Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.025212

ARTICLE

A New Hybrid Approach Using GWO and MFO Algorithms to Detect Network Attack

Hasan Dalmaz*, Erdal Erdal and Halil Murat Ünver

Department of Computer Engineering, Faculty of Engineering and Architecture, Kırıkkale University, Kırıkkale, 71450, Turkey

*Corresponding Author: Hasan Dalmaz. Email: hasandalmaz@gmail.com

Received: 28 June 2022 Accepted: 30 September 2022

ABSTRACT

This paper addresses the urgent need to detect network security attacks, which have increased significantly in recent years, with high accuracy and avoid the adverse effects of these attacks. The intrusion detection system should respond seamlessly to attack patterns and approaches. The use of metaheuristic algorithms in attack detection can produce near-optimal solutions with low computational costs. To achieve better performance of these algorithms and further improve the results, hybridization of algorithms can be used, which leads to more successful results. Nowadays, many studies are conducted on this topic. In this study, a new hybrid approach using Gray Wolf Optimizer (GWO) and Moth-Flame Optimization (MFO) algorithms was developed and applied to widely used data sets such as NSL-KDD, UNSW-NB15, and CIC IDS 2017, as well as various benchmark functions. The ease of hybridization of the GWO algorithm, its simplicity, its ability to perform global optimal search, and the success of the MFO algorithm in obtaining the best solution suggested that an effective solution would be obtained by combining these two algorithms. For these reasons, the developed hybrid algorithm aims to achieve better results by using the good aspects of both the GWO algorithm and the MFO algorithm. In reviewing the results, it was found that a high level of success was achieved in the benchmark functions. It achieved better results in 12 of the 13 benchmark functions compared. In addition, the success rates obtained according to the evaluation criteria in the different data sets are also remarkable. Comparing the 97.4%, 98.3%, and 99.2% classification accuracy results obtained in the NSL-KDD, UNSW-NB15, and CIC IDS 2017 data sets with the studies in the literature, they seem to be quite successful.

KEYWORDS

Network; attack detection; hybrid; GWO; MFO

1 Introduction

Nowadays, it has become very popular to use machine learning and related techniques to find solutions to real-world problems. The main feature of machine learning methods is the extraction of characteristics from a large amount of data without human influence [1]. There are many different classification studies that use machine learning methods and apply them to real-world problems. Automated classification of epileptic Electroencephalogram (EEG) signals [2], anomaly detection [3], iris recognition [4] are some of these studies. Network security is also a very important issue. The

evolution of technology has increased network security in almost all fields. The continuous increase in network traffic and networked systems, while bringing convenience in implementation, sometimes results in the measures taken for network security being inadequate and creating a vulnerability for possible network attacks. The Internet, which has become an essential part of our daily and working lives, developments in Internet technologies, increasing use of Internet of Things (IoT) technology in many fields, and Internet-based social networks can be considered as one of the main reasons for the increase in network attacks [5]. Various reports on network security state that network attacks have reached a disturbing level of scale and complexity, especially in recent years. Traditional methods such as data encryption, user security, and the use of firewalls are used as initial security measures, but weak passwords or password security breaches do not prevent unauthorized use, and user authentication fails. However, firewalls may also suspect undefined or insecure security policies if there are errors in the configuration [6]. In particular, threats to targets such as commercial, military, and public network systems have made it necessary to increase cybersecurity, and awareness in this area [7].

Intrusion Detection Systems (IDS) aims detect and prevent attacks on the network from inside or outside. In IDS, two types of systems are distinguished: signature-based and anomaly-based. Signature-based systems store known and previously seen attack types in the database, while anomaly-based systems evaluate real-time packets based on their anomalies with regular packets. Various machine learning techniques are used for this evaluation [8]. In addition, IDS has some weaknesses. The main drawback is that events that do not pose a threat are counted as attacks (false positives), while events that pose a threat to the system are not counted as attacks (false negatives). For these reasons, it is important to interpret the information received about the attacks.

Due to the anonymous structure of the Internet and the increasing ease of use, attacks on systems have become relatively easy nowadays. The need for IDS is increasing day by day to protect important and critical data [9]. As computer network attacks and their methods evolve, there are new approaches to prevent these attacks. In particular, the algorithm used is of great importance to find the most appropriate solution for network attack detection, which is considered as an optimization problem. Algorithms for finding the optimal solution are divided into deterministic and stochastic classes. In deterministic algorithms, the solution obtained does not change if the input value does not change. However, structural difficulties may arise in finding the solution, and the desired solution may not be obtained. For these reasons, metaheuristic algorithms, which are a nature-inspired type of stochastic algorithm, are used. They are easy to construct, can be hybridized with metaheuristic algorithms, can be easily applied to different problems, and avoid local optimal values [10,11]. Metaheuristic algorithms are swarm and population-based algorithms. Examples of these algorithms include Salp Swarm Algorithm (SSA), Genetic Algorithm (GA), Moth Flame Optimization (MFO), and Gray Wolf Optimizer (GWO). While these metaheuristic algorithms have advantages, they also have some disadvantages. Their weaknesses are in local search, stuckness, non-repeatable exact solutions, and convergence uncertainties. In order to obtain a better solution, hybrid methods should be developed that combine successful features by using more than one algorithm together [12]. In a developed hybrid algorithm, an attempt is made to eliminate the weaknesses of the existing algorithms and achieve more successful results.

In recent years, the topics of network attack detection and prevention have become very popular. However, there are few studies in the literature on how algorithms can become more efficient in classifying attacks. When developing a hybrid algorithm that uses different algorithms, it is necessary to learn more about how the solution presented using benchmark functions can work more effectively on which types of problems. New studies in this area continue to be of great importance, as network security and the detection and prevention of network attacks is an area that needs constant

improvement. Each new study improves on the shortcomings of previous studies and sheds light on future studies. For this reason, our work is significant because it applies not only to network security but also too many problems that machine learning is designed to solve. In this study, a new hybrid algorithm was developed and tested with NSL-KDD, UNSW-NB15, and CIC IDS 2017 data sets. NSL-KDD one of these data sets widely used in the literature is an improved version of the KDD Cup'99 data set. Although this dataset is an improved version of the KDD Cup'99 dataset, it has limitations in handling modern attack types. The UNSW-NB15 and CIC IDS 2017 datasets, which have had these shortcomings further reduced and their suitability for today's attack types increased, are also datasets that were evaluated as part of this study. To test the generalizability and robustness of the algorithm developed in this study, these datasets, which are widely used in the literature, were examined.

Unlike many similar studies, this study uses a newly introduced hybrid metaheuristic optimization method called GWOMFO to solve benchmark problems and detect network intrusions. GWOMFO contains the best features of GWO and MFO. The study aims to develop a new hybrid algorithm, to show its effectiveness by achieving more successful results than existing algorithms, to introduce a new hybrid algorithm into the literature so that it can be used on various problems in the scientific community, and the study is of the original value in this regard. Briefly, the main contributions of the study are as follows:

- We propose a new structure for network attack detection.
- The proposed hybrid approach is based on GWO and MFO algorithms. A hybrid solution is presented that achieves higher accuracy by eliminating the weaknesses of these algorithms.
- In the development of a new algorithm, the contribution of tests against benchmark functions is presented, and the developed algorithm has proven its success through successful results on these functions.
- The newly developed hybrid algorithm has been tested on 3 different datasets and has achieved successful results. The obtained results were compared with the studies in the literature and showed that it is a successful algorithm with its performance.

The remainder of this paper is arranged as follows. Related works in the literature can be found in Section 2, theoretical background in Section 3, materials and methods used in Section 4, and experimental results in Section 5. Conclusions and future work can be found in the last part of the study.

2 Related Works

Data mining and machine learning in network attack detection using various classification methods are now widely used. The main reason for this is to detect different attacks and their types. For effective and successful attack detection, the classification technique used is crucial. So far, several techniques have been proposed to detect attacks and improve existing systems. The use of hybrid methods is one of these techniques.

Researchers continue to develop hybrid IDS to detect network attacks. The most important reason is that a hybrid classifier can improve its ability to detect unknown attacks, its threat detection performance, and its detection speed depending on the underlying algorithms. This section presents some studies in the literature on network attack detection and the limitations of these studies. In addition, some recent studies on network security are mentioned and information about the trend is given.

In the first of these studies, Teng et al. [13] proposed a hybrid method created with a 2-class support vector mechanism and a decision tree. In the results obtained, the training time is much shorter than the traditional Support Vector Machine (SVM) algorithm. However, the detection accuracy of unknown or new attacks is quite low. Guo et al. [14] proposed a framework consisting of two stages, anomaly detection and misuse detection. First, network connections pass through stage 1, and if an anomaly is detected, it is forwarded to the anomaly detection component. If it was reported as normal in the first stage, it is forwarded to the misuse detection component. Clustering and k-NN algorithms are used in this structure. In the results obtained, 91.86% TPR and 93.29% accuracy were achieved. Khraisat et al. [15] proposed a hybrid structure using One Class Support Vector Machine (OCSVM) and C5.0 decision tree classifiers. They found that in this study, which aimed at a low false discovery rate and high recognition accuracy, they achieved quite good results compared to existing studies. Ahmim et al. [16] proposed a new IDS called HCPTCIDS that combines the probability estimates of the classification tree. In this structure, which consists of two layers, the first layer contains a tree structure and the second layer contains the final classifier structure of the first layer, which consists of different probability estimates. It was found that the proposed system achieves higher performance than most studies in the literature. Al-Yaseen et al. [17] proposed a new hybrid structure consisting of Extreme Learning Machine, K-means Clustering and SVM algorithms. It was highlighted that this proposed structure significantly improved the network intrusion detection results and they stated that they achieved 95.75% detection accuracy. Kevric et al. [18] proposed a new hybrid structure consisting of the combination of 3 different algorithms. In this study, using C4.5 decision tree, random tree, and Naive Bayes algorithms, it was found that the hybrid structure gave better results than the individual classifiers. Aslahi-Shahri et al. [19] developed a hybrid method consisting of GA and SVM algorithms. They stated that they reached 97.3% True Positive Rate (TPR) in their results.

The main idea in developing a hybrid algorithm, which is also used in this study, is to create a better combined model and obtain more accurate, better, and more reliable results. In another study based on the idea of creating a new model by integrating multiple algorithms, Mahmod et al. [20] proposed a hybrid system based on a multilayer perceptron. In this study, using Multi Layer Perceptron (MLP) and Artificial Bee Colony (ABC) algorithms, it was found that the detection accuracy of new attacks increased with the increase of food sources and colony size, but the success of intrusion detection in multiple classes was low in the NSL-KDD. Elbasiony et al. [21] proposed a new hybrid solution based on K-means and random forest algorithms for network attack detection. Reviewing the obtained results, it was stated that the detection rate and false positive rate were better than many of the compared studies. Panda et al. [22] proposed a hybrid attack detection algorithm consisting of random forest, decision tree, and SVM algorithms to classify network attacks. However, the proposed technique is insufficient to respond to all intrusion attempts. It does not achieve a high detection rate and a low false alarm rate. Mohamad Tahir et al. [23] developed a hybrid intelligent system based on Support Vecor Machine and K-means clustering algorithms for network attack detection. In the results studied, it was found that they achieved 96.24% accuracy and 3.72% false alarm rate. Gupta et al. [24] developed a hybrid algorithm in their study by combining the ABC and SVM algorithm. This study found that the developed hybrid method achieves better average accuracy than both ABC and SVM.

In general, only the accuracy values are given as a result in the studies in the literature. In this study, additional analyzes were made on 3 different data sets according to the attack type. Comparisons were also made with metrics such as accuracy, True Positive Rate (TPR), False Positive Rate (FPR). In addition to these, calculations were made by keeping the number of features low in most studies, which makes the reliability of the obtained result questionable. In addition to these studies, many

have appeared in the literature, especially in recent years that address network security and provide solutions to security problems from various aspects. In recent years, many studies have appeared in the literature on popular topics such as IoT and Fog/Cloud Computing that address security issues. Regarding these issues, Kumar et al. [25] proposed a unified IDS for use in IoT environments. Depending on the threshold confidence factor, the rules created from different decision tree models are selected, and the analysis of the data set is performed. In this study, using the UNSW-NB15 data set, they performed better than decision tree models in attack detection rates. Mousavi et al. [26] proposed a hybrid model using ABC, Secure Hash Algorithm 256 (SHA256), and Elliptic Curve Cryptography (ECC) to improve data security in IoT applications. It was found that the efficiency of encryption and decryption operations can be increased by more than 50% and the execution time for encryption is 52.31% lower than the method RSA-AES, when a private key is generated using the algorithm ABC. Al-Qerem et al. [27] studied fog/cloud networks. In their study, they propose concurrency control protocols for this subject. This study, which aims to reduce the communication and computation load in fog and cloud nodes, shows that they succeed in improving network quality. In another study, Bhushan et al. [28] proposed a new approach for flowchart sharing. This proposed method aims to protect Software-Defined Networking (SDN) networks from DDoS attacks with flow table overloading. The increased resilience of the network to the flow table has also been shown to be successful in preventing DDoS attacks. Stergiou et al. [29] discussed the issue of privacy and security in fog environments. They presented a cloud computing-based solution for working with Big Data. In this solution, a security wall was inserted between the cloud server and the Internet to create an architecture to improve network security. Mousavi et al. [30] proposed a study for IoT-based irrigation systems. A cryptographic algorithm is presented to improve the security of these systems. This study highlights that cryptography is one of the most important solutions to protect confidentiality and integrity between nodes. Rivest Cipher (RS4), SHA−256 and ECC are used in this approach. Summary of related works are shown below in Table 1.

