

Dynamic Ant Colony Optimization–Driven Feature Selection for GRU-Based Network Intrusion Detection

Zhaosheng Yang

Department of Information Engineering, Shijiazhuang College of Applied Technology,
Shijiazhuang City, Hebei Province, 050800, China

Abstract—Modern network environments generate large-scale, high-dimensional traffic data that can be effectively interpreted as complex temporal signals. Efficiently analyzing such data requires robust feature selection and accurate temporal modeling techniques. In this study, we propose a signal-driven intrusion detection framework that integrates a Dynamic Ant Colony Optimization (DyACO) algorithm with deep recurrent neural modeling. The proposed DyACO performs adaptive wrapper-based feature selection by dynamically adjusting pheromone evaporation, heuristic guidance, and reinforcement strategies to enhance exploration in high-dimensional signal spaces. Selected features are then processed using a gated recurrent unit network to capture temporal dependencies in network traffic signals. The experiments on the CSE-CIC-IDS2018, UNSW-NB15, and NSL-KDD datasets demonstrate that our proposed method not only effectively compresses the feature space but also enhances classification accuracy, precision, recall, and F-score. The results demonstrate that when applied to network traffic, structured temporal signals, along with adaptive optimization modeling, yield an effective solution for intelligent, signal-based anomaly detection.

Keywords—Ant colony optimization; feature selection; network intrusion detection; deep learning

I. INTRODUCTION

Nowadays, networked environments produce enormous amounts of diverse traffic data, complicating intrusion detection [1]. Owing to the increasing volume and complexity of cyberattacks, Intrusion Detection Systems (IDSs) face the difficulty of processing highly-dimensional spaces rich in redundant, noisy, and correlated attributes [2]. These factors create challenging classification tasks, high computational requirements, and rising false-positive rates. Similar challenges have also been observed in other security-sensitive domains, where machine learning-based detection approaches require effective feature selection, preprocessing strategies, and imbalance handling mechanisms to improve detection reliability [3]. Moreover, multi-class distributions and class imbalance in standard IDS benchmark sets can further degrade performance because detection results are unbalanced across attack classes. This demonstrates the limited performance of the standard architecture of IDSs under increasing data volume [4].

Feature Selection (FS) is an important technique for improving IDS reliability in high-dimensional network traffic data. Feature Selection is a process of choosing the most relevant features of a dataset [5]. FS enhances efficiency and

generalization of the model by discarding the redundant features in the input data [6]. Wrapper-based FS provides an easy way to evaluate classifiers using the subset of features with respect to the original features of the dataset, and it also achieves higher classification performance when compared to filter-based methods [7]. The removal of irrelevant features not only enhances the accuracy of classification models but also reduces their complexity when these models are applied in real time for intrusion detection in large-scale network environments [8].

Despite widespread use, existing wrapper-based and metaheuristic FS approaches exhibit notable shortcomings. Algorithms like GA, PSO, ACO, and CSA, and many of their variants, often suffer from premature convergence, causing the search to get stuck in local optima. Their performance worsens as problem dimensionality increases. In addition, many metaheuristics keep control parameters constant during optimization. The wrapper technique is precise but increases computational time, as it must repeatedly score each possible subset against a classifier.

To address these challenges, it is important to apply optimization algorithms that can adapt their search strategies during execution. Dynamic optimization approaches show that algorithms should detect environmental changes and adjust their search behavior by exploiting previous search knowledge to improve convergence performance [9]. In high-dimensional IDS problems involving FS, the search landscape will be rough and Sparse. The approach needs to adjust dynamically to expand the search region when diversity is required and to shift toward intensified exploitation as the algorithm converges toward the optimal solution. The approach will fail when the algorithms employed have constant parameters, which often results in poor performance as data sets grow.

The Dynamic Ant Colony Optimization (DyACO) algorithm is an adaptive extension of feature selection techniques based on the ACO algorithm, which overcomes the limitations of static search mechanisms (i.e., fixed pheromone evaporation rates, predefined heuristic information, and unchanged transition strategies) in high-dimensional IDS environments. Traditional ACO-based FS techniques usually have fixed pheromone evaporation rates, static heuristic information, and stationary transition rules throughout the entire optimization process. In DyACO, among other mechanisms, heuristic information is dynamic, pheromone evaporation rates vary with iterations, pheromone updates are based on reward and punishment

strategies, and state transition rules are adaptive. Such methods help the algorithm achieve a better balance between exploration and exploitation, prevent pheromone trail stagnation, and improve its capability to discover more compact and discriminative feature subsets. The adaptive parameter control strategy allows DyACO to change its search patterns as the optimization process proceeds, making the feature selection framework less static for high-dimensional IDS datasets, where avoiding local optima and selecting informative attributes are crucial.

Although DyACO was originally designed for continuous optimization tasks, adapting it into a binary framework presents new opportunities for IDS FS. Binary DyACO enables ants to construct inclusion–exclusion vectors representing candidate feature subsets, allowing dynamic search mechanisms to operate in a discrete domain. Embedding this binary DyACO within a wrapper architecture further enhances its utility by enabling direct feedback from classification performance during optimization. Such integration leverages both DyACO’s adaptive search strengths and the classifier’s discriminative capabilities, resulting in a more refined and practical approach to selecting high-impact features for network intrusion detection.

Based on the aforementioned interests, this work proposes a DyACO-based wrapper FS approach, named DyACO. This algorithm aims to optimize IDSs. DyACO stands out for the following reasons: 1) the proposed algorithm uses an adaptive pheromone function, heuristic function, and transition function for the binary variant of DyACO; 2) the algorithm employs the wrapper approach, combining DyACO for the search process with the CART evaluation function and the GRU classifier for the classifier selection process; 3) the algorithm outperforms current methods in terms of the solutions’ compactness, speed of convergence, and detection capabilities on the CIC-IDS2018, UNSW-NB15, and NSL-KDD datasets.

II. LITERATURE REVIEW

Shwaysh, et al. [10] introduced a hybrid IDS model that combines information gain with an autoencoder for dimensionality reduction and an ensemble of gradient-boosted classifiers. Zouhri, et al. [11] proposed a comprehensive comparison of univariate and multivariate filter-based FS methods across a wide array of IDS datasets. Barbosa, et al. [12] developed a filtering technique based on graph modeling of Pareto-dominant, informative, and correlated features. This technique discovers informative features using betweenness centrality on a feature-constructed graph, yielding a compact, discriminative subset with high accuracy using only a few features. Habeeb and Babu [13] proposed a two-level FS method that combines coarse correlation filtering with hybridization of the Whale Optimization Algorithm (WOA) and Genetic Algorithm (GA).

