Binary Tuna Swarm Optimization Algorithm-Based Feature Selection for Intrusion Detection Systems

Abdelouaheb Khiar, Smaine Mazouzi, Rohallah Benaboud, and Hichem Haouassi

Original scientific contribution

Abstract—**Feature selection is crucial for improving intrusion detection systems by addressing the curse of dimensionality and eliminating irrelevant features. However, applying continuous metaheuristics—such as the Tuna Swarm Optimization (TSO) algorithm—to this inherently binary problem requires effective binarization strategies. This paper presents TUNA-FS, a novel feature selection framework that employs a binary variant of the TSO algorithm. The proposed method introduces an adaptive V-shaped transfer function that dynamically manages the binarization process, maintaining a balance between exploration and exploitation throughout the search. Additionally, a multi-objective fitness function is used to jointly optimize key objectives: enhancing detection accuracy, reducing false alarms, and minimizing the number of selected features. The effectiveness of the approach is validated through comprehensive experiments on the NSL-KDD and CIC-IDS2017 benchmark datasets. Results demonstrate that the method achieves substantial feature reduction while maintaining high detection performance across multiple classifiers, including support vector machines, decision trees, random forests, and k-nearest neighbors. Comparative analysis against state-of-the-art methods confirms the competitiveness and balanced performance of the proposed framework, positioning it as an effective technique for enhancing intrusion detection efficiency and accuracy.**

Index Terms—**Feature Selection, Binary Tuna Swarm Optimization, Intrusion Detection System, Adaptive Transfer Function, Multi-objective Optimization, Network Security.**

I. Introduction

In today's interconnected digital landscape, cybersecurity threats have reached unprecedented levels, with global cybercrime costs projected to exceed $10.5 trillion annually by 2025 [1]. Intrusion Detection Systems (IDS) have become indispensable tools for safeguarding network infrastructures. These systems play a critical role in identifying and mitigating malicious activities, ensuring the integrity, confidentiality, and availability of sensitive data [2].

Intrusion Detection Systems are broadly categorized into tow types: signature based and anomaly-based approaches. Signature-based, also known as misuse detection, relies on predefined patterns or signatures of known attacks. While highly effective at detecting well-documented threats, this approach is inherently limited in its ability to identify zero-day attacks or polymorphic malware, which do not match existing signatures [3]. Furthermore, signature-based systems require frequent updates to their rule sets, often necessitating manual intervention by security experts, which can be both time-consuming and resource-intensive [4].

In contrast, anomaly-based systems leverage statistical analysis and machine learning to detect deviations from normal network behavior, enabling the identification of zero-day attacks [5]. Despite their advantages, anomaly-based systems face significant challenges, including high false-positive rates when distinguishing between legitimate traffic variations and actual threats [6], effectiveness heavily dependent on training feature quality, and increasing complexity due to exponential growth in network traffic from IoT devices, cloud computing, and big data applications [4].

The ”curse of dimensionality” presents a particular challenge, as network traffic data contains numerous features—from basic connection details to complex metrics—many of which are redundant or irrelevant [2]. Feature selection (FS) has emerged as a key strategy to address these issues by identifying the most relevant attributes, thereby improving detection accuracy, computational efficiency, and model interpretability [2].

Feature selection plays a pivotal role in the development of efficient and accurate IDS. In real-world network traffic datasets, the presence of a large number of features—many of which may be irrelevant or redundant—can hinder the performance of machine learning models. These high-dimensional datasets not only increase computational cost but also introduce noise, leading to overfitting and degraded detection accuracy. Therefore, identifying a minimal yet informative subset of features is essential to reduce complexity while maintaining or improving the system's detection capability.

Given the combinatorial nature of feature selection, traditional exhaustive search methods are computationally infeasible, especially for large datasets. As a result, metaheuristic optimization techniques have gained significant attention. These algorithms aim to efficiently explore the search space of feature subsets to find near-optimal solutions. Popular metaheuristics such as Genetic Algorithms, Particle Swarm Optimization (PSO), Ant Colony Optimization, and their many variants have been successfully applied to feature selection tasks [7]–[9]. Their ability to balance exploration and ex-

Manuscript received June 4, 2025; revised June 27, 2025. Date of publication September 30, 2025. Date of current version September 30, 2025. The associate editor prof. Hrvoje Karna has been coordinating the review of this manuscript and approved it for publication.

A. Khiar is with the Department of Mathematics and Computer Science, University of Oum El Bouaghi, Oum El Bouaghi, Algeria and ICOSI Laboratory, Abbes Laghrour University, Khenchela, Algeria (email: khiar_abdelouaheb@univ-khenchela.dz).

S. Mazouzi is with the LICUS Laboratory, University of Skikda, Skikda, Algeria (e-mail: s.mazouzi@univ-skikda.dz).

R. Benaboud is with the ReLa(CS)2 Laboratory, University of Oum El Bouaghi, Oum El Bouaghi, Algeria (e-mail: benaboud.rohallah@univ-oeb.dz).

H. Haouassi is with the ICOSI Laboratory, Abbes Laghrour University, Khenchela, Algeria (e-mail: haouassi.hichem@univ-khenchela.dz).

Digital Object Identifier (DOI): 10.24138/jcomss-2025-0083

ploitation makes them particularly suitable for navigating the complex and high-dimensional feature spaces encountered in intrusion detection applications.

Inspired by the collective foraging behaviour of tuna, the TSO algorithm has emerged as a powerful metaheuristic for FS in intrusion detection. Its global search capabilities and adaptability to high-dimensional optimization problems make it particularly suited for addressing modern cybersecurity challenges [10]–[12]. Recent studies highlight TSO's ability to balance exploration and exploitation, enabling efficient FS while maintaining detection accuracy [13].

A fundamental limitation in applying continuous optimization algorithms like TSO to feature selection —a binary decision problem— lies in the ineffective exploration of discrete search spaces. Conventional binarization methods, such as fixed thresholding [14], [15], force continuous solutions into binary formats, often resulting in premature convergence or suboptimal feature subsets. Our Binary TSO (BTSO) algorithm introduces enhanced transfer functions that adapt dynamically to the binarization process based on the TSO evolution to address this limitation.

Furthermore, a critical aspect of feature selection in intrusion detection is the definition of the fitness function, which guides the optimization process by evaluating the quality of feature subsets. The fitness function must balance multiple objectives, such as maximizing detection accuracy and minimizing the number of selected features [16], [17].

This paper presents a new feature selection framework for intrusion detection, based on a binary version of the Tuna Swarm Optimization algorithm. The proposed method efficiently explores high-dimensional search spaces to identify compact, relevant feature subsets that improve detection performance. The key contributions include:

- A novel Binary Tuna Swarm Optimization algorithm with an adaptive V-shaped transfer function for effective feature selection in IDS.
- A dynamic binarization strategy that balances exploration and exploitation during optimization.
- A multi-objective fitness function optimizing detection accuracy, false alarm rate, and feature sparsity.
- Comprehensive validation on NSL-KDD [18] and CIC-IDS2017 [19] datasets, demonstrating superior performance over state-of-the-art methods.

The remainder of the paper is structured as follows: Section II reviews related work, Section III details the TSO algorithm, and Section IV introduces the proposed BTSO-based feature selection methodology and TUNA-FS framework. Experimental results and discussions are presented in Section V, and conclusions and future directions are given in Section VI.

II. Related Works

Feature selection is a critical preprocessing step in developing high-performance IDS, particularly in environments characterized by high-dimensional, noisy, and redundant network traffic data. Effective FS enhances detection accuracy, reduces false positives, and lowers computational overhead—making IDS models more suitable for real-time and resource-constrained environments. This section reviews the evolution of FS techniques, including conventional methods, metaheuristic algorithms, hybrid models, and recent applications of TSO.

