

Article

DGWO: A Deep Reinforcement Learning-Driven Grey Wolf Optimizer for Feature Selection in Network Intrusion Detection Systems

Qianqian Zhang, Ting Shu *  and Jinsong Xia

School of Computer Science and Technology, Zhejiang Sci-Tech University, Hangzhou 310018, China; 2024220603109@mails.zstu.edu.cn (Q.Z.); summerx@zstu.edu.cn (J.X.)

* Correspondence: shuting@zstu.edu.cn

Abstract

With the continuous evolution of network attack techniques, efficiently selecting the most discriminative feature subset from massive network traffic data has become a key issue for improving the performance of intrusion detection systems. Metaheuristic algorithms, as a core approach for wrapper-based feature selection, directly determine the quality of the selected feature subset through their optimization capability. The Grey Wolf Optimizer (GWO) is popular due to its simple structure and few parameters, where three leader wolves guide the search through weighted cooperation. However, its static weight mechanism cannot adapt to dynamic changes in individual search states and population evolution stages, limiting optimization capability and convergence performance. To address this issue, this study proposes a Deep Reinforcement Learning-based Grey Wolf Optimizer (DGWO), which pre-trains a weight adjustment decision model offline and dynamically adjusts the guiding weights of leader wolves during the online search process, thereby improving the optimization ability of the algorithm. Experimental results on NSL-KDD, UNSW-NB15, and CIC-IDS-2017 datasets show that DGWO outperforms seven comparative feature selection methods. It achieves classification accuracies of 93.59%, 93.40%, and 94.84%, respectively, demonstrating superior performance in accuracy, precision, recall, and F1-score. DGWO promotes symmetry between cybersecurity requirements and reliable intrusion detection.

Keywords: network intrusion detection; feature selection; grey wolf optimizer; deep reinforcement learning

1. Introduction

With the rapid advancement of Internet technologies and the widespread proliferation of network applications, cybersecurity threats have become increasingly severe. Various forms of network attacks continue to evolve and escalate, imposing significant economic losses and privacy risks on enterprises, governments, and individual users. As a core component of network security defense, the intrusion detection system (IDS) [1] is considered essential for real-time network traffic monitoring, anomaly identification, and alert generation, serving as a critical technical measure against network intrusions [2]. However, modern network traffic data typically exhibit high dimensionality and substantial redundancy. The presence of numerous irrelevant or redundant features not only significantly increases the computational overhead of model training but also degrades classifier performance [3,4]. Consequently, efficiently identifying the most discriminative feature



Received: 23 June 2026

Revised: 17 July 2026

Accepted: 22 July 2026

Published: 23 July 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

subset from a large pool of raw features has emerged as a key challenge in enhancing the effectiveness of intrusion detection systems [5].

Feature selection (FS) [6] is defined as the process of identifying and retaining the most informative features from a high-dimensional feature space that contribute most to classification tasks. By reducing feature dimensionality and minimizing noise interference, FS improves model generalization capability and training efficiency. Feature selection methods are typically classified into three categories: filter-based, wrapper-based, and embedded methods [7]. Among these, wrapper-based methods directly employ classifier performance as the evaluation criterion for feature subsets, yielding feature combinations that are highly aligned with specific classification tasks. However, since the search space grows exponentially with the number of features, wrapper-based feature selection is inherently an NP-hard problem, making it infeasible for traditional exhaustive search or greedy algorithms to obtain high-quality solutions within a reasonable time frame. To address this challenge, metaheuristic optimization algorithms have been widely introduced for feature selection. Inspired by biological collective behaviors or physical phenomena observed in nature, these algorithms are characterized by simple heuristic mechanisms and strong global exploration capabilities, making them broadly applicable across various domains [8]. Representative examples include the particle swarm optimization (PSO) [9], whale optimization algorithm (WOA) [10], harris hawks optimization (HHO) [11], sparrow search algorithm (SSA) [12], and grey wolf optimizer (GWO) [13].

GWO is a metaheuristic optimization algorithm that simulates the social hierarchy and cooperative hunting behavior of grey wolf packs in nature [13]. Owing to its simple structure, minimal parameterization, and fast convergence speed, GWO has been widely applied in the field of feature selection [14,15]. To further enhance its performance, a series of improved strategies have been proposed from different perspectives. These include a dimensional learning mechanism to enhance global exploration capability [16], a random-walk-based position update strategy to improve local exploitation performance [17], and fuzzy logic-based parameter control for adaptive adjustment of the convergence factor [18]. Furthermore, a competitive binary variant has been developed to accelerate the solution of binary optimization problems [14]. However, these methods still suffer from fixed leader wolf weight allocation, which fails to respond to the dynamic changes in individual search states and population evolution stages, thereby limiting the optimization capability of the algorithm.

To overcome these limitations, a Deep Reinforcement Learning-based grey wolf optimizer (DGWO) is proposed in this paper. The proposed method constructs a three-dimensional state representation that integrates individual fitness ranking, search stagnation degree, and population diversity, enabling the agent to simultaneously perceive both the local search state of individuals and the macroscopic evolutionary dynamics of the population. On this basis, the guidance weights of the leader wolves are dynamically adjusted throughout the iterative process. Accordingly, this study hypothesizes that dynamically regulating the guidance weights of the α , β , and δ wolves according to individual and population states can improve the optimization performance of GWO and enable the selection of more compact and discriminative feature subsets. This hypothesis is evaluated through comparative experiments, ablation studies, and convergence analysis. The primary contributions of this work are as follows:

- A Deep Reinforcement Learning (DRL)-driven adaptive weight adjustment mechanism is proposed to enhance the GWO. Specifically, a Deep Q-Network is pre-trained offline to dynamically regulate the influence weights of the three leader wolves during the optimization process, thereby improving the search capability and convergence performance of the algorithm.

- A three-dimensional state representation is designed to guide the decision-making process of the DRL agent. It jointly encodes individual fitness ranking, search stagnation degree, and population diversity, enabling the model to capture both local search behavior and global evolutionary dynamics.
- The proposed DGWO is applied to intrusion detection feature selection, where a standard discretization strategy and a composite fitness function are adopted to evaluate candidate solutions.
- Experiments on the NSL-KDD, UNSW-NB15, and CIC-IDS-2017 datasets against seven algorithms show that DGWO consistently achieves superior performance across all evaluation metrics, while requiring the fewest features on average, thereby validating its effectiveness for intrusion detection feature selection.

The subsequent sections are structured as follows: Section 2 reviews related work. Section 3 introduces the foundational techniques of GWO and DQN. Section 4 presents the detailed design of the DGWO algorithm and its application to feature selection. Section 5 discusses the experimental evaluation. Section 6 summarizes the main findings of this study and discusses possible directions for future work.

2. Related Work

Feature selection aims to identify the most discriminative feature subset from a high-dimensional feature space, and is recognized as a critical component in enhancing the performance of intrusion detection systems. Existing feature selection methods are broadly categorized into three groups: filter-based, wrapper-based, and embedded methods [19,20].

Filter-based methods evaluate features independently of any classifier using statistical criteria such as information gain and correlation coefficients. While these approaches are computationally efficient, they tend to overlook feature interactions and classifier compatibility, which often results in suboptimal classification performance for the selected subset [21].

Embedded methods integrate the feature selection process directly into model training, allowing them to account for both feature relevance and model performance simultaneously. However, their outcomes are tightly coupled to specific learning algorithms, which limits their generalizability across different tasks [22].

In contrast, wrapper-based methods directly employ classifier performance as the evaluation criterion for candidate feature subsets, enabling the identification of feature combinations that are highly tailored to specific classification tasks. Nevertheless, since the search space grows exponentially with the number of features, wrapper-based feature selection is inherently an NP-hard problem, making traditional exhaustive and greedy strategies insufficient for obtaining high-quality solutions within a reasonable time frame [20].

To address this challenge, metaheuristic optimization algorithms have been widely adopted as search strategies in wrapper-based feature selection, leveraging swarm intelligence mechanisms to efficiently explore optimal feature combinations in high-dimensional spaces [23]. Kunhare et al. [24] combined particle swarm optimization (PSO) with random forest-based feature pre-screening for wrapper-based optimization of intrusion detection feature subsets on the NSL-KDD dataset, achieving effective compression of the feature space while significantly improving detection accuracy and reducing false alarm rates. Alazzam et al. [25] introduced a wrapper-based feature selection approach grounded in the pigeon-inspired optimizer (PIO), proposing two variants, SPIO and CPIO, that guide position updates via cosine similarity strategies. Their method demonstrated notable reductions in selected feature count alongside considerable gains in intrusion detection classification accuracy. Safaldin et al. [26] developed GWOSVM-IDS, an intrusion detection framework combining an improved binary grey wolf optimizer with a support

vector machine, which explores optimal feature subsets by adjusting the pack size and effectively reduces false alarm rates while enhancing detection accuracy on the NSL-KDD dataset. Alzaqebah et al. [27] proposed a modified grey wolf optimizer (MGWO) that integrates filter-based and wrapper-based concepts, employing an intelligent initialization mechanism to preferentially retain information-rich features during the early stages of optimization, thereby improving the IDS capability for detecting anomalous network traffic. Kareem et al. [28] designed GTO-BSA, a hybrid feature selection approach that fuses the gorilla troops optimizer (GTO) with the bird swarm algorithm (BSA). By leveraging the strong local exploitation capability of BSA to compensate for GTO's limitations, this method achieves superior convergence speed and solution quality compared to multiple competing algorithms across four benchmark datasets, including NSL-KDD, CICIDS-2017, UNSW-NB15, and BoT-IoT. Sezgin et al. [29] conducted a comparative analysis of multi-objective feature selection methods for intrusion detection systems using several representative metaheuristic algorithms, including GWO and PSO, demonstrating that different optimization strategies lead to significantly different trade-offs between detection accuracy and feature reduction. Mansouri and Hosseini [30] proposed a chaos-enhanced hybrid metaheuristic feature selection method combined with an artificial neural network classifier for intrusion detection. Their approach integrates opposition-based learning for population initialization, a hybrid Gray Wolf Optimization–Whale Optimization Algorithm to balance exploration and exploitation, and a chaos-based local search strategy to improve convergence behavior and avoid local optima. Experimental results demonstrate that the proposed method can effectively reduce feature dimensionality while improving detection accuracy and robustness in complex network environments. Although these metaheuristic-based methods have improved intrusion detection feature selection to varying degrees, their search mechanisms are still largely governed by fixed parameters or manually designed strategies. Consequently, they cannot effectively adapt to changes in individual search states and population evolution, thereby limiting further improvements in optimization performance.