Table 1: Summary of related works

Authors	Year	Method	Dataset	Advantages	Disadvantages
Teng et al. [13]	2017	2-Class SVM and Decision Tree	KDD99	The training time is much shorter than SVM.	The accuracy of new or unknown attack is very low.
Guo et al. [14]	2016	Clustering and k-NN algorithm	KDD99 and Kyoto	Two level hybrid solution consisting of misuse detection and anomaly detection.	TPR and accuracy values are low.
Khraisat et al. [15]	2020	One Class Support Vector Machine (OCSVM) and C5.0 decision tree	NSL-KDD and ADFA	This study is the integration of the signature and anomaly intrusion detection systems, which takes advantage of the respective strengths of SIDS and AIDS.	There is no detailed analysis according to attack types.
Ahmim et al. [16]	2018	HCPTCIDS	KDD99 and NSL-KDD	Low false alarm rate.	Low detection rate for DoS and Probe attacks.

(Continued)

Table 1 (continued)

Authors	Year	Method	Dataset	Advantages	Disadvantages
Al-Yaseen et al. [17]	2017	Extreme Learning Machine, K-means Clustering and Support Vector Machine algorithms	KDD99	High detection accuracy.	Only KDD99 dataset was used in the study. There is no detailed analysis according to attack types.
Kevric et al. [18]	2017	Random Tree, C4.5 decision Tree, NBTree	NSL-KDD	Improved results according to individual classifiers.	There is no detailed analysis according to attack types.
Aslahi-Shahri et al. [19]	2016	Genetic Algorithm and SVM	KDD99	High detection accuracy.	Only KDD99 dataset was used in the study. There is no detailed analysis according to attack types.
Mahmod et al. [20]	2015	MLP and Artificial Bee Colony	NSL-KDD	Easy to understand and simple.	Low accuracy rate.
Elbasiony et al. [21]	2013	A hybrid approach using weighted K-means and random forest algorithms	KDD99	The classifier can detect new attack types.	Only KDD99 dataset was used in the study.
Panda et al. [22]	2012	Random forest, decision tree, and SVM	NSL-KDD	High detection rate and low false alarm rate.	Only binary classification has been studied.
Mohamad Tahir et al. [23]	2015	Support Vecor Machine and K-means clustering	NSL-KDD	High TPR and low False alarm rate.	There is no detailed analysis according to attack types.
Gupta et al. [24]	2015	Artificial Bee Colony Algorithm (ABC) and SVM	KDD99	Improved results according to individual classifiers.	Low detection rate and only KDD99 dataset was used.
Yin et al. [31]	2017	Recurrent neural networks	NSL-KDD	With feature extraction, a better representation of the data set was obtained.	The training is performed several times to get the best learning rate and number of hidden nodes.
Almi'ani et al. [32]	2018	K-means clustering and self organized map	NSL-KDD	Easy to understand and simple.	Weak sensitivity and time consuming method.
Kamarudin et al. [33]	2017	Hybrid feature selection using wrapper and filter Methods	NSL-KDD and UNSW-NB15	It can detect unknown attacks.	Uses only five features and eight classes of dataset.
Naoum et al. [34]	2013	Resilient backpropagation artificial neural network	NSL-KDD	High generalization Fast converge.	Low detection rate.
Lei [35]	2017	Support vector machine and partical swarm optimization	KDD99 and DARPA	Improved SVM parameters with PSO.	The system uses only 6 features.

3 Theoretical Background of Hybrid Approach

The following subsections provide the necessary background for hybridization. The most prominent feature of the GWO algorithm is its social hierarchy. This hierarchy is also well adapted for solving

complex problems. However, to increase the performance of the algorithm, hybridization with the MFO algorithm was considered. In general, the MFO algorithm can achieve good results in finding the best solution, but it is not sufficient to refine the optimal solution in each iteration. For this reason, it is aimed to improve performance by hybridizing with the GWO algorithm. By hybridizing these two algorithms, the goal is to arrive at a precise solution and improve the convergence feature instead of being trapped in local optima. GWO and MFO algorithms are examined in detail in this section.

3.1 Gray Wolf Optimizer (GWO) Algorithm

This algorithm, which considers the social leadership and hunting strategy of gray wolves, was presented in 2014 by Mirjalili [11]. In gray wolves, which generally live in groups, groups consist of 5–12 individuals. Wolves referred to as alpha wolves are defined as leader wolves. There are four main groups: Omega, delta, beta, and alpha wolves. Alpha wolves are the best wolves, while omega wolves are the lowest. Alpha wolves are the decision-makers in managing the other wolves in the group, hunting areas, and sleeping times.

The priority in the hunting strategy of gray wolves is to find the prey. Then the prey is besieged under the leadership of the alpha wolf. Delta, beta, and alpha wolves are used as the three best solutions for updating the location of wolves because they are believed to provide better information about the location of the prey [36,37].

The GWO algorithm is based on the hierarchy of wolves as a model. The best solution found is alpha, the second and third solutions are beta and delta. The remaining solutions are called omegas. In this algorithm, alpha, beta, and delta give commands to omega, and omega searches the solution space for these commands.

In this algorithm, the first step of the hunting process is to encircle the prey. Eqs. (1)–(4) used in the mathematical model and their explanations can be found below.

$$\mathrm{X}\,(\mathrm{t}+1) = X_p\,(\mathrm{t}) - A.D \tag{1}$$

In Eq. (1), X represents the position of the gray wolves, while X_p represents the position of the prey and t represents the number of iterations. D is distance from prey. D in the equation is calculated with Eq. (2), and A and C are the coefficients calculated with Eqs. (3) and (4), respectively. Using Eqs. (1) and (2), the gray wolfs can randomly update its position around its prey.

$$D = \left|C.X_p\,(t) - \ X\,(t)\right| \tag{2}$$

$$A = 2\alpha.r_1 - \alpha \tag{3}$$

$$C = 2r_2 \tag{4}$$

The value α in Eq. (3) is a linearly reduced value from 2 to 0. It is calculated by Eq. (5). While the TotIter value indicates the total number of iterations, r1 and r2 are the values randomly assigned between 0 and 1 and used for the optimal solution. During the phase of attacking the prey, the value of α is decreased and so is the range of change of A. If A has random values in the range [−1, 1], the next position of the search agent is somewhere between its current position and the position of the prey. Gray wolves usually search for alpha, beta, and delta positions. They separate from each other to search for their prey and come together at the moment of attacking their prey. To model the distribution mathematically, the parameter A is used with random values greater or less than 1. This makes the search important and supports the global search of the GWO algorithm. Alpha, beta, and delta wolves have good information about the potential location. However, the alpha, beta, and

delta solutions help the other wolves to update their positions. Eq. (6) contains the equation used for position update.

$$\alpha = 2 - t\,(2/TotIter) \tag{5}$$

$$X\,(t+1) = (X_1 + X_2 + X_3)/3 \tag{6}$$

$$X_1 = |X_\alpha - A_1.D_\alpha| \tag{7}$$

$$X_2 = |X_\beta - A_2.D_\beta| \tag{8}$$

$$X_3 = |X_\delta - A_3.D_\delta| \tag{9}$$

The 3 best solutions at time t are represented by X_1, X_2, and X_3. The values A_1, A_2 and A_3 in their equations are calculated according to Eq. (3). Eqs. (10)–(12) also calculate D_α, D_β and D_δ. The values for C_1, C_2, and C_3 in these equations are calculated according to Eq. (4).

$$D_\alpha = |C_1.X_\alpha - X| \tag{10}$$

$$D_\beta = |C_2.X_\beta - X| \tag{11}$$

$$D_\delta = |C_3.X_\delta - X| \tag{12}$$

The alpha, beta, and delta species of gray wolves have exceptional knowledge of the current location of their prey. Therefore, the three best solutions are recorded, and the other wolves can update their positions relation to the positions of the best search agents. Eqs. (6)–(12) can be used in this context. The pseudocode of the GWO algorithm is shown below in Table 2.

Table 2: GWO pseudo-code

Algorithm	
01:	Set the max value for iterations I
02:	Create the population P_i $(i = 1, 2, 3, \ldots, k)$
03:	Assign initial values
04:	Calculate fitness values for wolves
05:	X_α = assign best agent
06:	X_β = assign second best agent
07:	X_δ = assign third best agent
08:	**While** (t < I) **do**
09:	**For each** agent **do**
10:	Perform update current agent with specified equation $X_{t+1} = (X_1 + X_2 + X_3)/3$
11:	**End For**
12:	Update parameters
13:	Calculate fitness values

(Continued)

Table 2 (continued)

Algorithm	
14:	Update X_a, X_β *and* X_δ
15:	T = t + 1
16:	**End** While
17:	**Return** X_a

3.2 Moth-Flame Optimization (MFO) Algorithm

This algorithm, presented by Seyedali Mirjalili, was developed by exploiting the particular navigation behavior of moths in nature, called cross-orientation [38]. When the moth moves, it determines a fixed angle to the moon and flies at that angle. This movement enables the moth to fly straight for long distances. However, if you observe moths, you will notice that they make a spiral movement around the light. This situation is caused by artificial light [39]. The extreme proximity to the light source and the attempt to maintain the fixed angle with respect to the light source reveals the spiral trajectory.

In the mathematical model of this algorithm, the moth cluster is represented by the matrix M. The fitness values are stored in an array called AM. At the beginning of the algorithm, a moth population is assigned at a random location based on lower and upper bounds. The flame matrix is updated by calculating the fitness values. It represents the flames and contains the best position each moth has ever reached. The flames are also updated when the moth finds a better solution. Each moth must update its position with only one of the flames. After each iteration, the positions are updated relative to the flame. The position of the moths is updated with Eq. (13). In the equation given in Eq. (13), the positions of the moths in the search space are updated based on the logarithmic spiral function.

$$S\left(M_i, F_j\right) = D_i.e^b t.\cos\left(2\pi t\right) + F_j \tag{13}$$

Here i'th moth M_i, j'th flame F_j and distance between moth and flame in each iteration is expressed in D_i. In the equation, t is a randomly generated number between −1 and 1 and b is a constant. After the moths are updated, the fitness values are re-evaluated. After this process, the best solutions are updated and stored in the flame matrix. For the position update process to be more successful and effective, the number of flames is reduced at each iteration step. The variation of the reduction in the number of flames as a function of iteration, where N is the maximum number of flames, is shown in Eq. (14). The number of flames for each iteration can be calculated using this equation.

$$FlameNo\left(N, I, T\right) = \text{round}(N - I(N - I)/T) \tag{14}$$

In the equation, N expresses the maximum number of flames. In other equation expressions, l represents the number of iterations, and T represents the maximum number of iterations. The pseudocode of the MFO algorithm is shown in Table 3.

Table 3: MFO pseudo-code

Algorithm	
01:	Initialize the parameters to be used for Moth-flame
02:	Randomly generate position for Moth M_t
03:	**For each** i = 1:k **do**
04:	Calculate the fitness values
05:	**End For**
06:	**While** (iteration ≤ max_iteration) **do**
07:	Update Moth position of M_i
08:	Using $FlameNo = round(N - l(N-l)/T)$ equation detect number of flames
09:	Check the results for fitness values
10:	**If**(iterations == 1) **then**
11:	F = sort(Moth M)
12:	AF = sort(Moth Array AM)
13:	**Else**
14:	F = sort(M_{t-1}, M_t)
15:	AF = sort(M_{t-1}, M_t)
16:	**End If**
17:	**For each** i = 1:k **do**
18:	**For each** j = 1:m **do**
19:	Update the values
20:	Using $D_i = \lvert M_i - F_j \rvert$ equation calculate the value
21:	Using $M_i = S(M_i, F_j)$ equation update M(i, j)
22:	**End For**
23:	**End For**
24:	**End While**
25:	Print best solution

4 Materials and Methods

4.1 Research Framework

In this part, the framework of the proposed study, which consists of 5 phases, is presented.