Maseno and Wang [14] proposed a sequential wrapper approach that combines an optimized extreme learning machine with SVM classification. Umer, et al. [15] suggested a wrapper-based FS pipeline integrated with a Multi-Head Attention transformer classifier. Anagha, et al. [16] proposed a framework

for jointly incorporating SHAP values, correlation analysis, and information gain to select relevant features in deep learning-based IDS classification tasks. Al-Yaseen, et al. [17] proposed a wrapper method that applies Differential Evolution (DE) for FS and Extreme Learning Machine (ELM) for subset evaluation. Halim, et al. [18] suggested an enhanced GA-based FS strategy with tuned parameters and a novel fitness function to preserve essential information while minimizing dimensionality.

Nazir and Khan [19] introduced a wrapper approach using a Tabu Search (TS) algorithm coupled with an RF process to evaluate feature subsets. Pourardebil Khah, et al. [20] proposed a hybrid multi-phase approach that incorporates heuristic filters, fuzzy TOPSIS, and a Discrete Gray Wolf Optimizer (DGWO) for metaheuristic refinement. Hosseini, et al. [21] proposed a hybrid optimizer for OSMOGA that couples the Slime Mold Algorithm (SMA) with GA, using opposition-based learning to improve GA population diversity. This optimizer provides efficient exploration, along with the exploitation strengths of GA and SMA, ensuring accuracy and accelerating learning within GA for the IDS problem. Laamari and Kamel [22] suggested a multi-objective binary bat algorithm within a wrapper FS framework using Decision Trees (DTs) for evaluation.

Although these studies have demonstrated the effectiveness of feature selection and optimization techniques for IDS, their limitations remain evident. Filter-based approaches generally evaluate features independently and may ignore classifier-related dependencies, while wrapper-based methods provide better classification feedback at the cost of higher computational complexity. Furthermore, optimization-based approaches using GA, PSO, DE, and other metaheuristic algorithms can improve feature selection performance; however, they may suffer from premature convergence, limited adaptability, and difficulty in exploring high-dimensional feature spaces. Hybrid optimization strategies can enhance search capability, but they often introduce additional parameters and computational complexity.

III. BACKGROUND AND PRELIMINARIES

A. Gated Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are commonly used to process temporal data; vanilla RNNs may suffer from unstable gradients and loss of long-term dependencies. Gated Recurrent Units (GRUs) were introduced to address the pitfalls of vanilla RNNs, using an innovative gating mechanism to control information flow over time. As shown in Fig. 1, the main difference between GRUs and LSTMs is that GRUs use both an update gate and a reset gate, which reduces the total number of parameters and helps prevent overfitting. GRUs are an excellent candidate for intrusion detection, given the inherent temporal dependencies in network traffic data and the need for efficient real-time processing. A GRU cell receives an input vector x_t at time step t and combines it with the previous internal state h_{t-1} . The first gating component controls how much past information to preserve. Its activation is computed using a sigmoid-based transformation as expressed in Eq. (1).

$$\gamma_t^{(u)} = \sigma(W_u[h_{t-1}, x_t]) \quad (1)$$

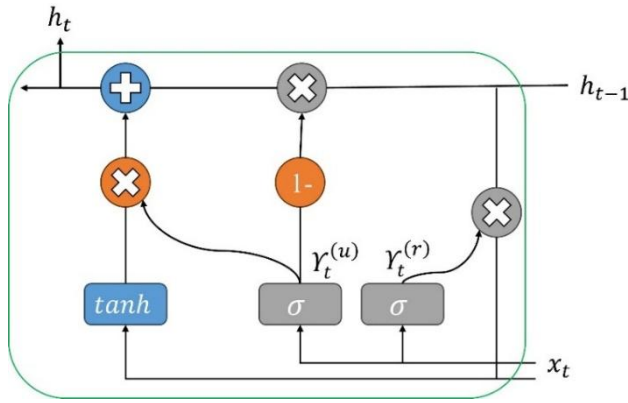


Fig. 1. Internal computational structure of a GRU cell.

The second gate, the reset controller, regulates how much previously forgotten hidden content is used to produce a new candidate state. This mechanism allows the model to selectively ignore irrelevant past information, promoting the sensitivity needed for network attack behaviors with abrupt changes in behavior sequences. Eq. (2) defines the reset gate.

$$Y_t^{(r)} = \sigma(W_r[h_{t-1}, x_t]) \quad (2)$$

Using the reset-modulated hidden state, the GRU computes a candidate activation that represents a potential new memory value, incorporating both fresh input and selectively filtered historical data. This candidate is generated via a nonlinear transformation, as expressed in Eq. (3).

$$\hat{h}_t = \tanh(W_c[Y_t^{(r)} \odot h_{t-1}, x_t]) \quad (3)$$

The final hidden state at a given time step is a function of an interpolation between the preserved old hidden state and the candidate hidden state. This function is modulated by the update gate, which can be viewed as a control gate that enables the cell to maintain a smooth flow of features rather than introduce novelty, as described by Eq. (4).

$$h_t = (1 - Y_t^{(u)}) \odot h_{t-1} + Y_t^{(u)} \odot \hat{h}_t \quad (4)$$

B. Decision Tree Classifier

DTs are hierarchical machine learning algorithms that mimic a human analyst's decision-making process by iteratively dividing a dataset into increasingly homogeneous groups. A DT consists of a root node that is initialized with the dataset, which is later split into branches based on a series of decision statements applied to the values of specific features, one by one. A split within a decision tree occurs on a particular attribute, and the terminal nodes contain the predicted classes, known as leaves.