Reinforcement learning (RL) [31] is a machine learning paradigm in which an agent learns optimal decision-making policies through continuous interaction with its environment. Its inherent capacity to extract knowledge from historical experience and enable dynamic parameter adjustment provides a promising avenue for overcoming the aforementioned limitations. In the domain of enhancing metaheuristic search mechanisms via reinforcement learning, Seyyedabbasi et al. [32] proposed three Q-learning-based hybrid algorithms that separately integrate reinforcement learning with the whale optimization algorithm (RLWOA), the incremental grey wolf optimizer (RLI-GWO), and the extended grey wolf optimizer (RLEx-GWO). By employing a reward-punishment mechanism to enable the agent to adaptively switch between exploration and exploitation phases, the effectiveness of these hybrid strategies was validated on 30 benchmark functions and a robot inverse kinematics problem. Seyyedabbasi [33] further combined Q-learning with the sand cat swarm optimization algorithm to develop RLSCSO, demonstrating the advantage of reinforcement learning in enhancing global optimization accuracy across 20 benchmark functions and a mobile sensor node localization problem. Ming et al. [34] incorporated deep reinforcement learning into the operator selection process of a constrained multi-objective evolutionary algorithm, characterizing the state using the convergence, diversity, and feasibility of the population, and dynamically selecting the most appropriate evolutionary operator for the current search phase via DQN. This approach achieved superior performance over manually designed strategies on constrained multi-objective optimization benchmarks.

The above studies demonstrate that reinforcement learning can effectively improve the adaptability of metaheuristic algorithms. However, existing reinforcement learning-assisted methods mainly employ agents for global parameter adjustment, exploration–exploitation

switching, or predefined operator selection, and therefore pay limited attention to the differentiated search states of individuals and the cooperative guidance of multiple leader solutions. In contrast, DGWO jointly considers individual states and population states, and uses DQN to dynamically adjust the guidance weights of the α , β , and δ wolves for each individual, thereby providing a more targeted adaptive search mechanism for GWO.

3. Preliminary

3.1. GWO

The GWO is a swarm intelligence optimization algorithm proposed by Mirjalili et al. [13]. GWO draws inspiration from the social hierarchy and cooperative hunting strategies of grey wolf packs in nature, organizing the population into four hierarchical tiers designated as α , β , δ , and ω . The three leading wolves, namely α , β , and δ , are responsible for guiding the population search, while ω wolves update their positions by following the directives of the three leaders. Owing to its straightforward structure and minimal parameterization, GWO has demonstrated robust optimization performance across a broad range of engineering and combinatorial problems.

In GWO, the encircling behavior of grey wolves toward their prey is mathematically formulated as follows. Let t denote the current iteration, $\vec{X}_p(t)$ the position of the prey, and $\vec{X}(t)$ the current position of a grey wolf. The distance between the wolf and the prey is computed according to Equation (1), and the position update rule is given by Equation (2):

$$\vec{D} = \left| \vec{C} \cdot \vec{X}_p(t) - \vec{X}(t) \right| \quad (1)$$

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} \cdot \vec{D} \quad (2)$$

The coefficient vectors $\vec{A}$ and $\vec{C}$ are calculated according to Equations (3) and (4), respectively, where $\vec{r}_1$ and $\vec{r}_2$ are random vectors uniformly sampled from $[0, 1]$, and $\vec{a}$ decreases linearly from 2 to 0 over the course of the iterations:

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \quad (3)$$

$$\vec{C} = 2\vec{r}_2 \quad (4)$$

$$\vec{a} = 2 - \frac{2t}{T_{\max}} \quad (5)$$

During the hunting phase, the α , β , and δ wolves are assumed to possess the most accurate knowledge of the prey's location. Each ω wolf computes its distances to the three leader wolves according to Equations (6)–(8), and derives three candidate step directions according to Equations (9)–(11):

$$\vec{D}_\alpha = \left| \vec{C}_1 \cdot \vec{X}_\alpha(t) - \vec{X}(t) \right| \quad (6)$$

$$\vec{D}_\beta = \left| \vec{C}_2 \cdot \vec{X}_\beta(t) - \vec{X}(t) \right| \quad (7)$$

$$\vec{D}_\delta = \left| \vec{C}_3 \cdot \vec{X}_\delta(t) - \vec{X}(t) \right| \quad (8)$$

$$\vec{X}_1 = \vec{X}_\alpha(t) - \vec{A}_1 \cdot \vec{D}_\alpha \quad (9)$$

$$\vec{X}_2 = \vec{X}_\beta(t) - \vec{A}_2 \cdot \vec{D}_\beta \quad (10)$$

$$\vec{X}_3 = \vec{X}_\delta(t) - \vec{A}_3 \cdot \vec{D}_\delta \quad (11)$$

Finally, the ω wolf's position for the next iteration is updated as the arithmetic mean of the three candidate positions, as expressed in Equation (12), progressively steering the population toward the global optimum:

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \quad (12)$$

3.2. Deep Q Network

DQN proposed by Mnih et al. [35] is a seminal algorithm in deep reinforcement learning that approximates the action-value function using a deep neural network, enabling effective decision-making in environments characterized by high-dimensional state spaces. The reinforcement learning framework is formally described by a Markov Decision Process (MDP), in which an agent observes a state $s \in \mathcal{S}$, selects an action $a \in \mathcal{A}$ according to policy π , receives a scalar reward r from the environment, and transitions to a subsequent state s' . The objective of the agent is to learn an optimal policy π^* that maximizes the expected cumulative discounted reward:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (13)$$

where $\gamma \in [0, 1]$ is the discount factor governing the relative importance of future rewards. The action-value function $Q^\pi(s, a)$ quantifies the expected cumulative return obtained by executing action a in state s and subsequently following policy π . The optimal action-value function $Q^*(s, a)$ satisfies the Bellman optimality equation:

$$Q^*(s, a) = \mathbb{E}_{s'}[R(s, a) + \gamma \max_{a'} Q^*(s', a')] \quad (14)$$

Traditional Q-learning maintains a tabular representation of Q-values for all state-action pairs; however, this approach becomes computationally intractable in high-dimensional or continuous state spaces. DQN addresses this limitation by employing a deep neural network $Q(s, a; \theta)$ to approximate the Q-function, and introduces two key mechanisms to stabilize the training process. The first is experience replay, wherein each interaction tuple (s_t, a_t, r_t, s_{t+1}) is stored in a replay buffer $\mathcal{D}$, and mini-batches of transitions are randomly sampled during training to break the temporal correlations between consecutive experiences. The second is the target network, which maintains a separate network parameterized by θ^- for computing target Q-values; its parameters are periodically synchronized from the online network to prevent rapid fluctuations in the optimization target. The loss function of DQN is defined as:

$$L(\theta) = \mathbb{E}_{(s, a, r, s') \sim \mathcal{D}}[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2] \quad (15)$$

By minimizing this loss, DQN progressively converges toward accurate Q-value estimates, yielding a stable and high-quality decision policy capable of handling complex sequential decision tasks.

4. Proposed Method

In this section, the detailed design of the proposed enhanced heuristic optimization algorithm is presented, along with its application to feature selection in network intrusion detection systems.

4.1. Reinforcement Learning Design for DGWO

To enable GWO to dynamically and adaptively regulate the guidance weights of the leader wolves throughout the optimization process, the underlying decision-making problem is formulated as a MDP and addressed within a deep reinforcement learning framework. The MDP is characterized by three core components: the state space, the action space, and the reward function, each of which is elaborated in detail below.

4.1.1. State

The principled construction of the state space is a prerequisite for the agent to make effective decisions. To endow the agent with simultaneous local perception of individual search states and global awareness of the overall population dynamics, a three-dimensional state vector is designed as follows:

$$S_i^t = [\text{rank}_i, \text{stag}_i, \text{div}^t] \quad (16)$$

rank_i characterizes the relative quality of the i -th grey wolf within the current population. At each generation, all individuals are first sorted from best to worst according to their fitness values, and the resulting ranks are then linearly normalized to the interval $[0, 1]$:

$$\text{rank}_i = \frac{i}{\text{pop} - 1}, \quad i \in \{0, 1, \dots, \text{pop} - 1\} \quad (17)$$

where pop denotes the population size. A value of rank_i approaching 0 indicates that the individual is among the top-performing members of the current population, whereas a value approaching 1 signals that its fitness is relatively poor and its position deviates substantially from the higher-quality region of the search space.

stag_i quantifies the degree to which an individual has been trapped in a stagnant search state over recent iterations. An adaptive window length $h_t = \min(h, t)$ is introduced to handle the insufficient historical data available in the early stages of optimization. The stagnation metric is subsequently computed by using the fitness at the start of the window, $f_i^{t-h_t}$, as the normalization reference:

$$\text{stag}_i^t = 1 - \frac{\max(f_i^{t-h_t} - f_i^t, 0)}{f_i^{t-h_t} + \varepsilon} \quad (18)$$

where f_i^t is the fitness value of the i -th individual at generation t , and $\varepsilon = 10^{-10}$ is a small constant introduced for numerical stability. A value of stag_i^t approaching 1 indicates that the fitness improvement within the window is negligible and the individual has effectively stalled, whereas a value approaching 0 indicates that the individual is in a productive search state.

div^t reflects the degree of spatial dispersion of the entire population across the search space, serving as a global indicator of the evolutionary phase in which the algorithm currently operates. Following the median-based dimensional diversity measure proposed by Hussain et al. [5], the per-dimension dispersion is first computed and then averaged after normalization:

$$\text{div}_j^t = \frac{1}{\text{pop}} \sum_{i=1}^{\text{pop}} |X_{i,j}^t - \text{median}(X_{:,j}^t)| \quad (19)$$

$$\text{div}^t = \frac{1}{\text{dim}} \sum_{j=1}^{\text{dim}} \frac{\text{div}_j^t}{|ub_j - lb_j|} \quad (20)$$

where $X_{i,j}^t$ is the position of the i -th individual along the j -th dimension at generation t , ub_j and lb_j are the upper and lower search bounds of dimension j , respectively, and $\dim$ is the problem dimensionality. The normalization eliminates scale discrepancies arising from different search ranges across dimensions, ensuring $\text{div}^t \in [0, 1]$. A higher value signifies a dispersed population characteristic of the global exploration phase, while a lower value indicates population convergence and a transition into the local exploitation phase. This global feature complements the individual-level features, enabling the agent to integrate information from both levels and make more targeted weight adjustment decisions.