Traditional feature selection techniques are generally categorized into filter, wrapper, and embedded methods, each offering distinct advantages and trade-offs. Filter-based methods select features by applying statistical criteria independently of the learning algorithm, which allows for fast execution and good scalability [20]. However, they may overlook feature interactions essential for detecting complex or stealthy attacks. For instance, Information Gain (IG) has been widely adopted for its computational efficiency. Mazighi et al. [21] applied IG on the CICIDS-2017 dataset, achieving 93.4% accuracy with 19 selected features, though the false positive rate remained high at 12%. Thaseen et al. [22] used Chi-squared filtering on the NSL-KDD dataset, achieving a notable 99.23% accuracy. Similarly, Thockchom et al. [23] used correlation-based filtering on UNSW-NB15 to select 18 features, resulting in 99.2% accuracy. Rahman et al. [24] conducted an exploratory study on the impact of filter-based FS for malware detection using simple classifiers, showing that optimized feature subsets improve both efficiency and accuracy. Another study [25] applied IG, Pearson correlation, and ANOVA F-test on KDD99, UNSW-NB15, and CICIDS-2017 datasets, further confirming the effectiveness of filter-based FS.

Wrapper methods evaluate the quality of feature subsets using a predictive model, often leading to higher accuracy but at greater computational cost due to repeated model training [26]. Lee et al. [27] applied sequential forward selection with a random forest classifier, achieving 99.89% accuracy and a 4% false alarm rate using only 10 features. Yin et al. [28] used recursive feature elimination on UNSW-NB15, selecting 23 features and improving classification accuracy to 84.24%, though at the expense of increased computational time.

Embedded methods, in contrast, perform FS during model training, balancing between performance and computational efficiency. These techniques are model-dependent but offer good generalization [17]. Karthick Kumar et al. [29] used LASSO and ridge regression for FS in the context of botnet detection on UNSW-NB15, leveraging regularization for automatic feature pruning. Mohamed Yusof et al. [30] developed a hybrid filter-wrapper model using Bayesian networks on NSL-KDD, achieving detection rates between 86% and 100%, depending on the attack category. Although these conventional FS methods are widely applied, they struggle to scale in modern network environments characterized by high data volumes and dynamic behavior due to IoT, cloud computing, and 5G technologies.

To address the scalability and search-space limitations of traditional FS methods, metaheuristic algorithms have gained popularity for their global search capabilities and adaptability [9]. These algorithms mimic natural or physical processes to explore large solution spaces effectively. Alazzam et al. [31] applied the binary pigeon-inspired optimizer, reducing the feature set to five features while maintaining a true positive rate of 97.14%, although the false positive rate remained high at 14.95%. Saeed and Jameel [32] employed PSO for

DDoS detection, selecting 19 features and achieving 99.52% accuracy when combined with decision trees. Although PSO converges quickly, it may suffer from premature convergence. Selvakumar and Muneeswaran [33] proposed a hybrid FS approach combining mutual information filtering and the firefly algorithm, selecting 10 features and achieving high accuracy with C4.5 and Bayesian classifiers. Mazini et al. [34] utilized the artificial bee colony algorithm combined with AdaBoost to attain 98.9% accuracy and a 99.61% detection rate on the NSL-KDD and ISCXIDS2012 datasets. Almomani [9] provided a comparative evaluation of PSO, genetic algorithms, grey wolf optimizer, and the firefly algorithm, showing that genetic algorithms offered the highest true positive rate, while PSO achieved the best overall accuracy. However, all methods exhibited trade-offs in convergence speed, diversity preservation, and scalability.

Among recent advances, the Tuna Swarm Optimization algorithm has emerged as a bio-inspired metaheuristic modeled on the spiral and random foraging behaviors of tuna fish. TSO effectively balances exploration and exploitation, making it suitable for challenging optimization tasks like FS. Harwalkar et al. [12] applied TSO for feature selection in an IoT-based IDS, integrating it with LSTM networks, and achieved 99.98% accuracy on NSL-KDD. Gowthami and Priscila [11] used TSO with deep neural networks on the CICIDS-2017 dataset and reported 99.5% accuracy with a reduced feature set. Qin et al. [13] proposed a modified version of TSO to optimize decision tree parameters for software-defined networking scenarios, significantly improving classification accuracy and reducing detection latency.

The current review highlights a wide range of feature selection approaches applied to IDS. Among them, TSO demonstrates strong potential due to its dynamic search behavior and adaptability. However, its binarization for FS problems and its integration into multi-objective frameworks remain under-explored. Addressing these challenges represents a promising direction for developing more effective, scalable, and adaptive intrusion detection systems and forms the core motivation for the present research.

III. Tuna Swarm Optimization Algorithm (TSO)

Tuna swarm optimization algorithm is a metaheuristic algorithm inspired by tuna foraging behavior [10]. It mimics cooperative hunting, balancing exploration and exploitation, suitable for high-dimensional optimization [12]. TSO is based on two main hunting patterns: spiral foraging (forming spirals to drive prey) and parabolic foraging (encircling prey in a parabolic formation). The effectiveness of TSO is demonstrated in various domains, including feature selection for IDS [13]. Its simplicity makes it a competitive metaheuristic.

A. Mathematical Formulation

1) Initialization: The algorithm initializes NP tuna individuals (X_i) randomly within bounds (lb, ub):

$$X_i = \text{rand} \cdot (ub - lb) + lb, \quad i = 1, 2, ..., NP \tag{1}$$

2) Spiral Foraging: Positions are updated based on spiral movement towards the best (X_{best}) or a random tuna (X_{rand}):

$$X_i^{t+1} = \begin{cases} \alpha_1 \cdot (X_{rand}^t + \beta \cdot |X_{rand}^t - X_i^t|) + \alpha_2 \cdot X_i^t, & i = 1 \\ \alpha_1 \cdot (X_{rand}^t + \beta \cdot |X_{rand}^t - X_i^t|) + \alpha_2 \cdot X_{i-1}^t, & i > 1 \end{cases} \quad \text{if } rand < \frac{t}{t_{max}} \tag{2}$$

$$X_i^{t+1} = \begin{cases} \alpha_1 \cdot (X_{best}^t + \beta \cdot |X_{best}^t - X_i^t|) + \alpha_2 \cdot X_i^t, & i = 1 \\ \alpha_1 \cdot (X_{best}^t + \beta \cdot |X_{best}^t - X_i^t|) + \alpha_2 \cdot X_{i-1}^t, & i > 1 \end{cases} \quad \text{if } rand \geq \frac{t}{t_{max}} \tag{3}$$

Weight coefficients α_1, α_2 and spiral parameter β are calculated dynamically:

$$\alpha_1 = a + (1 - a)\frac{t}{t_{max}} \tag{4}$$

$$\alpha_2 = (1 - a) - (1 - a)\frac{t}{t_{max}} \tag{5}$$

$$\beta = e^{bl} \cos(2\pi b) \tag{6}$$

$$l = e^{3\cos(((t_{max}+1/t)-1)\pi)} \tag{7}$$

where b is a random number uniformly distributed between 0 and 1.

3) Parabolic Foraging: Tunas follow a parabolic path towards X_{best}:

$$X_i^{t+1} = \begin{cases} X_{\text{best}}^t + \text{rand} \cdot (X_{\text{best}}^t - X_i^t) + TP \cdot p^2 \cdot (X_{\text{best}}^t - X_i^t), & \text{if } rand < 0.5 \\ X_{\text{best}}^t + TP \cdot p^2 \cdot X_i^t, & \text{if } rand \geq 0.5 \end{cases} \tag{8}$$

where $TP \in \{-1, 1\}$ and path parameter p is:

$$p = (1 - \frac{t}{t_{max}})^{(t/t_{max})} \tag{9}$$

B. Algorithmic Procedure

The TSO process is outlined in Algorithm 1.