Several variants of DT algorithms have been developed, including ID3 and C4.5, which employ multiway splits and entropy-based impurity measures. However, these variants can produce overly large structures with deep branching patterns, adding unnecessary complexity. In contrast, the Classification and Regression Tree (CART) methodology uses a binary division strategy, simplifying the tree's architecture and reducing training overhead. CART uses a bottom-up impurity-minimization approach that repeatedly selects the attribute that

best separates classes at each node. CART measures split quality using Gini impurity, which assesses the heterogeneity of the class distribution within a subset. For a dataset D consisting of K classes, the impurity is computed using Eq. (5).

$$\text{Gini}(D) = 1 - \sum_{k=1}^K \left(\frac{|C_k|}{|D|} \right)^2 \quad (5)$$

where, C_k denotes the collection of samples belonging to class k . A smaller impurity value indicates a more uniform class distribution within the node, thereby signaling a purer partition. When an attribute A is used to divide the dataset into subsets $D_1, D_2, \dots, D_n$, the overall impurity after splitting is obtained as a weighted combination of the individual impurities, as formulated in Eq. (6).

$$\text{Gini}(D | A) = \sum_{i=1}^n \left(\frac{|D_i|}{|D|} \right) \text{Gini}(D_i) \quad (6)$$

C. Wrapper Fitness Evaluation

In wrapper-based FS, the search process relies on a scoring mechanism that reflects how well a candidate subset supports classification accuracy while controlling model complexity. Unlike filter approaches, which use statistical proxies, wrapper strategies directly measure a classifier's predictive performance for each subset under evaluation. This lets the optimization algorithm favor feature sets with better real-world detection performance. Because selecting feature sets with very large subset sizes can lead to overfitting and high computational costs, the fitness function must simultaneously weigh predictive accuracy and dimensionality reduction. To formalize this balance, the optimizer assesses each candidate solution using a composite metric that integrates both classification precision and the proportion of selected features. Let P_{cls} represent the accuracy delivered by the classifier (in this case, CART), and let F_{sel} and F_{tot} denote the number of chosen features and the total available attributes, respectively. The resulting fitness score is minimized and is defined in Eq. (7).

$$\text{Fitness} = \min \left[\alpha \cdot \frac{1}{P_{\text{cls}}} + \beta \cdot \frac{F_{\text{sel}}}{F_{\text{tot}}} \right] \quad (7)$$

D. Data Preprocessing Pipeline

Intrusion detection datasets typically contain heterogeneous attributes, missing entries, duplicated flows, and large numerical disparities. If not handled systematically, these irregularities can significantly distort classifier behavior. Therefore, a structured preprocessing workflow is essential to ensure that the learning algorithms receive stable, unbiased, and properly scaled data representations. In this study, a three-stage pipeline comprising data cleansing, categorical encoding, and feature normalization is adopted to prepare all traffic samples for evaluation and optimization.

The first stage focuses on cleaning the raw dataset. Common issues include incomplete records, inconsistent formats, and repeated observations introduced during collection. Eliminating null entries and duplicates ensures that the subsequent optimization process does not overfit to noisy or redundant

patterns. This step is particularly important for intrusion detection since replicated benign flows or malformed attack records can bias the classifier, leading to artificially inflated accuracy or unstable convergence in deep models.

After cleaning, the second stage transforms categorical attributes. Many IDS datasets contain symbolic fields describing protocol categories, service types, connection states, or flag indicators. To incorporate these attributes into numerical learning systems, a translation mechanism is required. In this work, label encoding is applied, assigning a unique integer index to each category.

The third stage addresses feature scaling, a crucial step for stabilizing gradient-based optimization and reducing the influence of attributes with disproportionately large ranges. Large eigenvalue discrepancies can cause erratic cost-surface behavior, producing significant fluctuations during training and slowing model convergence. To mitigate this issue, a min–max normalization scheme is used, mapping each feature value v to a bounded range $[0,1]$. The transformation is computed according to Eq. (8).

$$v_{norm} = \frac{v - v_{min}}{v_{max} - v_{min}} \quad (8)$$

IV. PROPOSED METHOD

The proposed DyACO framework extends classical ACO by integrating dynamic adaptation mechanisms tailored for high-dimensional FS in network intrusion detection. Fig. 2 provides a detailed procedural flow of the DyACO optimization process, beginning with parameter initialization and progressing through iterative solution construction, pheromone updating, and termination. In conventional ACO, pheromone trails, heuristic values, and state-transition rules remain fixed across iterations, often leading to stagnation or premature convergence when searching large binary spaces. To address these limitations, DyACO incorporates four intertwined dynamic mechanisms: adaptive heuristics, iteration-aware pheromone volatility, reward–penalty reinforcement, and parameter-adjusted state transitions.

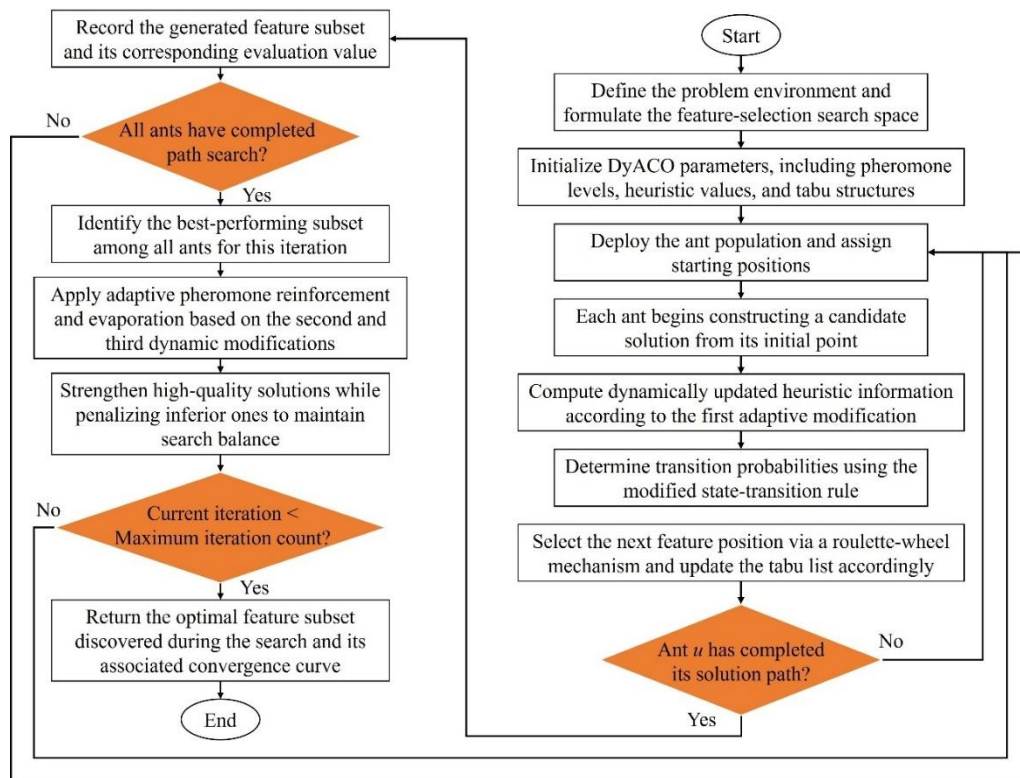


Fig. 2. Flowchart of the DyACO framework.