4.1.2. Action

As described above, the agent must dynamically regulate the influence weights of the three leader wolves in response to both the evolutionary phase of the population and the real-time search state of each individual. Within the hierarchical guidance mechanism of GWO, the α , β , and δ wolves represent decreasing tiers of guidance quality. Amplifying the weight assigned to a particular leader wolf strengthens its dominant role in steering the position update of each individual. Specifically, increasing w_α concentrates the individual's search around the current global best region, whereas increasing w_β or w_δ promotes exploration of new directions and enhances population diversity, thereby facilitating escape from local optima. The weight configuration is subject to the equality constraint $w_\alpha + w_\beta + w_\delta = 1$. Based on these considerations, the action space is defined as a discrete set of four candidate actions:

$$\mathcal{A} = \{\text{Inc}_{w_\alpha}, \text{Inc}_{w_\beta}, \text{Inc}_{w_\delta}, \text{Stay}\} \quad (21)$$

where Inc_{w_α} , Inc_{w_β} , and Inc_{w_δ} denote amplification of the guidance weight of the corresponding leader wolf, and Stay indicates that the current weight configuration is maintained without modification. The three amplification actions share a unified structural form. Let $k \in \{\alpha, \beta, \delta\}$ denote the target leader wolf whose weight is to be increased, and let j and l denote the remaining two leader wolves ($j, l \neq k$). The weight update rules following action execution are then:

$$w_k^{\text{new}} = \min(w_k^{\text{old}} + \Delta w, w_{\max}) \quad (22)$$

$$w_j^{\text{new}} = (1 - w_k^{\text{new}}) \cdot \frac{w_j^{\text{old}}}{w_j^{\text{old}} + w_l^{\text{old}}} \quad (23)$$

$$w_l^{\text{new}} = (1 - w_k^{\text{new}}) \cdot \frac{w_l^{\text{old}}}{w_j^{\text{old}} + w_l^{\text{old}}} \quad (24)$$

where Δw is the adjustment step size and $w_{\max}$ is the upper bound imposed on any individual weight. Equation (22) increments the target weight by one step and clips the result at $w_{\max}$, while Equations (23) and (24) proportionally reduce the remaining two weights while preserving their relative ratio. The Stay action applies no modification to any weight, i.e., $w_k^{\text{new}} = w_k^{\text{old}}$ for all $k \in \{\alpha, \beta, \delta\}$. The leader wolf influence weights for each individual are initialized to $1/3$, and are subsequently adjusted in an adaptive manner by the agent according to the real-time state of that individual.

4.1.3. Reward

The reward function serves as the central mechanism that guides the agent's learning direction within the reinforcement learning framework [36], and its formulation directly determines the quality of the policy acquired by the agent. With the minimization of the fitness function as the optimization objective, and adhering to the fundamental principle

that a decrease in fitness corresponds to an improvement in solution quality, the following reward function is designed:

$$R_i^t = \begin{cases} +1, & \text{if } f(X_i^{t+1}) < f(X_i^t) \\ -1, & \text{if } f(X_i^{t+1}) > f(X_i^t) \\ 0, & \text{if } f(X_i^{t+1}) = f(X_i^t) \end{cases} \quad (25)$$

where $f(\cdot)$ is the fitness function to be optimized, and X_i^t and X_i^{t+1} are the position vectors of the i -th individual before and after the weight adjustment action is executed, respectively.

This reward design offers three distinct advantages. First, the reward signal is directly aligned with the optimization objective, requiring no auxiliary rewards or hand-crafted heuristic rules; the semantic interpretation of the reward is unambiguous, with no risk of goal misspecification. Second, the reward values are restricted to the discrete set $\{-1, 0, +1\}$, rendering them independent of the specific scale and units of the fitness function; this scale-invariant formulation enables the acquired decision policy to generalize effectively across optimization problems of different types. Third, the reward is returned immediately following each position update, allowing the agent to rapidly perceive the impact of weight direction adjustments on fitness improvement, which in turn accelerates the convergence of policy learning.

4.2. Training Procedure of the Decision Model

To equip the agent with an effective weight adjustment policy, a systematic training procedure centered on experience replay and a target network is designed, with policy generalizability further enhanced through a diverse collection of training functions.

The training process is organized in episodes, where each episode represents a complete optimization run. The benchmark functions used for training are drawn from the CEC2017 test suite [37], which comprises unimodal, simple multimodal, hybrid, and composition functions. One function was selected from each major category of the CEC2017 test suite for training: F1 (Shifted and Rotated Bent Cigar Function) from the unimodal category, F5 (Shifted and Rotated Rastrigin's Function) from the simple multimodal category, F11 (Hybrid Function 1, $N = 3$) from the hybrid category, and F21 (Composition Function 1, $N = 3$) from the composition category. This setting enables the DQN agent to learn weight adjustment strategies under different search landscapes. We use these functions for training due to their stable and low-cost optimization environment, which avoids noisy reward signals and high computational overhead in intrusion detection datasets.

The offline DQN model was trained for 300 episodes. The four selected functions were used in a cyclic manner during training. Specifically, in each episode, one of the four functions was selected in turn, and one complete DGWO optimization run was performed on the selected function. In each episode, the maximum number of iterations was set to 200, and the population size was set to 30.

During the execution of a single episode, the agent observes the three-dimensional state vector S_i^t of each individual and selects an action using an ϵ -greedy strategy. The exploration rate ϵ was linearly annealed from 0.9 to 0.01 over the first 200 episodes and was subsequently maintained at 0.01 for the remaining 100 episodes. This annealing schedule encouraged the agent to perform more exploratory actions in the early training stage and gradually rely more on the learned Q-value policy in the later stage. The resulting transition tuple (s, a, r, s') is stored in the experience replay buffer. Batches are randomly sampled to update network parameters, breaking temporal correlations and improving training stability. Training targets are constructed from the Bellman optimality equation using the target network, and parameters are updated by minimizing the mean squared error loss

defined in Equation (26). After training, the online network parameters are saved as θ for deployment in DGWO.

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (Q_{\text{target}}(s_i, a_i) - Q_{\text{eval}}(s_i, a_i))^2 \quad (26)$$

To evaluate the learning behavior of the DQN decision model, Figure 1 presents the cumulative episode reward and its 20-episode moving average during offline training. The raw rewards fluctuate because the four functions are cyclically selected and exploratory actions are performed under the ϵ -greedy strategy. Nevertheless, the moving-average reward increases gradually and becomes relatively stable in the later episodes, indicating that the agent learns an effective leader-weight adjustment policy within 300 training episodes.

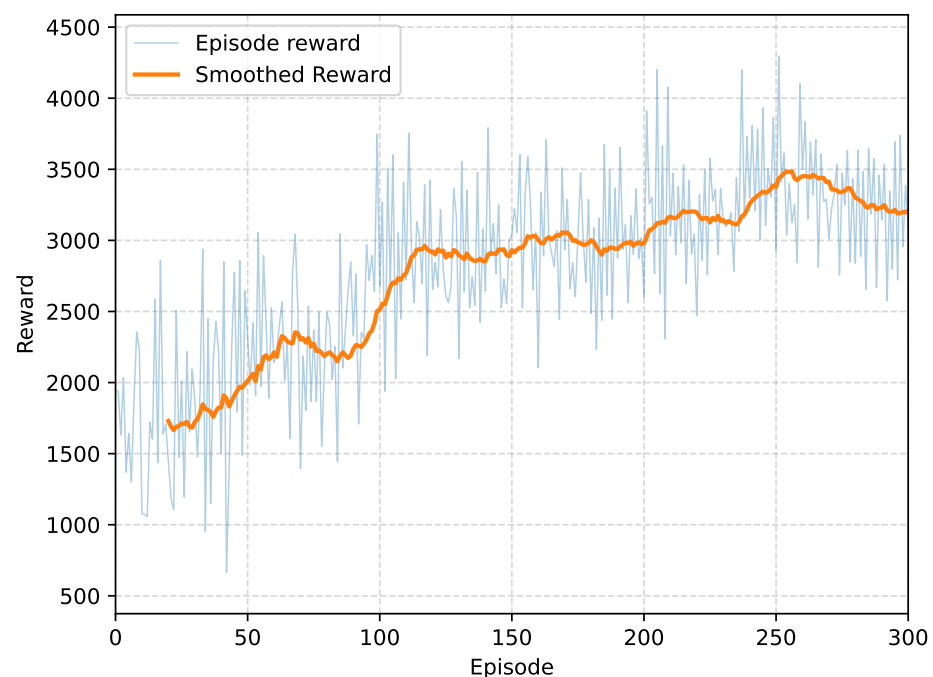


Figure 1. Reward convergence of the DQN decision model.

4.3. DGWO

Drawing upon the components introduced in the preceding sections, the complete DGWO algorithm is presented in this section. The overall procedure is illustrated in Figure 2 and comprises three principal phases.

In the initialization phase, the algorithm loads the pre-trained decision model θ and randomly initializes the population $P = \{X_i \mid i = 1, 2, \dots, PS\}$ within the search space, evaluates the fitness of each individual, ranks the population according to fitness, and identifies the α , β , and δ wolves. The leader wolf influence weights for each individual are initialized to $w_{\alpha,i} = w_{\beta,i} = w_{\delta,i} = 1/3$, and a fitness history queue $\mathcal{H}_i$ of length h is established for each individual.

