



ARTICLE

Metaheuristic-Based SDN Intrusion Detection Model Based on Adaptive Chinese Pangolin Optimizer Algorithm

Hui Xu, Yonglei Yang^{*} and Shuang Qu

School of Computer Science and Artificial Intelligence, Hubei University of Technology, Wuhan, China

^{*}Corresponding Author: Yonglei Yang, Email: 102401121@hbut.edu.cn

Received: 09 June 2026; Accepted: 04 August 2026; Published: 22 October 2026

ABSTRACT: With the widespread deployment of Software-Defined Networking (SDN), its centralized control architecture faces increasingly severe security threats despite improving network flexibility and programmability. Due to the high dimensionality, redundancy, and nonlinearity of SDN traffic data, existing intrusion detection methods often suffer from high computational cost, unstable feature selection, and limited generalization ability. To address these challenges, this paper proposes an SDN intrusion detection model based on an Adaptive Chinese Pangolin Optimizer (ACPO), termed ACPO-IDM. Unlike the original Chinese Pangolin Optimizer (CPO), ACPO integrates hierarchical initialization, adaptive inertia weight, elite local search, and precomputation-based caching strategies. These improvements enhance population diversity, balance global exploration and local exploitation, strengthen local refinement, and reduce repeated computational operations. Therefore, ACPO provides a more stable and lightweight feature selection mechanism for high-dimensional SDN traffic data. The selected compact feature subsets are then used to train intrusion detection classifiers. Experimental results on benchmark functions, UCI datasets, and the InSDN dataset show that ACPO achieves competitive or superior performance compared with several representative metaheuristic algorithms. Statistical tests further confirm the robustness of the proposed optimizer. On the full InSDN dataset, ACPO-IDM achieves 98.952% accuracy in binary classification and 98.653% accuracy in multiclass classification while selecting compact feature subsets. Additional comparisons with recent SDN-oriented and deep learning baselines, as well as multiple classifiers, further verify the effectiveness, robustness, and practical potential of the proposed framework.

KEYWORDS: Software-defined networking; intrusion detection; feature selection; metaheuristic algorithm; Chinese pangolin optimizer

1 Introduction

1.1 Research Background

As a novel network architecture, Software-Defined Networking (SDN) decouples the control plane from the data plane, improving network flexibility, programmability, and management efficiency [1]. However, this centralized architecture also introduces new security challenges [2]. Since the SDN controller is the core component of the network, its compromise may lead to network-wide failures and severe security threats [3]. Therefore, developing efficient Intrusion Detection Systems (IDSs) for SDN has become an important research topic [4].

In SDN intrusion detection, traffic data are typically high-dimensional and contain redundant features [5]. These features increase computational overhead, aggravate the curse of dimensionality, and reduce

detection performance. Moreover, redundant features may lead to unstable feature selection, affecting the consistency and generalization of both binary and multiclass detection tasks [6]. Therefore, selecting a compact and discriminative feature subset is essential for improving SDN intrusion detection performance [7].

1.2 Related Work

In recent years, feature selection has been widely studied to improve the efficiency and accuracy of intrusion detection systems. Traditional machine learning-based feature selection methods can reduce redundant features to some extent, but their search capability is often limited when dealing with high-dimensional and nonlinear traffic data [8]. Therefore, metaheuristic algorithms have attracted increasing attention because of their strong global search ability and flexibility in solving combinatorial optimization problems [9].

Many metaheuristic and hybrid optimization methods have been proposed for intrusion detection feature selection. For example, Basfar et al. [10] developed an enhanced intrusion detection method based on EMRMR feature selection. Wang et al. [11] proposed an ensemble framework combining improved grey wolf optimization with multi-level feature selection, while Alqahtany et al. [12] integrated enhanced grey wolf optimization with random forest for intrusion detection. Zhao et al. [13] introduced a hybrid metaheuristic dynamic optimization algorithm for network intrusion feature selection. In addition, several studies have explored multi-strategy optimization methods for intrusion detection, such as enhanced Marine Predator Algorithm [14], multi-strategy RIME optimization [15], and hybrid machine learning and metaheuristic optimization frameworks [16,17].

Deep learning-based IDS models have also been widely investigated. Cui et al. [18] developed a multiclass SDN intrusion detection system based on a hybrid LSTM model. Elsayed et al. [19] proposed a hybrid deep learning model for DDoS attack detection using consensus-based feature selection. Elsaid et al. [20] combined Grey Wolf Optimizer with GRU and LSTM networks for intrusion detection. da Silva Ruffo et al. [21] provided a systematic review of deep learning-based anomaly and intrusion detection in SDN. More recently, Li and Xiu [22] explored the use of large language models for feature selection, providing new ideas for automated feature engineering.

Recent studies have introduced multi-objective metaheuristic feature selection that jointly optimizes classification accuracy and subset size, along with adaptive and hybrid strategies to balance exploration and exploitation [23,24]. These advances underscore the need for an optimizer that achieves high detection performance while maintaining an effective trade-off between global search, local refinement, and feature reduction.

In addition, our previous studies have also explored several metaheuristic algorithms for feature selection and network traffic detection, demonstrating the effectiveness of optimization-based feature selection in improving detection efficiency and accuracy [25,26].

Although existing studies have achieved promising results, several challenges remain. Many metaheuristic algorithms still suffer from slow convergence and premature convergence in high-dimensional optimization problems [27], while deep learning-based IDS models often rely on large feature spaces and complex architectures, limiting lightweight deployment in SDN environments. In addition, comprehensive evaluation under realistic SDN scenarios, including full-dataset testing, minority attack analysis, statistical significance testing, and multi-classifier validation, is still limited. Although the recently proposed Chinese Pangolin Optimizer (CPO) [28] provides a promising optimization framework, it still requires further adaptation for SDN intrusion detection.

1.3 Contributions of This Paper

Motivated by the aforementioned challenges, the main contributions of this paper are summarized as follows:

- (1) An Adaptive Chinese Pangolin Optimizer (ACPO) is proposed by integrating hierarchical initialization, adaptive inertia weight, elite local search, and precomputation-based caching to improve convergence accuracy, search stability, and optimization efficiency.
- (2) An ACPO-based SDN intrusion detection model, termed ACPO-IDM, is constructed to perform feature selection and remove redundant traffic features while preserving discriminative information.
- (3) Comprehensive experiments, including benchmark optimization, UCI validation, full InSDN evaluation, statistical analysis, and comparison with recent methods, are conducted to comprehensively assess the effectiveness and robustness of the proposed framework.
- (4) Experimental results show that ACPO-IDM achieves competitive detection performance with compact feature subsets, demonstrating strong robustness, generalization ability, and potential for lightweight SDN intrusion detection.

The remainder of this paper is organized as follows. [Section 2](#) introduces the Chinese Pangolin Optimizer. [Section 3](#) presents the proposed improvement strategies. [Section 4](#) describes the ACPO-based SDN intrusion detection model. [Section 5](#) reports the experimental results and analysis. [Section 6](#) concludes this paper and discusses future research directions.

2 Chinese Pangolin Optimizer

CPO is a bio-inspired metaheuristic algorithm that mimics the hunting behavior of Chinese pangolins. Compared with conventional population-based optimization algorithms, its aroma-guided switching mechanism provides a more adaptive balance between global exploration and local exploitation, making it particularly suitable for SDN traffic feature selection. This balanced search strategy effectively explores the search space while refining promising candidate solutions, thereby improving the quality of the selected feature subsets. As a result, CPO can effectively remove redundant features while preserving discriminative traffic information, providing a solid foundation for accurate and lightweight SDN intrusion detection.

2.1 Aroma Diffusion Model

To describe aroma propagation in the environment, CPO employs a Gaussian-plume-based aroma diffusion model. The ground-level aroma concentration is defined as

$$M(x, 0, 0) = \frac{Q}{\pi u \sigma_y \sigma_z} \exp\left(-\frac{H^2}{2\sigma_z^2}\right) \quad (1)$$

where Q denotes the aroma source intensity, u is the average wind speed, H is the effective source height, and σ_y and σ_z are the horizontal and vertical diffusion parameters, respectively.

To constrain the aroma concentration within $[0, 1]$, normalization is performed as

$$C_M(t) = \frac{M(t) - \min(M)}{\max(M) - \min(M)} \quad (2)$$

The normalized aroma concentration $C_M(t)$ is used to guide behavioral selection during the optimization process.

2.2 Population Initialization

Similar to other population-based metaheuristic algorithms, CPO initializes a population of candidate solutions in the search space:

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{bmatrix} \quad (3)$$

where n and d denote the population size and problem dimension, respectively. The initial position of each individual is generated by

$$X_{ij} = \text{rand} \times (UB_j - LB_j) + LB_j, \quad i = 1, 2, \dots, n, \quad j = 1, 2, \dots, d \quad (4)$$

where UB_j and LB_j are the upper and lower bounds of the j th dimension, respectively.

2.3 Behavior Selection Mechanism

The CPO algorithm selects its search behavior according to the aroma concentration and a random factor:

$$\begin{cases} C_M \leq 0.2 \ \&\& \ r_1 \leq 0.5, & \text{luring behavior,} \\ \text{otherwise,} & \text{predation behavior.} \end{cases}$$

where r_1 is a random number in $[0, 1]$. This mechanism allows the algorithm to balance global exploration and local exploitation adaptively.