A. Mathematical Model of Traditional ACO

Traditional ACO simulates cooperative behavior observed in real ant colonies. Artificial ants traverse a solution graph, a graph where selecting or discarding each feature corresponds to choosing one of two nodes, while depositing virtual pheromones that bias future ant decisions. This behavior is characterized by two fundamental processes: the transition probability, which determines how ants choose their next binary decision, and pheromone revision, which regulates the colony's memory. When constructing a feature subset, each ant begins with an

empty solution and progressively decides whether to include or exclude each feature. For a given ant u at decision point f , the chance of selecting option $o \in \{0,1\}$ (exclude/include) is determined by the classical transition model denoted by Eq. (9).

$$\Pi_{f,o}^u = \begin{cases} \frac{\Lambda_{f,o}(t)^{\alpha_o} \cdot \Gamma_{f,o}(t)^{\beta_o}}{\sum_{o' \in \{0,1\}} \Lambda_{f,o'}(t)^{\alpha_{o'}} \cdot \Gamma_{f,o'}(t)^{\beta_{o'}}} & \text{if } o \text{ is feasible} \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where, $\Lambda_{f,o}(t)$ is the pheromone intensity for choosing option o for feature f during iteration t , $\Gamma_{f,o}(t)$ is the heuristic

bias for option o , α_0 and β_0 are weight pheromone memory and heuristic guidance, respectively. This formulation governs initial exploration but is insufficient for high-dimensional intrusion datasets because static parameters cannot adapt to evolving search conditions. The pheromone values are updated after each iteration to reinforce desirable decisions. Classical evaporation and deposition follow Eq. (10).

$$\Lambda_{f,o}(t+1) = (1 - \delta_0)\Lambda_{f,o}(t) + \sum_{u=1}^U \Delta\Lambda_{f,o}^u(t) \quad (10)$$

The deposit amount for ant u , is determined by Eq. (11).

$$\Delta\Lambda_{f,o}^u(t) = \begin{cases} \frac{C}{L_u} & \text{if ant } u \text{ selected } o \text{ for } f \\ 0 & \text{otherwise} \end{cases} \quad (11)$$

where, L_u is the solution quality (e.g., inverse fitness), C is a constant reward, and δ_0 is the fixed evaporation rate. In complex FS scenarios, fixed evaporation and simple reinforcement can cause stagnation due to over-concentration of pheromones on early-discovered subsets.

To overcome the previously described limitations, DyACO introduces adaptive mechanisms that evolve over the course of the search. Rather than treating each decision as static, DyACO continuously adjusts heuristics, pheromone decay, reinforcement, and transition probabilities over iterations.

In classical ACO, heuristic information is often based on simple metrics (e.g., inverse distance). In FS, the “heuristic” should instead reflect a feature’s usefulness. DyACO introduces a dynamic weighting function using a Gaussian-shaped modulation that varies with iteration count. The dynamic heuristic is defined in Eq. (12).

$$\Gamma_{f,o}(t) = \Psi_t \cdot \frac{X}{d_{f,o}^2} \quad (12)$$

The temporal scaling factor follows a modified normal distribution, expressed by Eq. (13).

$$\Psi_t = \exp\left(-3\left(\frac{t}{T}\right)^2\right) \quad (13)$$

where, t is the current iteration, T is the maximum number of iterations, $d_{f,o}$ is a domain-specific heuristic score (e.g., relevance measure), and X is a constant. This mechanism ensures strong heuristic guidance early on while diminishing its effect later, promoting better global exploration. Traditional ACO uses a fixed pheromone evaporation parameter, which is unsuitable for high-dimensional FS because it either over-strengthens or over-weakens pheromone memory. DyACO adopts a sigmoid-based dynamic evaporation rate, allowing the pheromone memory to adapt as the search progresses. The dynamic pheromone volatility is computed according to Eq. (14).

$$\delta_t = \frac{1}{1 + e^{-\left(A\frac{T-t}{T} + B\frac{t}{T}\right)}} \quad (14)$$

A and B control the transition shape. Early iterations maintain lower evaporation (thus promoting exploitation), while later iterations increase evaporation to encourage diversification

and overcome stagnation. Classical reinforcement accumulates pheromone only on good solutions. DyACO modifies this by rewarding strong feature subsets and penalizing the worst ones, encouraging the colony to abandon inferior search regions. The updated pheromone model is expressed in Eq. (15).

$$\Lambda_{f,o}(t+1) = (1 - \delta_t)\Lambda_{f,o}(t) + \sum_{u=1}^U \Delta\Lambda_{f,o}^u(t) \quad (15)$$

The deposit term is adjusted as stated in Eq. (16).

$$\Delta\Lambda_{f,o}^u(t) = \begin{cases} \frac{3C}{L_u} + \mu \cdot \frac{L_{best} - L_u}{L_{avg} - L_{best}}, & L_u \leq L_{best} \\ \frac{C}{L_u} + \varepsilon \cdot \frac{L_u - L_{worst}}{L_{worst} - L_{avg}}, & L_u \geq L_{best} \\ \frac{C}{L_u}, & \text{otherwise} \end{cases} \quad (16)$$

where, L_{best} , L_{worst} , and L_{avg} are iteration-level statistics, and μ and ε are reinforcement weights. This formulation accelerates convergence while avoiding the amplification of poor decisions. To balance exploration and exploitation, DyACO introduces two mechanisms, namely, the feature-smoothness heuristic and the updated transition rule. A turning-angle heuristic from path planning is reinterpreted here as subset smoothness, discouraging abrupt changes in FS patterns that may lead to unstable learning performance. Its scaling factor is computed using Eq. (17).

$$\Theta_t = \frac{1}{e^{(\omega_t/2)}} \quad (17)$$

where, ω_t quantifies inconsistency in consecutive selections in the current ant’s partial feature subset. Eq. (18) presents the improved transition probability combining pheromone, heuristic, and subset-smoothness.

$$\begin{aligned} & \prod_{f,o}^u(t) \\ &= \begin{cases} \frac{\Lambda_{f,o}(t)^{\alpha_1} \cdot \Gamma_{f,o}(t)^{\beta_1} \cdot \Theta_t^{\gamma_1}}{\sum_{o' \in \{0,1\}} \Lambda_{f,o'}(t)^{\alpha_1} \cdot \Gamma_{f,o'}(t)^{\beta_1} \cdot \Theta_t^{\gamma_1}}, & o \in \{0,1\} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (18)$$

A dynamically decaying exploitation parameter q_t is adopted, computed according to Eq. (19).

$$q_t = q^* \cdot \exp\left(-0.5\left(\frac{t}{T}\right)^2\right) \quad (19)$$

B. Training and Evaluation of Classifiers

As a result of the proposed FS procedure, a classification model can be trained to distinguish malicious traffic from legitimate network traffic using the selected features. In this work, a GRU-based neural architecture serves as the primary classifier because it efficiently learns temporal dependencies in flow-based intrusion data. After feature selection, the selected attributes of each network flow are organized by temporal occurrence to construct sequential input representations for the

GRU model. Each traffic sample is therefore represented as a feature sequence, allowing the GRU network to capture temporal dependencies and behavioral patterns associated with different network activities. The trained model is then validated on an independent test partition to assess its generalization ability.