TSO uses two parameters: a controls the balance between exploration/exploitation via α_1, α_2; z controls the probability of random re-initialization versus foraging. Empirical studies suggest $a = 0.7$ and $z = 0.05$ are often optimal [10]. TSO's robustness and efficiency have been validated [10], [12], [13], establishing it as competitive for complex tasks like FS.

IV. TUNA-FS Framework

This section details our proposed framework, TUNA-FS, which is designed to be highly practical and applicable in real-world scenarios. TUNA-FS is an adaptation of the TSO algorithm that specifically addresses the challenges of feature selection in intrusion detection. We first introduce our Binary

Algorithm 1 Tuna Swarm Optimization (TSO)

```
1: Initialize population $X_i$ $(i = 1..N)$ randomly.
2: Set parameters $a, z$.
3: while termination criterion not met do
4:     Calculate fitness for each tuna.
5:     Update $X_{best}$ position and value.
6:     for each tuna do
7:         Update $\alpha_1, \alpha_2, p$ using Eqs. (4), (5), (9).
8:         if $rand < z$ then                        ▷ Random initialization
9:             Update position using Eq. (1).
10:        else                                      ▷ Foraging
11:            if $rand < 0.5$ then                  ▷ Spiral Foraging
12:                if $t/t_{max} < rand$ then
13:                    Update using Eq. (2)
14:                else
15:                    Update using Eq. (3)
16:                end if
17:            else                                  ▷ Parabolic Foraging
18:                Update using Eq. (8).
19:            end if
20:        end if
21:    end for
22:    $t = t + 1$.
23: end while
24: return $X_{best}$ and $f(X_{best})$.
```

TSO variant, which features an adaptive V-shaped transfer function and binarization rule designed to navigate the discrete search space effectively. Finally, we present the multi-objective fitness function, a key component of our framework tailored to meet the conflicting demands of real-world IDS deployment.

A. Binary TSO Algorithm (BTSO)

In a binary problem, solutions must be encoded as binary vectors where each bit represents whether a feature is selected (1) or not (0). This binary nature creates a fundamental challenge when applying continuous-domain algorithms to discrete search spaces.

The key challenge lies in the movement mechanism of the original TSO algorithm. While tuna in TSO can move freely in continuous space using equations (2), (3) and (8) to update real-valued position, binary optimization requires tuna to navigate a hypercube where only two values (0 and 1) are permitted for each dimension. In this discrete landscape, tuna cannot simply adjust their positions incrementally but must instead make binary decisions to flip or maintain each bit.

To solve this problem, we used a mechanism to map real-valued positions to {0,1}. Transfer functions have been successfully employed to binarize various metaheuristic optimization algorithms, including PSO [14], [35], WOA [36], and GWO [37].

1) Binarization Strategy: Adaptive V-Shaped Transfer Function: Transfer functions were first introduced by Kennedy et al. [14] to bridge continuous and binary optimization spaces. These functions map continuous metaheuristic outputs to probabilities in [0,1], preserving search dynamics while enabling binary solutions. While conventional TF have proven useful [37]–[39], recent studies demonstrate that adaptive – time-varying – transfer functions offer superior performance in avoiding local optima stagnation [40]–[42].

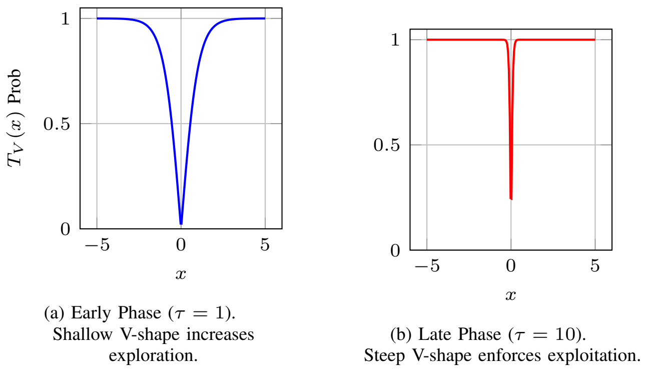


Fig. 1. Evolution of the adaptive V-shaped TF $T_V(x) = |\tanh(\tau \cdot x)|$.

a) Mathematical Formulation: The adaptive V-shaped TF is:

$$T_V(x_i^t) = \left|\tanh\left(\tau(t) \cdot x_i^t\right)\right| \tag{10}$$

where $\tau(t)$ controls the slope steepness, evolving over time t:

$$\tau(t) = \tau_{\text{min}} + (\tau_{\text{max}} - \tau_{\text{min}}) \cdot \left(\frac{t}{t_{\text{max}}}\right) \tag{11}$$

We set $\tau_{\text{min}} = 1$ and $\tau_{\text{max}} = 10$. Figure 1 illustrates its evolution.

b) Binarization Rule: While transfer functions map continuous values to probabilities, binarization rules determine how these probabilities are converted to binary solutions. We propose a decision behavior that preserves exploitation solution while enabling controlled exploration. The following rule converts TF probabilities values to binary decisions:

$$Bx_i^{(t+1)} = \begin{cases} \neg Bx_i^{(t)} & \text{if } rand < T_V(X_i^{t+1}) \\ Bx_i^{(t)} & \text{otherwise} \end{cases} \tag{12}$$

where $Bx_i^{(t)}$ is the current binary value of the j-th dimension of tuna i, X_i^{t+1} is the updated continuous position from TSO, T_V is the adaptive V-shaped transfer function, $\neg$ is the bit flip (logical NOT), and $rand \in [0, 1]$.

2) Algorithm Integration: The BTSO algorithm integrates the TSO foraging strategies with the adaptive transfer function and binarization rule, as shown in Algorithm 2.

The BTSO algorithm (Algorithm 2) maintains the core behavioral principles of the TSO algorithm but introduces specialized mechanisms to handle binary solution encoding and discrete movement patterns.

In terms of computational complexity, TUNA-FS has a time complexity of approximately $O(NP \times t_{\text{max}} \times C)$, where NP is the population size, t_{max} is the number of iterations, and C is the cost of evaluating a candidate solution using a classifier. The binarization process and the adaptive transfer function introduce negligible computational overhead.

Algorithm 2 Binary Tuna Swarm Optimization (BTSO)

```
Require: $NP$ tunas, $t_{\max}$, $a$, $z$, $\tau_{\min}$, $\tau_{\max}$
Ensure: Optimal binary feature subset $BX_{best}$
 1: Initialize continuous positions $X_i \in [-L, L]^D$ and binary positions $BX_i$.
 2: Evaluate initial fitness $f(BX_i)$, find $BX_{best}$.
 3: for $t = 1$ to $t_{\max}$ do
 4:     Compute $\tau(t)$ using Eq. (11).
 5:     for each tuna $i$ do
 6:         Update continuous position $X_i^{t+1}$ via TSO foraging (Eqs. 2, 3, 8).
 7:         for each dimension $j = 1$ to $D$ do
 8:             Calculate probability $P_i = T_V(X_i^{t+1})$ using Eq. (10).
 9:             Generate binary value $BX_i^{(t+1)}$ using Eq. (12).
10:         end for
11:         Evaluate fitness $f(BX_i^{t+1})$ using Eq. (13).
12:         if $f(BX_i^{t+1})$ is better than $f(BX_{best})$ then
13:             Update $BX_{best} = BX_i^{t+1}$.
14:         end if
15:     end for
16: end for
17: return $BX_{best}$ and $f(BX_{best})$.
```

B. FS with Binary TSO for IDS

Our TUNA-FS framework applies the BTSO algorithm to identify the most relevant feature subset for IDS. The search is driven by evaluating candidate subsets using a tailored fitness function described below. The subset yielding the best fitness upon BTSO convergence is chosen as the optimal set for training the final IDS classifier.