4.1.1 Data Preprocessing

This study used the data sets NSL-KDD, UNSW-NB15, and CIC IDS 2017, which are commonly used for network attack detection. Usually, the data sets contain numerical and categorical data that humans can read and understand. However, some machine learning and deep learning models cannot handle categorical data. Therefore, normalization and transformation processes are performed to clean unnecessary data and improve performance. The result is a more suitable data set.

In the conversion processes, the nominal values in the data set were converted to numeric values. One-to-one digitization of the data used label coding and assigned a numeric value to each categorical value. The result of these processes is a data set consisting of numeric values.

In normalization processes, a linear transformation was made on the data by using min-max normalization, and the data was spread between 0–1. Eq. (15) used for this process is given below.

$$X' = \frac{(X - Min(X))}{(Max(X) - Min(X))} \tag{15}$$

The min.-max. normalization process examines at how far the field value is from the minimum value and ranks these differences [40].

4.1.2 Hybrid Classification (GWOMFO)

Solutions in the literature for network attack detection systems face many problems such as low detection accuracy, unbalanced detection rates, and difficulty of attack detection in real-time networks. As a result, hybrid algorithms were developed to achieve the best possible accuracy by combining several algorithms to solve these problems. In hybrid techniques, in general, each component has tasks in different phases, such as preprocessing, classification, clustering, and they can also be used together in the same phase. In this study, the hybrid structure obtained by combining the GWO and MFO algorithms was applied and tested in the classification phase. Our aim is to develop a new metaheuristic approach for training artificial neural network (ANN) and to show that the method we have developed is also successful in the network attack detection system using the models trained with the algorithms we have compared. The values in the population of the hybrid algorithm form the weights and biases used in training the ANN. The data we observed and obtained show that the network was well trained and successfully detects attacks compared to studies trained with other algorithms.

In this section, the developed hybrid algorithm is presented. We propose a method to develop a hybrid IDS that combines two machine learning algorithms. In this hybrid model, GWO and MFO are combined to create a hybrid model. Metaheuristic algorithms are algorithms that aim to find global or near-optimal solutions. The ability of the GWO algorithm to find a global optimum and the success of the MFO algorithm in achieving the best result suggest that better performance can be achieved by combining these two algorithms.

In GWO, wolves update their positions according to the food source. During this update, the alpha wolves and the other wolves try to be closest to the food source. Poorly obtained positions are not considered in the GWO calculations. In the MFO algorithm, the best solutions are included in the calculations to obtain the new solution. It is assumed that in this way the best solution is found to reach the food source. The GWOMFO algorithm was developed based on the GWO algorithm, whose pseudocode is given in Table 4. The equations developed for the GWOMFO algorithm, and the explanations of these equations are as follows.

Table 4: GWOMFO pseudo-code

Algorithm	
01:	Initialize the parameters to be used
02:	Create the population X_i $(i = 1, 2, \ldots, k)$

(Continued)

Table 4 (continued)

```
Calculate the fitness values
X_a = assign best agent
X_b = assign second best agent
X_d = assign third best agent
X_m = assign Moth's best agent
While (t < L) do
    For each agent do
        If (p ≤ 0, 5)
            Using Eq. (6) update the position for current agent
        Else
            Using Eq. (16) update the position for current agent
    End For
    Update parameters
    Recalculate the fitness values for all agents
    Update X_a, X_b, X_d and X_m
    t++
End While
Return solution
```

The data set obtained after the data preprocessing described in the previous Section 4.1.1 was used to train and test the proposed system. In this algorithm, the effect of the agents of the GWO algorithm is improved based on the MFO algorithm. This method aims to improve the global convergence, discovery, and exploitation performance instead of running the variant for countless generations without improvement.

The flowchart of the developed hybrid algorithm is shown below in Fig. 1.

In the mathematical model for gray wolf hunting strategy, delta, beta, and alpha wolves are assumed to provide better information about prey location. Therefore, the first three best solutions are used to update the positions of wolves in the GWO algorithm. Moreover, the best solution from the MFO algorithm was evaluated with the found solutions, and a new best solution was created. The equation given by Eq. (16) is used to improve the position in the GWOMFO algorithm. In this equation, the position obtained with the GWO agents is evaluated together with the position from the MFO algorithm. The 4 best solutions at time t are represented by X_a, X_b, X_d and X_m.

$$X_{t+1} = (X_a + X_b + X_d + X_m)/4 \tag{16}$$

In this algorithm, hybridization was achieved by interfering with the standard GWO algorithm in 2 stages. First, a condition was added in the exploitation stage to improve hunting performance. Then, Eq. (16) was applied. A new mechanism was added to improve the solution. This improved the hunting mechanism of the GWO. After each iteration, the solution is evolved. Also, the added condition improves the search ability and the exploration phase and increases the quality of the solution.

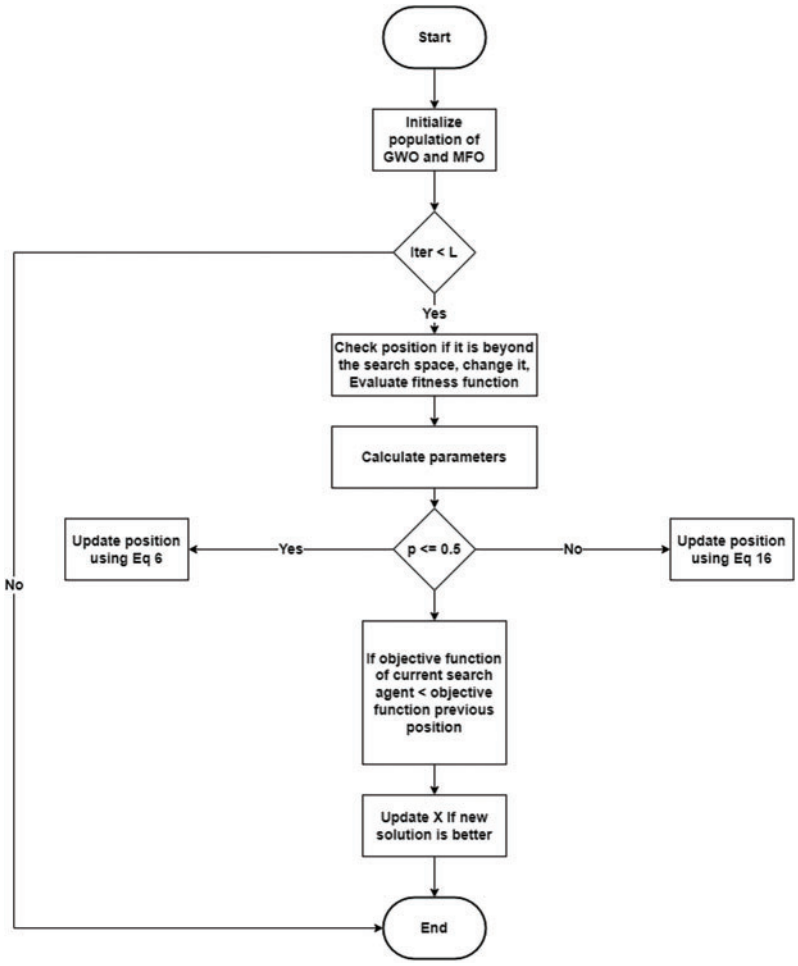


Figure 1: Flowchart of the proposed algorithm (GWOMFO)

GWOMFO is started by initializing the population size of both the MFO and GWO search agents. After this process, the fitness value is calculated. After the value assignments for the agents are made, the position updates of the existing search agents are performed using the appropriate equations according to the value of parameter p up to the maximum number of iterations (L) in the loop. After these operations, the fitness value is calculated, and the process continues until a suitable solution is found.

The parameter p is a randomly generated number. If the random value is less than or equal to 0.5, position updates are performed using alpha, beta, and delta position equations. The calculation is performed using Eq. (6) in this condition. The positions are updated if the new position is better than the old one. If it is greater than 0.5, Eq. (16) is used and updated again, comparing it with the old positions. As a result, the fitness value is calculated again. The best fitness value is returned.

In metaheuristic approaches, there are two main components directly related to the search ability of an algorithm: exploration and exploitation. Exploration seeks to find promising solutions by going deep into unknown territory. In other words, exploration aims to increase the diversity of solutions. In contrast, exploitation aims to improve the quality of solutions by searching locally around

discoverable and promising solutions. These components conflict with each other and influence each other. Therefore, an optimization algorithm should be designed with a correct and appropriate balance between exploration and exploitation. With this hybrid algorithm, an improvement in exploration and exploitation performance was observed. This is also reflected in the results obtained. The pseudocode of the GWOMFO algorithm is shown in Table 4.

4.1.3 Training

Before modeling or estimation, data sets are separated into training and test data sets. In this separation, the hold-out method was used, and 2/3 of the original data was reserved as the training data set and 1/3 as the test data set [41]. Training of the algorithm was performed using the training sets obtained in this way.

4.1.4 Testing

During the testing phase, eight swarm intelligence-based algorithms (GWO, MFO, DE, PSO, MVO, JAYA, SSA, and SCA) and the algorithm developed in this study were applied separately and tested on the NSL-KDD, UNSW-NB15, and CIC IDS 2017 data sets. The results obtained by each algorithm were studied separately.

4.1.5 Evaluation

Several quantitative metrics are used to interpret the results obtained by the testing process performed in the study. These metrics are:

Accuracy: Shows the percentage of correct estimates. It is calculated based on the Confusion Matrix.

$$\text{Accuracy} = \text{TPs} + \text{TNs/n}$$

True Positive Rate (TPR): Total number of true positives.

$$\text{TPR} = \text{TPs}/(\text{FNs} + \text{TPs})$$

False Positive Rate (FPR): The formula used to calculate this ratio is below.

$$\text{FPR} = \text{FPs}/\text{TNs} + \text{FPs}$$

Sensitivity (Recall): Shows how well positive situations are predicted.

$$\text{Sensitivity} = \text{TPs}/(\text{TPs} + \text{FNs})$$

Precision: It is the criterion that shows the success in positive predictions.

$$\text{Precision} = \text{TPs}/(\text{TPs} + \text{FPs})$$

F-measure: Harmonic average of the values for sensitivity and precision.

Studies were assessed and compared using these metrics.

4.2 Data Set Descriptions

In this study, 3 different data sets were examined. The data sets used in this study are known in the literature as commonly used data sets for network attack detection. These data sets, which differ in terms of attack types and number and type of features, were evaluated in the study to obtain more accurate results about the generalizability and robustness of the developed algorithm. By using these three different data sets, the performance changes and consistency of the algorithm developed in this study will be examined. These data sets are listed below, respectively.

4.2.1 NSL-KDD

NSL-KDD is a commonly used data set for network attack detection and can also be represented as a model. This data set has been used in many studies in the literature [42–46]. It is an extended version of the data set known as KDD Cup'99. Several problems with the KDD Cup'99 data set have been fixed with the NSL-KDD data set [47]. In particular, biased results should be avoided by removing unnecessary records [48]. The NSL-KDD data set is the result of an evolution of the KDD Cup'99 data set, but has the same characteristics as the KDD Cup'99 data set in terms of features. It contains no unnecessary data, but enough data to train and test. These data sets, defined as attacks, are classified into four main classes. These are DOS, Probe, User to Root (U2R), and Remote to Local (R2L) attacks. The distribution of the data set used is shown in Table 5.

Table 5: Data distribution of dataset NSL-KDD

Class	Record
DoS	45,927
U2R	52
Probe	11,656
R2L	995
Normal	67,343
Total	125,973

In the NSL-KDD data set, which contains 125,973 records, 67,343 were classified as normal, and 58,630 as attack. With 41 features (32 numeric, six binary, three nominal), this data set contains 24 different attacks and records labeled as normal. Features in NSL-KDD data set shown in Table 6.