2.3.1 Luring Behavior

When $C_M \leq 0.2$ and $r_1 \leq 0.5$, the pangolin adopts luring behavior, which consists of two stages. In the *attraction and capture stage*, ants are attracted by the aroma and move toward the pangolin:

$$D_A(t) = |a \cdot X_A(t) - X_M(t)| \quad (5)$$

$$X_A(t+1) = X(t) + X_A(t) - A_1 \cdot D_A(t) \quad (6)$$

In the *movement and feeding stage*, the pangolin carries the prey toward the best position, and a Lévy flight term is introduced to enhance exploration:

$$D_M(t) = |C_1 \cdot D_A(t) - X_M(t)| + L_{levy} \cdot \left(1 - \frac{t}{T}\right) \quad (7)$$

$$X_M(t+1) = X(t) + X_M(t) - A_1 \cdot D_M(t) \quad (8)$$

2.3.2 Predation Behavior

Otherwise, the pangolin performs predation behavior, including three stages. In the *search and localization stage*, the pangolin searches for prey through random walking:

$$D_M(t) = |L_{levy} \cdot X_M(t) - X(t)| \quad (9)$$

$$X_M(t+1) = \sin(C_1 \cdot X(t) + A_1 \cdot |X_M(t) - L_{levy} \cdot D_M(t)|) \quad (10)$$

In the *rapid approach stage*, the pangolin follows the aroma trajectory to approach the prey:

$$D_M(t) = |a \cdot X_M(t) - X(t)| \quad (11)$$

$$X_M(t+1) = X(t) - A_1 \cdot |X_M(t) - \exp(-a) \cdot \sin(\text{rand} \cdot \pi) \cdot D_M(t)| \quad (12)$$

In the *digging and feeding stage*, local exploitation is strengthened near the target region:

$$D_M(t) = |C_1 \cdot X_M(t) - X(t)| \quad (13)$$

$$X(t+1) = X(t) + A_1 \cdot |X_M(t) - D_M(t)| \quad (14)$$

3 Improvement Strategies

Although CPO has competitive search capability, it may still suffer from uneven population initialization, slow convergence, and local optima in high-dimensional feature selection tasks. To address these limitations, four improvement strategies are introduced, including hierarchical initialization, adaptive inertia weight, elite local search, and precomputation-based caching.

3.1 Hierarchical Initialization Strategy

To address the problem that traditional random initialization may lead to an uneven population distribution, a hierarchical initialization strategy is introduced in this paper. This strategy systematically partitions the search space $[lb, ub]$ into L equal-width subregions, and initial solutions are independently generated within each subregion.

Assume that the population size is N , the problem dimension is D , and the number of layers is $L = 5$. The number of individuals assigned to each layer is given by

$$\text{layer_size} = \lceil N/L \rceil \quad (15)$$

The boundaries of the l th layer ($l = 1, 2, \dots, L$) are defined as

$$\text{layer_lb}_l = lb + (l-1) \cdot \frac{ub-lb}{L} \quad (16)$$

$$\text{layer_ub}_l = lb + l \cdot \frac{ub-lb}{L} \quad (17)$$

Within each sublayer, the initialization formula for the individual position is

$$X_{\text{index},j} = \text{layer_lb}_{l,j} + \xi \cdot (\text{layer_ub}_{l,j} - \text{layer_lb}_{l,j}) \quad (18)$$

where $\xi \sim U(0,1)$, and $\text{index} = (l-1) \times \text{layer_size} + i$. This strategy distributes the initial individuals across several predefined intervals, thereby reducing excessive concentration caused by purely random initialization and improving the diversity of the initial population.

3.2 Adaptive Inertia Weight Strategy

To better balance the global exploration and local exploitation capabilities of the algorithm, an adaptive inertia weight w is introduced into the position update equation. This weight decreases linearly from 0.9 with the increase of the iteration number t , and its update formula is given by

$$w = 0.9 - 0.5 \times \frac{t}{T_{\max}} \quad (19)$$

where $T_{\max}$ denotes the maximum number of iterations.

During each position update, the new position combines the candidate position generated by the standard CPO rule with the current position. As the weight decreases, the update gradually becomes more conservative. The new position is calculated as

$$X_{\text{new}} = w \times X_{\text{updated}} + (1 - w) \times X_{\text{current}} \quad (20)$$

where X_{current} is the current position and X_{updated} is generated by the standard CPO rule.

3.3 Elite Local Search Strategy

To enhance the ability of ACPO to escape from local optima, an elite local search strategy is performed around the current global best solution X_{best} . This strategy is triggered with a certain probability or at fixed iteration intervals, and imposes a Gaussian perturbation on the current best solution. The perturbation strength σ decreases adaptively with the iterative process:

$$\sigma = (ub - lb) \times \left(1 - \frac{t}{T_{\max}}\right) \times 0.05 \quad (21)$$

The local search position is generated by

$$X_{\text{local}} = X_{\text{best}} + \sigma \times N(0, 1) \quad (22)$$

where $N(0, 1)$ denotes a random number following the standard normal distribution. This strategy adopts a dual-trigger mechanism (periodic every 10 generations or probabilistic with 10% probability). It enhances fine-grained search around the best solution while preserving population diversity; if the perturbed solution is better, it replaces the original, thereby improving local exploitation and accuracy.

3.4 Precomputation and Caching Optimization Strategy

In the original algorithm, the Aroma_concentration function needs to be called in each iteration to calculate the aroma concentration factor C_m , which is computationally expensive. In the improved version, all C_m values required throughout the optimization process are precomputed before the iteration begins and stored in the array Cm_values_all. During the iterative process, the precomputed values are directly accessed through random indexing, thereby avoiding a large number of repeated calculations and effectively reducing practical computational overhead.

The individual strategies incorporated into ACPO are not claimed to be independently novel. The main contribution lies in coordinating hierarchical initialization, adaptive inertia weighting, elite local search, and caching within CPO's aroma-guided search framework. These components act at different stages of the optimization process to improve population distribution, search balance, local refinement, and computational efficiency. Therefore, ACPO is not a simple aggregation of existing techniques. The ablation results further confirm the complementary effects of these components. Compared with ICOA and ITROA, which modify different base optimizers, ACPO retains CPO's aroma-guided behavior-switching mechanism and coordinates the four components across initialization, position updating, local refinement, and computational acceleration. A detailed comparison is provided in [Table 1](#).

Table 1: Comparison of ACPO with recent SDN feature-selection and intrusion-detection methods.

Method	Mechanism	Dataset	Feat.	Classifier	Imbalance	Cost	Limitation
ACPO	CPO + HI + AIW + ELS + caching	InSDN subset and full dataset	3.2–7.3	KNN; SVM, RF, XGB, MLP	subset sampling; training-only oversampling	532.1 s; 752 MB	Offline wrapper cost
ICOA	Four improvement strategies on COA	InSDN, NSL-KDD, UNSW-NB15	4	KNN	NR	NR	Limited cost analysis
ITROA	Four improvement strategies on TROA	InSDN	5–6	KNN	NR	NR	Limited cost analysis
CBDNN	CNN-BiLSTM	InSDN and other IDS datasets	48	CNN-BiLSTM	NR	NR	Full features; complex model
GAN-IDS	GAN + chi-square selection	InSDN	48	XGB; NB, ET, RF	GAN-generated samples	NR	Complex training pipeline

3.5 Adaptive Chinese Pangolin Optimizer Algorithm

ACPO integrates hierarchical initialization, adaptive inertia weight, elite local search, and precomputation-based caching to improve population diversity, balance exploration and exploitation, enhance local refinement, and reduce computational overhead. The pseudo-code of ACPO is shown in Algorithm 1.

Algorithm 1: Pseudo-code of the ACPO algorithm

Input: Population size n , maximum iterations $T_{\max}$, bounds lb, ub , dimension dim , objective function $fobj(\cdot)$

Output: Best solution X^* , best fitness F^* , convergence curve

- 1: Initialize the population using the hierarchical initialization strategy in Eq. (18)
- 2: Precompute and cache the aroma concentration factors C_m
- 3: Evaluate the initial population and determine X^* and F^*
- 4: **for** $t = 1$ to $T_{\max}$ **do**
- 5: Update the adaptive inertia weight w using Eq. (19)
- 6: **for** $i = 1$ to n **do**
- 7: Generate a candidate position according to the standard CPO update rules
- 8: Refine the candidate position using Eq. (20)
- 9: Apply boundary control and evaluate the fitness
- 10: Update X^* and F^* if a better solution is found
- 11: **end for**
- 12: **if** $\text{mod}(t, 10) = 0$ or $\text{rand} < 0.1$ **then**

(Continued)

Algorithm 1 (continued)

```

13:     Perform elite local search around  $X^*$  using Eq. (22)
14:     Update  $X^*$  and  $F^*$  if the perturbed solution is better
15:   end if
16:   Record  $Convergence(t) = F^*$ 
17: end for
18: return  $X^*, F^*, Convergence$ 

```

3.6 Complexity Analysis and Scalability Discussion

To evaluate the computational efficiency of ACPO, let M denote the number of samples, d the original feature dimension, d_s the number of selected features, n the population size, $T_{\max}$ the maximum number of iterations, and C_f the cost of evaluating one candidate feature subset.

The hierarchical initialization requires $O(nd)$ time. In each iteration, ACPO updates all individuals and evaluates their fitness values, leading to a complexity of $O(n(d + C_f))$. Therefore, the overall time complexity can be expressed as

$$T_{ACPO} = O(nd + T_{\max} \cdot n \cdot (d + C_f)). \quad (23)$$

The precomputation and caching strategy stores aroma concentration factors before iteration, reducing repeated calculations during the search process. Although it does not change the dominant complexity order, it decreases practical runtime overhead. For the KNN-based fitness evaluation used in the main experiments, the fitness evaluation cost can be approximated as

$$C_f = O(M_{tr}M_{val}d_s), \quad (24)$$

where M_{tr} and M_{val} denote the numbers of training and validation samples, respectively. Thus, reducing d_s directly lowers the fitness evaluation and detection costs.

The memory complexity mainly consists of data storage, population storage, cached values, and the convergence curve:

$$S_{ACPO} = O(Md + nd + K + T_{\max}), \quad (25)$$

where K is the number of cached aroma concentration values. After feature selection, the classifier only uses d_s selected features, reducing the storage and inference burden from $O(Md)$ to $O(Md_s)$.