1) Detection Metrics for Final Evaluation: To evaluate the final performance of the IDS using the TUNA-FS selected feature subset, we adopt standard metrics [9]:

- *Accuracy*: Overall correctness.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- *Precision*: Proportion of predicted intrusions that are correct.

$$Precision = \frac{TP}{TP + FP}$$

- *Detection Rate/Recall*: Proportion of actual intrusions detected.

$$Recall = \frac{TP}{TP + FN}$$

- *False Alarm Rate (FAR)*: False Positive Rate (FP-Rate), proportion of normal instances misclassified as intrusions.

$$FAR = \frac{FP}{FP + TN}$$

- *F1-Score*: Harmonic mean of Precision and Recall.

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

where TP, TN, FP, and FN are True Positives, True Negatives, False Positives and False Negatives, respectively.

2) Fitness Function Design: The selection of evaluation metrics for IDS depends on several key factors: Characteristics of the dataset, Security Requirements, Operational Context, and Cost Considerations. In typical network traffic, normal instances vastly outnumber intrusion attempts. In such scenarios, a naive classifier that always predicts "normal" would achieve nearly 100% accuracy, despite being utterly useless for detecting intrusions. On the other hand, the emphasis is often on minimizing FAR while effectively detecting intrusions. Our fitness function integrates three core objectives:

- Detection Performance & Balance: Using macro-averaged F1-Score ($F1_{\text{macro}}$) for robustness to imbalance.
- False Alarm Minimization: Rewarding low FAR using $(1 - FAR)$.
- Feature Sparsity: Encouraging fewer features ($|S|$) using $(1 - |S|/D)$, where D is the total feature count.

The fitness function to be maximized is:

$$\text{Max Fitness} = \omega_1 \cdot F1_{\text{macro}} + \omega_2 \cdot (1 - FAR) + \omega_3 \cdot (1 - |S|/D) \quad (13)$$

where $\omega_1, \omega_2, \omega_3$ are configurable weights ($\sum \omega_i = 1$) calibrated via sensitivity analysis to prioritize objectives.

C. Algorithm complexity

The computational complexity of the BTSO algorithm can be approximated by analyzing its key components. Let C be the cost of the objective function, typically dominated by the classification algorithm used for fitness evaluation. During initialization, each of the NP tuna agents is assigned a continuous and a binary position vector of length D, resulting in a complexity of $O(NP \cdot D)$. In the subsequent step, the initial fitness of each agent is evaluated, contributing an additional cost of $O(NP \cdot C)$.

The primary computational load arises from the iterative optimization loop, which executes for $t_{\max}$ iterations. Within each iteration, the continuous positions of all tuna agents are updated over D dimensions, incurring a cost of $O(t_{\max} \cdot NP \cdot D)$. After each update, the binarization process is performed, involving an adaptive transfer function and thresholding, followed by fitness evaluation for each agent. These steps also contribute $O(D)$ per tuna per iteration. Across the population and all iterations, this results in an additional cost of $O(t_{\max} \cdot NP \cdot D \cdot C)$.

Assuming C is constant and dominated by the classifier, the overall time complexity of BTSO can be approximated as $O(t_{\max} \cdot NP \cdot D)$, indicating that the algorithm scales linearly with the number of features, population size, and iterations—making it computationally efficient for medium- to large-scale feature selection tasks. This theoretical analysis is supported by the empirical runtime results reported in Tables IV and V, which show that TUNA-FS significantly reduces execution time across multiple classifiers and datasets.

V. Experimental Setup and Discussion

This section details the experimental validation of the TUNA-FS framework. We conducted comprehensive experiments on benchmark datasets in a computational environment with 2x Intel Xeon PLATINUM 8160 @ 2.40GHz, 128GB RAM, NVIDIA RTX 3500 with 12GB memory, Python 3.9.10, scikit-learn 1.4. The BTSO algorithm was implemented from MealPy in Python [43].

A. Datasets

For this study, we utilized two widely recognized datasets for IDS: NSL-KDD [18] and CIC-2017 [19]. Both datasets have been extensively used as benchmarks in the cybersecurity research community for evaluating intrusion detection systems.

B. Evaluation Classifiers

A two-stage classifier approach was used. During the BTSO optimization phase, feature subset fitness was evaluated using a Linear Support Vector Classifier (SVC) with L2 regularization for efficiency. To assess the final performance and generalizability of the feature subsets selected by TUNA-FS, we employed a panel of diverse classifiers: Random Forest (RF), k-Nearest Neighbors (k-NN), and Decision Tree (DT).

C. Convergence Analysis and Parameter Sensitivity

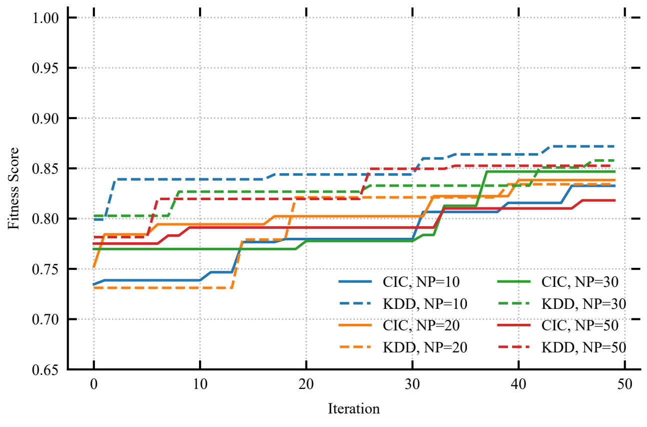
The convergence characteristics and parameter sensitivity of TUNA-FS are examined via the fitness evolution plots in Figure 2. These compare performance on NSL-KDD (dashed lines) and CIC-IDS2017 (solid lines) datasets under varying population sizes (NP, by color) and maximum iterations (t_{max}=50, 100, 200 in sub-figures a, b, c).

The convergence plots in Figure 2 confirm that TUNA-FS effectively improves solutions over time. Examining the influence of maximum iterations (t_{max}), a substantial improvement in fitness is evident when increasing from $t_{max} = 50$ (Fig. 2 (a)) to $t_{max} = 100$ (Fig. 2 (b)). For instance, top configurations (NP=30/50) on CIC-IDS2017 typically rise from fitness levels around 0.85-0.87 to approximately 0.92-0.94. The convergence rate often slows noticeably after 70-80 iterations within the $t_{max} = 100$ runs. Further extending the search to $t_{max} = 200$ (Fig. 2 (c)) results in relatively minor additional fitness gains, with top performance stabilizing near 0.93-0.95 for CIC-IDS2017, highlighting diminishing returns for iterations beyond 100-150.

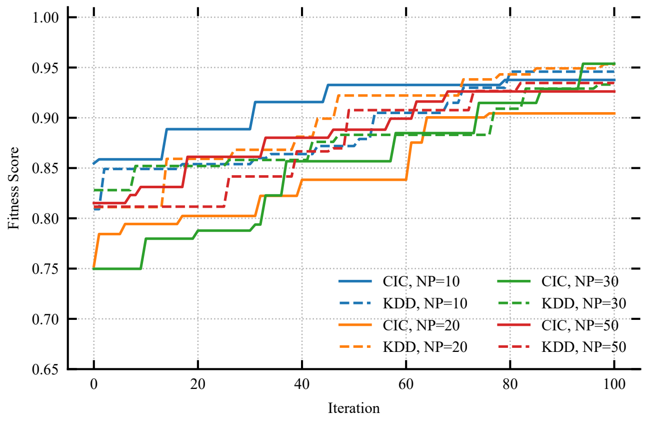
Population size plays a significant role in avoiding local optima and achieving high-quality solutions. Smaller populations, especially NP=10, consistently demonstrate premature convergence, often stagnating well below the potential optimum (e.g., failing to exceed 0.89 fitness on CIC-IDS2017 in Fig. 2 (c)). In contrast, larger populations (NP=30 and NP=50) facilitate more thorough exploration, leading to significantly better final fitness values across all t_{max} settings compared to NP=10 and NP=20. A crucial observation, especially in Figures 2 (b) and (c), is that the peak performance achieved by NP=30 is often very close to, or indistinguishable from,

TABLE I
Characteristics of the Benchmark Datasets.