Table 6: Features in NSL-KDD data set

Feature no.	Feature name	Feature no.	Feature name	Feature no.	Feature name
1	Duration	15	Su_attempted	29	Same_srv_rate
2	Protocol Type	16	Num_root	30	Diff_srv_rate
3	Service	17	Num_file_creations	31	Srv_diff_host_rate
4	Flag	18	Num_shells	32	Dst_host_count
5	Src_Bytes	19	Num_access_files	33	Dst_host_srv_count
6	Dst_Bytes	20	Num_outbound_cmds	34	Dst_host_same_srv_rate
7	Land	21	Is_host_login	35	Dst_host_diff_srv_rate
8	Wrong_Fragment	22	Is_guest_login	36	Dst_host_same_src_port_rate
9	Urgent	23	Count	37	Dst_host_srv_diff_host_rate
10	Hot	24	Srv_count	38	Dst_host_serror_rate
11	Num_Failed_Logins	25	Serror_rate	39	Dst_host_srv_serror_rate
12	Logged_in	26	Srv_serror_rate	40	Dst_host_rerror_rate
13	Num_compromised	27	Rerror_rate	41	Dst_host_srv_rerror_rate
14	Root_shell	28	Srv_rerror, rate	42	Label

One of the most important factors in evaluating the performance of IDS and developing more effective and efficient IDSs is the data sets used [49]. The most commonly used data sets to measure the performance of IDS are KDD99 and NSL-KDD. Data sets are essential for developing IDS and measuring its performance. The data set used should meet the time requirements and include current attack types. The literature's most commonly used KDD99 and NSLKDD data sets are different from current conditions in terms of attack types and normal traffic scenarios, and the distribution of training and testing data sets is also different. It is now assumed that current data sets should be used in studies. To address these issues, data sets such as UNSW-NB15, CIC IDS 2017 have been developed to capture current and modern attack types [50].

4.2.2 UNSW-NB15

This data set was created using the IXIA PerfectStorm tool at the Australian Cyber Security Center (ACCS) Cyber Range Lab to merge modern, realistic normal network activity and attack behavior from network traffic [50]. This data set has been used in many studies in the literature [51–55]. Unlike NSLKDD, this data set contains original versions of the various identity states that are common today. Attacks in this data set include fuzzer, analytics, backdoor, DoS, exploit, general, reconnaissance, shellcode, and worm attacks. The distribution of the data set used is shown in Table 7.

Table 7: Data distribution of dataset UNSW-NB15

Class	Record
DoS	12,264
Analytics	2,000
Exploits	33,393
General	40,000
Reconnaissance	10,491
Worm	130
Shellcode	1,133
Fuzzer	18,184
Backdoor	1,746
Normal	56,000
Total	175,341

This data set contains 175,341 records and 49 extracted features [56]. Features in UNSW-NB15 data set shown in Table 8.

Table 8: Features in UNSW-NB15 data set

Feature no.	Feature name	Feature no.	Feature name	Feature no.	Feature name
1	Srcip	18	Dpkts	35	Ackdat
2	Sport	19	Swin	36	Is_sm_ips_ports
3	Dstip	20	Dwin	37	Ct_state_ttl

(Continued)

Table 8 (continued)

Feature no.	Feature name	Feature no.	Feature name	Feature no.	Feature name
4	Dsport	21	Stcpb	38	Ct_flw_http_mthd
5	Proto	22	Dtcpb	39	Is_ftp_login
6	State	23	Smeansz	40	Ct_ftp_cmd
7	Dur	24	Dmeansz	41	Ct_srv_src
8	Sbytes	25	Trans_depth	42	Ct_srv_dst
9	Dbytes	26	Res_bdy_len	43	Ct_dst_ltm
10	Sttl	27	Sjit	44	Ct_src_ltm
11	Dttl	28	Djit	45	Ct_src_dport_ltm
12	Sloss	29	Stime	46	Ct_dst_sport_ltm
13	Dloss	30	Ltime	47	Ct_dst_src_ltm
14	Service	31	Sintpkt	48	Attack_cat
15	Sload	32	Dintpkt	49	Label
16	Dload	33	Tcprtt		
17	Spkts	34	Synack		

4.2.3 CIC IDS 2017

This data set was developed by the University of New Brunswick, School of Computer Science, in 2017. It is an improved version of the ISCX 2012 data set [57]. This data set has been used in many studies in the literature [57–61]. It consists of a generalization of real network traffic. The total number of records in this dataset is 2,829,463. 2,358,036 of these records are normal, and the remaining records are in the attack class. In this dataset, attacks are examined in 7 categories. These are DoS, DDoS, Botnet, Port Scan, Infiltration, Web Attack and HeartBleed attacks. The distribution of the data set used is shown in Table 9.

Table 9: Data distribution of dataset CIC IDS 2017

Class	Record
DoS	252,661
DDoS	41,835
Botnet	1,966
Port Scan	158,930
Infiltration	36
Web Attack	15,988
HeartBleed	11
Normal	2,358,036
Total	2,829,463

There are more than 80 features in this dataset. Features in CIC IDS 2017 data set shown in Table 10.

Table 10: Features in CIC IDS 2017 data set

Feature no.	Feature name	Feature no.	Feature name	Feature no.	Feature name
1	Flow ID	29	Fwd IAT Std	57	ECE Flag Count
2	Source IP	30	Fwd IAT Max	58	Down/Up Ratio
3	Source Port	31	Fwd IAT Min	59	Average Packet Size
4	Destination IP	32	Bwd IAT Total	60	Avg Fwd Segment Size
5	Destination Port	33	Bwd IAT Mean	61	Avg Bwd Segment Size
6	Protocol	34	Bwd IAT Std	62	Fwd Avg Bytes/Bulk
7	Time stamp	35	Bwd IAT Max	63	Fwd Avg Packets/Bulk
8	Flow Duration	36	Bwd IAT Min	64	Fwd Avg Bulk Rate
9	Total Fwd Packets	37	Fwd PSH Flags	65	Bwd Avg Bytes/Bulk
10	Total Backward Packets	38	Bwd PSH Flags	66	Bwd Avg Packets/Bulk
11	Total Length of Fwd Pck	39	Fwd URG Flags	67	Bwd Avg Bulk Rate
12	Total Length of Bwd Pck	40	Bwd URG Flags	68	Subflow Fwd Packets
13	Fwd Packet Length Max	41	Fwd Header Lenth	69	Subflow Fwd Bytes
14	Fwd Packet Length Min	42	Bwd Header Lenth	70	Subflow Bwd Packets
15	Fwd Pck Length Mean	43	Fwd Packets/s	71	Subflow Bwd Bytes
16	Fwd Packet Length Std	44	Bwd Packets/s	72	Init_Win_bytes_fwd
17	Bwd Packet Length Max	45	Min Packet Length	73	Act_data_pkt_fwd
18	Bwd Packet Length Min	46	Max Packet Length	74	Min_seg_size_fwd
19	Bwd Packet Length Mean	47	Packet Length Mean	75	Active Mean
20	Bwd Packet Length Std	48	Packet Length Std	76	Active Std
21	Flow Bytes/s	49	Packet Len. Variance	77	Active Max
22	Flow Packets/s	50	FIN Flag Count	78	Active Min
23	Flow IAT Mean	51	SYN Flag Count	79	Idle Mean
24	Flow IAT Std	52	RST Flag Count	80	Idle Packet
25	Flow IAT Max	53	PSH Flag Count	81	Idle Std
26	Flow IAT Min	54	ACK Flag Count	82	Idle Max
27	Flow IAT Total	55	URG Flag Count	83	Idle Min
28	Fwd IAT Mean	56	CWE Flag Count	84	Label

5 Experimental Results

In this section, information about the simulation and its environment is given, and the results obtained with the hybrid algorithm are explained and discussed.

5.1 Simulation Setup

To briefly discuss the working environment and the programs used in this study, the computer configurations used for the proposed method are as follows. Two computers are used for development and deployment. The development computer is a home PC with an Intel Core i7-4500U CPU, 16 GB RAM, and a 1 TB SATA HDD. The deployment computer has Intel Xeon E5-2620 v4 2.10 GHz processors (8 cores) server, 64 GB RAM, 512 GB SSD, and 1 TB SATA HDD. In addition, Python version 3.7.6 was used as the programming language and programmed with Spyder IDE.

In all experiments, the values of common parameters such as total number of iterations and population size used in each algorithm are chosen to be the same. Equal search agent, equal run, and equal iteration are set in all run algorithms. The search agent used in all algorithms is set to 30, the number of runs is set to 30, and the number of iterations is set to 100. The developed algorithm was first tested in benchmark functions and its results were examined. Then, it was tested on 3 different data sets that are commonly used in the literature. It is aimed to ensure the consistency and generalizability of the algorithm by running the developed algorithm under the same conditions on 3 different data sets.

In the study, Evolopy framework which developed for ANN training, optimization problem solving, clustering operations and feature selection was used. Each algorithm studied may use its own parameters, may use various constant values, and these may vary according to the applied problems. The values in the Evolopy framework are used in the parameter values of the algorithms [62]. In this study, the method in which the number of hidden neurons is $(2 \times N + 1)$ was chosen; N is the number of features in each dataset. For each dataset, all input features values are normalized in the range [0, 1] with normalization technique Eq. (15).

5.2 Benchmark Functions and Results

Benchmark functions to test and validate newly created optimization algorithms with different properties. One of the most important properties of these functions is that they are differentiable, decomposable, scalable, continuous, unimodal, bimodal, continuous, and discontinuous. These properties can determine which problems an algorithm will succeed on.

The hybrid algorithm developed in this study was tested using benchmark functions with different characteristics and compared with various optimization algorithms from the literature. These functions, studied in 2 groups as single-mode and multi-mode, are characterized as single-mode functions containing only one global optimum, and functions with more than one local and global optimum are called multi-mode benchmark functions. Functions F1–F7 are called unimodal functions, which have a single solution. This makes the convergence rate very suitable for testing and applying algorithms. As a result, the GWOMFO exploitation capability can be evaluated by using these unimodal functions. In other functions, such as F8–F13, which are multimodal functions and they are useful to assess our proposed algorithm in terms of exploration. For multimode benchmark functions, there is more than one optimal value, and one of them is the global optimum while the other values are defined as local optima. This is a more complex problem than for single-mode benchmark functions. If the discovery process of an evolved algorithm is poorly designed, no effective wide-angle search can be performed, resulting in the algorithm getting stuck in the local optimum. For this reason, these functions seem to be the most difficult problem for many algorithms.

In this study, 13 functions were investigated as single-mode and multi-mode. These functions provide an important starting point to test the reliability of a developed algorithm. There are many local optimal points in the solution spaces for these functions. As the complexity increases, the number of local optima also increases. In the following, we describe in detail each of the benchmark functions used in this study.

These studied functions are listed in Table 11.

Table 11: Characteristics of the benchmark test functions

Mathematical formulation	Name	D	Mode	Range	fmin
$F1(x) = \sum_{i=1}^{n} X_i^2$	Sphere	30	Single	[−100, 100]	0
$F2(x) = \sum_{i=1}^{n} \mid x^i \mid + \prod_{i=1}^{n} \mid x^i \mid$	Schwefel 2.22	30	Single	[−10, 10]	0
$F3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j\right)^2$	Schwefel 1.2	30	Single	[−100, 100]	0
$F4(x) = max_i \{\mid x_i \mid, \ 1 \le i \le n\}$	Schwefel 2.21	30	Single	[−100, 100]	0
$F5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	Rosenbrock	30	Single	[−30, 30]	0
$F6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	Step	30	Single	[−100, 100]	0
$F7(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1]$	Quartic	30	Single	[−1.28, 1.28]	0
$F8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\mid x_i \mid}\right)$	Schwefel	30	Multi	[−500, 500]	–418.9829xB
$F9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	Rastrigin	30	Multi	[−5.12, 5.12]	0
$F10(x) = 20\exp(-0.2\sqrt{\frac{1}{n}\sum_1^n x_i^2}) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	Ackley	30	Multi	[−32, 32]	0
$F11(x) = \frac{1}{4000}\sum_1^n x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Griewank	30	Multi	[−600, 600]	0
$F12(x) = \frac{\pi}{n}\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\ sin^2(\pi y_{i+1})]$ $+y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i,\ 10,\ 100,\ 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{pmatrix} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{pmatrix}$	Penalized	30	Multi	[−50, 50]	0
$F13(x) = 0.1\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)]$ $+(x_n - 1)^2[1 + sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	Penalized2	30	Multi	[−50, 50]	0

In these functions, F1 is continuous, differentiable, separable and convex. x is a n-dimensional vector located within the range [−100; 100]. The global minimum is located at the origin with a function value of zero. F2 is continuous, differentiable, non-separable and scalable. x is a n-dimensional vector located within the range [−10.0; 10.0]. The global minimum is located at the origin with a function value of zero. F3 is continuous, differentiable, non-separable and scalable. x is a n-dimensional vector located within the range [−100.0; 100.0]. The global minimum is located at the origin with a function value of zero. F4 is continuous, non-differentiable, separable and scalable. x is a n-dimensional vector located within the range [−100.0; 100.0]. The global minimum is located at the origin with a function value of zero. F5 is continuous, differentiable, non-separable and scalable. x is a n-dimensional vector located within the range [−30.0; 30.0]. The global minimum is located at (1, ..., 1) with a function value of zero. F6 is discontinuous, non-differentiable, separable and scalable. x is a n-dimensional vector located within the range [−100.0; 100.0]. The global minimum is located at (0.5, ..., 0.5) with a function value of zero. F7 is continuous, differentiable, separable and scalable. x is a n-dimensional vector located within the range [−1.28; 1.28]. The global minimum is located at (0, ..., 0) with a function value of zero. F8 is continuous, differentiable, separable and scalable. x is a n-dimensional vector located within the range [−500; 500]. The global minimum is located at $\pm[\pi(0.5 + k)]^2$ with a function value of −418.983. F9 is continuous, differentiable, convex and separable. In this function, x

is a n-dimensional vector located within the range [−5.12; 5.12]. The global minimum is located at the origin and its function value is zero. F10 is continuous, differentiable, non-convex and non-separable. x is a n-dimensional vector that is normally located within the range [−32.0; 32.0]. Ackley's function is a highly multimodal function with regularly distributed local optima. The global minimum is located at the origin and its value is zero. F11 is non-separable and non-convex with several local optima within the search region defined by [−600, 600]. The global minimum is located at the origin and its value is zero. F12 is discontinuous, no separable, and non-convex functions. x is a n-dimensional vector located within the range [−50.0; 50.0]. The exact global minimum for all of these problems is achieved for an objective function value of zero. F13 is discontinuous, nonseparable, and non-convex functions. x is a n-dimensional vector located within the range [−50.0; 50.0]. The exact global minimum for all these problems is obtained with an objective function value of zero. The developed algorithm was applied to these 13 benchmark functions.