Overall, ACPO scales linearly with the population size, iteration number, and feature dimension, except for the classifier-dependent fitness evaluation term. Since candidate solutions are evaluated independently, the fitness evaluation process can be parallelized in large-scale SDN deployments. The full InSDN experiments further verify that ACPO maintains low training and prediction costs, supporting its scalability for larger SDN intrusion detection scenarios.

4 Proposed SDN Intrusion Detection Model

To address high-dimensional feature redundancy and unstable detection performance in SDN intrusion detection, this paper proposes an intrusion detection model based on the Adaptive Chinese Pangolin Optimizer, termed ACPO-IDM. As shown in Fig. 1, the model consists of four main stages: data preprocessing, feature selection, classifier training, and intrusion detection.

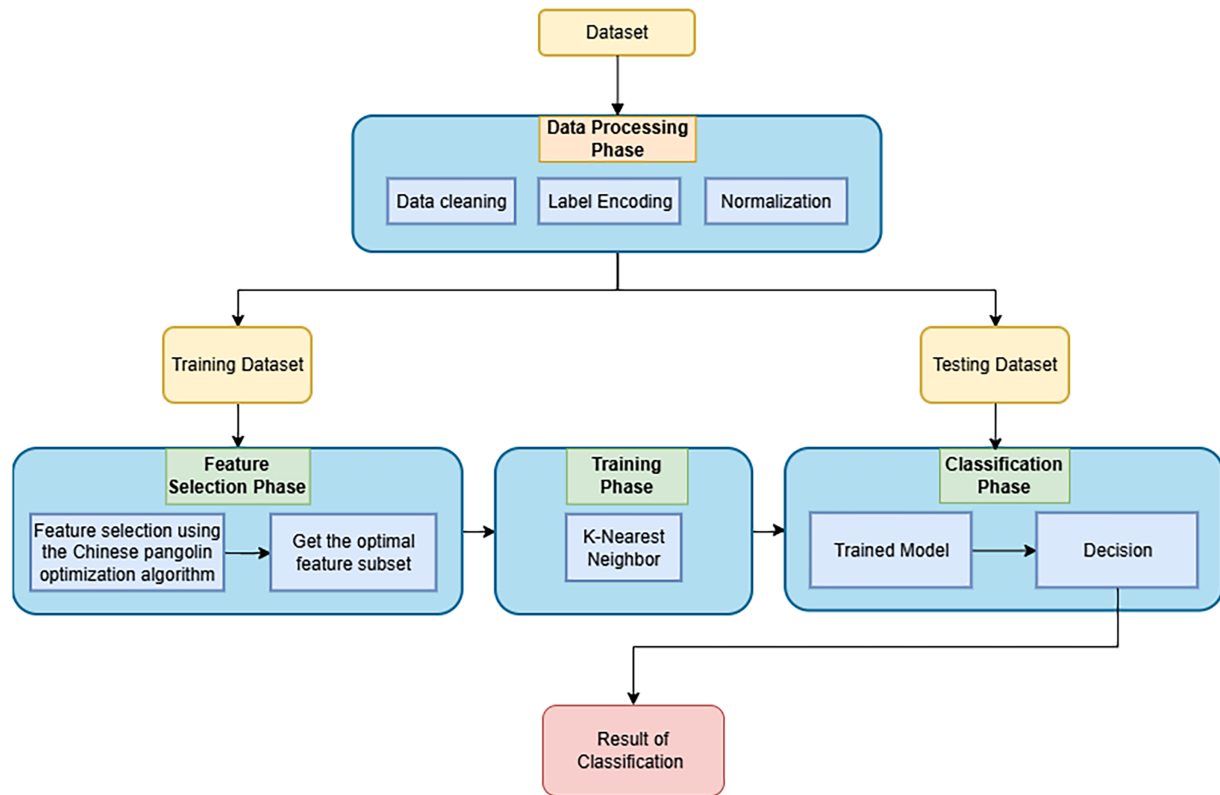


Figure 1: Proposed ACPO-IDM for SDN.

4.1 Data Preprocessing

The preprocessing stage aims to transform raw traffic data into a suitable format for model training. It mainly includes data cleaning, label encoding, data splitting, and normalization. Invalid, duplicate, or incomplete records are processed first, and categorical labels are then converted into numerical values. After data cleaning and label encoding, the dataset is divided into stratified training and testing sets at a ratio of 7:3. Normalization parameters are calculated from the training set only and then applied to the validation and testing sets.

4.2 Feature Selection

Feature selection is the core stage of ACPO-IDM. In this paper, the feature selection task is formulated as a combinatorial optimization problem, where each candidate solution represents a feature subset. ACPO is employed to remove redundant and weakly relevant features while preserving discriminative information. The detailed workflow of ACPO-based feature selection is shown in [Fig. 2](#).

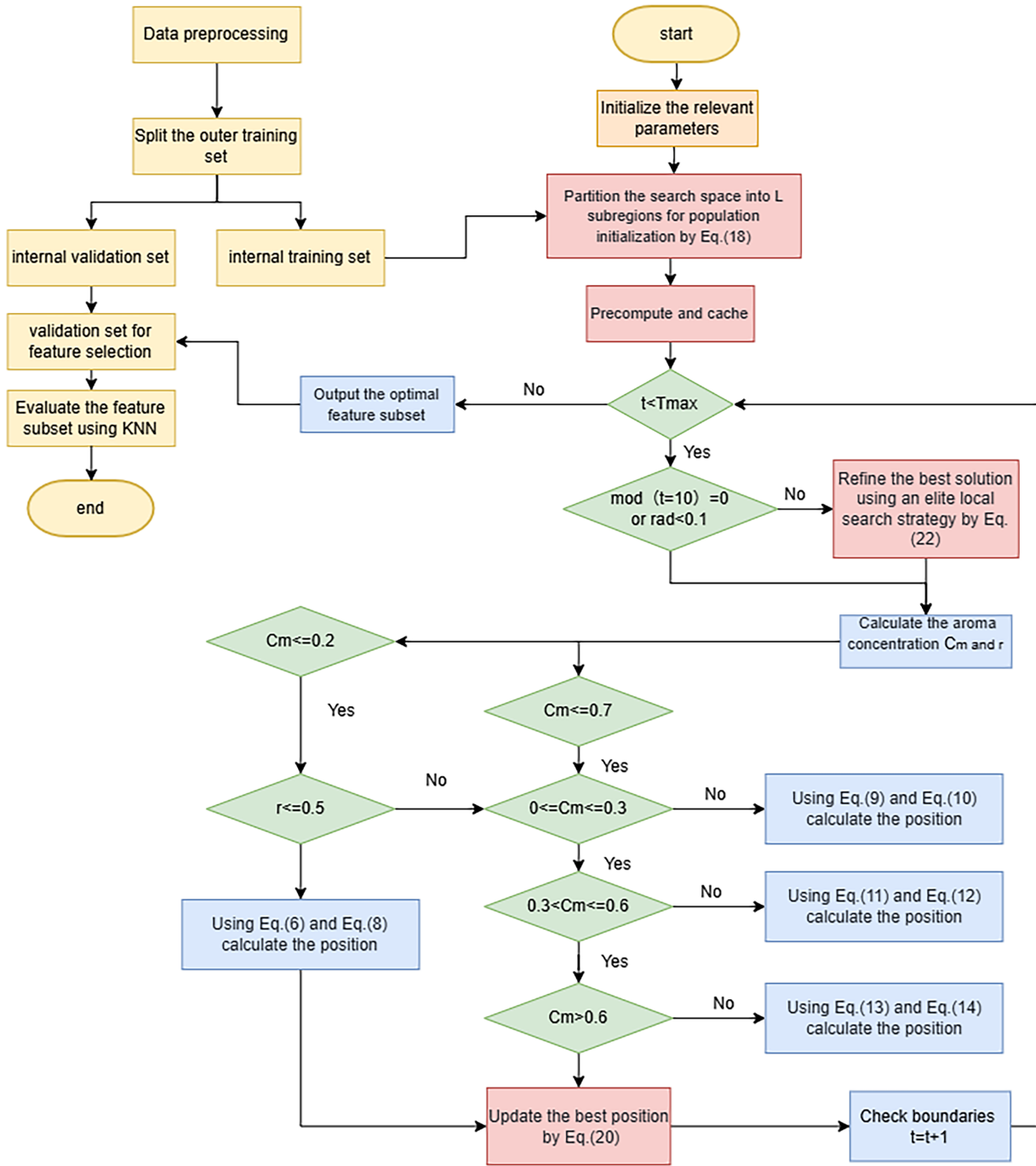


Figure 2: Feature selection flowchart of the ACPO algorithm.

Binary Encoding and Fitness Evaluation

Each candidate solution is represented by a continuous vector $\mathbf{x} = (x_1, x_2, \dots, x_d)$, where $x_j \in [0, 1]$ and d denotes the total number of candidate features.

$$b_j = \begin{cases} 1, & x_j \geq 0.5, \\ 0, & x_j < 0.5, \end{cases} \quad (26)$$

where $b_j = 1$ indicates that the j -th feature is selected, while $b_j = 0$ indicates that it is discarded. The feature-selection objective is defined as

$$Fitness = \alpha (1 - Acc_{val}(\mathbf{b})) + (1 - \alpha) \frac{N_s}{d}, \quad (27)$$

where Acc_{val} is the classification accuracy obtained on the internal validation set, N_s is the number of selected features, and $\alpha = 0.9$ controls the trade-off between classification performance and feature reduction. A candidate solution that selects no feature is assigned the worst fitness value.

The complete dataset is first divided into stratified outer training and testing sets at a ratio of 7:3. The outer training set is further split into internal training and validation subsets using the same ratio. Candidate feature subsets are evaluated only on the internal validation set, while the outer testing set is reserved for final evaluation. After feature selection, the final classifier is trained on the complete outer training set and tested on the outer testing set.