To ensure a fair comparison across competing FS strategies, the same GRU configuration is used throughout the experiments. The implemented intrusion detection network consists of five hidden layers constructed around GRU cells, forming a deep sequential learning pipeline tailored for network traffic analysis. Table I summarizes the architectural specifications, including the number of hidden units, selected feature dimensionality (#), and output class cardinality (*) for each dataset.

TABLE I. HYPERPARAMETER CONFIGURATION OF THE IMPLEMENTED GRU-BASED IDS MODEL

Component	Specification	Value
Input layer	Feature dimension	#
Dense Layer 1	Linear neurons	100
Dense Layer 2	Linear neurons	64
Recurrent Layer 1	GRU cells	256
Recurrent Layer 2	GRU cells	128
Dense Layer 3	Linear neurons	100
Output layer	Linear units (classification)	*
Training batch size	–	1024
Total epochs	–	300
Dropout probability	–	0.5
Learning rate	–	0.0003
Optimization algorithm	–	Adam (Neural Network variant)
Activation function	–	ReLU
Loss function	–	Cross-Entropy

V. RESULTS AND DISCUSSION

This section reports the empirical evaluation of the proposed DyACO framework across two complementary experimental dimensions. First, DyACO is benchmarked using widely adopted optimization test functions from the CEC suite to validate its global search ability and robustness in heterogeneous landscapes. Then, DyACO is assessed within the intrusion detection domain, examining three representative datasets, including CSE-CIC-IDS2018, UNSW-NB15, and NSL-KDD, where its FS efficiency and classification performance are compared against several well-established metaheuristic baselines.

To comprehensively evaluate DyACO, three publicly available benchmark datasets were employed, including CSE-CIC-IDS2018, UNSW-NB15, and NSL-KDD. Collectively, these databases span both legacy and modern network environments, include diverse attack families, and exhibit variations in feature dimensionality and class imbalance, conditions that challenge both FS algorithms and IDS classifiers.

The NSL-KDD dataset refines the well-known KDDCup99 dataset by removing redundant and duplicate traffic flows, thereby minimizing training bias and mitigating the dominance of majority attack classes. NSL-KDD contains four attack types and normal traffic instances and provides 125,973 records for training and 22,543 for testing. UNSW-NB15 was generated using the IXIA PerfectStorm testbed to emulate realistic contemporary cyber threats.

CSE-CIC-IDS2018 is an enterprise-grade repository developed by the Canadian Network Security Institute that models realistic operational environments. It includes normal activity, along with a range of modern cyberattacks such as brute-force attacks, infiltration, botnet behavior, Web exploits, and distributed Denial-of-Service (DDoS). Given its high dimensionality and complex traffic dynamics, CSE-CIC-IDS2018 is a particularly rigorous testbed for evaluating advanced FS systems like DyACO. As summarized in Table II, the dataset comprises benign traffic alongside diverse contemporary attack categories.

TABLE II. DISTRIBUTION OF ATTACK CATEGORIES AND FLOW VOLUMES IN THE CSE-CIC-IDS2018 DATASET

Attack category	Example attack instances	Number of flows
Normal traffic	Benign	13,484,708
Web-based exploits	Brute Force-Web, Brute Force-XSS, SQL Injection	611; 230; 87
DoS incidents	SlowHTTPTest, GoldenEye, Slowloris, Hulk	139,890; 41,508; 10,990; 461,192
Infiltration threats	Infiltration	161,934
Brute-force attempts	FTP-BruteForce, SSH-Bruteforce	193,360; 187,589
Botnet activity	Bot	286,191
DDoS-related attacks	HOIC, LOIC-HTTP, LOIC-UDP	686,012; 576,191; 1,730
Overall total	–	16,232,223

In this study, the performance of the GRU-based intrusion classifier, fed with the feature subsets recommended by DyACO, is quantified using a set of well-established metrics derived from the confusion matrix. The confusion matrix summarizes the model's predictions by categorizing samples into four groups: True Positives (TPs) representing correctly detected attacks, True Negatives (TNs) for correctly identified normal traffic, False Positives (FPs) referring to benign samples incorrectly flagged as malicious, and False Negatives (FNs) denoting attacks that the classifier failed to identify. To comprehensively evaluate the classifier, four primary measures are adopted: accuracy, recall, precision, and F1-score. Accuracy reflects the overall proportion of correctly classified samples and serves as a broad indicator of predictive performance, computed using Eq. (20).

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (20)$$

To better capture the algorithm's ability to identify malicious behavior, recall [(Eq. (21))] is employed. Recall quantifies the fraction of actual attacks the algorithm detects.

$$Recall = \frac{TP}{TP + FN} \quad (21)$$

Complementing recall, precision measures the reliability of positive predictions by evaluating how many detected attacks are indeed malicious. It is calculated according to Eq. (22).

$$Precision = \frac{TP}{TP + FP} \quad (22)$$

Since high recall with low precision (or vice versa) is undesirable, the F1-score provides a harmonic mean of these two metrics, yielding a balanced indicator of IDS performance. Eq. (23) defines the F1-score.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (23)$$