Each function was run 30 times for the benchmark functions, and the standard deviation and mean were calculated based on the obtained results. The developed hybrid algorithm was applied to benchmark functions, and the results were compared with algorithms based on swarm intelligence such as DE, PSO, MVO, JAYA, SSA, SCA. Table 12 contains the comparison results. This table contains the mean and standard deviation of the algorithms used in the benchmark functions. When examining the means and standard deviations, it was found that GWOMFO generally achieved better means and standard deviations for all benchmark functions.

In addition to these results, the average convergence curves of the benchmark functions optimized by GWOMFO are shown in Fig. 2.

The graphs in Fig. 2 represent the convergence curves generated for benchmark functions. In these graphs, the horizontal axis represents the index of repetitions, and the vertical axis is the divergence between two consecutive objective function values. The convergence curve shows the value of the objective function compared to the computation time during minimization (model calibration). This testing procedure aims to solve the optimization problems in earlier iterations, reduce the convergence time and obtain a better solution. Thirteen benchmark functions known in the literature were used for the experimental studies. The ultimate goal of optimization is either to reduce the convergence time to obtain the best solution or to increase the efficiency of the algorithm. In this context, the main goal of the developed GWOMFO algorithm focuses on reducing the convergence time, obtaining effective results and improving the local search capability. In this way, the algorithm is designed to reach the solution in fewer iterations. By looking at the change on the y-axis, you can see how fast it converges to the optimal result. In this case, we can say that the most suitable individuals are selected in early iterations to reduce the computational cost of the algorithm.

Table 12: Comparison of benchmark function results of the proposed algorithm (GWOMFO) and various swarm intelligence based algorithms

F	Proposed algorithm (GWOMFO)		DE		PSO		MVO		JAYA		SSA		SCA	
	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std	avg	std
F1	0	0	1263.87	400.4283	6.88	2.27534	36.04	10.386851	786.1	517.815	305.25	60.019	4344.5	2731.652186
F2	0	0	28.97	6.490931	14.1	7.132155	75.63	61.952689	12.26	4.894991	8.24	0.9479	5.6	2.97684102
F3	9.72	53.23197	47715.65	8485.433	1236.11	485.9593	6671.85	1747.6725	36812	11799.82	2853.3	1056.6	31771	12076.90141
F4	0	0	75.67	6.841525	5.31	1.668242	22.46	8.2750305	55.13	9.855786	12.87	2.8782	64.94	10.07644517
F5	28.84	0.256805	603539.7	377497.8	5109.08	16475.11	2690.73	2216.8964	869149	749194.9	13073	6020.2	8E + 06	8702090.465
F6	3.32	0.416849	1208.85	298.9181	6.17	3.0117	35.96	8.9201995	847.74	374.5994	319.89	70.395	3996	2521.325613
F7	0	0.00032	0.59	0.235218	5.3	4.663797	0.14	0.0410475	0.89	0.69831	0.15	0.056	5.97	5.758827531
F8	−3917.17	466.6463	−4597.06	341.6723	−3254.98	492.1964	−7187.4	627.55625	−4247	528.4435	−5825.1	318.35	−3283	292.6505876
F9	7.5	31.89674	246.02	19.02749	213.72	31.3239	164.88	36.019504	200.97	36.83116	125.4	22.725	158.98	53.80637744
F10	0	1.5E−31	10.06	2.134396	3.11	0.440712	4.82	4.154069	9.44	2.635121	5.49	0.3557	17.47	4.587843638
F11	0	0	11.71	3.326986	18.58	4.774702	1.34	0.1050404	8.32	4.738595	3.85	0.4845	36.31	22.94477716
F12	0.5	0.091182	115920.1	299382.4	1.11	1.199464	8.75	3.4550983	497755	930442.9	8.47	3.6403	4E + 07	34016524.72
F13	2.8	0.193006	653490.9	587023.1	2.28	1.074185	24.91	17.691802	2E + 06	3341609	20.53	7.3844	7407E	80031189.99

Notes: *avg* average; *STD* standard deviation = Best Result.

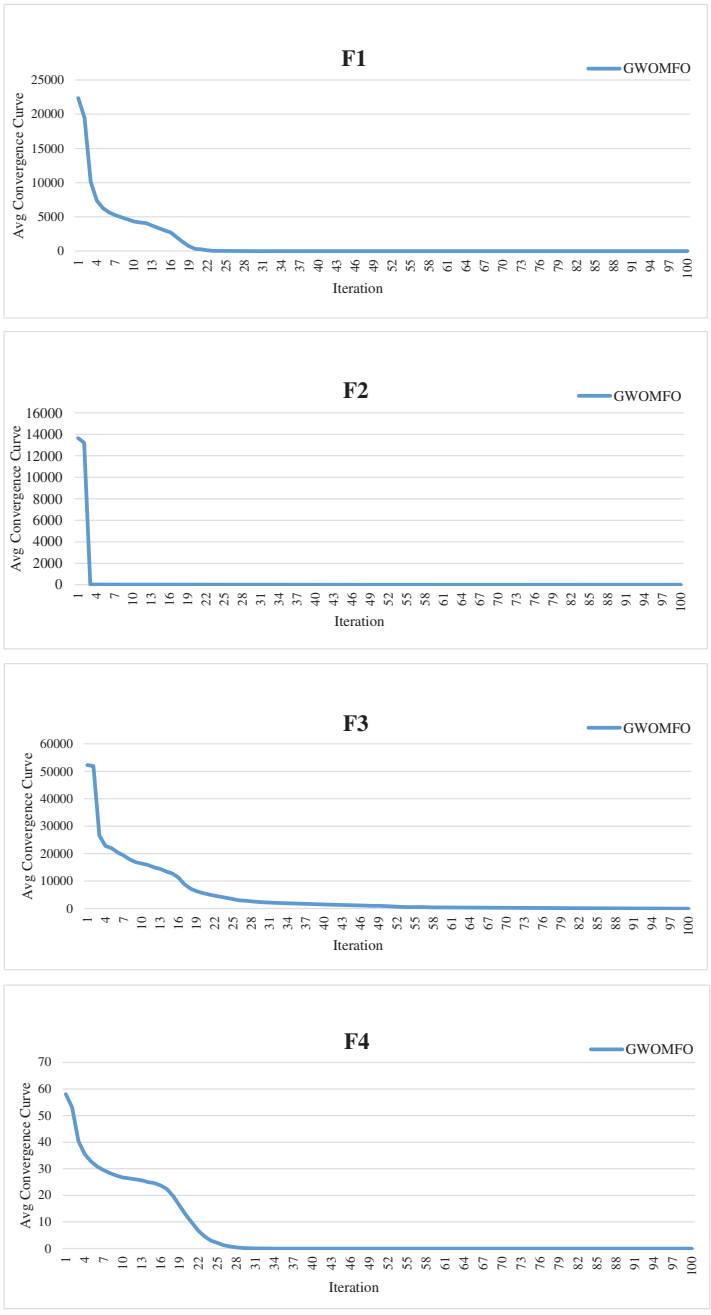


Figure 2: (Continued)

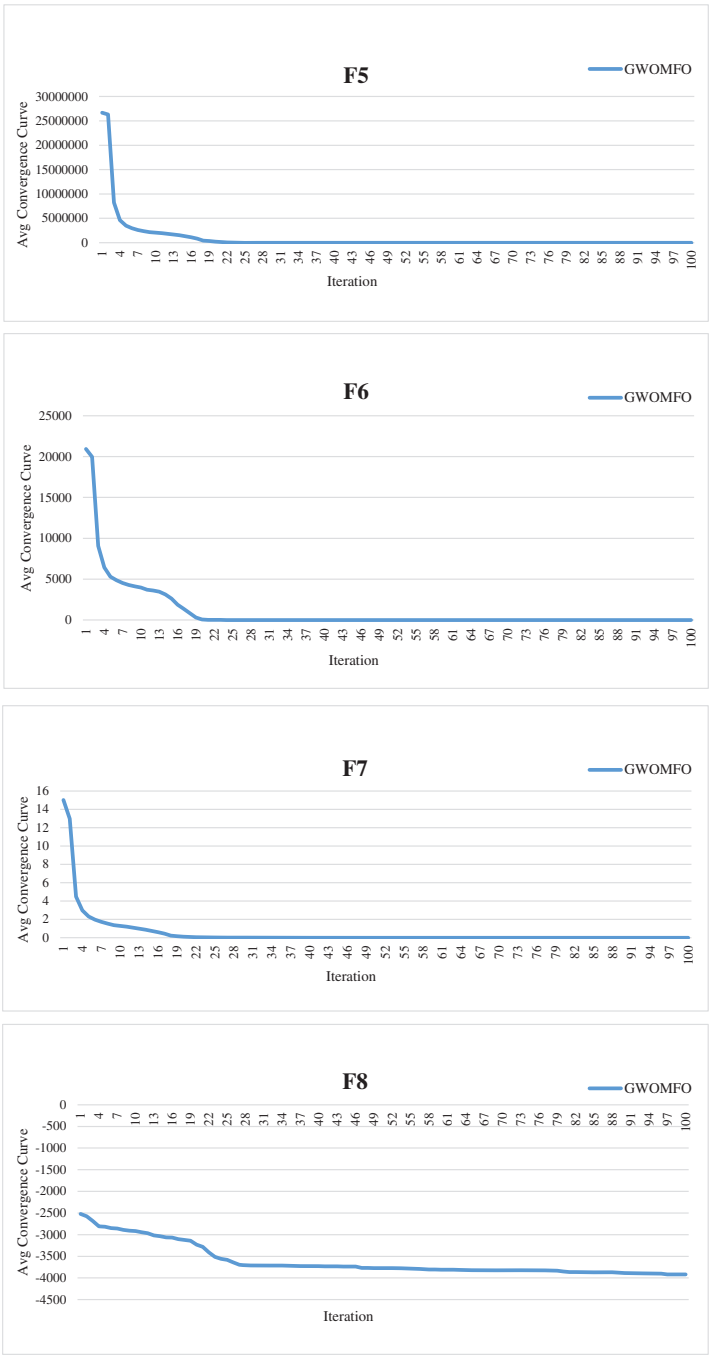


Figure 2: (Continued)