Upon completion of initialization, the algorithm enters the dynamic search phase. At the beginning of each iteration, the current population diversity div^t is computed according to Equation (20). For each individual X_i in the population, the fitness rank rank_i and stagnation degree stag_i are calculated according to Equations (17) and (18), respectively. The resulting three-dimensional state vector S_i^t , as defined in Equation (16), is then constructed and fed into the pre-trained decision model to obtain the optimal weight adjustment action a_i^t for the current state. Based on this action, the leader wolf weights of

individual X_i are adaptively updated according to Equations (22)–(24). Subsequently, the three guided candidate positions $\vec{X}_1$, $\vec{X}_2$, and $\vec{X}_3$ produced by the respective leader wolves are computed according to Equations (6)–(11), and the final position update is carried out as a weighted combination of these candidates:

$$\vec{X}_i(t+1) = w_{\alpha,i} \cdot \vec{X}_1 + w_{\beta,i} \cdot \vec{X}_2 + w_{\delta,i} \cdot \vec{X}_3 \quad (27)$$

where $\vec{X}_1$, $\vec{X}_2$, and $\vec{X}_3$ are the guided candidate position vectors contributed by the α , β , and δ wolves for individual X_i , respectively, and $w_{\alpha,i}$, $w_{\beta,i}$, $w_{\delta,i}$ are the leader wolf influence weights dynamically determined by the DQN decision model for individual X_i . Following the position update, the fitness of the new position is evaluated and the history queue $\mathcal{H}_i$ is updated accordingly. At the end of each iteration, the population is re-ranked and the α , β , and δ wolves are updated. When the number of iterations reaches $I_{\max}$, the algorithm terminates and returns the position of the α wolf as the best solution found.

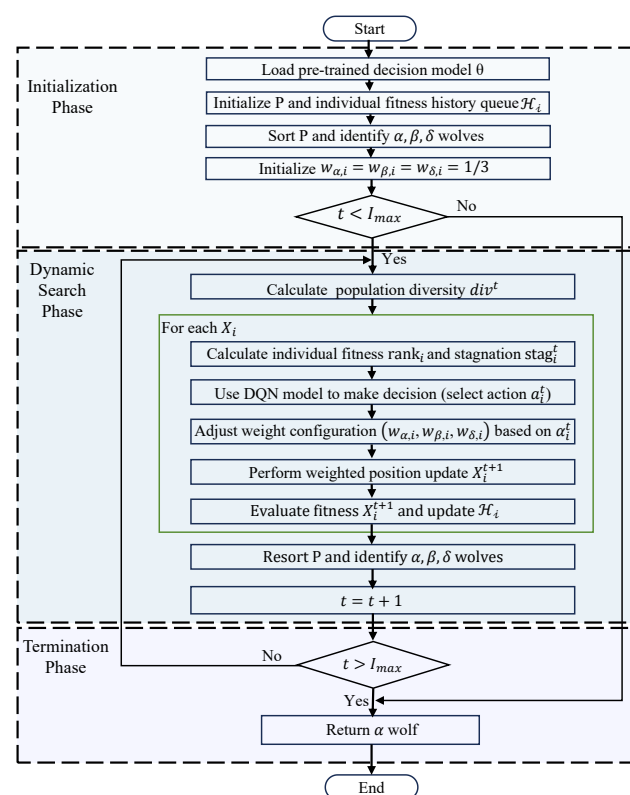


Figure 2. Flowchart of the DGWO.

4.4. Feature Selection Using the Proposed DGWO

FS is a critical step in the construction of intrusion detection systems, aiming to identify the most discriminative feature subset from the original feature pool in order to enhance detection performance, reduce computational complexity, and improve the generalization capability of the classification model. This section describes in detail how the DGWO algorithm is applied to the feature selection task in the context of network intrusion detection. The overall framework of the proposed DGWO-based feature selection method for network intrusion detection is illustrated in Figure 3.

The fitness function serves as the core criterion for assessing the quality of candidate solutions, guiding the optimization algorithm in its search for the optimal configuration within the solution space. In the feature selection context, an ideal solution should simultaneously satisfy two objectives: achieving high classification accuracy and encompassing as

few features as possible. To pursue this dual-objective optimization, the fitness function is formulated as follows:

$$Fitness = \lambda_1 \times Error + \lambda_2 \times \frac{SF}{NF} \quad (28)$$

where *Error* denotes the classification error rate, *SF* is the number of selected features, and *NF* is the total number of features. The weights satisfy $\lambda_1 + \lambda_2 = 1$, with $\lambda_1 = 0.99$ and $\lambda_2 = 0.01$ to prioritize classification performance while encouraging smaller feature subsets. The objective is to minimize the fitness value.

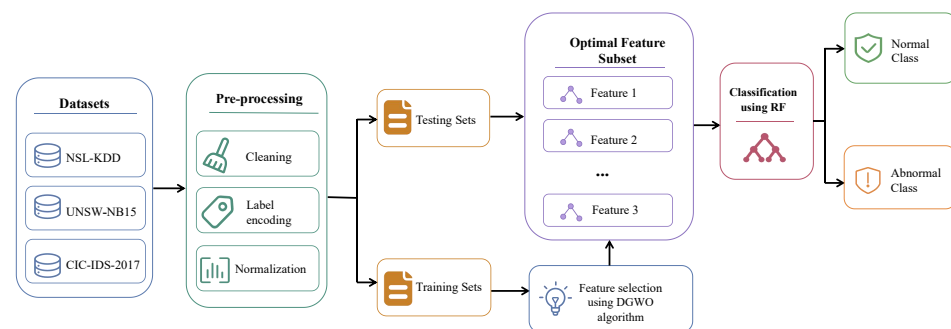


Figure 3. Overall framework of the DGWO-based feature selection method for network intrusion detection.

Data preprocessing constitutes an indispensable component of the machine learning pipeline, as its quality directly governs the performance of downstream models. The preprocessing workflow adopted in this study encompasses three core steps: data cleaning, feature encoding, and data normalization. To prevent data leakage, the training and testing sets were separated before data-dependent preprocessing. The encoding mappings and normalization parameters were fitted using only the training set and then applied unchanged to the testing set. In the data cleaning step, sample records containing missing values are removed alongside duplicate instances, thereby improving data quality and mitigating the risk of model overfitting. For feature encoding, categorical attributes present in the intrusion detection datasets, including protocol type, service type, and connection flag, are mapped to consecutive integers via label encoding, converting them into a numerically tractable representation. The original class labels were transformed into binary labels, where normal traffic was labeled as 0 and all attack types were uniformly labeled as 1. Finally, given the substantial variation in the value ranges across different features, all feature values are uniformly scaled to the interval $[0, 1]$ through min-max normalization, as expressed in Equation (29):

$$X_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \quad (29)$$

where X is the original feature value, $X_{\min}$ and $X_{\max}$ are the minimum and maximum values of that feature across the entire dataset, respectively, and X_{norm} is the normalized feature value. Through these preprocessing steps, noise and inconsistencies in the raw data are eliminated, feature scales are unified, and favorable conditions are established for the efficient operation of both the feature selection algorithm and the classification model.

In the DGWO-based feature selection framework, each grey wolf individual represents a candidate solution vector whose length equals the total number of features NF . Since feature selection requires each feature to be assigned one of two binary states, namely selected or discarded, the continuous solution vector must be discretized into a binary feature selection scheme. To this end, a threshold-based binarization method is employed,

whereby each continuous position value is compared against a fixed threshold to determine whether the corresponding feature is included:

$$B_{i,j} = \begin{cases} 1, & \text{if } X_{i,j} \geq \text{threshold} \\ 0, & \text{otherwise} \end{cases} \quad (30)$$

where $\text{threshold} = 0.5$. Specifically, $B_{i,j} = 1$ indicates that the j -th feature is selected, whereas $B_{i,j} = 0$ indicates that it is excluded from the candidate subset. For fitness evaluation, a K-Nearest Neighbor (KNN) classifier is employed owing to its low computational overhead, which makes it well-suited for the frequent fitness evaluations required during iterative optimization. It should be emphasized that KNN is used only as the internal fitness evaluator during the feature subset search process, rather than as the final classifier for performance assessment. To reduce overfitting, five-fold cross-validation is performed within the training set. The average validation error across the five folds is used as the classification error in the fitness function. The complete procedure of the DGWO-based feature selection algorithm is presented in Algorithm 1.

Algorithm 1 DGWO-based Feature Selection for Intrusion Detection Systems

Input: Preprocessed training set D_{train} , population size PS , maximum iterations I_{max} , pre-trained DQN model θ

Output: Optimal feature subset F^*

- 1: Load dataset D_{train} and pre-trained DQN model θ
 - 2: Randomly initialize population $P = \{X_i \mid i = 1, \dots, PS\}$
 - 3: Binarize each X_i to B_i via Equation (30)
 - 4: Evaluate fitness $f(X_i)$ via Equation (28) using KNN on D_{train} with features indicated by B_i
 - 5: Sort population P by fitness and identify α, β, δ wolves
 - 6: Initialize $w_{\alpha,i} = w_{\beta,i} = w_{\delta,i} = 1/3$ and $\mathcal{H}_i = [f(X_i)]$ for all i
 - 7: **while** $t \leq I_{\text{max}}$ **do**
 - 8: Compute population diversity div^t via Equation (20)
 - 9: **for** $i = 1$ to PS **do**
 - 10: Compute rank_i and stag_i^t via Equations (17)–(18)
 - 11: Construct state $S_i^t = [\text{rank}_i, \text{stag}_i^t, \text{div}^t]$ via Equation (16)
 - 12: Select action $a_i^t = \arg \max_a Q_\theta(S_i^t, a)$
 - 13: Update weights $(w_{\alpha,i}, w_{\beta,i}, w_{\delta,i})$ via Equations (22)–(24)
 - 14: Compute guided positions $\bar{X}_1, \bar{X}_2, \bar{X}_3$ via Equation (9)–(11)
 - 15: Update position $X_i(t+1) = w_{\alpha,i} \bar{X}_1 + w_{\beta,i} \bar{X}_2 + w_{\delta,i} \bar{X}_3$
 - 16: Binarize $X_i(t+1)$ to B_i^{t+1} via Equation (30)
 - 17: Evaluate $f(X_i(t+1))$ via Equation (28) using KNN on D_{train} with features indicated by B_i^{t+1}
 - 18: Update fitness history $\mathcal{H}_i$
 - 19: **end for**
 - 20: Sort population P by fitness and update α, β, δ wolves
 - 21: $t = t + 1$
 - 22: **end while**
-

4.5. Complexity Analysis

The computational cost of DGWO consists of offline DQN training and online feature selection. During offline training, population evolution and state construction require $O(PSd)$ operations per iteration, while the additional cost associated with action selection, experience replay, and network updating is denoted by C_{train} . Therefore, the offline training time complexity is $O(ET_{\text{tr}}(PSd + C_{\text{train}}))$, where E is the number of training episodes, T_{tr} is the number of iterations per episode, PS is the population size, and d is the benchmark dimension.