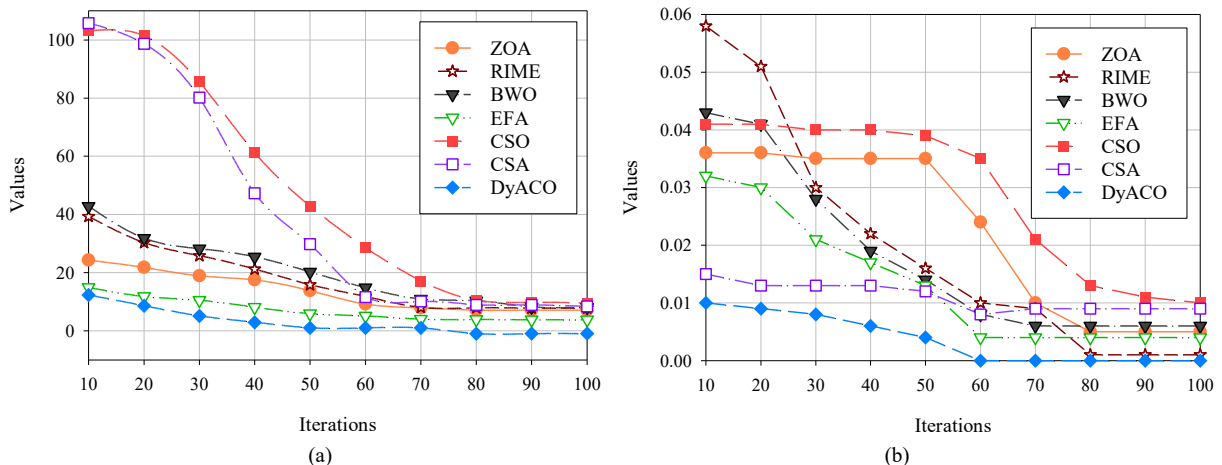
To establish the optimization capability of DyACO, the framework was initially benchmarked using a collection of well-known continuous test functions widely adopted in CEC competitions. As summarized in Table III, these functions span diverse search landscapes, varying in modality, curvature, and search-space boundaries. DyACO was evaluated against six competitive baseline optimizers, including Zebra Optimization Algorithm (ZOA) [23], Rime Optimization Algorithm (RIME) [24], Beluga Whale Optimization (BWO) [25], Enhanced Fireworks Algorithm (EFA) [26], Chicken Swarm Optimization (CSO) [27], and Crow Search Algorithm (CSA) [28]. All experiments used uniform settings, including a population size of $N=60$, a 30-dimensional search space, and a maximum of 100 iterations.

TABLE III. BENCHMARK OPTIMIZATION FUNCTIONS USED IN THIS STUDY, ALONG WITH THEIR INPUT DOMAINS AND GLOBAL SOLUTIONS

Benchmark	Reformulated expression	Search interval	Optimal value
Rosenbrock	$g(u) = \sum_{j=1}^{m-1} [100(u_{j+1} - u_j^2)^2 + (1 - u_j)^2]$	$[-5, 10]$	0
Schaffer	$g(v_1, v_2) = 0.5 + \frac{\sin^2(v_1^2 - v_2^2) - 0.5}{[1 + 0.001(v_1^2 + v_2^2)]^2}$	$[-100, 100]$	0
Booth	$g(v_1, v_2) = (v_1 + 2v_2 - 7)^2 + (2v_1 + v_2 - 5)^2$	$[-10, 10]$	0
Easom	$g(v_1, v_2) = -\cos(v_1)\cos(v_2)\exp(-[(v_1 - \pi)^2 + (v_2 - \pi)^2])$	$[-100, 100]$	-1
Rastrigin	$g(u) = 10m + \sum_{j=1}^m [u_j^2 - 10\cos(2\pi u_j)]$	$[-5.12, 5.12]$	0
Ackley	$g(u) = -20\exp(-0.2\sqrt{\frac{1}{m}\sum_{j=1}^m u_j^2}) - \exp(\frac{1}{m}\sum_{j=1}^m \cos(2\pi u_j)) + 20 + e$	$[-5, 5]$	0

Fig. 3 visually presents convergence behavior and final optimization performance, highlighting the superiority of the proposed method. The intrusion detection experiments on CSE-CIC-IDS2018, UNSW-NB15, and NSL-KDD collectively demonstrate DyACO's effectiveness in improving FS quality and classification performance across heterogeneous network environments. DyACO consistently outperforms baseline

metaheuristics, demonstrating its ability to navigate high-dimensional search spaces while maintaining stable convergence. The visual comparisons in Fig. 4–7 further reinforce these findings by clearly separating DyACO from competing strategies across accuracy, precision, recall, and F1-score distributions.