4.3 Training Stage

The selected feature subset is used to train the classifier. K-nearest neighbor (KNN) is adopted as the default classifier, while additional classifiers are evaluated to verify the generality of the selected features. Feature reduction improves representation quality and classification performance.

4.4 Classification and Detection

The trained classifier predicts the labels of testing samples and distinguishes normal traffic from different attack categories. The resulting predictions are used to evaluate the effectiveness of ACPO-IDM.

5 Experimental Results and Analysis

5.1 Experimental Setup

All experiments were conducted in MATLAB 2023b on an Intel Core i7-10875H CPU under Windows 10. The population size was set to 30. Unless otherwise stated, each algorithm was independently run 30 times. The maximum number of iterations was set to 1000 for benchmark tests and 50 for feature-selection experiments. Classification experiments used 30 stratified 7:3 train-test splits with different random seeds, and normalization parameters were estimated from the corresponding training set only. All reported results are averages over 30 runs.

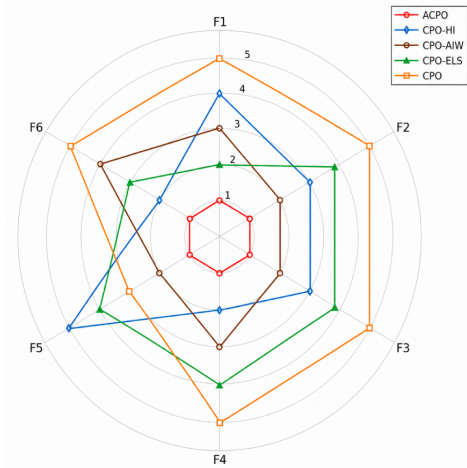
Statistical comparisons were performed using Wilcoxon signed-rank tests based on 30 runs, followed by Holm correction for multiple comparisons. For classification experiments using matched data splits and random seeds, paired Cohen's d_z was used to quantify effect size. Cohen's d was used for independent benchmark-run comparisons. The 95% confidence intervals were reported for the principal classification results. Friedman average ranking was used only in the ablation study.

5.2 Ablation Study on Key Components

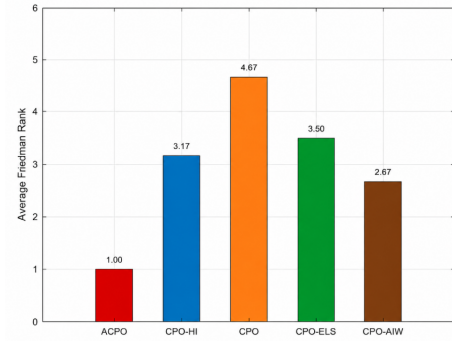
To evaluate the contribution of each component, three ablated variants were constructed by individually incorporating hierarchical initialization (HI), elite local search (ELS), and adaptive inertia weight (AIW) into the baseline CPO. All variants were tested on six benchmark functions under identical settings.

Fig. 3 shows the ranking results of the ablation variants, and Table 2 reports their mean fitness values. As can be seen, each component improves the baseline CPO on most benchmark functions, while the complete

ACPO achieves the best performance on all six benchmark functions, indicating that the three strategies are complementary.



(a) Radar ranking chart



(b) Friedman average rank

Figure 3: Ablation study results on benchmark functions.

Table 2: Numerical results of ablation variants on benchmark functions.

Variant	F1	F2	F3	F4	F5	F6
CPO	2.44×10^1	9.14×10^{-3}	7.27×10^{-4}	2.38×10^{-3}	3.51×10^{-10}	1.80×10^0
CPO-HI	1.86×10^1	3.62×10^{-3}	2.74×10^{-4}	2.08×10^{-4}	8.40×10^{-10}	8.37×10^{-2}
CPO-ELS	2.63×10^0	5.82×10^{-3}	5.11×10^{-4}	1.07×10^{-3}	5.20×10^{-10}	4.63×10^{-1}
CPO-AIW	8.74×10^0	1.02×10^{-3}	9.86×10^{-5}	5.46×10^{-4}	4.13×10^{-11}	9.26×10^{-1}
ACPO	5.64×10^{-3}	7.68×10^{-5}	2.35×10^{-7}	1.23×10^{-6}	3.51×10^{-12}	5.91×10^{-6}

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

The results show that HI improves population diversity, AIW balances exploration and exploitation, and ELS enhances local refinement. Caching mainly improves efficiency, reducing the runtimes of CPO-HI, CPO-ELS, and CPO-AIW from 15.42, 15.87, and 15.63 s to 12.71, 13.06, and 12.85 s, respectively. Overall, ACPO achieves the best performance, confirming the complementary effects of its components.

5.3 Benchmark Function Experiments

Six widely used benchmark functions with different characteristics were selected to evaluate the global exploration and local exploitation capabilities of ACPO. Their definitions are listed in Table 3.

ACPO was compared with five metaheuristic algorithms: Particle Swarm Optimization (PSO), Whale Optimization Algorithm (WOA), Chinese Pangolin Optimizer (CPO), Energy Valley Optimizer (EVO) [29], and Hybrid Metaheuristic Dynamic Optimization Algorithm (HMDOA). Under the same settings, each algorithm used a population size of 30 and was run 30 times for 1000 iterations on each benchmark function. Table 4 reports the mean and standard deviation, while Fig. 4 shows the convergence curves.

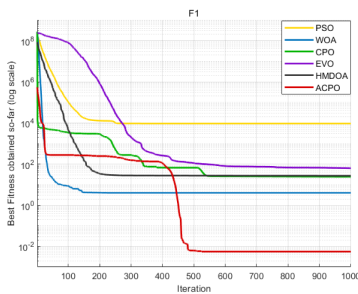
Table 3: Benchmark functions.

Function Expression	Dim	Range	Min
$F_1(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
$F_2(x) = \sum_{i=1}^n ix_i^4 + \varepsilon, \varepsilon \sim U(0, 1)$	30	$[-1.28, 1.28]$	0
$F_3(x) = \frac{\pi}{n} \{10 \sin^2(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})]\}$	30	$[-50, 50]$	0
$F_4(x) = 0.1 \{ \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + \sin^2(3\pi y_{i+1})] \}$	30	$[-50, 50]$	0
$F_5(x) = (\sum_{i=1}^n x_i) \exp(-\sum_{i=1}^n \sin(x_i^2))$	30	$[-2\pi, 2\pi]$	0
$F_6(x) = \sin^2(\pi w_1) + \sum_{i=1}^{n-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_{i+1} + 1)]$	30	$[-10, 10]$	0

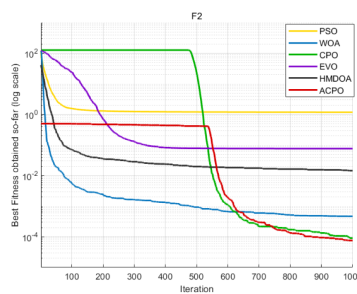
Table 4: Benchmark function results.

Function	Type	PSO	WOA	CPO	EVO	HMDOA	ACPO
F1	Avg	9.37×10^3	4.02×10^0	2.44×10^1	6.24×10^1	2.71×10^1	5.64×10^{-3}
	Std	2.74×10^4	9.82×10^0	8.27×10^0	4.99×10^1	1.56×10^1	6.11×10^{-3}
F2	Avg	1.19×10^0	4.71×10^{-4}	9.14×10^{-3}	7.70×10^{-2}	1.47×10^{-2}	7.68×10^{-5}
	Std	2.70×10^0	7.17×10^{-4}	8.34×10^{-3}	5.74×10^{-2}	9.09×10^{-3}	6.41×10^{-5}
F3	Avg	1.00×10^0	3.99×10^{-4}	7.27×10^{-4}	3.54×10^{-1}	2.06×10^{-4}	2.35×10^{-7}
	Std	1.27×10^0	4.13×10^{-4}	2.00×10^{-3}	5.92×10^{-1}	7.76×10^{-4}	2.12×10^{-7}
F4	Avg	9.14×10^{-2}	3.94×10^{-3}	2.38×10^{-3}	3.97×10^{-2}	1.02×10^{-3}	1.23×10^{-6}
	Std	7.97×10^{-2}	3.26×10^{-3}	3.52×10^{-3}	6.76×10^{-2}	1.43×10^{-3}	1.58×10^{-6}
F5	Avg	2.29×10^{-11}	3.53×10^{-10}	3.51×10^{-10}	2.86×10^{-11}	8.76×10^{-10}	3.51×10^{-12}
	Std	5.03×10^{-12}	5.59×10^{-14}	9.35×10^{-17}	8.70×10^{-13}	7.32×10^{-12}	4.43×10^{-19}
F6	Avg	3.23×10^0	3.55×10^{-3}	1.80×10^0	3.24×10^{-1}	3.62×10^{-1}	5.91×10^{-6}
	Std	5.15×10^{-1}	8.18×10^{-3}	9.60×10^{-1}	6.31×10^{-1}	5.87×10^{-1}	5.87×10^{-6}

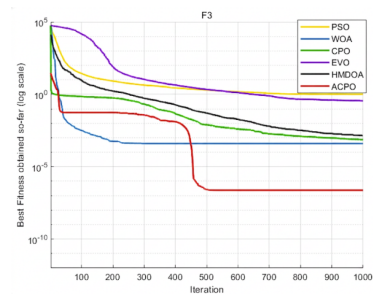
Note: Bold values indicate the optimal results. Tied optimal values are not bolded.