During online feature selection, the pre-trained DQN performs only forward inference without gradient calculation or parameter updating. Population-state calculation and position updating require $O(PS \times NF)$ operations per iteration, while DQN-based action selection requires $O(PSC_D)$, where C_D denotes the cost of one DQN forward pass. Therefore, the online time complexity of DGWO is $O(TPS(NF + C_D))$, where T is the maximum number of iterations and NF is the total number of features. Since the DQN architecture is fixed, C_D can be regarded as a constant. Thus, DGWO retains the same asymptotic time complexity, $O(TPSNF)$, as the standard GWO, although it introduces additional constant computational overhead.

In terms of space complexity, the offline training stage requires storage for the population, individual fitness-history queues, experience replay buffer, and DQN parameters. Let h denote the length of the fitness-history queue, R denote the capacity of the replay buffer, d_s denote the state dimension, and P_D denote the number of DQN parameters. The offline space complexity is therefore $O(PSd + PSh + Rd_s + P_D)$. During online feature selection, the replay buffer and network optimization information are no longer required. The main storage components include the population, individual fitness histories, state and weight information, and the pre-trained DQN parameters. Thus, the online space complexity is $O(PSNF + PSh + PSd_s + P_D)$. Since h , d_s , and the DQN architecture are fixed, the asymptotic online space complexity remains $O(PSNF)$, which is of the same order as that of GWO, although DGWO requires a larger constant amount of storage.

The DQN is trained only once, and the resulting decision model can be reused in subsequent feature selection runs. Therefore, the offline training cost is separated from the online computational cost when evaluating the efficiency of DGWO. The additional computational overhead remains controllable, making DGWO feasible for practical deployment in feature selection tasks with fixed computing resources.

5. Experimentation and Results

5.1. Datasets

Three publicly available benchmark datasets widely adopted in the intrusion detection domain, namely NSL-KDD, UNSW-NB15, and CIC-IDS-2017, are employed to evaluate the performance of the proposed method. The NSL-KDD dataset [38] is an improved version of the KDDCUP99 dataset, addressing issues of redundant and duplicate records to mitigate classifier bias toward high-frequency instances. It comprises 125,973 training records and 22,543 test records, each described by 41 features, and encompasses normal traffic along with four attack categories: DoS, R2L, U2R, and Probe. The UNSW-NB15 dataset [39], developed at the University of New South Wales, contains 49 features and provides 175,341 training records and 82,332 test records covering nine attack types, including backdoor, fuzzing, exploits, and shellcode, among others. The CIC-IDS-2017 dataset [40], generated by the Canadian Institute for Cybersecurity using the CICFlowMeter tool, consists of approximately 2.84 million traffic records, each characterized by 78 features across eight attack categories, and is widely regarded as one of the most representative modern benchmarks in the intrusion detection field. Considering the large scale of CIC-IDS-2017, 10% of the original records were randomly sampled for the experiments. The sampled data were then divided into training and testing sets at a ratio of 8:2. The training set was used for feature selection, while the testing set was used only for the final classification performance evaluation.

5.2. Experimental Setup

In this study, all experiments were conducted using Python 3.6.15 on a 64-bit Windows 10 operating system, equipped with an 11th-generation Intel Core i5-1135G7 processor running at 2.40 GHz and 16 GB of memory. After the feature selection process, the classifi-

cation performance of the optimal feature subset obtained by each algorithm was verified using a Random Forest (RF) classifier [41] implemented in the scikit-learn library. The RF classifier was used as the final evaluation model to assess the detection performance of the selected feature subsets under identical evaluation conditions. For all metaheuristic algorithms, the maximum number of iterations was set to 100, the population size was set to 30, this configuration is commonly used in metaheuristic feature selection experiments. Each algorithm was independently executed 30 times to reduce the impact of randomness and ensure statistical reliability. The average performance over all runs is reported for comparison. The proposed DGWO algorithm is benchmarked against seven competitive feature selection methods, including Particle Swarm Optimization (PSO) [9], Genetic Algorithm (GA) [42], Grey Wolf Optimizer (GWO) [13], Whale Optimization Algorithm (WOA) [10], Butterfly Optimization Algorithm (BOA) [43], RIME [44], and Sparrow Search Algorithm (SSA) [12]. The parameter settings of all compared algorithms are summarized in Table 1.

Table 1. Parameter settings.

Algorithm	Parameter	Value
PSO	$w_{\max}, w_{\min}, c_1, c_2$	0.8, 0.4, 2, 2
GA	crossover rate, mutation rate	0.67, 0.33
GWO	a	linearly decreased from 2 to 0
WOA	a, b	linearly decreased from 2 to 0, 1
BOA	power exponent, sensory modality	0.8, 0.1
SSA	PD, SD	20%, 10%
RIME	w	5
DGWO	learning rate	0.01
	discount factor γ	0.9
	ε -greedy	annealed linearly from 0.9 to 0.01
	batch size	32
	replay memory size	1000
	Δw	0.1
	$w_{\max}$	0.9

5.3. Evaluation Metrics

In this study, Accuracy, Precision, Recall, and F1-score are used to evaluate classification performance based on the confusion matrix. The confusion matrix consists of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Accuracy measures the proportion of correctly classified samples, as defined in Equation (31). Precision represents the proportion of actual attacks among all samples predicted as attacks, as given in Equation (32), whereas Recall measures the proportion of actual attacks that are correctly detected, as defined in Equation (33). The F1-score, shown in Equation (34), is the harmonic mean of Precision and Recall and provides a balanced evaluation of intrusion detection performance, particularly for imbalanced datasets.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (31)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (32)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (33)$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (34)$$

5.4. Experimental Results and Analysis

5.4.1. Results on NSL-KDD Dataset

Table 2 presents the comparative results of all algorithms on the NSL-KDD dataset. As shown, DGWO consistently outperforms all competing algorithms across every evaluation metric. Specifically, DGWO achieves an accuracy of 0.9359, a precision of 0.9454, a recall of 0.9406, and an F1-score of 0.9356, while selecting an average of only 12.5 features, representing a reduction of 33.86% relative to the standard GWO. These results indicate that DGWO is capable of substantially compressing the feature space while simultaneously maintaining superior classification performance.

Figure 4a illustrates the accuracy and precision of each algorithm on the NSL-KDD dataset. Compared with GWO, DGWO achieves improvements of 0.72% in accuracy and 0.77% in precision. Although the standard deviation of DGWO is marginally higher than that of certain competing algorithms, its overall performance remains the best among all evaluated methods. Figure 4b reports the recall and F1-score comparisons across all algorithms. DGWO attains the highest values on both metrics, demonstrating a clear advantage over all other methods. Compared with the second-best algorithm RIME, DGWO not only maintains a competitive recall rate but also achieves a further gain in F1-score, reflecting a more favorable balance between precision and recall. The consistently well-rounded performance of DGWO across all metrics, particularly its strong precision and recall, makes it especially well-suited for application scenarios that demand both classification accuracy and comprehensive detection coverage.

Table 2. Comparison of algorithms on NSL-KDD dataset in terms of classification performance and the number of selected features.

Method	Accuracy	Std	Precision	Std	Recall	Std	F1-Score	Std	SF
PSO	0.9245	0.0072	0.9270	0.0068	0.9275	0.0105	0.9270	0.0146	18.6
GA	0.9218	0.0122	0.9260	0.0093	0.9210	0.0100	0.9204	0.0129	23.2
WOA	0.9292	0.0042	0.9326	0.0054	0.9318	0.0045	0.9286	0.0056	19.5
BOA	0.9308	0.0058	0.9245	0.0067	0.9372	0.0047	0.9216	0.0053	20.8
SSA	0.9256	0.0047	0.9375	0.0047	0.9350	0.0045	0.9320	0.0060	20.4
RIME	0.9305	0.0061	0.9432	0.0052	0.9301	0.0042	0.9337	0.0062	15.3
GWO	0.9287	0.0041	0.9377	0.0048	0.9358	0.0049	0.9295	0.0064	18.9
DGWO	0.9359	0.0046	0.9454	0.0050	0.9406	0.0042	0.9356	0.0050	12.5

Bold values indicate the best result in each column.

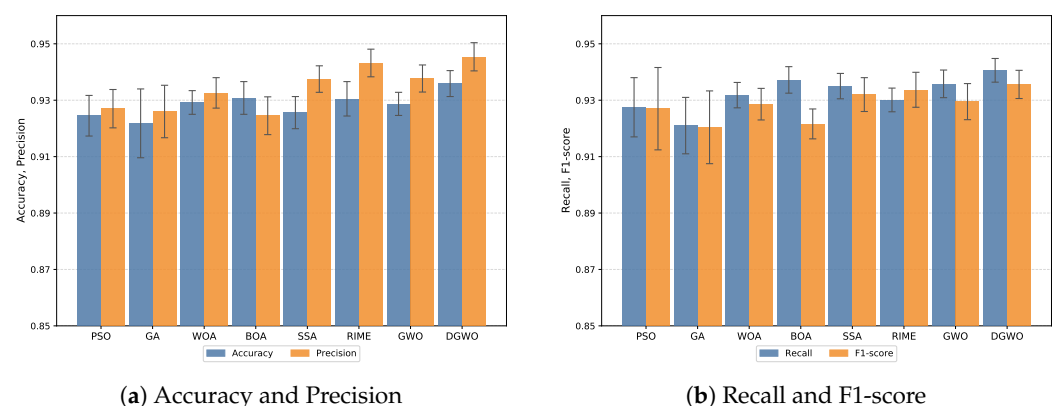


Figure 4. Performance comparison on NSL-KDD dataset.