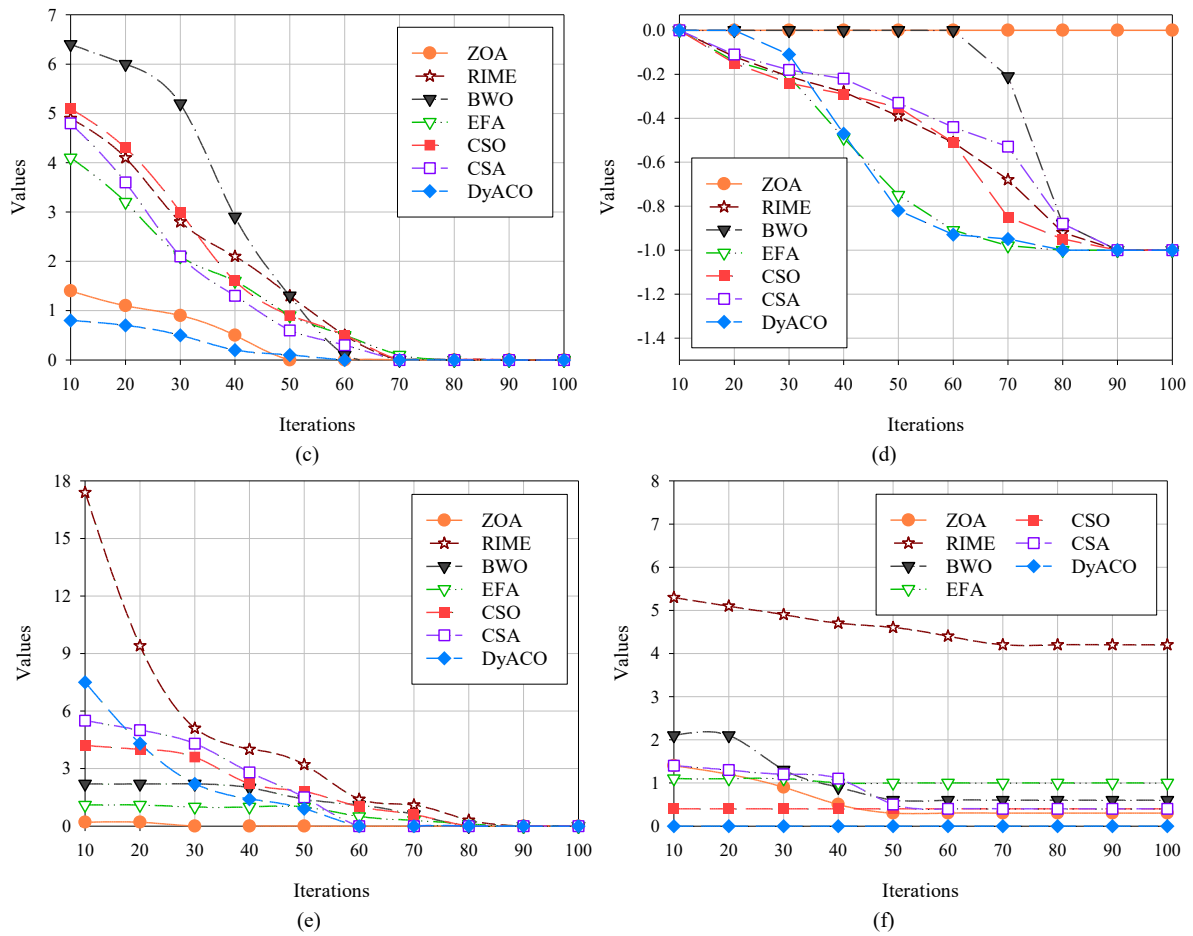


Fig. 3. Convergence curves comparing DyACO with baseline optimizers across benchmark test functions: Rosenbrock (a), Schaffer (b), Booth (c), Easom (d), Rastrigin (e), Ackley (f).

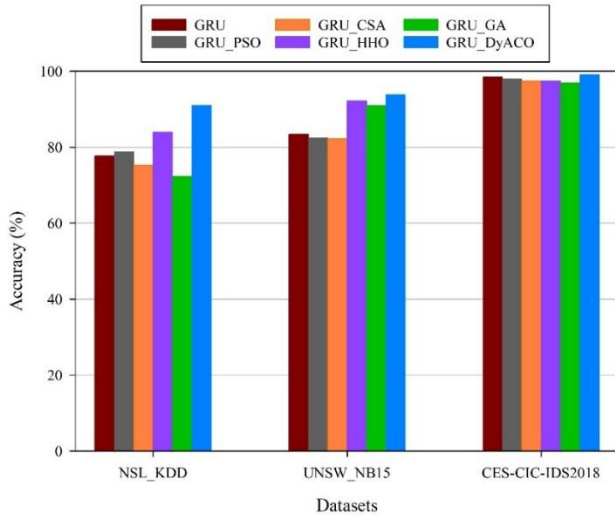


Fig. 4. Accuracy comparison of GRU-driven IDS models across benchmark datasets.

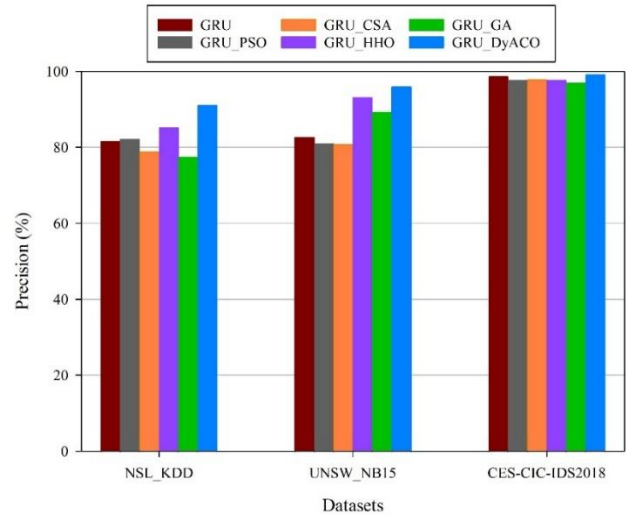


Fig. 5. Precision comparison of GRU-driven IDS models across benchmark datasets.

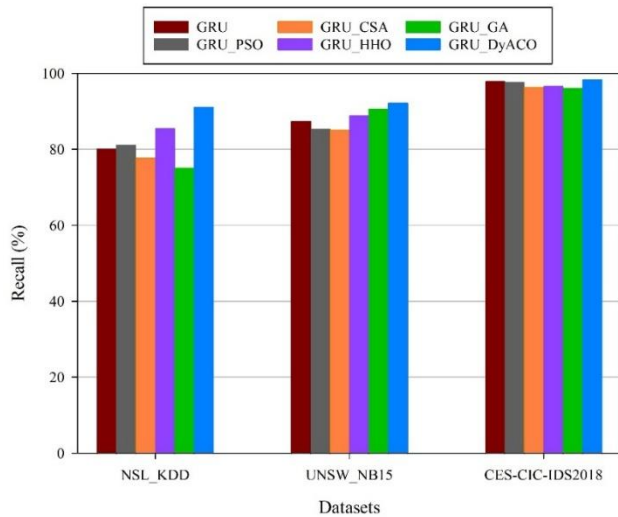


Fig. 6. Recall comparison of GRU- driven IDS models across benchmark datasets.

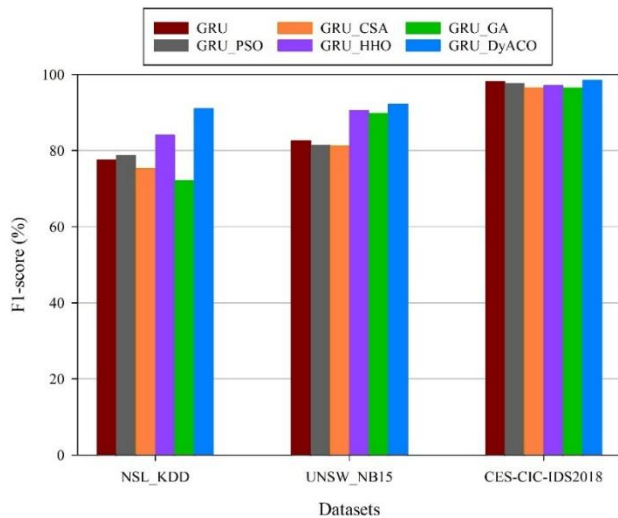


Fig. 7. F1-score comparison of GRU- driven IDS models across benchmark datasets.

VI. CONCLUSION

In this study, a dynamic wrapper-based method called DyACO, based on the ACO algorithm, is proposed to efficiently address high-dimensional FS problems for NID. By introducing adaptive heuristics, iteration-dependent pheromone evaporation rates, magnitude-based reward-penalty functions in the reinforcement scheme, and adaptive state transition rules, the DyACO algorithm overcomes the limitations of conventional metaheuristic algorithms. Experimental evaluations on the CSE-CIC-IDS2018, UNSW-NB15, and NSL-KDD datasets demonstrate that DyACO can obtain compact subsets of discriminative features, reduce feature dimensionality, and achieve high detection performance. Based on accuracy, precision, recall, and F1-score, DyACO demonstrates superior performance compared with the considered optimization algorithms while maintaining the computational efficiency required for wrapper-based methods.