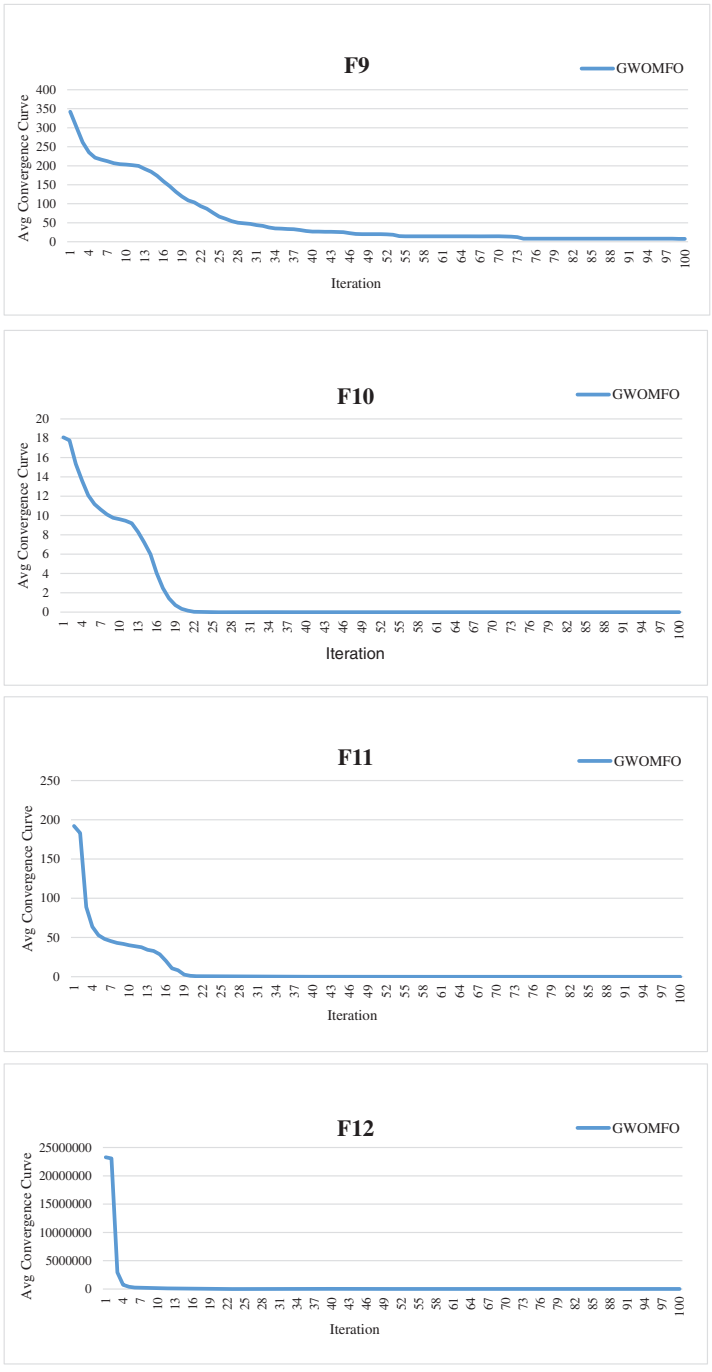


Figure 2: (Continued)

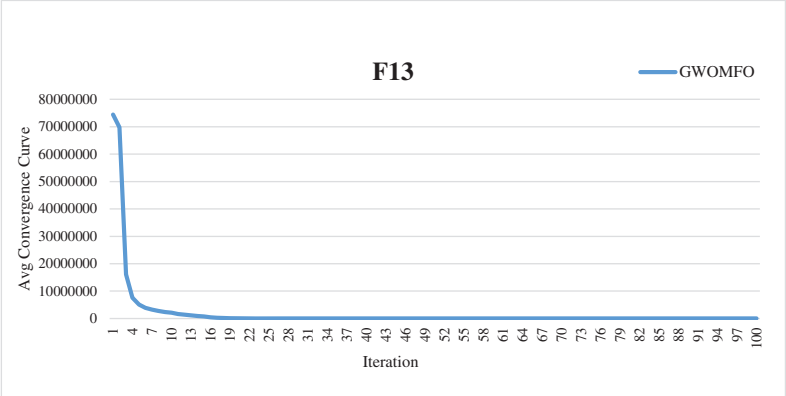


Figure 2: Average convergence curves for benchmark functions optimized with GWOMFO

The GWOMFO algorithm appears to be quite successful in testing benchmark functions. However, the success of the algorithm needs to be demonstrated statistically. For this purpose, Wilcoxon rank sum, a test for data analysis, was applied. It was assumed that the *p*-value is smaller than 0.05 to express a significant difference. The results obtained are shown in Table 13.

Table 13: Wilcoxon rank sum test results

F	Proposed algorithm (GWOMFO)	DE	PSO	MVO	JAYA	SSA	SCA
F1	N/A	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11
F2	N/A	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11
F3	N/A	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11
F4	N/A	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11
F5	N/A	5.30E-10	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11
F6	N/A	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11
F7	N/A	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11
F8	3.69E-01	5.15E-01	N/A	2.69E-09	2.84E-11	4.19E-01	2.84E-11
F9	N/A	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11
F10	N/A	2.84E-11	2.84E-11	6.17E-03	3.16E-07	5.12E-09	4.19E-11
F11	N/A	5.46E-06	2.78E-11	2.84E-11	4.46E-03	2.84E-11	2.84E-11
F12	N/A	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11
F13	N/A	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11	2.84E-11

Note: NA = Not applicable.

The best algorithm was compared with other algorithms in each statistical test. When examining the results, it was found that the developed algorithm has statistically significant differences. For this reason, it can be concluded that the algorithm is quite successful.

5.3 Network Intrusion Detection Results

In this section, the results of the hybrid algorithm developed in detecting network attacks are shared. The developed GWOMFO algorithm was applied to 3 different data sets. Confusion matrix was used to evaluate the performance of the proposed technique. The results obtained with 3 different datasets using Accuracy, TPR and FPR metrics are shown in Table 14. The population size was chosen as 30, the number of iterations as 100, and applied.

Table 14: Performance of GWOMFO (%)

Datasets	Accuracy	TPR	FPR
NSL-KDD	97.4	95.7	8.64
UNSW-NB15	98.3	97.2	5.71
CIC IDS 2017	99.2	98.3	4.11

The developed algorithm was compared with the algorithms on which it is based. The accuracy results obtained are shown in Fig. 3.

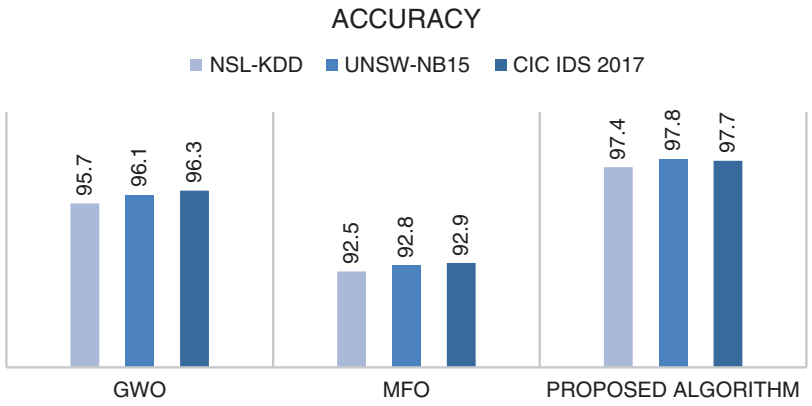


Figure 3: Comparison of accuracy results of GWO, MFO and the proposed algorithm (GWOMFO) on 3 different datasets

It was found that the proposed algorithm achieved the best result in terms of accuracy in all 3 data sets. The proposed algorithm has achieved more successful results than GWO algorithm and MFO algorithm. In other words, better results were obtained by combining GWO and MFO algorithms. However, other criteria within the scope of the study were examined, and the results are shown in Table 15.

Table 15: Comparison of the proposed algorithm (GWOMFO), GWO and MFO on 3 different datasets

Classifiers	Measure	NSL-KDD	UNSW-NB15	CIC IDS 2017
GWO	Accuracy	95.7	96.1	96.3
	Precision	94.9	95.3	95.5
	Sensitivity	94.7	94.8	95.1

(Continued)

Table 15 (continued)

Classifiers	Measure	NSL-KDD	UNSW-NB15	CIC IDS 2017
MFO	F-measure	94.8	95.0	95.3
	Accuracy	92.5	92.8	92.9
	Precision	92.0	92.4	93.1
	Sensitivity	91.4	92.0	92.7
	F-measure	91.7	92.2	92.9
Proposed Algorithm (GWOMFO)	Accuracy	**97.4***	**97.8***	**97.7***
	Precision	**97.7***	**97.4***	**97.8***
	Sensitivity	**96.5***	**97.2***	**97.4***
	F-measure	**97.2***	**97.3***	**97.6***

Note: *: Best Results.

It is found that the developed hybrid algorithm achieves better results in terms of accuracy, precision, sensitivity and F-measure. The successes of various swarm intelligence-based algorithms performing classification processes in the literature are compared to demonstrate the success of the developed hybrid algorithm in network attack detection. For comparison, these swarm intelligence-based algorithms and proposed algorithm were run with the same parameters on 3 datasets. The comparison results considering the accuracy measure are shown in the graph in Fig. 4.

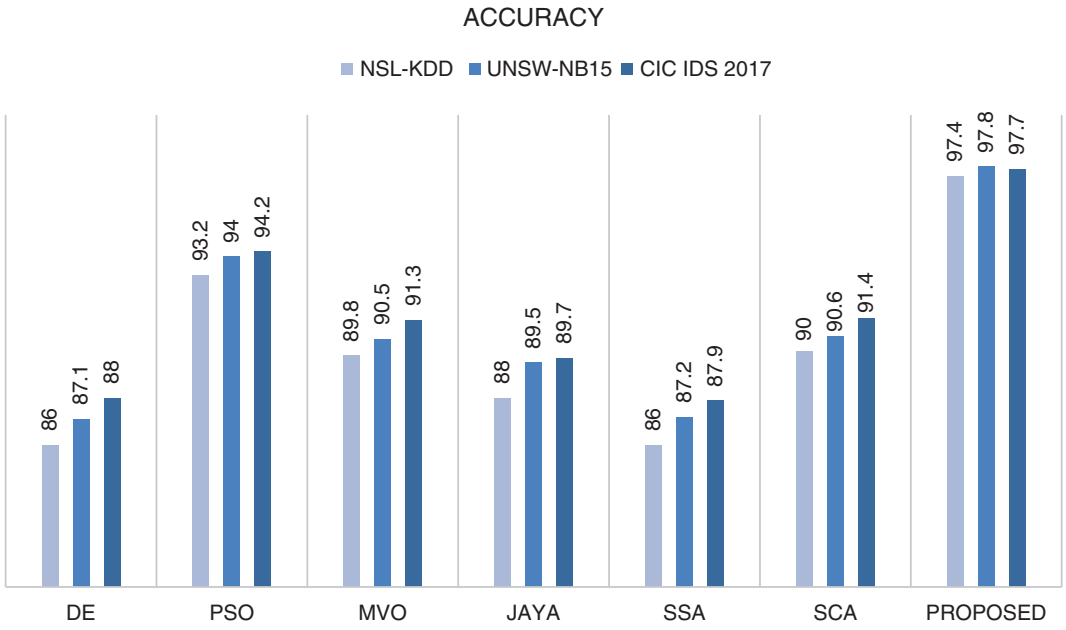


Figure 4: Comparison of the accuracy results of various swarm intelligence based algorithms and proposed algorithm (GWOMFO) on 3 different datasets

It has been shown that the algorithm proposed in the accuracy measurement is the most successful among all the algorithms compared. The results of the comparison considering other metrics are shown in Table 16.

Table 16: Comparison of various swarm intelligence based algorithms and the proposed algorithm (GWOMFO) on 3 different datasets

Data sets	Measure	DE	PSO	MVO	JAYA	SSA	SCA	Proposed algorithm (GWOMFO)
NSL-KDD	Accuracy	86.0	93.2	89.8	88.0	86.0	90.0	**97.4***
	Precision	88.8	94.1	88.0	90.8	87.9	90.8	**97.7***
	Sensitivity	88.4	93.7	87.5	92.3	86.7	92.0	**96.5***
	F-measure	88.6	93.9	88.0	91.5	87.3	91.4	**97.2***
UNSW-NB15	Accuracy	87.1	94.0	90.5	89.5	87.2	90.6	**97.8***
	Precision	89.3	94.3	90.2	89.7	88.0	91.5	**97.4***
	Sensitivity	89.1	94.6	91.3	90.3	88.5	91.6	**97.2***
	F-measure	89.2	94.4	90.7	90.0	88.2	91.5	**97.3***
CIC-IDS 2017	Accuracy	88.0	94.2	91.3	89.7	87.9	91.4	**97.7***
	Precision	89.3	94.3	90.7	90.2	88.3	91.6	**97.8***
	Sensitivity	88.7	93.6	91.8	90.5	88.4	91.9	**97.4***
	F-measure	89.0	93.9	90.5	90.3	88.3	91.7	**97.6***

Note: *Best Results.

In this study, the performance of the developed algorithm was investigated in three different datasets depending on the type of attack. The obtained results were compared with studies performed on these datasets. Overall, GWOMFO proved to be quite successful and achieved a high detection rate for all types of attacks. The results are shown in Figs. 5–7.