(a) F1



(b) F2



(c) F3

Figure 4: (Continued)

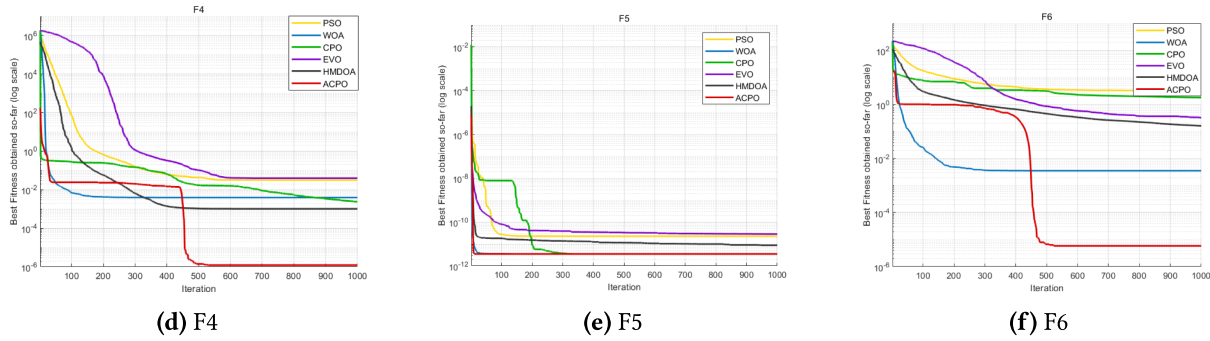


Figure 4: Convergence curves of fitness.

Overall, ACPO achieves the lowest mean fitness values on all six benchmark functions. The values reported in Table 5 are the original p -values obtained from the Wilcoxon signed-rank tests based on 30 independent runs. Holm correction was subsequently applied to the five pairwise comparisons within each benchmark function. After correction, all adjusted p -values remained below 0.05, with the largest adjusted p -value being 0.0132. The effect-size analysis using Cohen's d yielded values ranging from 0.58 to 4.32, corresponding to medium to very large effects. These results demonstrate that the improvements achieved by ACPO are both statistically significant and practically meaningful.

Table 5: Wilcoxon signed-rank test results on benchmark functions.

Algorithm	F1	F2	F3	F4	F5	F6
PSO	1.73×10^{-6}	1.73×10^{-6}	2.60×10^{-5}	4.73×10^{-6}	1.73×10^{-6}	1.32×10^{-2}
WOA	1.73×10^{-6}	1.59×10^{-3}	1.73×10^{-6}	1.73×10^{-6}	7.69×10^{-6}	1.73×10^{-6}
CPO	2.88×10^{-6}	5.30×10^{-3}	3.49×10^{-3}	2.41×10^{-3}	1.73×10^{-6}	2.35×10^{-6}
EVO	1.73×10^{-6}	1.73×10^{-6}	1.89×10^{-4}	2.60×10^{-5}	1.73×10^{-6}	1.73×10^{-6}
HMDOA	1.73×10^{-6}	1.73×10^{-6}	1.73×10^{-6}	1.48×10^{-3}	4.99×10^{-3}	3.93×10^{-5}

5.4 UCI Dataset Tests

To further evaluate the feature selection capability of ACPO, experiments were conducted on four representative UCI datasets. The dataset characteristics and corresponding experimental results are presented in Tables 6 and 7, respectively.

Table 6: UCI datasets.

No.	Dataset	Samples	Features
1	Ionosphere	351	34
2	WDBC	569	30
3	HeartStatlog	270	13
4	Sonar	208	60

Table 7: UCI dataset results.

Algorithm	Ionosphere				WDBC				HeartStatlog				Sonar			
	Acc	Rec	Prec	F1	Acc	Rec	Prec	F1	Acc	Rec	Prec	F1	Acc	Rec	Prec	F1
PSO	89.524	99.125	85.897	92.038	98.235	93.546	98.387	95.905	83.931	82.857	80.556	81.690	82.258	82.353	84.848	83.582
WOA	92.381	98.507	90.411	94.286	98.121	94.356	98.825	96.539	85.185	83.323	83.216	83.269	90.323	86.486	95.457	90.750
CPO	91.429	99.083	89.561	94.082	98.815	95.234	98.856	97.011	85.185	86.384	83.333	84.831	91.925	88.504	96.235	92.208
EVO	90.812	98.613	88.924	93.518	98.102	94.137	98.214	96.132	84.537	84.732	82.517	83.610	89.764	87.137	96.974	91.793
HMDOA	91.205	98.731	89.861	94.087	98.415	95.607	99.050	97.298	85.126	85.375	83.913	84.638	90.537	88.279	96.398	92.160
ACPO	93.373	98.871	90.541	94.523	98.824	96.825	98.314	97.564	86.424	85.714	83.361	84.521	91.935	88.889	96.053	92.332

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

For each matched run, the mean accuracy across the four UCI datasets was calculated, and the resulting 30 paired values were used for the statistical comparisons. Table 8 reports the Wilcoxon signed-rank test results for the UCI classification accuracy.

Table 8: Wilcoxon signed-rank test results for UCI accuracy.

Statistic	WOA	PSO	CPO	EVO	HMDOA
Raw p -value	2.17×10^{-4}	8.43×10^{-5}	3.91×10^{-4}	1.56×10^{-4}	6.72×10^{-4}
Holm-adjusted p -value	6.51×10^{-4}	4.22×10^{-4}	7.82×10^{-4}	6.24×10^{-4}	7.82×10^{-4}

Table 7 shows that ACPO achieved the highest accuracy on all four UCI datasets. Its mean accuracy was 92.639%, with a 95% confidence interval of [92.574%, 92.704%]. All Holm-adjusted p -values were below 0.001, while the paired Cohen's d_z values ranged from 0.42 to 1.87, indicating statistically significant and practically meaningful improvements.

5.5 InSDN Dataset

The InSDN dataset is a public benchmark designed for evaluating intrusion detection systems in SDN environments. It contains normal traffic and seven attack categories, including Denial of Service (DoS), Distributed Denial of Service (DDoS), Probe, Brute Force Attack (BFA), Web attack, Botnet (BOTNET), and User-to-Root (U2R). The complete dataset contains 343,889 records and exhibits a highly imbalanced class distribution, particularly for the rare Web, BOTNET, and U2R classes.

Two independent experimental settings were considered. The first used a sampled InSDN subset to evaluate ACPO under a reduced-scale scenario. The sampled subset was constructed using class-wise random sampling. Approximately 10% of the records from the Normal, DoS, DDoS, Probe, and BFA classes were randomly selected, whereas all records from the rare Web, BOTNET, and U2R classes were retained.

The second setting used the complete InSDN dataset containing all 343,889 original records to evaluate the scalability and robustness of the proposed method. No resampling was applied in the baseline full-dataset experiment, and all eight traffic categories were retained. An additional experiment using training-only random oversampling was conducted to evaluate the effect of class imbalance handling. Table 9 reports the complete class distributions and total sample sizes before and after subset sampling.

In the binary setting, normal traffic was labeled as 0, whereas all attack samples were labeled as 1. In the multiclass setting, all traffic categories were encoded as numerical labels from 0 to 7, corresponding to Normal, DoS, DDoS, Probe, BFA, Web, BOTNET, and U2R.

Table 9: Class distributions of the complete InSDN dataset and the sampled subset.

Dataset	Normal	DoS	DDoS	Probe	BFA	Web	BOTNET	U2R	Total
Complete InSDN dataset	68,424	53,616	121,942	98,129	1405	192	164	17	343,889
Approx. full-dataset test set	20,527	16,085	36,583	29,439	421	58	49	5	103,167
Sampled InSDN subset	6800	5300	12,000	9800	140	192	164	17	34,413

5.5.1 Binary Classification Results

Redundant and non-informative columns were removed before model training. To avoid data leakage, normalization parameters were calculated only from the training set and then applied to the testing set. The selected feature subsets obtained by different algorithms were evaluated using the same classifier and preprocessing procedure over the same sequence of 30 stratified 7:3 train–test splits, with matched random seeds across algorithms. [Tables 10](#) and [11](#) report the binary classification results on the InSDN subset and the full InSDN dataset, respectively. [Fig. 5](#) shows the corresponding convergence curves.

Table 10: Binary classification results on the sampled InSDN subset.

Method	Acc	Recall	Pre	F1	FPR	FNR	MCC	Feat	Train (s)	Pred (s)	Infer (ms)
ACPO	99.142	99.931	99.009	99.468	4.062	0.069	0.973	3.2	0.009	0.004	0.0004
WOA	98.444	99.170	98.894	99.032	4.504	0.830	0.951	4.2	0.011	0.005	0.0005
PSO	98.542	99.101	99.082	99.092	3.728	0.899	0.954	11.5	0.011	0.009	0.0009
CPO	98.556	99.862	98.363	99.107	6.747	0.138	0.954	7.3	0.011	0.010	0.0010
EVO	98.361	99.240	98.724	98.981	5.208	0.760	0.948	6.1	0.011	0.008	0.0008
HMDOA	99.053	99.343	99.476	99.410	2.125	0.657	0.970	6.7	0.010	0.007	0.0007

Note: FPR: False Positive Rate; FNR: False Negative Rate; MCC: Matthews Correlation Coefficient. Bold: best performance.

Table 11: Binary classification results on the full InSDN dataset.

Method	Acc	Recall	Pre	F1	FPR	FNR	MCC	Feat	Train (s)	Pred (s)	Infer (ms)
ACPO	98.952	99.802	98.900	99.349	4.470	0.198	0.967	3.4	0.255	0.113	0.0011
WOA	98.282	99.032	98.826	98.929	4.737	0.968	0.946	4.5	0.312	0.142	0.0014
PSO	98.389	98.972	99.016	98.994	3.958	1.028	0.949	11.9	0.318	0.538	0.0052
CPO	98.407	99.735	98.301	99.013	6.939	0.265	0.950	7.8	0.312	0.283	0.0027
EVO	98.202	99.112	98.650	98.880	5.462	0.888	0.943	6.4	0.318	0.227	0.0022
HMDOA	98.897	99.214	99.408	99.311	2.379	0.786	0.966	6.1	0.297	0.198	0.0019

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

To evaluate feature stability, selection frequencies were recorded over 30 independent runs. As shown in [Table 12](#), the top three features were selected in more than 90% of the runs, indicating stable feature-selection behavior. FlowDuration represents the duration of a network flow, FwdPktLenMean denotes the average forward-packet length, and FlowPkts_s measures the number of packets transmitted per second. These features characterize the temporal, packet-size, and traffic-rate properties of network flows.