5.4.2. Results on UNSW-NB15 Dataset

As reported in Table 3, DGWO achieves marked superiority across all evaluation metrics on the UNSW-NB15 dataset, attaining an accuracy of 0.9340, a precision of 0.9313, a

recall of 0.9227, and an F1-score of 0.9284. The algorithm selects an average of 13.2 features, corresponding to a 27.87% reduction compared with GWO.

Figure 5a presents the accuracy and precision comparisons among the eight algorithms. Relative to GWO, DGWO yields improvements of 2.25% in accuracy and 2.04% in precision. Figure 5b depicts the recall and F1-score comparisons. DGWO attains the highest recall among all algorithms by a notable margin. Its F1-score likewise surpasses all competing methods, confirming that DGWO achieves a well-calibrated balance between precision and recall. Beyond its classification performance, DGWO also exhibits strong stability: its standard deviations across all evaluation metrics are consistently among the lowest of all compared algorithms, rendering it particularly suitable for deployment environments that require both high detection accuracy and sustained system reliability.

Table 3. Comparison of algorithms on UNSW-NB15 dataset in terms of classification performance and the number of selected features.

Method	Accuracy	Std	Precision	Std	Recall	Std	F1-Score	Std	SF
PSO	0.9269	0.0073	0.9295	0.0076	0.9151	0.0071	0.9211	0.0078	18.6
GA	0.8909	0.0072	0.9191	0.0060	0.8761	0.0083	0.8812	0.0086	23.8
WOA	0.9264	0.0043	0.9299	0.0073	0.9140	0.0067	0.9203	0.0055	20.1
BOA	0.9206	0.0045	0.9288	0.0044	0.9093	0.0041	0.9150	0.0044	21.7
SSA	0.9150	0.0083	0.9112	0.0067	0.9027	0.0091	0.9085	0.0082	17.4
RIME	0.9206	0.0046	0.9311	0.0047	0.9087	0.0040	0.9146	0.0036	17.9
GWO	0.9115	0.0043	0.9109	0.0045	0.8995	0.0041	0.9051	0.0039	18.3
DGWO	0.9340	0.0042	0.9313	0.0044	0.9227	0.0039	0.9284	0.0037	13.2

Bold values indicate the best result in each column.

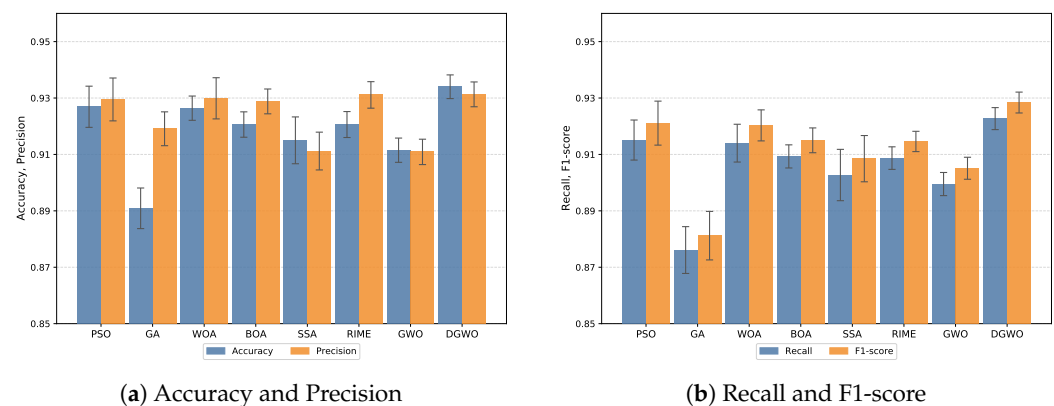


Figure 5. Performance comparison on UNSW-NB15 dataset.

5.4.3. Results on CIC-IDS-2017 Dataset

Table 4 summarizes the performance comparison of the eight algorithms on the CIC-IDS-2017 dataset. As shown, DGWO achieves the best results across all evaluation metrics, recording an accuracy of 0.9484, a precision of 0.9551, a recall of 0.9472, and an F1-score of 0.9511, while selecting an average of 18.2 features, which represents a reduction of 32.84% relative to GWO.

Figure 6a presents the accuracy and precision results for all eight algorithms. Compared with the standard GWO, DGWO raises accuracy from 0.9257 to 0.9484, an improvement of 2.27%, and precision from 0.9339 to 0.9551, an improvement of 2.12%. Figure 6b displays the recall and F1-score comparisons. DGWO achieves the highest recall (0.9472) and F1-score (0.9511) among all methods, with the F1-score exceeding that of GWO by 2.18%. Notably, the F1-score standard deviation of DGWO (0.0020) is the lowest across all

algorithms, indicating that DGWO delivers particularly stable and consistent classification performance.

Overall, these results demonstrate that DGWO maintains superior predictive performance while effectively reducing feature dimensionality, highlighting a better balance between accuracy and compactness. Given that the CIC-IDS-2017 dataset is characterized by higher feature dimensionality and a more diverse range of attack categories, the strong performance of DGWO on this dataset further substantiates its search effectiveness and generalization capability in large-scale, high-dimensional feature spaces.

Table 4. Comparison of algorithms on CIC-IDS-2017 dataset in terms of classification performance and the number of selected features.

Method	Accuracy	Std	Precision	Std	Recall	Std	F1-Score	Std	SF
PSO	0.9244	0.0037	0.9312	0.0033	0.9231	0.0061	0.9271	0.0049	26.4
GA	0.9338	0.0034	0.9396	0.0032	0.9324	0.0057	0.9360	0.0051	28.9
WOA	0.9218	0.0061	0.9287	0.0052	0.9205	0.0067	0.9246	0.0063	24.7
BOA	0.9294	0.0033	0.9357	0.0029	0.9286	0.0061	0.9321	0.0054	27.3
SSA	0.9181	0.0038	0.9003	0.0042	0.9174	0.0057	0.9087	0.0046	29.6
RIME	0.9401	0.0021	0.9474	0.0024	0.9398	0.0062	0.9435	0.0052	25.8
GWO	0.9257	0.0026	0.9339	0.0030	0.9248	0.0041	0.9293	0.0024	27.1
DGWO	0.9484	0.0024	0.9551	0.0028	0.9472	0.0039	0.9511	0.0020	18.2

Bold values indicate the best result in each column.

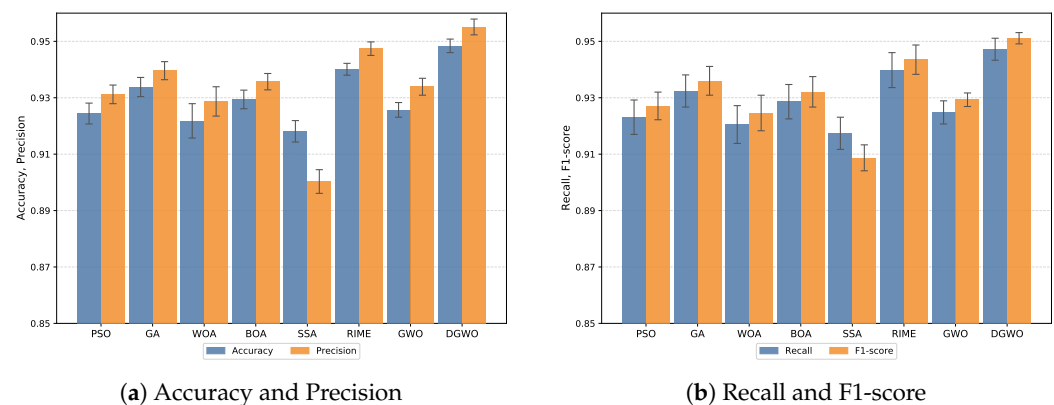


Figure 6. Performance comparison on CIC-IDS-2017 dataset.

5.4.4. Statistical Significance Analysis

To assess the statistical significance of performance differences, non-parametric tests were conducted on the F1-score results from 30 independent runs. Since F1-score balances Precision and Recall, it was used as the primary metric for statistical comparison. The Wilcoxon rank-sum test was applied for pairwise comparisons between DGWO and each competing algorithm at the 0.05 significance level. As shown in Table 5, “+”, “=”, and “−” indicate that DGWO is significantly better than, statistically comparable to, and significantly worse than the corresponding algorithm, respectively. DGWO significantly outperformed the competing methods in 20 of the 21 pairwise comparisons, providing statistical evidence for the overall performance advantage of DGWO.

Friedman ranking was further used to assess the overall ordering of all algorithms based on the mean F1-score values across the three datasets. For each dataset, algorithms were ranked in descending order of F1-score, and the average rank was then calculated. As shown in Table 6, DGWO obtains the lowest average rank and final rank among all compared algorithms. This further demonstrates the consistent advantage of DGWO across different intrusion detection datasets.

Table 5. Wilcoxon rank-sum test results.

Dataset	PSO	GA	WOA	BOA	SSA	RIME	GWO
NSL-KDD	$6.84 \times 10^{-3}+$	$1.27 \times 10^{-4}+$	$2.15 \times 10^{-3}+$	$8.91 \times 10^{-5}+$	$4.72 \times 10^{-2}+$	$1.64 \times 10^{-1}=$	$3.86 \times 10^{-3}+$
UNSW-NB15	$7.46 \times 10^{-4}+$	$3.02 \times 10^{-11}+$	$1.89 \times 10^{-4}+$	$2.37 \times 10^{-6}+$	$4.81 \times 10^{-8}+$	$1.16 \times 10^{-6}+$	$6.42 \times 10^{-10}+$
CIC-IDS-2017	$3.02 \times 10^{-11}+$	$8.73 \times 10^{-10}+$	$1.64 \times 10^{-10}+$	$1.94 \times 10^{-9}+$	$3.02 \times 10^{-11}+$	$2.61 \times 10^{-5}+$	$4.85 \times 10^{-11}+$
+ / = / −	3/0/0	3/0/0	3/0/0	3/0/0	3/0/0	2/1/0	3/0/0

Table 6. Friedman ranking results of the compared algorithms.

Algorithm	Average Rank	Final Rank
DGWO	1.00	1
RIME	3.00	2
PSO	4.67	3
WOA	5.00	4
BOA	5.00	4
GWO	5.33	6
SSA	5.67	7
GA	6.33	8

5.4.5. Time Efficiency

To further quantify the actual computational cost of the compared algorithms, their execution times on the three datasets are reported in Table 7. The time unit is seconds. Since the DQN decision model is trained only once offline and can be reused in subsequent feature selection tasks, only the online feature selection stage is considered in this comparison.