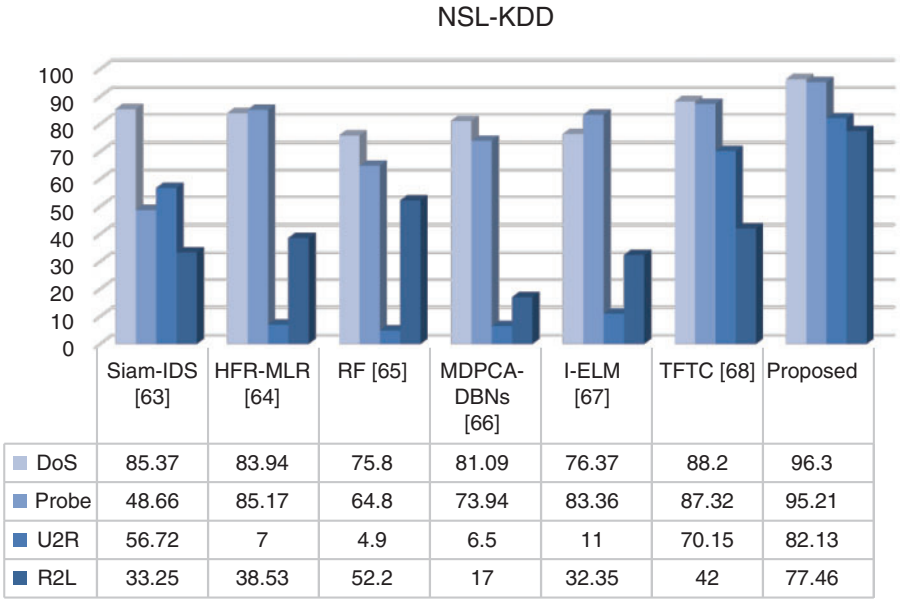
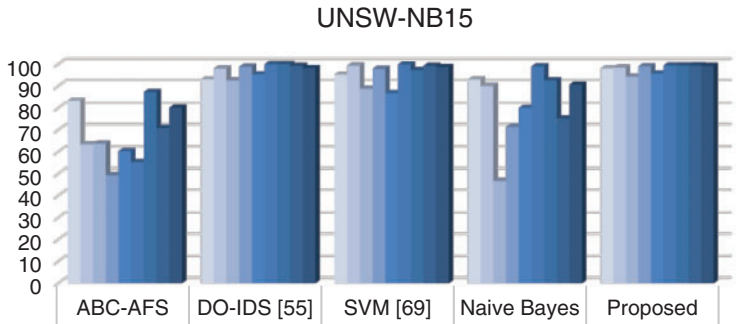
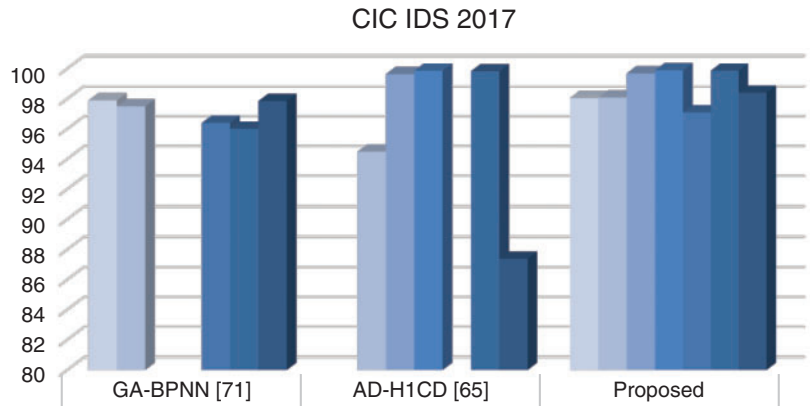


Figure 5: Comparison of TPR results on NSL-KDD dataset



	ABC-AFS [52]	DO-IDS [55]	SVM [69]	Naive Bayes [70]	Proposed
DoS	83.3	93.1	95.2	92.97	98.11
Backdoor	63.4	98	99.3	90.02	98.41
Exploits	63.7	92.6	88.8	46.7	94.33
Reconnaissance	49.3	98.8	97.9	71.42	99.02
Fuzzers	60.3	95.3	86.8	80.03	95.72
Worms	55.3	100	99.9	99	99.4
General	87.3	100	97.3	92.61	99.37
Shellcode	70.9	99.2	99.2	75.24	99.46
Analytics	80.11	98.2	98.7	90.66	99.21

Figure 6: Comparison of TPR results on UNSW-NB15 dataset



	GA-BPNN [71]	AD-H1CD [65]	Proposed
Brute Force	97.89		98.05
Botnet	97.5	94.48	98.08
DoS		99.63	99.68
DDoS		99.86	99.89
Infiltration	96.4		97.1
Port Scan	96.01	99.82	99.86
Web Attack	97.85	87.38	98.4

Figure 7: Comparison of TPR results on CIC IDS 2017 dataset

The hybrid algorithm developed in this study was compared with studies using the publicly available data sets NSL-KDD, UNSW-NB15, and CIC IDS 2017. In this study, studies conducted

separately on 3 different data sets were examined in detail. The results of these studies were compared using accuracy criteria. The results obtained are presented in Table 17.

Table 17: Comparison of the proposed algorithm (GWOMFO) with various studies using the NSL-KDD, UNSW-NB15 and CIC IDS 2017 data set

Algorithm or model	Authors	Datasets	Accuracy
Quasi-optimal algorithm	Nasr et al. [72]	NSL-KDD	0.933
Canonical correlation + feature association impact scale	Jyothsna et al. [73]	NSL-KDD	0.910
PCA and optimized SVM	Ikram et al. [74]	NSL-KDD	0.965
Deep learning approach	Javaid et al. [75]	NSL-KDD	0.791
Random forest	Aljawarneh et al. [76]	NSL-KDD	0.806
Naïve bayes	Aljawarneh et al. [76]	NSL-KDD	0.765
Decision tree	Aljawarneh et al. [76]	NSL-KDD	0.810
Information gain and RepTree	Belouch et al. [77]	UNSW-NB15	0.900
ABC and AFS algorithms	Hajisalem et al. [52]	UNSW-NB15	0.980
Constrained-optimization-based extreme learning machines	Wang et al. [78]	UNSW-NB15	0.830
Skip-gram model classifier	Carrasco et al. [79]	UNSW-NB15	0.910
Hyper clique improved binary gravitational search algorithm as FS and SVM	Gauthama Raman et al. [80]	UNSW-NB15	0.960
Gradient boosted machine	Tama et al. [81]	UNSW-NB15	0.930
Random forest as feature selection and k-nearest neighbors	Alrowaily et al. [82]	CIC IDS 2017	0.990
Negative selection algorithm and classifiers	Hosseini et al. [83]	CIC IDS 2017	0.970
Random forest classifier	Bindra et al. [84]	CIC IDS 2017	0.960
LSTM, CNN AND FNN	Lee et al. [85]	CIC IDS 2017	0.980
Self-adaptive grasshopper optimization algorithm	Shukla [86]	CIC IDS 2017	0.990
LSTM	Kaur et al. [87]	CIC IDS 2017	0.990
BRS	Prasad et al. [57]	CIC IDS 2017	0.970
DBN	Elmasry et al. [88]	CIC IDS 2017	0.980
Proposed agorithm (GWOMFO)		NSL-KDD	**0.974***
		UNSW-NB15	**0.983***
		CIC IDS 2017	**0.992***

Note: *Best Results.

6 Conclusions and Future Work

In this study, the topic of network attacks and detection is discussed, the data sets commonly used to solve this problem are examined, and the studies in the literature and the results obtained are mentioned. The importance of hybrid solutions to improve the performance of the proposed algorithms for the solution has been highlighted. It has been shown that the use of benchmark functions in the preliminary evaluation of the performance of a developed algorithm provides insight into the performance of the algorithm. This study developed and tested a new hybrid algorithm with

NSL-KDD, UNSW-NB15, and CIC IDS 2017 data sets. There is always a need to improve existing methods' solution quality and develop new methods. Hybrid algorithms can be used to improve the solution quality or performance of the algorithm. In the algorithm developed in this study, the effect of the search agents of the GWO algorithm was improved by hybridizing it with the MFO algorithm. To this algorithm, which was developed based on the GWO algorithm, a new mechanism was added to improve the hunting mechanism of the GWO algorithm. It was observed that the search ability and exploration phase were improved after each iteration, which improved the solution. The obtained results show that the location updates are more successful and efficient depending on the search agents. Unlike studies in the literature, the hybrid algorithm developed in this study was first tested in benchmark functions and then tested on data sets used for network attacks. Performance analyzes were performed by adapting the developed algorithm to 13 benchmark functions. The results obtained were compared with many different studies in this study and the following conclusions were drawn:

i. GWOMFO achieved the best average results on 12 of the 13 benchmark functions compared to other algorithms.
ii. The best classification accuracy was achieved in the NSL-KDD data set with 97.4%.
iii. The best classification accuracy was achieved in the UNSW-NB15 data set with 98.3%.
iv. The best classification accuracy was achieved in the CIC IDS 2017 data set with 99.2%.
v. In general, GWOMFO performed well and achieved high detection rates for all attack types.

The results were compared with the results of different optimization algorithms from the literature. It was found that the developed hybrid model showed a successful result. The success of GWOMFO is due to the newly developed equation, the structure of the algorithm and the parameters used. The obtained results proved that the GWOMFO algorithm is successful in the benchmark functions compared to the algorithms with which it is compared. The successful results obtained on the tested datasets also support the algorithm's success. By bringing this developed algorithm to the literature, one of the expected goals is to extend techniques that can be adapted to real-world, hard-to-solve problems.

Network security will continue to be a topic that will always be alive. This is because network defense is a necessity. With many recent studies such as [25–28] listed in the related works section of this study, network security is being discussed from many different angles and will continue to be addressed from many different angles and with many new methods. In addition to this study, we are currently working on simultaneous network attack detection, and other possible future work could include the following:

1. Studies may be conducted to further increase the success rates achieved in this study. For this purpose, hybridization with various improved algorithms from the literature can be performed. Also, various improved feature extraction algorithms can achieve better performance.
2. This hybrid approach can be applied to different data sets, and the results can be observed. It can also solve various optimization problems, such as feature selection. To improve network security infrastructure, Deep Learning can be used to develop next-generation intrusion detection systems that detect network threats instantly and with higher accuracy. Developing an IDS model for multi-class data problems using multiple networks in Deep Learning is also possible.

However, a possible limitation of the proposed method is that we tested GWOMFO on only three datasets. It will also be important to test the method on a more recent dataset. We believe that the proposed method can be extended to many domains in the future.

Funding Statement: This work is supported by the Kırıkkale University Department of Scientific Research Projects (2022/022).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Predić, B., Vukić, U., Saračević, M., Karabašević, D., Stanujkić, D. (2022). The possibility of combining and implementing deep neural network compression methods. *Axioms, 11(5),* 229. DOI 10.3390/axioms11050229.
2. Jukic, S., Saracevic, M., Subasi, A., Kevric, J. (2020). Comparison of ensemble machine learning methods for automated classification of focal and non-focal epileptic EEG signals. *Mathematics, 8(9),* 1481. DOI 10.3390/math8091481.
3. Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S. A. et al. (2018). Deep one-class classification. *International Conference on Machine Learning*, pp. 4393–4402. Stockholm, Sweden, PMLR.
4. Adamović, S., Miškovic, V., Maček, N., Milosavljević, M., Šarac, M. et al. (2020). An efficient novel approach for iris recognition based on stylometric features and machine learning techniques. *Future Generation Computer Systems, 107,* 144–157. DOI 10.1016/j.future.2020.01.056.
5. Resul, D. A. Ş., Bitikçi, B. (2020). Analysis of different types of network attacks on the GNS3 platform. *Sakarya University Journal of Computer and Information Sciences, 3(3),* 210–230.
6. Summers, R. C. (1997). *Secure computing: Threats and safeguards*. The McGraw-Hill, Inc., New York, U.S.A.
7. Baker, S., Filipiak, N., Timlin, K. (2011). *In the dark: Crucial industries confront cyberattacks McAfee annual critical infrastructure protection report*. 2nd edition, pp. 1–6. Santa Clara, CA: The Center for Strategic and International Studies (CSIS).
8. Pehlivanoglu, M. K., Atay, R., Odabaş, D. E. (2019). İki seviyeli hibrit makine Öğrenmesi yöntemi ile saldırı tespiti. *Gazi Mühendislik Bilimleri Dergisi, 5(3),* 258–272.
9. Radoglou-Grammatikis, P. I., Sarigiannidis, P. G. (2019). Securing the smart grid: A comprehensive compilation of intrusion detection and prevention systems. *IEEE Access, 7,* 46595–46620. DOI 10.1109/Access.6287639.
10. Gupta, D., Gupta, V. (2016). Test suite prioritization using nature inspired meta-heuristic algorithms. *International Conference on Intelligent Systems Design and Applications*, pp. 216–226. Cham, Porto, Portugal: Springer.
11. Mirjalili, S., Mirjalili, S. M., Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69,* 46–61. DOI 10.1016/j.advengsoft.2013.12.007.
12. Senel, F. A., Gökçe, F., Yüksel, A. S., Yigit, T. (2019). A novel hybrid PSO–GWO algorithm for optimization problems. *Engineering with Computers, 35,* 1359–1373.
13. Teng, S., Wu, N., Zhu, H., Teng, L., Zhang, W. (2017). SVM-DT-based adaptive and collaborative intrusion detection. *IEEE/CAA Journal of Automatica Sinica, 5(1),* 108–118. DOI 10.1109/JAS.2017.7510730.
14. Guo, C., Ping, Y., Liu, N., Luo, S. S. (2016). A two-level hybrid approach for intrusion detection. *Neurocomputing, 214,* 391–400. DOI 10.1016/j.neucom.2016.06.021.