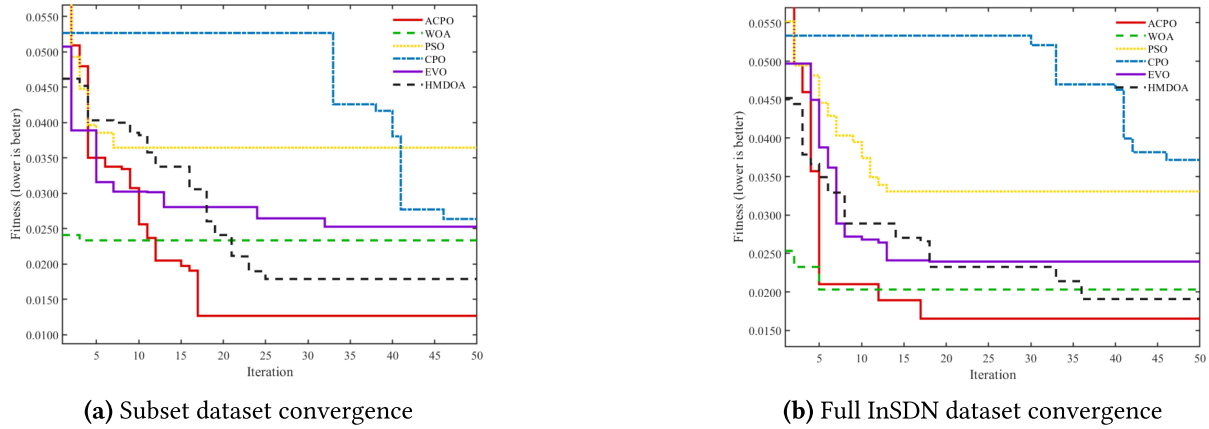


Figure 5: Convergence curves on subset and full InSDN datasets.

Table 12: Most frequently selected features in full-dataset binary classification.

Feature Name	Selection Frequency (%)
FlowDuration	96.7
FwdPktLenMean	96.7
FlowPkts_s	93.3

As shown in [Tables 10](#) and [11](#), ACPO achieved the best overall binary classification performance under both InSDN settings. On the full dataset, it obtained 98.95% accuracy, a 99.349% F1-score, and an MCC of 0.967 while selecting only 3.4 features. As reported in [Table 13](#), all Holm-adjusted p -values were below 0.001. The 95% confidence interval of the ACPO accuracy was [98.913%, 98.991%], and its comparison with HMDOA yielded a large effect size of approximately $d_z = 1.50$.

Table 13: Wilcoxon signed-rank test results for binary accuracy on the full InSDN dataset.

Statistic	WOA	PSO	CPO	EVO	HMDOA
Raw p -value	3.29×10^{-5}	9.57×10^{-6}	6.85×10^{-5}	2.40×10^{-5}	9.82×10^{-5}
Holm-adjusted p -value	9.87×10^{-5}	4.79×10^{-5}	1.37×10^{-4}	9.60×10^{-5}	1.37×10^{-4}

As shown in [Table 14](#), all algorithms used the same number of main-loop evaluations, ensuring a fair comparison. ACPO achieved the lowest optimization time (532.1 s), average fitness-evaluation time (0.324 s), and peak memory usage (752 MB), reducing runtime by approximately 3.8%–13.1% and memory consumption by 3.1%–11.0%. These results confirm the efficiency benefits of caching, compact feature subsets, and coordinated search.

Table 14: Computational cost comparison of all algorithms on the full InSDN dataset.

Metric	ACPO	PSO	WOA	CPO	EVO	HMDOA
Optimization time (s)	532.1	582.3	598.7	612.4	576.8	553.2
Average fitness-evaluation time (s/evaluation)	0.324	0.356	0.341	0.352	0.337	0.348
Main-loop population evaluations	1500	1500	1500	1500	1500	1500
Peak memory (MB)	752	812	834	845	798	776

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

5.5.2 Multiclass Classification Results

The multiclass experiments retained all eight traffic classes to evaluate ACPO under the highly imbalanced InSDN distribution. [Tables 15](#) and [16](#) report the class-wise results on the sampled subset and complete dataset, respectively. The class distributions and approximate test-set sample numbers are reported in [Table 9](#).

Table 15: Metrics summary for multiclass classification on the sampled InSDN subset.

Metric	Algo	Normal	DoS	DDoS	Probe	BFA	Web	BOTNET	U2R
Recall (%)	ACPO	97.437	98.949	99.985	98.072	91.886	73.684	100	66.667
	WOA	96.516	98.319	99.810	98.808	89.039	73.684	100	66.667
	PSO	96.982	98.914	99.995	97.665	91.530	73.684	100	66.667
	CPO	96.007	98.513	99.646	98.693	86.690	71.579	100	66.667
	EVO	97.010	98.774	99.959	97.940	91.388	73.684	100	66.667
	HMDOA	96.982	98.669	99.974	98.386	91.459	73.684	100	66.667
Precision (%)	ACPO	98.365	98.953	99.969	96.861	88.563	76.539	100	100
	WOA	97.562	98.287	99.345	95.548	85.998	74.145	99.412	93.333
	PSO	98.113	98.704	99.923	96.826	88.517	75.300	99.412	100
	CPO	97.364	97.542	98.713	94.315	85.266	69.334	98.824	100
	EVO	98.059	98.809	99.954	96.789	89.039	76.517	100	100
	HMDOA	98.059	98.846	99.944	96.769	88.991	77.357	100	93.333
F1-score (%)	ACPO	97.899	98.951	99.977	97.463	90.194	75.084	100	80.000
	WOA	97.036	98.303	99.577	97.151	87.492	73.914	99.705	77.778
	PSO	97.544	98.809	99.959	97.244	89.998	74.483	99.705	80.000
	CPO	96.681	98.025	99.177	96.454	85.972	70.439	99.409	80.000
	EVO	97.532	98.791	99.956	97.361	90.198	75.074	100	80.000
	HMDOA	97.518	98.757	99.959	97.571	90.208	75.476	100	77.778
Accuracy (%)	ACPO	98.581			5.7		0.009		0.1064
	WOA	98.439			9.4		0.010		0.1246
	PSO	98.372	Selected		12.7	Train	0.010	Pred.	0.1327
	CPO	98.257	features		10.2	time (s)	0.011	time (s)	0.1160
	EVO	98.421			8.5		0.010		0.1285
	HMDOA	98.532			12.1		0.011		0.1307

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

Table 16: Metrics summary for multiclass classification on the full InSDN dataset.

Metric	Algo	Normal	DoS	DDoS	Probe	BFA	Web	BOTNET	U2R
Recall (%)	ACPO	97.656	99.102	99.991	97.587	92.026	73.684	100	66.667
	WOA	96.422	98.256	99.781	98.855	89.154	73.684	100	66.667
	PSO	97.053	98.859	99.995	97.001	91.452	73.684	100	66.667
	CPO	96.135	98.411	99.593	97.847	86.558	71.579	100	66.667
	EVO	97.083	98.705	99.921	97.351	91.304	73.684	100	66.667
	HMDOA	97.052	98.586	99.941	97.717	91.384	73.684	100	66.667
Precision (%)	ACPO	98.557	99.101	99.983	97.051	88.807	76.539	100	100
	WOA	97.481	98.217	99.302	95.454	85.851	74.145	99.412	93.333
	PSO	98.187	98.661	99.894	96.754	88.435	75.300	99.412	100
	CPO	97.451	97.488	98.671	94.239	85.158	69.334	98.824	100
	EVO	98.121	98.759	99.927	96.728	89.057	76.571	100	100
	HMDOA	98.107	98.782	99.924	96.708	88.951	77.357	100	93.333
F1-score (%)	ACPO	98.104	99.101	99.987	97.318	90.388	75.084	100	80.000
	WOA	96.949	98.236	99.541	97.125	87.471	73.914	99.705	77.778
	PSO	97.617	98.760	99.944	96.877	89.918	74.483	99.705	80.000
	CPO	96.789	97.947	99.130	96.009	85.852	70.439	99.409	80.000
	EVO	97.599	98.732	99.924	97.039	90.167	75.100	100	80.000
	HMDOA	97.577	98.684	99.932	97.210	90.151	75.476	100	77.778
Accuracy (%)	ACPO	98.653			7.3		0.2553		3.0147
	WOA	98.551			12.7		0.2834		3.5307
	PSO	98.327		Selected	14.5	Train	0.2833	Pred.	3.7586
	CPO	98.152		features	11.6	time (s)	0.3127	time (s)	3.2861
	EVO	98.382			9.2		0.2833		3.6411
	HMDOA	98.469			12.5		0.3127		3.7043

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

For the complete dataset, an additional imbalance-aware experiment was conducted. After each stratified split, random oversampling was applied only to the training set, while the test set retained its original distribution to avoid data leakage. The Web, BOTNET, and U2R training samples were oversampled to the size of the BFA training class. Performance was evaluated using class-wise metrics, macro-F1, balanced accuracy, and 95% confidence intervals over 30 independent runs. Table 17 presents the Wilcoxon signed-rank test results for multiclass accuracy.

As shown in Table 16, ACPO achieved the highest overall accuracy on the full InSDN dataset while selecting the smallest feature subset. It also obtained competitive recall, precision, and F1-scores across the eight traffic classes.

Table 17: Wilcoxon test results for full InSDN multiclass accuracy.