As shown in Table 7, DGWO requires 12,312.84 s, 13,807.25 s, and 22,407.91 s on NSL-KDD, UNSW-NB15, and CIC-IDS-2017, respectively. Compared with GWO, DGWO increases the execution time by 470.17 s, 535.76 s, and 925.55 s on the three datasets, corresponding to relative increases of 3.97%, 4.04%, and 4.31%, respectively. Compared with the other competing algorithms, DGWO still maintains relatively high computational efficiency.

Table 7. Runtime of the compared algorithms on the three datasets.

Dataset	PSO	GA	GWO	WOA	BOA	RIME	SSA	DGWO
NSL-KDD	12,941.04	14,163.83	11,842.67	13,343.69	12,683.50	11,126.89	13,701.97	12,312.84
UNSW-NB15	13,704.81	16,018.69	13,271.49	14,076.41	14,359.75	13,656.36	14,514.37	13,807.25
CIC-IDS-2017	23,979.32	26,165.51	21,482.36	24,640.27	23,372.81	22,148.31	25,263.26	22,407.91

The additional runtime of DGWO mainly arises from individual-state construction, DQN forward inference, and adaptive leader-weight updating. However, during online feature selection, the DQN performs only forward inference without gradient computation, experience replay, or parameter updating. Therefore, the additional computational overhead introduced by the DRL mechanism remains limited. Considering the simultaneous improvements in classification performance and feature reduction reported above, DGWO achieves a favorable trade-off between optimization effectiveness and computational cost.

5.4.6. Convergence Performance Comparison

To further evaluate the convergence behavior of DGWO during feature selection, this section provides a comparative analysis of the convergence curves produced by each algorithm across the three benchmark datasets.

As shown in Figure 7a, DGWO converges faster on the NSL-KDD dataset and achieves the lowest final fitness value of 0.0654. In Figure 7b, several algorithms stagnate at relatively high fitness values on the UNSW-NB15 dataset, whereas DGWO maintains a stable downward trend and reaches the lowest final fitness value of 0.0705. As shown in Figure 7c,

the differences are more pronounced on the higher-dimensional CIC-IDS-2017 dataset, where DGWO maintains lower fitness values throughout the iterations and obtains the lowest final value of 0.0489.

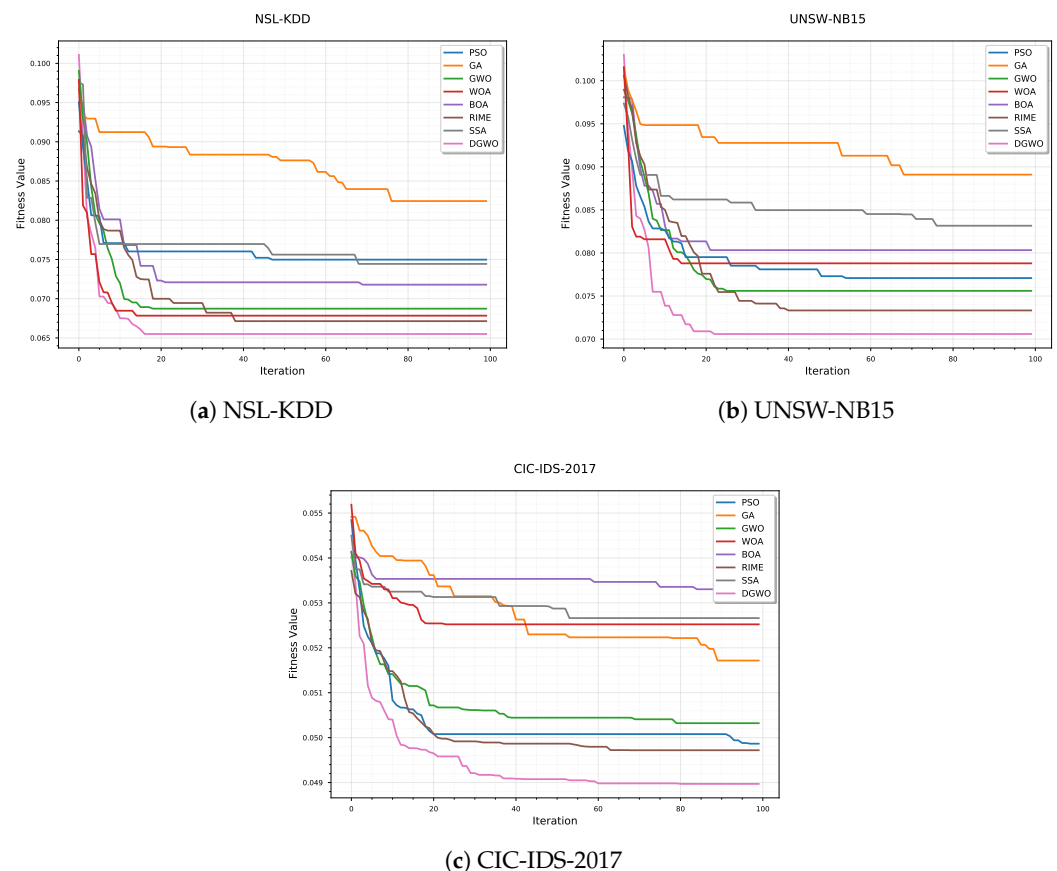


Figure 7. Convergence curves of the DGWO algorithm and other methods.

Benefiting from the pre-trained DQN model's ability to dynamically adjust the influence weights of the three leader wolves, DGWO consistently guides individuals toward higher-quality feature subsets throughout the optimization process. The lower final fitness values obtained on all three datasets further demonstrate the effectiveness of the proposed method in feature selection.

5.4.7. Ablation Study

To assess the contribution of each state component, three degraded variants are constructed by removing individual fitness ranking (rank_i), search stagnation degree (stag_i), and population diversity (div^t), yielding DGWO- rank_i , DGWO- stag_i , and DGWO- div^t , respectively. A fourth variant, DGWO-DQN, replaces the pre-trained DQN with random action selection to isolate the effect of the decision-making mechanism. Results are reported in Table 8.

As shown, removing rank_i leads to a significant increase in the number of selected features across all datasets, reaching increases of 49.6%, 27.3%, and 22.5%, respectively. This indicates that the absence of individual quality information prevents the agent from learning differentiated weight strategies for individuals of varying fitness levels. Removing div^t results in a noticeable decline in F1-score, decreasing by 0.34%, 0.42%, and 0.38% on the three datasets, as the agent loses awareness of global population diversity and becomes less effective in maintaining exploratory capability. Similarly, removing stag_i causes moderate performance degradation, with F1-score reductions of 0.25%, 0.27%, and 0.34%, while

still leading to a notable increase in feature size, suggesting that stagnation information is crucial for identifying trapped individuals and guiding them out of local optima.

Table 8. Ablation study results.

Dataset	Method	Accuracy	Precision	Recall	F1-Score	SF
NSL-KDD	DGWO-DQN	0.9296	0.9369	0.9348	0.9298	17.9
	DGWO-rank _{<i>i</i>}	0.9322	0.9412	0.9371	0.9319	18.7
	DGWO-stag _{<i>i</i>}	0.9337	0.9429	0.9391	0.9333	14.3
	DGWO-div ^{<i>t</i>}	0.9327	0.9419	0.9379	0.9324	16.4
	DGWO	0.9359	0.9454	0.9406	0.9356	12.5
UNSW-NB15	DGWO-DQN	0.9142	0.9131	0.9028	0.9077	17.8
	DGWO-rank _{<i>i</i>}	0.9307	0.9279	0.9194	0.9251	16.8
	DGWO-stag _{<i>i</i>}	0.9315	0.9291	0.9203	0.9259	15.1
	DGWO-div ^{<i>t</i>}	0.9303	0.9274	0.9191	0.9245	15.9
	DGWO	0.9340	0.9313	0.9227	0.9284	13.2
CIC-IDS-2017	DGWO-DQN	0.9274	0.9344	0.9259	0.9300	25.9
	DGWO-rank _{<i>i</i>}	0.9447	0.9514	0.9431	0.9471	22.3
	DGWO-stag _{<i>i</i>}	0.9455	0.9524	0.9443	0.9479	20.6
	DGWO-div ^{<i>t</i>}	0.9451	0.9519	0.9439	0.9475	22.9
	DGWO	0.9484	0.9551	0.9472	0.9511	18.2

Bold values indicate the best result in each column.

The DGWO-DQN variant exhibits the worst overall performance among all configurations, with F1-scores decreasing to 0.9298, 0.9077, and 0.9300 on the three datasets, respectively, and the highest average number of selected features across all variants, reaching 17.9, 17.8, and 25.9. This indicates that without an intelligent decision-making policy, the three-dimensional state representation cannot be effectively translated into adaptive weight control, and random actions may further disrupt the convergence behavior of the optimizer. Overall, the complete DGWO achieves the best balance between classification performance and feature compactness, demonstrating that rank_{*i*}, stag_{*i*}, div^{*t*}, and the DQN-based decision mechanism are all essential components of the proposed framework.

6. Conclusions

In this work, we proposed a Deep Reinforcement Learning-driven Grey Wolf Optimizer for feature selection in network intrusion detection systems. Specifically, a weight adjustment decision model is first pre-trained offline using deep reinforcement learning and then deployed to dynamically regulate the influence weights of the α , β , and δ wolves during the optimization process. This mechanism enables adaptive guidance of the search behavior and effectively enhances the algorithm's optimization capability.

Experiments on the NSL-KDD, UNSW-NB15, and CIC-IDS-2017 datasets demonstrate that DGWO consistently outperforms seven competitive feature selection methods in terms of classification performance. Specifically, DGWO achieves the highest accuracy across all datasets, reaching 93.59%, 93.40%, and 94.84% on NSL-KDD, UNSW-NB15, and CIC-IDS-2017, respectively. Compared with the standard GWO, DGWO improves accuracy by approximately 0.72%, 2.25%, and 2.27%, while also achieving consistent improvements in F1-score of 0.61%, 2.33%, and 2.18%, respectively. In addition, DGWO significantly outperforms the standard GWO in terms of the average number of selected features. Specifically, the average feature subset size is reduced from 18.9 to 12.5 on NSL-KDD, from 18.3 to 13.2 on UNSW-NB15, and from 27.1 to 18.2 on CIC-IDS-2017, demonstrating that DGWO is capable of consistently producing more compact feature subsets while maintaining strong discriminative power. Ablation studies further validate the effectiveness of the key components of DGWO, particularly the DRL-based dynamic weight adjustment strategy and the designed state representation, both of which contribute to improved optimization perfor-

mance and convergence behavior. These results confirm the effectiveness of incorporating adaptive decision-making into swarm intelligence optimization.