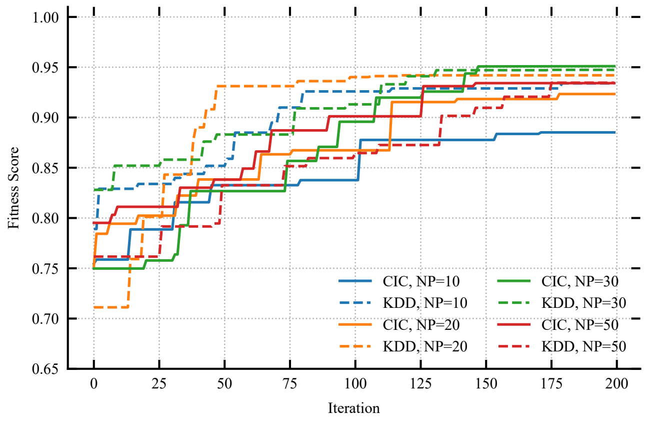
Dataset	Instances	Features	Attack Types	Attack Ratio (%)
NSL-KDD	125,973	41	4	46.5
CIC-IDS2017	2,830,743	78	14	19.7



(a) Fitness Evolution $t_{max} = 50$



(b) Fitness Evolution $t_{max} = 100$



(c) Fitness Evolution $t_{max} = 200$

Fig. 2. TUNA-FS convergence for varying t_{max} and NP values. in CIC-IDS2017, NSL-KDD
The fitness values are obtained using a Linear SVC with L2 regularization

that achieved by NP=50 (e.g., both hover around 0.94-0.95 on NSL-KDD in Fig. 2 (c)).

The underlying dataset also impacts performance. TUNA-FS generally converges faster and reaches slightly higher fitness scores on the NSL-KDD dataset (dashed lines) than on the larger, more complex CIC-IDS2017 dataset (solid lines). For example, comparing NP=30 runs at $t_{max} = 100$ (Fig. 2 (b)), fitness reaches approximately 0.95 on NSL-KDD versus around 0.93 on CIC-IDS2017. The differentiation in performance between the various population sizes is also more evident on the CIC-IDS2017 dataset, suggesting its optimization landscape is more sensitive to the search agent count.

Considering the trade-off between convergence quality and computational cost, $t_{max} = 100$ captures the majority of performance gains efficiently. Since NP=30 achieves robust results similar to NP=50 with less overhead, the configuration $t_{max} = 100$ and $NP = 30$ was selected as a balanced and effective setting for subsequent comparative experiments.

D. Sensitivity Analysis of Transfer Function Parameters

The adaptive V-shaped transfer function's parameters, τ_{min} and τ_{max}, govern the dynamic behavior of the binarization process. Their impact on feature selection performance was systematically analyzed on the NSL-KDD dataset, with results summarized in Table II. This analysis focused on the trade-offs between overall fitness, F1-score, FP-Rate, and feature subset cardinality.

Examination of $\boldsymbol{\tau}_{\text{min}}$ (while τ_{max} was held at 6.0) reveals distinct performance characteristics. A τ_{min} value of 1.5 emerged as particularly effective, achieving the highest overall fitness (0.9539) and a desirable balance by selecting only 8 features with a competitive F1-Score (0.9726) and FP-Rate (0.0374). Lowering τ_{min} to 0.5, while yielding a slightly higher F1-Score, resulted in a substantially larger feature set (19 features) and an increased FP-Rate, suggesting overly broad initial exploration. Conversely, a τ_{min} of 1.0, though achieving a low FP-Rate, compromised overall fitness. This indicates that τ_{min} critically tunes the initial exploration phase: a value of 1.5 provides a robust starting point for exploration without over-selecting features on the NSL-KDD dataset.

The influence of τ_{max} was investigated with τ_{min} fixed at 1.0. The configuration with $\tau_{\text{max}} = 8.0$ demonstrated superior performance, attaining the highest fitness (0.9605), the lowest FP-Rate (0.0237), and a compact 9-feature subset. This significantly outperformed the non-adaptive baseline (where τ_{max} was also 1.0), which, despite good performance, could not match the balance achieved when allowing τ to adapt to a higher maximum. Increasing τ_{max} beyond 8.0 (e.g., to 10.0 or 12.0) did not yield further improvements and, in some cases, slightly degraded performance, particularly the FP-Rate. This underscores the benefit of a sufficiently high τ_{max} to drive the algorithm towards stronger exploitation in its later stages, thereby refining the selected feature subset effectively.

Collectively, these results affirm the utility of the adaptive V-shaped transfer function. The dynamic adjustment of the TF's slope, controlled by τ_{min} and τ_{max}, allows for a more sophisticated balance between exploration and exploitation compared to a static TF. Based on this sensitivity analysis, the parameter settings of $\tau_{\text{min}} = 1.5$ and $\tau_{\text{max}} = 8.0$ were identified as providing a compelling trade-off between high fitness, low false alarm rates, and feature sparsity for the NSL-KDD dataset, guiding their selection for subsequent evaluations.

TABLE II
SENSITIVITY ANALYSIS OF TF PARAMETERS ON NSL-KDD.

Varying $\boldsymbol{\tau}_{\text{min}}$ ($\boldsymbol{\tau}_{\text{max}} = \mathbf{6.0}$ fixed)				
$\boldsymbol{\tau}_{\text{min}}$	Fitness	F1-Score	FP-Rate	Features
0.5	0.9337	0.9861	0.0512	19
1.0	0.9120	0.9361	0.0215	16
<u>1.5</u>	<u>0.9539</u>	<u>0.9726</u>	<u>0.0374</u>	<u>8</u>
2.0	0.9511	0.9848	0.0452	12
Varying $\boldsymbol{\tau}_{\text{max}}$ ($\boldsymbol{\tau}_{\text{min}} = \mathbf{1.0}$ fixed)				
$\boldsymbol{\tau}_{\text{max}}$	Fitness	F1-Score	FP-Rate	Features
1.0*	0.9497	0.9801	0.0480	11
4.0	0.9434	0.9960	0.0493	18
6.0	0.9503	0.9899	0.0422	14
<u>8.0</u>	<u>0.9605</u>	<u>0.9817</u>	<u>0.0237</u>	<u>9</u>
10.0	0.9422	0.9797	0.0594	13
12.0	0.9485	0.9815	0.0709	10

*Non-adaptive baseline. Metrics averaged over 20 runs.

E. Fitness Function Weights Analysis

The sensitivity analysis for the fitness function weights (ω_1 for performance/balance, ω_2 for low FP-Rate, ω_3 for sparsity), presented in Table III, demonstrates the inherent trade-offs in feature selection for IDS. For instance, the perfectly balanced configuration [0.33, 0.33, 0.33] prioritizes sparsity most effectively, yielding the minimal feature set size of 7 features. However, this comes at a significant cost to other metrics, resulting in the lowest accuracy (96.58%) and the highest, potentially unacceptable, FP-Rate (10.98%) among the tested configurations. Conversely, prioritizing only performance ($\omega_3 = 0$) leads to high accuracy (99.44%) but requires a large number of features (25). These results highlight that the weight configuration $\omega_1 = 0.8, \omega_2 = 0.1, \omega_3 = 0.1$ achieves high accuracy (99.14%), the lowest observed FP-Rate (3.07%), and excellent sparsity using only 9 features. Although the ideal weight selection depends on specific application priorities, the [0.8, 0.1, 0.1] setting shows a superior combination of desirable metrics in this evaluation.

In summary, our parameter analysis establishes optimal settings for BTSO in feature selection for intrusion detection.

TABLE III
SENSITIVITY ANALYSIS OF FITNESS FUNCTION WEIGHTS ON NSL-KDD.