Statistic	WOA	PSO	CPO	EVO	HMDOA
Raw p -value	1.56×10^{-5}	6.25×10^{-5}	7.80×10^{-4}	1.09×10^{-3}	6.25×10^{-4}
Holm-adjusted p -value	7.80×10^{-5}	2.50×10^{-4}	1.88×10^{-4}	1.88×10^{-3}	1.88×10^{-3}

Training-only oversampling increased the macro-F1 from 92.498% to 93.671% and the balanced accuracy from 90.839% to 93.685%, as shown in Table 18. Table 19 and Fig. 6 further show improved recognition of Web and U2R traffic, while BOTNET maintained stable performance. Because each test split contained only approximately five U2R samples, its confidence interval remained relatively wide, and the corresponding result should be interpreted cautiously. All Holm-adjusted Wilcoxon p -values were below 0.01, while the paired Cohen's d_z values ranged from 0.35 to 2.10, confirming both statistical significance and practical relevance.

Table 18: Imbalance-aware multiclass results on the full InSDN dataset.

Setting	Macro-F1 (%)	Balanced Acc. (%)
Original ACPO-KNN	92.498	90.839
Training-only oversampling + ACPO-KNN	93.671	93.685

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

Table 19: Minority-class results with training-only oversampling.

Class	Test Samples	Recall (%)	F1-Score (%)	95% CI of F1
Web	58	82.800	79.738	[76.210, 83.270]
BOTNET	49	100.000	100.000	[98.500, 100.000]
U2R	5	80.000	86.073	[74.800, 97.340]

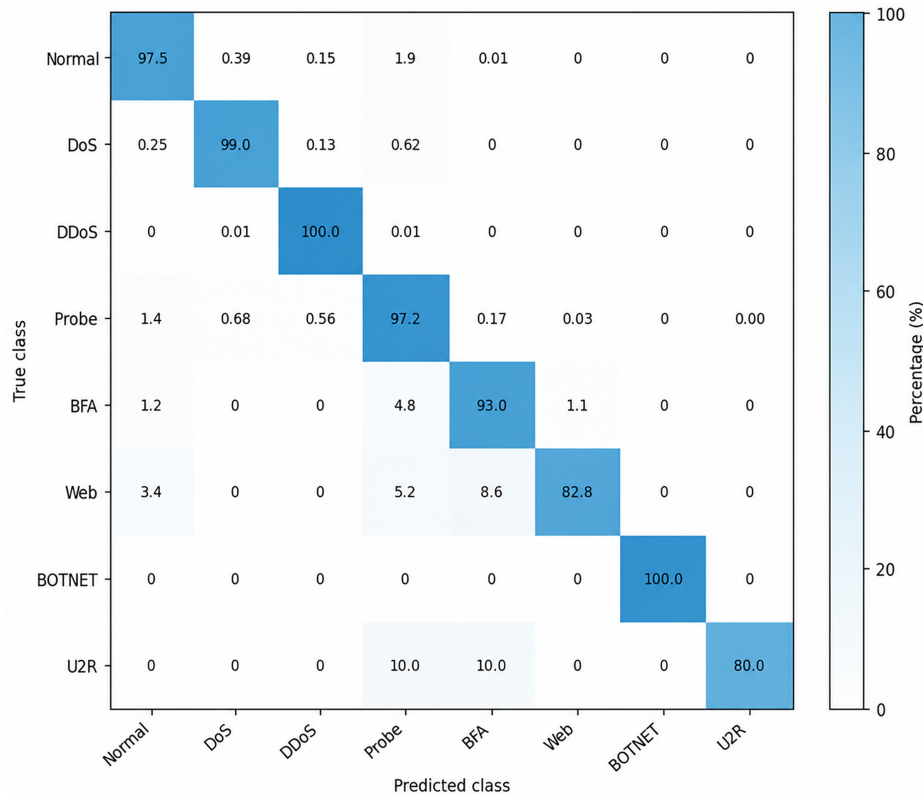


Figure 6: Row-normalized confusion matrix of ACPO-KNN on the full InSDN dataset.

5.6 Comparison with State-of-the-Art SDN-Oriented and Deep Learning Baselines

To further evaluate ACPO, it was compared with two recent SDN-oriented metaheuristic feature-selection methods, ICOA and ITROA, and two deep-learning IDS methods, CBDNN and GAN-IDS [30–33].

However, complete source codes, identical data partitions, preprocessing procedures, and computational settings were not consistently available. Therefore, these results are treated as literature-level references rather than strictly controlled reimplementations. Unavailable information is denoted by “NR”, and computational costs across different hardware environments are not directly compared. Table 1 summarizes the methodological characteristics reported for the five methods.

Table 1 shows that ACPO provides a compact feature subset while jointly improving initialization, search balance, local refinement, and computational efficiency. The external methods are included as literature-level references because identical implementations and experimental settings are unavailable.

As shown in Table 20, ACPO achieves competitive binary classification performance while selecting fewer features than the recent SDN-oriented feature-selection methods. Although CBDNN and GAN-IDS report slightly higher accuracy, they use the full feature space and more complex deep-learning architectures.

Table 20: Reported binary classification results on the InSDN dataset.

Method	Acc	Recall	Pre	F1	FPR	FNR	MCC	Feat
ACPO	99.142	99.931	99.009	99.468	4.062	0.069	0.973	3.2
ICOA	98.900	99.000	98.802	98.901	–	–	–	4
ITROA	99.37	99.35	99.67	99.51	–	–	–	5
CBDNN	99.98	99.98	99.98	99.98	–	–	–	48
GAN-IDS	99.95	99.91	99.99	99.95	–	–	–	48

As shown in Table 21, ACPO achieves higher overall accuracy than ITROA with a slightly smaller feature subset, although ITROA performs better on several minority classes.

Table 21: Reported multiclass classification results on the InSDN dataset.

Metric	Algo	Normal	DoS	DDoS	Probe	BFA	Web	BOTNET	U2R
Recall (%)	ACPO	97.437	98.949	99.985	98.072	91.886	73.684	100	66.667
	ITROA	99.30	93.53	99.60	96.34	92.95	63.64	100	80.0
Precision (%)	ACPO	98.365	98.953	99.969	96.861	88.563	76.539	100	100
	ITROA	98.46	92.29	99.53	98.64	95.01	74.47	90.74	80.0
F1-score (%)	ACPO	97.899	98.951	99.977	97.463	90.194	75.084	100	80.0
	ITROA	98.88	92.29	99.34	97.48	94.97	68.63	95.15	80.0
Accuracy (%)	ACPO	98.581	Selected	5.7					
	ITROA	96.73	features	6					

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

5.7 Evaluation with Multiple Classifiers

To evaluate the classifier-independence of the proposed ACPO-based feature selection method, four additional classifiers, including Support Vector Machine (SVM), Random Forest (RF), eXtreme Gradient

Boosting (XGBoost), and multilayer perceptron (MLP), were employed besides K-nearest neighbor (KNN). All classifiers were evaluated using the same ACPO-selected feature subset, preprocessing procedure, training–testing split, and random seeds to ensure a fair comparison.

The hyperparameters were selected through preliminary validation using the training data and were fixed for all runs. SVM used an RBF kernel with $C = 1$, and RF used 100 trees. XGBoost used 100 estimators, a learning rate of 0.1, a maximum depth of 6, and subsample and column-sampling ratios of 1.0. The MLP contained two hidden layers with 64 and 32 neurons, used ReLU activation and the Adam optimizer, and was trained for at most 200 epochs with a batch size of 256. Training stopped if the validation loss did not improve for 10 consecutive epochs. The settings with the highest validation F1-score were selected, without using the test set.

Table 22 compares the binary classification performance of five classifiers using the ACPO-selected features on the full InSDN dataset. KNN, SVM, RF, and XGBoost achieved accuracies above 98.6%. KNN obtained the best overall performance, with 98.952% accuracy, a 99.349% F1-score, and an MCC of 0.967, while XGBoost achieved comparable results. Although MLP showed lower performance, the results indicate that the selected features can effectively support multiple downstream classifiers.

Table 22: Binary classifier comparison on the full InSDN dataset.

Classifier	Acc	Recall	Pre	F1	FPR	FNR	MCC	Train (s)	Pred (s)	Infer (ms)
KNN	98.952	99.802	98.900	99.349	4.470	0.198	0.967	0.255	0.113	0.0011
SVM	98.650	99.385	98.940	99.162	4.356	0.615	0.957	114.6	0.368	0.0036
RF	98.800	99.723	98.795	99.257	4.979	0.277	0.962	18.43	2.046	0.0198
XGBoost	98.928	99.793	98.883	99.336	4.611	0.207	0.966	5.351	0.243	0.0024
MLP	88.994	99.122	88.775	93.594	52.447	0.878	0.603	127.4	0.093	0.0009

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

Table 23 compares the corresponding multiclass classification results. KNN, RF, and XGBoost achieved accuracies above 98%. KNN obtained the highest accuracy and F1-score, reaching 98.653% and 98.658%, respectively, whereas XGBoost achieved the highest MCC of 0.974. These results further demonstrate that the ACPO-selected features maintain strong discriminative capability across different classification models.

Table 23: Multiclass classifier comparison on the full InSDN dataset.

Classifier	Accuracy	Weighted Recall	Weighted Precision	Weighted F1	MCC	Train (s)	Pred (s)
KNN	98.653	98.653	98.666	98.658	0.969	0.2553	3.015
SVM	95.686	95.686	95.946	95.536	0.929	1775.2	5.462
RF	98.045	98.045	98.095	97.942	0.968	32.57	4.810
XGBoost	98.414	98.414	98.386	98.383	0.974	40.19	1.556
MLP	95.301	95.301	95.348	94.973	0.923	392.7	0.113

Note: Bold values indicate the optimal results. Tied optimal values are not bolded.