In future work, the proposed DGWO framework can be further extended in several directions. First, the current method can be generalized to multi-objective feature selection scenarios by jointly optimizing classification performance and additional objectives such as computational cost and model robustness. Finally, the proposed framework can be integrated with other advanced swarm intelligence or deep learning models to further enhance its adaptability in complex and dynamic intrusion detection scenarios.

Author Contributions: Q.Z. conducted all experiments, prepared all figures, and wrote and revised the manuscript. T.S. and J.X. reviewed the manuscript and provided valuable feedback for revision. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Zhejiang Provincial Natural Science Foundation of China under Grant No. LY22F020019, Public-welfare Technology Application Research of Zhejiang Province in China under Grant LGG22F020032, the Zhejiang Science and Technology Plan Project under Grant No. 2022C01045, and the National Natural Science Foundation of China under Grants (Nos. 61101111 and 62132014).

Data Availability Statement: The data presented in this study are available in NSL-KDD, UNSW-NB15, and CIC-IDS-2017. These data were derived from the following resources available in the public domain: <https://ieee-dataport.org/documents/nsl-kdd-0> (accessed on 3 October 2025), <https://research.unsw.edu.au/projects/unswnb15-dataset> (accessed on 3 October 2025), and <https://www.unb.ca/cic/datasets/ids-2017.html> (accessed on 3 October 2025).

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Thakkar, A.; Lohiya, R. A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions. *Artif. Intell. Rev.* **2022**, *55*, 453–563.
2. Buczak, A.L.; Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 1153–1176. [\[CrossRef\]](#)
3. Injadat, M.; Moubayed, A.; Nassif, A.B.; Shami, A. Multi-stage optimized machine learning framework for network intrusion detection. *IEEE Trans. Netw. Serv. Manag.* **2020**, *18*, 1803–1816. [\[CrossRef\]](#)
4. Di Mauro, M.; Galatro, G.; Fortino, G.; Liotta, A. Supervised feature selection techniques in network intrusion detection: A critical review. *Eng. Appl. Artif. Intell.* **2021**, *101*, 104216. [\[CrossRef\]](#)
5. Hussain, K.; Salleh, M.N.M.; Cheng, S.; Shi, Y. On the exploration and exploitation in popular swarm-based metaheuristic algorithms. *Neural Comput. Appl.* **2019**, *31*, 7665–7683.
6. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. [\[CrossRef\]](#)
7. Miao, J.; Niu, L. A survey on feature selection. *Procedia Comput. Sci.* **2016**, *91*, 919–926. [\[CrossRef\]](#)
8. Korani, W.; Mouhoub, M. Review on nature-inspired algorithms. *SN Oper. Res. Forum.* **2021**, *2*, 36. [\[CrossRef\]](#)
9. Kennedy, J.; Eberhart, R. Particle swarm optimization. In *Proceedings of the ICNN'95-International Conference on Neural Networks*; IEEE: New York, NY, USA, 1995; Volume 4, pp. 1942–1948.
10. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [\[CrossRef\]](#)
11. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [\[CrossRef\]](#)
12. Xue, J.; Shen, B. A novel swarm intelligence optimization approach: Sparrow search algorithm. *Syst. Sci. Control Eng.* **2020**, *8*, 22–34. [\[CrossRef\]](#)
13. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [\[CrossRef\]](#)
14. Too, J.; Abdullah, A.R.; Mohd Saad, N.; Mohd Ali, N.; Tee, W. A new competitive binary grey wolf optimizer to solve the feature selection problem in EMG signals classification. *Computers* **2018**, *7*, 58. [\[CrossRef\]](#)
15. Emary, E.; Zawbaa, H.M.; Grosan, C.; Hassenian, A.E. Feature subset selection approach by gray-wolf optimization. In *Proceedings of the Afro-European Conference for Industrial Advancement: Proceedings of the First International Afro-European Conference for Industrial Advancement AECIA 2014*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 1–13.

16. Long, W.; Jiao, J.; Liang, X.; Tang, M. An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization. *Eng. Appl. Artif. Intell.* **2018**, *68*, 63–80. [\[CrossRef\]](#)
17. Gupta, S.; Deep, K. A novel random walk grey wolf optimizer. *Swarm Evol. Comput.* **2019**, *44*, 101–112. [\[CrossRef\]](#)
18. Rodríguez, L.; Castillo, O.; Soria, J.; Melin, P.; Valdez, F.; Gonzalez, C.I.; Martinez, G.E.; Soto, J. A fuzzy hierarchical operator in the grey wolf optimizer algorithm. *Appl. Soft Comput.* **2017**, *57*, 315–328. [\[CrossRef\]](#)
19. Maldonado, J.; Riff, M.C.; Neveu, B. A review of recent approaches on wrapper feature selection for intrusion detection. *Expert Syst. Appl.* **2022**, *198*, 116822. [\[CrossRef\]](#)
20. Dokeroglu, T.; Deniz, A.; Kiziloz, H.E. A comprehensive survey on recent metaheuristics for feature selection. *Neurocomputing* **2022**, *494*, 269–296. [\[CrossRef\]](#)
21. Zare, F.; Mahmoudi-Nasr, P. Feature engineering methods in intrusion detection system: A performance evaluation. *Int. J. Eng. Trans. A Basics* **2023**, *36*, 1343–1353. [\[CrossRef\]](#)
22. Kavitha, C.; M., S.; Gadekallu, T.R.; K., N.; Kavim, B.P.; Lai, W.-C. Filter-based ensemble feature selection and deep learning model for intrusion detection in cloud computing. *Electronics* **2023**, *12*, 556. [\[CrossRef\]](#)
23. Pourardbilkah, Y.; Shirvani, M.H.; Taheri, J. A survey study on meta-heuristic-based feature selection approaches of intrusion detection systems in distributed networks. *Comput. Stand. Interfaces* **2025**, *96*, 104074.
24. Kunhare, N.; Tiwari, R.; Dhar, J. Particle swarm optimization and feature selection for intrusion detection system. *Sādhanā* **2020**, *45*, 109. [\[CrossRef\]](#)
25. Alazzam, H.; Sharieh, A.; Sabri, K.E. A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer. *Expert Syst. Appl.* **2020**, *148*, 113249. [\[CrossRef\]](#)
26. Safaldin, M.; Otair, M.; Abualigah, L. Improved binary gray wolf optimizer and SVM for intrusion detection system in wireless sensor networks. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 1559–1576.
27. Alzaqebah, A.; Aljarah, I.; Al-Kadi, O.; Damaševičius, R. A modified grey wolf optimization algorithm for an intrusion detection system. *Mathematics* **2022**, *10*, 999. [\[CrossRef\]](#)
28. Kareem, S.S.; Mostafa, R.R.; Hashim, F.A.; El-Bakry, H.M. An effective feature selection model using hybrid metaheuristic algorithms for IoT intrusion detection. *Sensors* **2022**, *22*, 1396. [\[CrossRef\]](#) [\[PubMed\]](#)
29. Sezgin, A.; Ulaş, M.; Boyacı, A. Multi-Objective Feature Selection for Intrusion Detection Systems: A Comparative Analysis of Bio-Inspired Optimization Algorithms. *Sensors* **2025**, *25*, 6099. [\[CrossRef\]](#) [\[PubMed\]](#)
30. Mansouri, S.; Hosseini, S. A Chaos-Enhanced Hybrid Metaheuristic Feature Selection for Intrusion Detection Using Artificial Neural Networks. *Int. J. Comput. Intell. Syst.* **2026**. [\[CrossRef\]](#)
31. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, UK, 1998; Volume 1.
32. Seyyedabbasi, A.; Aliyev, R.; Kiani, F.; Gulle, M.U.; Basyildiz, H.; Shah, M.A. Hybrid algorithms based on combining reinforcement learning and metaheuristic methods to solve global optimization problems. *Knowl.-Based Syst.* **2021**, *223*, 107044. [\[CrossRef\]](#)
33. Seyyedabbasi, A. A reinforcement learning-based metaheuristic algorithm for solving global optimization problems. *Adv. Eng. Softw.* **2023**, *178*, 103411. [\[CrossRef\]](#)
34. Ming, F.; Gong, W.; Wang, L.; Jin, Y. Constrained multi-objective optimization with deep reinforcement learning assisted operator selection. *IEEE/CAA J. Autom. Sin.* **2024**, *11*, 919–931. [\[CrossRef\]](#)
35. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [\[CrossRef\]](#) [\[PubMed\]](#)
36. Thrun, S.; Littman, M.L. Reinforcement learning: An introduction. *AI Mag.* **2000**, *21*, 103.
37. Wu, G.; Mallipeddi, R.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization*; Technical Report; National University of Defense Technology: Changsha, China; Kyungpook National University: Daegu, Republic of Korea; Nanyang Technological University: Singapore, 2017; Volume 9, p. 2017.
38. Ahmed, M.; Mahmood, A.N.; Hu, J. A survey of network anomaly detection techniques. *J. Netw. Comput. Appl.* **2016**, *60*, 19–31. [\[CrossRef\]](#)
39. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*; IEEE: New York, NY, USA, 2015; pp. 1–6.
40. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSP* **2018**, *1*, 108–116. [\[CrossRef\]](#)
41. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [\[CrossRef\]](#)
42. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [\[CrossRef\]](#)

43. Arora, S.; Singh, S. Butterfly optimization algorithm: A novel approach for global optimization. *Soft Comput.* **2019**, *23*, 715–734.
44. Su, H.; Zhao, D.; Heidari, A.A.; Liu, L.; Zhang, X.; Mafarja, M.; Chen, H. RIME: A physics-based optimization. *Neurocomputing* **2023**, *532*, 183–214. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.