REFERENCES

- [1] B. Pourghebleh, K. Wakil, and N. J. Navimipour, "A comprehensive study on the trust management techniques in the Internet of Things," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 9326-9337, 2019, doi: <https://doi.org/10.1109/JIOT.2019.2933518>.
- [2] C. Hamroun, A. Fladenmuller, M. Pariente, and G. Pujolle, "Intrusion detection in 5G and Wi-Fi networks: A survey of current methods, challenges & perspectives," *IEEE Access*, 2025.
- [3] E. Rivandi, "Application of Machine Learning Techniques to Fraud Detection in Financial Transactions," *Journal of Intelligent Decision Making and Granular Computing*, vol. 2, no. 1, pp. 288-303, 2026, doi: <https://doi.org/10.31181/jidmgc21202647>.
- [4] D. Manivannan, "Recent endeavors in machine learning-powered intrusion detection systems for the internet of things," *Journal of Network and Computer Applications*, vol. 229, p. 103925, 2024.
- [5] V. Hayyolalam, B. Pourghebleh, and A. A. Pourhaji Kazem, "Trust management of services (TMoS): investigating the current mechanisms," *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 10, p. e4063, 2020, doi: <https://doi.org/10.1002/ett.4063>.
- [6] A. V. Turukmane and R. Devendiran, "M-MultiSVM: An efficient feature selection assisted network intrusion detection system using machine learning," *Computers & Security*, vol. 137, p. 103587, 2024.
- [7] E. Zorarpaci, "A fast intrusion detection system based on swift wrapper feature selection and speedy ensemble classifier," *Engineering Applications of Artificial Intelligence*, vol. 133, p. 108162, 2024.
- [8] J. Maldonado and M. C. Riff, "An evolutionary wrapper to support intrusion detection system configuration," *Computers & Security*, p. 104478, 2025.
- [9] A. Birashk, J. K. Kordestani, and M. R. Meybodi, "Cellular teaching-learning-based optimization approach for dynamic multi-objective problems," *Knowledge-Based Systems*, vol. 141, pp. 148-177, 2018, doi: <https://doi.org/10.1016/j.knsys.2017.11.016>.
- [10] M. M. Shwaysh et al., "Adaptive hybrid information gain and autoencoder-based feature selection with ensemble recurrent extreme learning machine for enhanced network intrusion detection systems," *Journal of Network and Systems Management*, vol. 34, no. 1, pp. 1-30, 2026.
- [11] H. Zouhri, A. Idri, and A. Ratnani, "Evaluating the impact of filter-based feature selection in intrusion detection systems," *International Journal of Information Security*, vol. 23, no. 2, pp. 759-785, 2024.
- [12] G. N. N. Barbosa, M. Andreoni, and D. M. F. Mattos, "Optimizing feature selection in intrusion detection systems: Pareto dominance set approaches with mutual information and linear correlation," *Ad Hoc Networks*, vol. 159, p. 103485, 2024.
- [13] M. S. Habeeb and T. R. Babu, "Coarse and fine feature selection for network intrusion detection systems (IDS) in IoT networks," *Transactions on Emerging Telecommunications Technologies*, vol. 35, no. 4, p. e4961, 2024.
- [14] E. M. Maseno and Z. Wang, "Hybrid wrapper feature selection method based on genetic algorithm and extreme learning machine for intrusion detection," *Journal of big data*, vol. 11, no. 1, p. 24, 2024.
- [15] M. Umer, M. Tahir, M. Sardaraz, M. Sharif, H. Elmannai, and A. D. Algarni, "Network intrusion detection model using wrapper based feature selection and multi head attention transformers," *Scientific reports*, vol. 15, no. 1, p. 28718, 2025.
- [16] A. Anagha, C. Thomas, and N. Balakrishnan, "Optimized intrusion predictions through feature selection methods," *Computers & Security*, p. 104541, 2025.
- [17] W. L. Al-Yaseen, A. K. Idrees, and F. H. Almasoudy, "Wrapper feature selection method based differential evolution and extreme learning machine for intrusion detection system," *Pattern Recognition*, vol. 132, p. 108912, 2022.
- [18] Z. Halim et al., "An effective genetic algorithm-based feature selection method for intrusion detection systems," *Computers & Security*, vol. 110, p. 102448, 2021.

- [19] A. Nazir and R. A. Khan, "A novel combinatorial optimization based feature selection method for network intrusion detection," *Computers & Security*, vol. 102, p. 102164, 2021.
- [20] Y. Pourardebil Khah, M. Hosseini Shirvani, and H. Motameni, "A hybrid machine learning approach for feature selection in designing intrusion detection systems (IDS) model for distributed computing networks," *The Journal of Supercomputing*, vol. 81, no. 1, p. 254, 2025.
- [21] S. Hosseini, M. Khorashadizade, and M. Jouyban, "A Hybrid Method Using Slime Mold Algorithm and Genetic Algorithm for Feature Selection Problems in Intrusion Detection Systems," *Engineering Reports*, vol. 7, no. 7, p. e70254, 2025.
- [22] M. A. Laamari and N. Kamel, "A New Multi-Objective Binary Bat Algorithm for Feature Selection in Intrusion Detection Systems," *Concurrency and Computation: Practice and Experience*, vol. 37, no. 4-5, p. e70000, 2025.
- [23] E. Trojovská, M. Dehghani, and P. Trojovský, "Zebra optimization algorithm: A new bio-inspired optimization algorithm for solving optimization algorithm," *Ieee Access*, vol. 10, pp. 49445-49473, 2022.
- [24] H. Su et al., "RIME: A physics-based optimization," *Neurocomputing*, vol. 532, pp. 183-214, 2023.
- [25] C. Zhong, G. Li, and Z. Meng, "Beluga whale optimization: A novel nature-inspired metaheuristic algorithm," *Knowledge-based systems*, vol. 251, p. 109215, 2022.
- [26] S. Zheng, A. Janeczek, and Y. Tan, "Enhanced fireworks algorithm," in *2013 IEEE congress on evolutionary computation*, 2013: IEEE, pp. 2069-2077.
- [27] X. Meng, Y. Liu, X. Gao, and H. Zhang, "A new bio-inspired algorithm: chicken swarm optimization," in *International conference in swarm intelligence*, 2014: Springer, pp. 86-94.
- [28] A. G. Hussien et al., "Crow search algorithm: theory, recent advances, and applications," *Ieee Access*, vol. 8, pp. 173548-173565, 2020.