6 Conclusion

In this paper, ACPO and an ACPO-based intrusion detection model, termed ACPO-IDM, were proposed for SDN environments to address feature redundancy and unstable detection performance. By incorporating multiple improvement strategies, ACPO effectively enhanced the convergence speed,

optimization accuracy, and stability of the original CPO algorithm. Experimental results on benchmark functions, UCI datasets, and the InSDN dataset demonstrated that the proposed method achieved competitive or superior performance in both feature selection and intrusion detection tasks. In particular, ACPO-IDM showed strong effectiveness in both binary and multiclass classification while reducing feature dimensionality and maintaining high detection accuracy. Despite these promising results, several limitations should be acknowledged. First, although full InSDN experiments were conducted, the evaluation is still based on public benchmark datasets, and the generalization ability of ACPO-IDM against unseen attacks requires further validation. Second, the feature selection process is mainly performed offline; therefore, in dynamic SDN environments, the selected feature subset may need to be periodically updated to adapt to traffic distribution changes. Third, although the compact feature subset reduces inference cost, the optimization process still introduces additional computational overhead during the training stage. In future work, we will investigate online and incremental feature selection strategies, evaluate ACPO-IDM in real SDN controller environments, and further improve its adaptability to large-scale and evolving network traffic.

Acknowledgement: None.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Study conception and design: Hui Xu, Yonglei Yang and Shuang Qu; data collection: Yonglei Yang; analysis and interpretation of results: Yonglei Yang and Hui Xu; draft manuscript preparation: Hui Xu, Yonglei Yang and Shuang Qu. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The dataset used in this study is openly accessible and reliable. It can be obtained from the following website: <https://www.kaggle.com/datasets/badcodebuilder/insdn-dataset> (accessed on 1 January 2025). The four UCI datasets (Ionosphere, WDBC, Heart Statlog, and Sonar) are publicly available at UCI Machine Learning Repository.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Kreutz D, Ramos FMV, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: a comprehensive survey. *Proc IEEE*. 2015;103(1):14–76. doi:10.1109/jproc.2014.2371999.
2. Ahmed DS, Abdulhameed AA, Gaata MT. A systematic literature review on cyber attack detection in software-defined networking (SDN). *Mesopotamian J CyberSecurity*. 2024;4(3):86–135. doi:10.58496/mjcs/2024/018.
3. Phan TV, Nguyen TG, Dao NN, Huong TT, Thanh NH, Bauschert T. DeepGuard: efficient anomaly detection in SDN with fine-grained traffic flow monitoring. *IEEE Trans Netw Serv Manag*. 2020;17(3):1349–62. doi:10.1109/tnsm.2020.3004415.
4. Mustafa Z, Amin R, Aldabbas H, Ahmed N. Intrusion detection systems for software-defined networks: a comprehensive study on machine learning-based techniques. *Clust Comput*. 2024;27(7):9635–61. doi:10.1007/s10586-024-04430-6.
5. Singh P, Kaur D. Multilayered SDN security with MAC authentication and GAN-based intrusion detection. *PLoS One*. 2025;20(9):e0331470. doi:10.1371/journal.pone.0331470.
6. Gaurav A, Gupta BB. Enhancing the security of SDN in 5G: a hybrid feature selection based ensemble machine learning framework for classification of cyber-attacks. *SN Comput Sci*. 2025;6(3):233. doi:10.1007/s42979-025-03771-2.
7. Eldhai AM, Hamdan M, Abdelaziz A, Hashem IAT, Babiker SF, Marsono MN, et al. Improved feature selection and stream traffic classification based on machine learning in software-defined networks. *IEEE Access*. 2024;12(2):34141–59. doi:10.1109/access.2024.3370435.

8. Gennaro A, Di Pietro R. Feature selection in ML-based SDN intrusion detection system. In: Proceedings of the 2024 11th International Conference on Future Internet of Things and Cloud (FiCloud); 2024 Aug 19–21; Vienna, Austria. p. 152–9. doi:10.1109/ficloud62933.2024.00031.
9. Xu H, Bai L, Huang W. An optimization-inspired intrusion detection model for software-defined networking. *Electron Res Arch.* 2025;33(1):1–24. doi:10.3934/era.2025012.
10. Basfar R, Dahab MY, Ali AM, Eassa F, Bajunaied K. Enhanced intrusion detection in software-defined networking using advanced feature selection: the EMRMR approach. *Eng Technol Appl Sci Res.* 2024;14(6):19001–8. doi:10.48084/etasr.9256.
11. Wang J, Zhang L, Yang W, Chen Y, Liu S, Wang T. An ensemble framework with improved grey wolf optimization algorithm and multi-level feature selection for IoT intrusion detection. *Clust Comput.* 2025;28(12):776. doi:10.1007/s10586-025-05374-1.
12. Alqahtany SS, Shaikh A, Alqazzaz A. Enhanced grey wolf optimization (EGWO) and random forest based mechanism for intrusion detection in IoT networks. *Sci Rep.* 2025;15(1):1–19. doi:10.1038/s41598-024-81147-x.
13. Zhao X, Sun T, Zhang L, Li H, Chen M, Wang P. Feature selection method for network intrusion based on hybrid meta-heuristic dynamic optimization algorithm. *Comput Secur.* 2025;156(10):104512. doi:10.1016/j.cose.2025.104512.
14. Wang Z, Zhang Y, Yu J, Gao Y, Zhao G, Houssein EH, et al. Multi-strategy enhanced marine predator algorithm: performance investigation and application in intrusion detection. *J Big Data.* 2025;12(1):38. doi:10.1186/s40537-025-01080-2.
15. Wang L, Xu J, Jia L, Wang T, Xu Y, Liu X. Multi-strategy RIME optimization algorithm for feature selection of network intrusion detection. *Comput Secur.* 2025;153(10):104393. doi:10.1016/j.cose.2025.104393.
16. Thakkar A, Lohiya R. Machine learning and metaheuristic optimization algorithms for feature selection and botnet attack detection. *Knowl Inf Syst.* 2025;67(4):1–30. doi:10.1007/s10115-024-02322-0.
17. Kumar S, Sharma V. A hybrid machine learning approach for feature selection in designing intrusion detection systems (IDS) model for distributed computing networks. *J Supercomput.* 2025;81(1):254. doi:10.1007/s11227-024-06677-7.
18. Cui M, Chen J, Qiu X, Lv W, Qin H, Zhang X. Multi-class intrusion detection system in SDN based on hybrid BiLSTM model. *Clust Comput.* 2024;27(7):9937–56. doi:10.1007/s10586-024-04477-5.
19. Elsayed MS, Le-Khac N-A, Jurcut AD. A hybrid deep learning model with consensus-based feature selection for DDoS attacks detection in SDN. *Procedia Comput Sci.* 2025;252(9):643–52. doi:10.1016/j.procs.2025.01.024.
20. Elsaid SA, Shehab E, Mattar AM, Azar AT, Hameed IA. Hybrid intrusion detection models based on GWO optimized deep learning. *Discov Appl Sci.* 2024;6(10):1–34. doi:10.1007/s42452-024-06209-1.
21. da Silva Ruffo VG, Lent DMB, Komarchesqui M, Schiavon VF, de Assis MVO, Carvalho LF, et al. Anomaly and intrusion detection using deep learning for software-defined networks: a survey. *Expert Syst Appl.* 2024;255(5):124982. doi:10.1016/j.eswa.2024.124982.
22. Li J, Xiu X. LLM4FS: leveraging large language models for feature selection and how to improve it. *arXiv:2503.24157.* 2025. doi:10.48550/arXiv.2503.24157.
23. Han C, Jia H, Li W, Li Y, Shi Q. BGR-FS: a bandit-guided redundancy-aware framework for high-dimensional multi-objective feature selection. *Complex Intell Syst.* 2026;38(6):433. doi:10.1007/s44443-026-00848-7.
24. Li H, Zhang Q. An adaptive initialization and multitasking based evolutionary algorithm for bi-objective feature selection in classification. *Complex Intell Syst.* 2025;11(7):310. doi:10.1007/s40747-025-01941-6.
25. Wu Q, Xu H, Liu M. Applying an improved dung beetle optimizer algorithm to network traffic identification. *Comput Mater Contin.* 2024;78(3):4091–107. doi:10.32604/cmc.2024.048461.
26. Xu H, Hu Y, Cao W, Han L. An improved jump spider optimization for network traffic identification feature selection. *Comput Mater Contin.* 2023;76(3):3239–55. doi:10.32604/cmc.2023.039227.
27. Zhang M, Chen H, Heidari AA, Chen Y, Wu Z, Cai Z, et al. WHHO: enhanced harris hawks optimizer for feature selection in high-dimensional data. *Clust Comput.* 2025;28(3):186. doi:10.1007/s10586-024-04770-3.
28. Guo Z, Liu G, Jiang F. Chinese pangolin optimizer: a novel bio-inspired metaheuristic for solving optimization problems. *J Supercomput.* 2025;81(4):517. doi:10.1007/s11227-025-07004-4.

29. Azizi M, Aickelin U, Khorshidi HA, Baghalzadeh Shishehgarkhaneh M. Energy valley optimizer: a novel meta-heuristic algorithm for global and engineering optimization. *Sci Rep.* 2023;13(1):226. doi:10.1038/s41598-022-27344-y.
30. Xu H, Huang W, Bai L. AI-integrated feature selection of intrusion detection for both SDN and traditional network architectures using an improved crayfish optimization algorithm. *Comput Mater Contin.* 2025;84(2):3053–73. doi:10.32604/cmc.2025.064930.
31. Xu H, Chen J, Hu Z. Metaheuristic-driven abnormal traffic detection model for SDN based on improved tyranosaurus optimization algorithm. *Comput Mater Contin.* 2025;83(3):4495–513. doi:10.32604/cmc.2025.062189.
32. Jia Y, Ren J, He H, Li X, Xu K. CBDNN: an efficient and generalized intrusion detection method in SDN environments. *Clust Comput.* 2026;29(4):263. doi:10.1007/s10586-026-06083-z.
33. Shamim SM, Kodera Y, Nogami Y. GAN-based IDS system for imbalanced datasets in SDN environments with offline and real-time traffic analysis. *IEEE Access.* 2025;13:190786–806. doi:10.1109/ACCESS.2025.3629710.