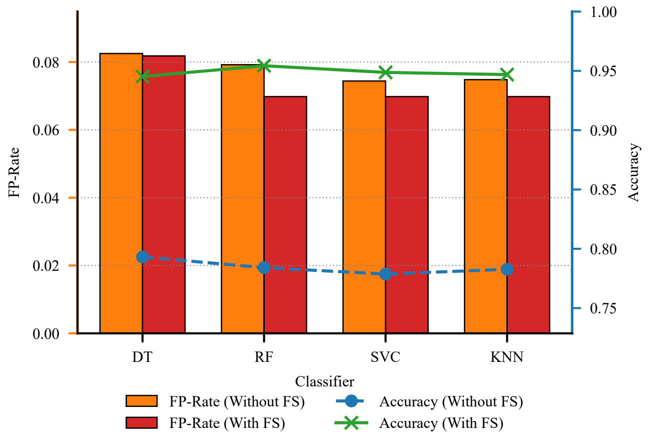
$\boldsymbol{\omega_1}$	$\boldsymbol{\omega_2}$	$\boldsymbol{\omega_3}$	Accuracy	FP-Rate (%)	# Feats
0.33	0.33	0.33	96.58	10.98	7
0.80	0.20	0.00	99.44	6.48	25
0.70	0.10	0.20	98.42	8.26	10
0.60	0.20	0.20	97.91	3.74	15
0.50	0.30	0.20	97.46	3.82	11
<u>0.80</u>	<u>0.10</u>	<u>0.10</u>	<u>99.14</u>	<u>3.07</u>	<u>9</u>
0.70	0.20	0.10	98.66	4.23	12
0.60	0.30	0.10	94.02	3.10	13

The parameters NP = 30, t_{max} = 100, τ_{min} = 1.5, τ_{max} = 8.0 (based on Table II best values), and fitness weights ω_1 = 0.8, ω_2 = 0.1, ω_3 = 0.1 provide a strong balance between detection accuracy, false alarm reduction, and computational efficiency, based on the sensitivity analyses performed on NSL-KDD.

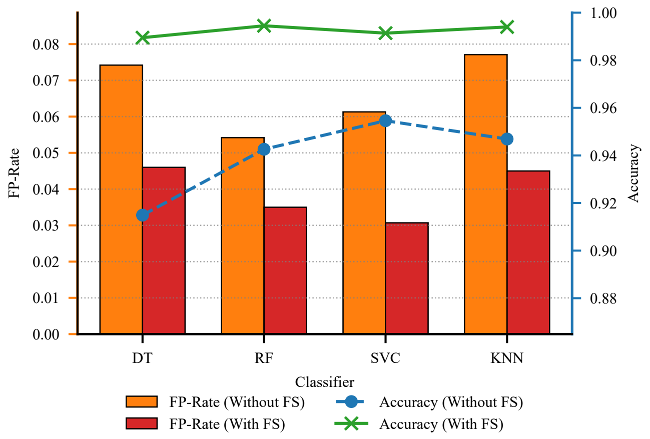
F. Performance Evaluation

The performance evaluation results, presented in Figure 3, illustrate the effectiveness of TUNA-FS feature selection across different classifiers and datasets. On the more complex CIC-IDS2017 dataset (Fig. 3(a)), applying TUNA-FS maintains the already high accuracy levels achieved with the full feature set (lines remain close, generally above 0.95) while slightly reducing the FP-Rate for most classifiers (red bars slightly lower or equal to orange bars). This demonstrates TUNA-FS's ability to significantly reduce dimensionality without sacrificing performance on contemporary datasets.

The benefits are more striking on the NSL-KDD dataset (Fig. 3(b)). Here, using the TUNA-FS selected features consistently leads to a marked improvement in accuracy across all classifiers (green line significantly above blue dashed line, often exceeding 0.99), particularly noticeable for DT and KNN, which performed less well with the full set. Concurrently, a substantial reduction in the FP-Rate is observed for all classifiers (red bars are much lower than orange bars, often halved or more). This strongly suggests TUNA-FS effectively eliminates noisy or redundant features from NSL-KDD that hindered both accuracy and generated excessive false alarms. Collectively, these results validate TUNA-FS as a valuable approach, enhancing classifier efficiency and effectiveness by identifying impactful feature subsets tailored to the dataset characteristics.



(a) CIC-IDS2017



(b) NSL-KDD

Fig. 3. Performance comparison using Full Features vs. TUNA-FS.

TABLE IV
TRAINING AND TESTING TIME ON NSL-KDD (SECONDS)

Classifier	Training Time		Testing Time	
	Full (41)	TUNA-FS (9)	Full (41)	TUNA-FS (9)
SVM	4.2325	1.2956 (-69.4%)	0.6872	0.2311 (-66.4%)
DT	1.1542	0.3517 (-69.5%)	0.1742	0.0541 (-69.0%)
k-NN	6.2174	1.3254 (-78.7%)	1.0759	0.2384 (-77.8%)
RF	9.9541	3.214 (-67.7%)	1.4355	0.5215 (-63.7%)

Note: Reduction percentages shown in parentheses.

Overall, these results validate the efficacy of the TUNA-FS framework. It demonstrably enhances classifier performance by selecting relevant features, leading to significantly improved accuracy and lower false alarm rates on datasets like NSL-KDD, while maintaining high performance levels and reducing dimensionality on more modern datasets like CIC-IDS2017.

We comprehensively evaluated runtime efficiency gains across all classifiers and datasets. Tables IV and V compare the training and testing times for each classifier for the NSL-KDD and CIC-IDS2017 datasets, before and after applying TUNA-FS, reporting absolute times and percentage reductions. As shown, the use of the proposed framework leads to substantial reductions in computational time across all classifiers. On NSL-KDD, training time was reduced by up to 78.7% (k-NN), and testing time by up to 77.8%. Similar improvements are observed on CIC-IDS2017, where training time dropped by over 79% and testing time by over 76%. These gains are primarily due to the dimensionality reduction achieved by TUNA-FS, which reduces the number of input features while preserving classification performance. This demonstrates that the framework not only improves detection quality but also significantly reduces computational overhead, making it suitable for time-sensitive or resource-constrained environments.

G. Comparison with state-of-the-art

Table VI compares TUNA-FS against contemporary FS methods. Direct comparison requires caution due to methodological variations.

TUNA-FS demonstrates clear advantages over filter methods [21], [22], offering comparable or better accuracy and significantly lower FP-Rates with fewer features. Against wrapper/hybrid methods [9], [28], [32], TUNA-FS provides competitive accuracy with substantial feature reduction on bench-

TABLE V
TRAINING AND TESTING TIME ON CIC-IDS2017 (SECONDS)

Classifier	Training Time		Testing Time	
	Full (78)	TUNA-FS (21)	Full (78)	TUNA-FS (21)
SVM	105.1652	29.3358 (-72.1%)	14.2181	4.1257 (-71.0%)
DT	26.4381	7.2145 (-72.7%)	3.2846	0.9354 (-71.5%)
k-NN	158.498	33.1504 (-79.1%)	21.541	5.1283 (-76.2%)
RF	219.2381	72.2519 (-67.0%)	29.1181	9.7462 (-66.5%)

Note: Reduction percentages shown in parentheses.

mark datasets. Compared to other metaheuristics, TUNA-FS achieves a superior balance; unlike Binary PIO [31] which sacrificed FP-Rate for sparsity, TUNA-FS maintains a low FP-Rate (e.g., 3.07% on NSL-KDD) with high accuracy (99.14%) and a compact feature set (9 features). Relative to other TSO-based methods often using deep learning [11] or improved TSO variants [13], TUNA-FS shows strong performance using standard classifiers, validating the effectiveness of the BTSO adaptation for FS.

