proposed SA-IDS demonstrates effective performance with regard to classification - the classification rate for NSL-KDD was equal to 99.34%, while it was 98.87% for CICIDS2017 and 98.61% for UNSW-NB15, with alarm rates amounting to 0.41%, 0.53%, and 0.68% respectively, which is more efficient than the five alternative metaheuristic optimizers with statistically important differences ($p < 0.01$, large effect sizes). During simulations of temporal drift, ABBS re-optimization limited the degradation of accuracy to 0.66%, which corresponds to a decrease of accuracy by over 93% when compared to static benchmarks. Overall, the discussed method contributes to the development of adaptive intrusion detection technologies and creates a framework that can be used during real life monitoring, guaranteeing its theoretical coherence and practical effectiveness.

11.2 Future Work

This research provides several new directions for further investigations. One area of study arises from the potential for implementing the ADBS in federated learning circumstances, allowing joint optimization of decentralized network nodes without collecting any sensitive traffic data. Another approach could be to develop quantum-inspired variant of ADBS that would make use of the principles of superposition in order to enhance the effectiveness of searching in extremely high-dimensional spaces with lots of features, typical for flow-based datasets implying hundreds of engineering features. In addition, the ADBS will become more efficient if the system outputs include means of explanation which will allow generating human-readable explanations for each alert produced by SA-IDS automatically, significantly enhancing the practical use of the system. Moreover, it would be interesting to test the SA-IDS system with encrypted packet datasets where normal payload-based features cannot be applied and detection is based on flow metadata exclusively. Finally, it would be great if detailed convergence analysis of the ABBS method was carried out based on real-life assumptions about the fitness landscape, thus providing an essential theoretical contribution to the field of metaheuristic optimization methods.

REFERENCES

[1] Cybersecurity Ventures, 'Cybercrime to Cost the World $10.5 Trillion Annually by 2025,' Cybersecurity Ventures Official Annual Cybercrime Report, 2021.
[2] I. Ghafir, J. Svoboda, V. Prenosil, and S. Alfadel, 'A Survey on Intrusion Detection and Prevention Systems,' in IEEE/ACS 14th Int. Conf. Computer Systems and Applications, 2017.
[3] C. Khammassi and S. Krichen, 'A GA-LR wrapper approach for feature selection in network intrusion detection,' Comput. Secur., vol. 70, pp. 255–277, 2017.
[4] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, 'Deep learning approach for intelligent intrusion detection system,' IEEE Access, vol. 7, pp. 41525–41550, 2019.
[5] G. Widmer and M. Kubat, 'Learning in the presence of concept drift and hidden contexts,' Mach. Learn., vol. 23, no. 1, pp. 69–101, 1996.
[6] E.-S. M. El-Alfy and M. A. Alsheikh, 'Toward designing efficient classifier ensembles for intrusion detection using hybrid feature selection and PSO,' Procedia Comput. Sci., vol. 50, pp. 265–272, 2015.
[7] D. Datta, A. Chaki, B. Bhattacharyya, and C. K. Maiti, 'Metaheuristic approaches for feature selection in intrusion detection systems: A comprehensive survey,' Neurocomputing, vol. 452, pp. 89–115, 2021.
[8] N. Hassan, A. Hameed, and A. Tahir, 'Comparative analysis of swarm intelligence algorithms for feature selection in intrusion detection,' J. Netw. Comput. Appl., vol. 189, p. 103126, 2023.
[9] B. Mukherjee and L. T. Heberlein, 'Network intrusion detection,' IEEE Netw., vol. 8, no. 3, pp. 26–41, 1994.
[10] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, 'A detailed analysis of the KDD CUP 99 data set,' in 2009 IEEE Symp. Computational Intelligence for Security and Defense Applications, 2009, pp. 1–6.
[11] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, 'A deep learning approach to network intrusion detection,' IEEE Trans. Emerg. Top. Comput. Intell., vol. 2, no. 1, pp. 41–50, 2018.
[12] C. Yin, Y. Zhu, J. Fei, and X. He, 'A deep learning approach for intrusion detection using recurrent neural networks,' IEEE Access, vol. 5, pp. 21954–21961, 2017.
[13] W.-L. Lo, W. Peng, and T.-H. Lin, 'E-GraphSAGE: A graph neural network based intrusion detection system for IoT,' in Proc. IEEE NDSS, 2022.
[14] M. A. Jabbar, B. L. Deekshatulu, and P. Chandra, 'Classification of intrusion detection system using KNN and identification of fuzzy rules,' in Proc. IEEE Int. Conf. Communication Systems and Network Technologies, 2013.
[15] S. Mirjalili and A. Lewis, 'The whale optimization algorithm,' Adv. Eng. Softw., vol. 95, pp. 51–67, 2016.
[16] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, 'Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system,' Expert Syst. Appl., vol. 67, pp. 296–303, 2017.
[17] S. Mirjalili, S. M. Mirjalili, and A. Lewis, 'Grey wolf optimizer,' Adv. Eng. Softw., vol. 69, pp. 46–61, 2014.
[18] H. H. Eid, A. I. Salama, A. S. Doaa, and H. Hamza, 'A new feature selection approach using GWO for intrusion detection classification,' Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 4, 2021.
[19] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, 'Harris hawks optimization: Algorithm and applications,' Future Gener. Comput. Syst., vol. 97, pp. 849–872, 2019.
[20] M. Canayaz, 'MH-COVIDNet: Diagnosis of COVID-19 using deep neural networks and meta-heuristic-based feature selection on X-ray images,' Biomed. Signal Process. Control, vol. 64, p. 102257, 2021.
[21] A. Tsymbal, 'The problem of concept drift: Definitions and related work,' Tech. Rep. TCD-CS-2004-15, Dept. Computer Science, Trinity College Dublin, 2004.
[22] R. Sommer and V. Paxson, 'Outside the closed world: On using machine learning for network intrusion detection,' in 2010 IEEE Symp. Security and Privacy, 2010, pp. 305–316.

[23] J. C. Schlimmer and R. H. Granger, 'Incremental learning from noisy data,' Mach. Learn., vol. 1, no. 3, pp. 317–354, 1986.
[24] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, 'Learning with drift detection,' in Advances in Artificial Intelligence, Springer, 2004, pp. 286–295.
[25] L. I. Kuncheva and I. Zliobaite, 'On the window size for classification in changing environments,' Intell. Data Anal., vol. 13, no. 6, pp. 861–872, 2009.
[26] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, 'Characterization of encrypted and VPN traffic using time-related features,' in Proc. ICISSP, 2016, pp. 407–414.
[27] A. Bifet and R. Gavalda, 'Learning from time-changing data with adaptive windowing,' in Proc. SIAM Int. Conf. Data Mining, 2007, pp. 443–448.
[28] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, 'Toward generating a new intrusion detection dataset and intrusion traffic characterization,' in Proc. ICISSP, 2018, pp. 108–116.
[29] N. Moustafa and J. Slay, 'UNSW-NB15: A comprehensive data set for network intrusion detection systems,' in 2015 Military Communications and Information Systems Conf., 2015, pp. 1–6.