The comprehensive experiments validate the effectiveness of the proposed TUNA-FS framework. Convergence analysis (Figure 2) showed robust optimization, guiding the selection of $t_{max} = 100$ and $NP = 30$ for balanced performance and efficiency. Sensitivity analyses (Tables II, III) confirmed the impact of the adaptive TF parameters and fitness weights, establishing optimal parameters for balancing accuracy, FAR, and sparsity. Performance evaluations (Figure 3) demonstrated clear benefits over using full feature sets, notably improving accuracy and reducing FAR on NSL-KDD, while efficiently reducing dimensionality on CIC-IDS2017. The SOTA comparison (Table VI) further established TUNA-FS's competitiveness, showcasing its ability to achieve a strong balance between high accuracy, low FAR, and significant feature reduction compared to diverse existing techniques.

VI. CONCLUSION

The present paper introduced TUNA-FS, a feature selection framework for intrusion detection, using a novel binary Tuna Swarm Bptimization algorithm. TUNA-FS effectively adapts the TSO algorithm for discrete feature selection tasks in IDS by incorporating an adaptive V-shaped TF and a multi-objective fitness function. Experimental validation on NSL-KDD and CIC-IDS2017 datasets confirmed robust convergence and identified optimal parameters. TUNA-FS significantly improves upon baseline performance, notably achieving 99.14% accuracy with only 9 features and 3.07% FP-Rate on NSL-KDD (using SVM), and 95.72% accuracy with 21 features and 7.78% FP-Rate on CIC-IDS2017. Comparisons with state-of-the-art methods established TUNA-FS as a competitive approach, offering a superior balance between accuracy, false alarm reduction, and dimensionality reduction. TUNA-FS provides an effective solution for enhancing IDS performance and efficiency.

In conclusion, the TUNA-FS framework offers a robust, effective, and well-balanced solution for feature selection in network intrusion detection. It successfully identifies compact, high-impact feature subsets that demonstrably enhance classifier performance and computational efficiency. While acknowledging limitations such as dataset dependency and the computational cost inherent in wrapper methods, the presented evidence strongly supports TUNA-FS as a valuable contribution to the network security field.

Future research directions could involve applying TUNA-FS to more recent and large-scale datasets, including those from IoT or industrial control system environments. Investigating online or incremental versions of TUNA-FS for adapting to evolving network traffic patterns would be highly relevant. Exploring hybridization strategies could yield further improvements, such as combining TUNA-FS with fast filter methods for initial dimensionality reduction or using ensemble FS techniques based on multiple TUNA-FS runs. Further investigation into different transfer function families or more sophisticated binarization rules within the BTSO context could also be fruitful. Finally, exploring the synergy between TUNA-FS and advanced deep learning architectures could further push the boundaries of intrusion detection performance.

REFERENCES

[1] S. Morgan, "2023 official cybercrime report," 2023. [Online]. Available: https://www.esentire.com/resources/library/2023-official-cybercrime-report

[2] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artificial Intelligence Review*, vol. 55, no. 1, pp. 453–563, 2022.

[3] S. Aljawarneh, M. Aldwairi, and M. B. Yassein, "Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model," *Journal of Computational Science*, vol. 25, pp. 152–160, 2018.

[4] A. Fatani, A. Dahou, M. Abd Elaziz, M. A. Al-Qaness, S. Lu, S. A. Alfadhli, and S. S. Alresheedi, "Enhancing intrusion detection systems for IoT and cloud environments using a growth optimizer algorithm and conventional neural networks," *Sensors*, vol. 23, no. 9, p. 4430, 2023.

[5] P. Pitre, A. Gandhi, V. Konde, R. Adhao, and V. Pachghare, "An intrusion detection system for zero-day attacks to reduce false positive rates," in *2022 International Conference for Advancement in Technology (ICONAT)*, 2022, pp. 1–6.

[6] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, p. 20, 2019.

[7] S. S. Kareem, R. R. Mostafa, F. A. Hashim, and H. M. El-Bakry, "An effective feature selection model using hybrid metaheuristic algorithms for iot intrusion detection," *Sensors (Basel, Switzerland)*, vol. 22, no. 4, p. 1396, 2022.

[8] N. K. Y. Gurukala and D. K. Verma, "Feature selection using particle swarm optimization and ensemble-based machine learning models for ransomware detection," *SN Computer Science*, vol. 5, no. 8, p. 1093, Nov. 2024.

[9] O. Almomani, "A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms," *Symmetry*, vol. 12, no. 6, p. 1046, 2020.

[10] L. Xie, T. Han, H. Zhou, Z.-R. Zhang, B. Han, and A. Tang, "Tuna swarm optimization: A novel swarm-based metaheuristic algorithm for global optimization," *Computational Intelligence and Neuroscience*, vol. 2021, pp. 1–22, 2021.

[11] G. Gowthami and S. S. Priscila, "Tuna swarm optimisation-based feature selection and deep multimodal-sequential-hierarchical progressive network for network intrusion detection approach," *International Journal of Critical Computer-Based Systems*, vol. 10, no. 4, pp. 355–374, 2023.

[12] S. S. Harwalkar, A. H. A. Hussein, B. V. Kumar, M. I. Habelalmateen, and R. M. Victoria, "Intrusion detection in iot platform using tuna swarm optimization with long short-term memory," in *International Conference on Ambient Intelligence, Knowledge Informatics and Industrial Electronics (AIKIIE)*, Nov. 2023, pp. 1–6.

TABLE VI
COMPARISON WITH STATE-OF-THE-ART FS METHODS FOR IDS.

Ref	Technique/Algo	Dataset(s)	Performance	# Feat.
[22]	Chi-Squared (ensemble)	NSL-KDD	Acc=99.23%, FPR=5.2%	22
[21]	Info. Gain (ensemble)	CIC-IDS2017 (20%)	Acc=93.4%, FPR=12%	19
[28]	IGRF-RFE (MLP)	UNSW-NB15	Acc=84.24%	23
[32]	B-PSO (DT)	Balanced dataset	Acc=99.52%	19
[9]	MI + Rules (J48)	UNSW-NB15	Acc=90.48%	30
[31]	Bin. PIO (DT)	Multi	TPR=97.1%, FPR=14.9%	5
[34]	ABC (AdaBoost)	NSL/ISCX	Acc=98.9%, DR=99.6%	15
[11]	TSO (DMS-HPN)	UNSW-NB15/ CIC- IDS2017	Acc=99.5(CIC)%, Acc=92.6(UNSW)%	18/22
[13]	ITSO (DT)	NSL-KDD	Acc=+4.1% vs TSO	-
This work	TUNA-FS (BTSO) (SVM)	NSL-KDD	Acc=99.14%, FPR=3.07%	9
		CIC-IDS2017	Acc=95.72%, FPR=7.78%	21

Note: Multi=Multiple Datasets Used, '-'=Not Available.

[13] Z. Qin, H. Xu, Y. Jin, and L. Huang, "Multi-strategy improved tuna swarm optimization algorithm for feature selection of network intrusion detection," in *3rd International Conference on Artificial Intelligence, Automation, and High-Performance Computing (AIAHPC)*, vol. 12717. Wuhan, China: SPIE, 2023, pp. 681–686.

[14] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Computational Cybernetics and Simulation 1997 IEEE International Conference on Systems , Man, and Cybernetics*, vol. 5, 1997, pp. 4104–4108 vol.5.

[15] C. Qiu, "A novel multi-swarm particle swarm optimization for feature selection," *Genetic Programming and Evolvable Machines*, vol. 20, no. 4, pp. 503–529, 2019.

[16] S.-S. Hong, E.-j. Lee, and H. Kim, "An advanced fitness function optimization algorithm for anomaly intrusion detection using feature selection," *Applied Sciences*, vol. 13, no. 8, p. 4958, 2023.

[17] M. Sharma and P. Kaur, "A comprehensive analysis of nature-inspired meta-heuristic techniques for feature selection problem," *Archives of Computational Methods in Engineering*, vol. 28, no. 3, pp. 1103–1127, 2021.

[18] L. Dhanabal and S. P. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International journal of advanced research in computer and communication engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[19] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116.

[20] J. Kittler, P. Pudil, and P. Somol, "Advances in statistical feature selection," in *Advances in Pattern Recognition — ICAPR 2001*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, vol. 2013, pp. 427–436.

[21] A. Mazighi, L. Ballihi, and G. Orhanou, "Improving intrusion detection using machine learning algorithms with feature selection based on information gain," in *IEEE International Conference on Advances in Data-Driven Analytics And Intelligent Systems (ADACIS)*, 2023, pp. 1–6.

[22] I. S. Thaseen, C. A. Kumar, and A. Ahmad, "Integrated intrusion detection model using chi-square feature selection and ensemble of Classifiers," *Arabian Journal for Science and Engineering*, vol. 44, no. 4, pp. 3357–3368, Apr. 2019.

[23] N. Thockchom, M. M. Singh, and U. Nandi, "A novel ensemble learning-based model for network intrusion detection," *Complex & Intelligent Systems*, vol. 9, no. 5, pp. 5693–5714, 2023.

[24] M. A. Rahman, S. Islam, Y. S. Nugroho, F. Y. Al Irsyadi, and M. J. Hossain, "An Exploratory Analysis of Feature Selection for Malware Detection with Simple Machine Learning Algorithms," *Journal of Communications Software and Systems*, vol. 19, no. 3, pp. 207–219, 2023.

[25] A. A. A. Mahmood, A. A. Hadi, and W. H. Al-Masoody, "Enhancing network security: A study on classification models for intrusion detection systems," *Journal of Communications Software and Systems*, vol. 21, no. 2, pp. 156–165, 2025.

[26] M. Samadi Bonab, A. Ghaffari, F. Soleimanian Gharehchopogh, and P. Alemi, "A wrapper-based feature selection for improving performance of intrusion detection systems," *International Journal of Communication Systems*, vol. 33, no. 12, p. e4434, 2020.

[27] J. Lee, D. Park, and C. Lee, "Feature selection algorithm for intrusions detection system using sequential forward search and random forest classifier," *KSII Transactions on Internet and Information Systems (TIIS)*, vol. 11, no. 10, pp. 5132–5148, 2017.

[28] Y. Yin, J. Jang-Jaccard, W. Xu, A. Singh, J. Zhu, F. Sabrina, and J. Kwak, "Igrf-rfe: A hybrid feature selection method for mlp-based network intrusion detection on unsw-nb15 dataset," *Journal of Big Data*, vol. 10, no. 1, p. 15, 2023.

[29] A. Karthick Kumar, K. Vadivukkarasi, R. Dayana, and P. Malarvezhi, "Botnet attacks detection using embedded feature selection methods for secure iomt environment," in *Pervasive Computing and Social Networking*, G. Ranganathan, R. Bestak, and X. Fernando, Eds. Singapore: Springer Nature, 2023, pp. 585–599.

[30] M. H. M. Yusof, M. R. Mokhtar, A. M. Zain, and C. Maple, "Embedded feature selection method for a network-level behavioural analysis detection model," *International Journal of Advanced Computer Science and Applications (ijacsa)*, vol. 9, no. 12, 2018.

[31] H. Alazzam, A. Sharieh, and K. E. Sabri, "A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer," *Expert systems with applications*, vol. 148, p. 113249, 2020.

[32] A. A. Saeed and N. G. M. Jameel, "Intelligent feature selection using particle swarm optimization algorithm with a decision tree for DDoS attack detection," *International Journal of Advances in Intelligent Informatics*, vol. 7, no. 1, pp. 37–48, 2021.

[33] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," *Computers & Security*, vol. 81, pp. 148–155, 2019.

[34] M. Mazini, B. Shirazi, and I. Mahdavi, "Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and adaboost algorithms," *Journal of King Saud University - Computer and Information Sciences*, vol. 31, no. 4, pp. 541–553, Oct. 2019.

[35] A. R. Jordehi, "Binary particle swarm optimisation with quadratic transfer function: A new binary optimisation algorithm for optimal scheduling of appliances in smart homes," *Applied Soft Computing*, vol. 78, pp. 465–480, 2019.

[36] M. M. Mafarja and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neurocomputing*, vol. 260, pp. 302–312, 2017.

[37] E. Emary, H. M. Zawbaa, C. Grosan, and A. E. Hassenian, "Feature subset selection approach by gray-wolf optimization," in *Afro-European Conference for Industrial Advancement*, A. Abraham, P. Krömer, and V. Snasel, Eds. Addis-Abeba, Éthiopia: Springer International Publishing, 2015, vol. 334, pp. 1–13.

[38] J. Lemus-Romani, B. Crawford, F. Cisternas-Caneo, R. Soto, and M. Becerra-Rozas, "Binarization of metaheuristics: Is the transfer function really important," *Biomimetics*, vol. 8, no. 5, p. 400, 2023.

[39] K. K. Ghosh, R. Guha, S. K. Bera, N. Kumar, and R. Sarkar, "S-shaped versus v-shaped transfer functions for binary manta ray foraging optimization in feature selection problem," *Neural Computing and Applications*, vol. 33, no. 17, pp. 11 027–11 041, Sep. 2021.

[40] N. Mohd Yusof, A. K. Muda, S. F. Pratama, and A. Abraham, "A novel nonlinear time-varying sigmoid transfer function in binary whale optimization algorithm for descriptors selection in drug classification," *Molecular Diversity*, vol. 27, no. 1, pp. 71–80, 2023.

[41] M. Mafarja, I. Aljarah, A. A. Heidari, H. Faris, P. Fournier-Viger, X. Li, and S. Mirjalili, "Binary dragonfly optimization for feature selection using time-varying transfer functions," *Knowledge-Based Systems*, vol. 161, pp. 185–204, 2018.

[42] M. J. Islam, X. Li, and Y. Mei, “A time-varying transfer function for balancing the exploration and exploitation ability of a binary pso,” *Applied Soft Computing*, vol. 59, pp. 182–196, 2017.
[43] N. Van Thieu and S. Mirjalili, “Mealpy: An open-source library for latest meta-heuristic algorithms in python,” *Journal of Systems Architecture*, vol. 139, p. 102871, 2023.

Abdelouaheb Khiar is currently pursuing his Ph.D. in Computer Science at the University of Oum El Bouaghi, Algeria, and conducting research at the ICOSI Laboratory, Abbes Laghrour University, Khenchela. He earned his Magister degree in 2011. Since 2012, he has served as an Assistant Professor at Abbes Laghrour University, where he contributes to research in artificial intelligence and cybersecurity. His work focuses on metaheuristic optimization, cybersecurity, and machine learning applications for network security.

Smaine Mazouzi received the M.S. and Ph.D. degrees in computer science from the University of Constantine, in 1996 and 2008, respectively. He is currently a professor in Université 20 Août 1955-Skikda, where he leads research at the LICUS Laboratory. His fields of interest are pattern recognition, machine vision, and computer security. His current research concerns using distributed and complex systems in image understanding and intrusion detection.

Rohallah Benaboud is a senior lecturer at the department of Mathematics and Computer Sciences - University of Oum El Bouaghi (Algeria). He obtained his PhD degree in Computer Science from University Abdelhamid Mehri Constantine 2, Algeria in 2016. He is currently a member of Distributed-Intelligent Systems Engineering (DISE) team at ReLa(CS)2 Laboratory - University of Oum El Bouaghi. He published many articles in many International Conferences and Journals. He supervises many Master and License students. His research interests include Internet of Things, Service Oriented Computing, Machine/Deep Learning and Multi-Agents Systems.

Hichem Haouassi received his Ph.D. degree in computer science from the University of Batna, Algeria in 2012, and now he is a full professor in Abbas Laghrour University, Khenchela, Algeria. His main research interests are the Artificial intelligence, Data mining, Methaheuristics and swarm-based optimization, Feature selection, and classification.