15. Khraisat, A., Gondal, I., Vamplew, P., Kamruzzaman, J., Alazab, A. (2020). Hybrid intrusion detection system based on the stacking ensemble of C5 decision tree classifier and one class support vector machine. *Electronics, 9(1),* 173. DOI 10.3390/electronics9010173.
16. Ahmim, A., Derdour, M., Ferrag, M. A. (2018). An intrusion detection system based on combining probability predictions of a tree of classifiers. *International Journal of Communication Systems, 31(9),* e3547. DOI 10.1002/dac.3547.
17. Al-Yaseen, W. L., Othman, Z. A., Nazri, M. Z. A. (2017). Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system. *Expert Systems with Applications, 67,* 296–303. DOI 10.1016/j.eswa.2016.09.041.
18. Kevric, J., Jukic, S., Subasi, A. (2017). An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications, 28(1),* 1051–1058. DOI 10.1007/s00521-016-2418-1.
19. Aslahi-Shahri, B. M., Rahmani, R., Chizari, M., Maralani, A., Eslami, M. et al. (2016). A hybrid method consisting of GA and SVM for intrusion detection system. *Neural Computing and Applications, 27(6),* 1669–1676. DOI 10.1007/s00521-015-1964-2.
20. Mahmod, M. S., Alnaish, Z. A. H., Al-Hadi, I. A. A. (2015). Hybrid intrusion detection system using artificial bee colony algorithm and multi-layer perceptron. *International Journal of Computer Science and Information Security, 13(2),* 1.
21. Elbasiony, R. M., Sallam, E. A., Eltobely, T. E., Fahmy, M. M. (2013). A hybrid network intrusion detection framework based on random forests and weighted k-means. *Ain Shams Engineering Journal, 4(4),* 753–762. DOI 10.1016/j.asej.2013.01.003.
22. Panda, M., Abraham, A., Patra, M. R. (2012). A hybrid intelligent approach for network intrusion detection. *Procedia Engineering, 30,* 1–9. DOI 10.1016/j.proeng.2012.01.827.
23. Mohamad Tahir, H., Hasan, W., Md Said, A., Zakaria, N. H., Katuk, N. et al. (2015). Hybrid machine learning technique for intrusion detection system. *5th International Conference on Computing and Informatics (ICOCI)*, Istanbul.
24. Gupta, M., Shrivastava, S. K. (2015). Intrusion detection system based on SVM and bee colony. *International Journal of Computer Applications, 111(10)*. DOI 10.5120/19576-1377.
25. Kumar, V., Das, A. K., Sinha, D. (2021). UIDS: A unified intrusion detection system for IoT environment. *Evolutionary Intelligence, 14(1),* 47–59. DOI 10.1007/s12065-019-00291-w.
26. Mousavi, S. K., Ghaffari, A., Besharat, S., Afshari, H. (2021). Improving the security of internet of things using cryptographic algorithms: A case of smart irrigation systems. *Journal of Ambient Intelligence and Humanized Computing, 12(2),* 2033–2051. DOI 10.1007/s12652-020-02303-5.
27. Al-Qerem, A., Alauthman, M., Almomani, A., Gupta, B. B. (2020). IoT transaction processing through cooperative concurrency control on fog–cloud computing environment. *Soft Computing, 24(8),* 5695–5711. DOI 10.1007/s00500-019-04220-y.
28. Bhushan, K., Gupta, B. B. (2019). Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment. *Journal of Ambient Intelligence and Humanized Computing, 10(5),* 1985–1997. DOI 10.1007/s12652-018-0800-9.
29. Stergiou, C., Psannis, K. E., Gupta, B. B., Ishibashi, Y. (2018). Security, privacy & efficiency of sustainable cloud computing for big data & IoT. *Sustainable Computing: Informatics and Systems, 19,* 174–184. DOI 10.1016/j.suscom.2018.06.003.
30. Mousavi, S. K., Ghaffari, A. (2021). Data cryptography in the internet of things using the artificial bee colony algorithm in a smart irrigation system. *Journal of Information Security and Applications, 61,* 102945. DOI 10.1016/j.jisa.2021.102945.
31. Yin, C., Zhu, Y., Fei, J., He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access, 5,* 21954–21961. DOI 10.1109/ACCESS.2017.2762418.

32. Almi'ani, M., Ghazleh, A. A., Al-Rahayfeh, A., Razaque, A. (2018). Intelligent intrusion detection system using clustered self organized map. *2018 Fifth International Conference on Software Defined Systems (SDS)*, pp. 138–144. Barcelona, Spain, IEEE.
33. Kamarudin, M. H., Maple, C., Watson, T., Safa, N. S. (2017). A logitboost-based algorithm for detecting known and unknown web attacks. *IEEE Access, 5,* 26190–26200. DOI 10.1109/ACCESS.2017.2766844.
34. Naoum, R. S., Abid, N. A., Al-Sultani, Z. N. (2012). An enhanced resilient backpropagation artificial neural network for intrusion detection system. *International Journal of Computer Science and Network Security, 12(3),* 11.
35. Lei, Y. (2017). Network anomaly traffic detection algorithm based on SVM. *2017 International Conference on Robots & Intelligent System (ICRIS)*, pp. 217–220. Huaian, China, IEEE.
36. Mirjalili, S. (2015). The ant lion optimizer. *Advances in Engineering Software, 83,* 80–98. DOI 10.1016/j.advengsoft.2015.01.010.
37. Doğan, L., Yüzgeç, U. (2018). Robot path planning using gray wolf optimizer. *Proceedings of International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES'18)*, pp. 70. Safranbolu, Turkey.
38. Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89,* 228–249. DOI 10.1016/j.knosys.2015.07.006.
39. Li, Z., Zhou, Y., Zhang, S., Song, J. (2016). Lévy-flight moth-flame algorithm for function optimization and engineering design problems. *Mathematical Problems in Engineering, 2016,* 1–22.
40. Larose, D. T., Larose, C. D. (2014). *Discovering knowledge in data: An introduction to data mining*, vol. 4, pp. 411–412. New Jersey, ABD, John Wiley & Sons.
41. Omary, Z., Mtenzi, F. (2010). Machine learning approach to identifying the dataset threshold for the performance estimators in supervised learning. *International Journal for Infonomics, 3(3),* 314–325. DOI 10.20533/iji.1742.4712.
42. Relan, N. G., Patil, D. R. (2015). Implementation of network intrusion detection system using variant of decision tree algorithm. *2015 International Conference on Nascent Technologies in the Engineering Field (ICNTE)*, pp. 1–5. Navi Mumbai, India, IEEE.
43. Chauhan, H., Kumar, V., Pundir, S., Pilli, E. S. (2013). A comparative study of classification techniques for intrusion detection. *2013 International Symposium on Computational and Business Intelligence*, pp. 40–43. New Delhi, India, IEEE.
44. Bhattacharjee, P. S., Fujail, A. K. M., Begum, S. A. (2017). A comparison of intrusion detection by K-means and fuzzy C-means clustering algorithm over the NSL-KDD dataset. *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pp. 1–6. Coimbatore, India, IEEE.
45. Ullah, I., Mahmoud, Q. H. (2017). A filter-based feature selection model for anomaly-based intrusion detection systems. *2017 IEEE International Conference on Big Data (Big Data)*, pp. 2151–2159. Boston, USA, IEEE.
46. Saleh, A. I., Talaat, F. M., Labib, L. M. (2019). A hybrid intrusion detection system (HIDS) based on prioritized k-nearest neighbors and optimized SVM classifiers. *Artificial Intelligence Review, 51(3),* 403–443. DOI 10.1007/s10462-017-9567-1.
47. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6. Ottawa, Canada, IEEE.
48. Revathi, S., Malathi, A. (2013). A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection. *International Journal of Engineering Research & Technology (IJERT), 2(12),* 1848–1853.
49. Tsai, C. F., Hsu, Y. F., Lin, C. Y., Lin, W. Y. (2009). Intrusion detection by machine learning: A review. *Expert Systems with Applications, 36(10),* 11994–12000. DOI 10.1016/j.eswa.2009.05.029.

50. Moustafa, N., Slay, J. (2016). The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective, 25(1–3),* 18–31. DOI 10.1080/19393555.2015.1125974.
51. Khammassi, C., Krichen, S. (2017). A GA-LR wrapper approach for feature selection in network intrusion detection. *Computers & Security, 70,* 255–277. DOI 10.1016/j.cose.2017.06.005.
52. Hajisalem, V., Babaie, S. (2018). A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Computer Networks, 136,* 37–50. DOI 10.1016/j.comnet.2018.02.028.
53. Boulaiche, A., Adi, K. (2018). An auto-learning approach for network intrusion detection. *Telecommunication Systems, 68(2),* 277–294. DOI 10.1007/s11235-017-0395-z.
54. Ahmad, U., Asim, H., Hassan, M. T., Naseer, S. (2019). Analysis of classification techniques for intrusion detection. *2019 International Conference on Innovative Computing (ICIC)*, pp. 1–6. Lahore, Pakistan, IEEE.
55. Ren, J., Guo, J., Qian, W., Yuan, H., Hao, X. et al. (2019). Building an effective intrusion detection system by using hybrid data optimization based on machine learning algorithms. *Security and Communication Networks, 2019*. DOI 10.1155/2019/7130868.
56. Nawir, M., Amir, A., Yaakob, N., Lynn, O. B. (2018). Multi-classi_cation of UNSW-NB15 dataset for network anomaly detection system. *Journal of Theoretical and Applied Information Technology, 96(15),* 5094–5104.
57. Prasad, M., Tripathi, S., Dahal, K. (2020). An efficient feature selection based Bayesian and rough set approach for intrusion detection. *Applied Soft Computing, 87,* 105980. DOI 10.1016/j.asoc.2019.105980.
58. Panigrahi, R., Borah, S. (2018). A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems. *International Journal of Engineering & Technology, 7(3),* 479–482.
59. D'hooge, L., Wauters, T., Volckaert, B., de Turck, F. (2019). Classification hardness for supervised learners on 20 years of intrusion detection data. *IEEE Access, 7,* 167455–167469. DOI 10.1109/Access.6287639.
60. Panwar, S. S., Negi, P. S., Panwar, L. S., Raiwani, Y. (2019). Implementation of machine learning algorithms on CICIDS-2017 dataset for intrusion detection using WEKA. *International Journal of Recent Technology and Engineering Regular Issue, 8(3),* 2195–2207. DOI 10.35940/ijrte.2277-3878.
61. D'hooge, L., Wauters, T., Volckaert, B., de Turck, F. (2019). In-depth comparative evaluation of supervised machine learning approaches for detection of cybersecurity threats. *4th International Conference on Internet of Things, Big Data and Security (IoTBDS)*, pp. 125–136. Heraklion, Greece.
62. Faris, H., Qaddoura, R., Aljarah, I., Bae, J. W., Fouad, M. M. et al. (2016). Evolopy, github. https://github.com/7ossam81/EvoloPy/blob/master/optimizers/.
63. Bedi, P., Gupta, N., Jindal, V. (2020). Siam-IDS: Handling class imbalance problem in intrusion detection systems using siamese neural network. *Procedia Computer Science, 171,* 780–789. DOI 10.1016/j.procs.2020.04.085.
64. Kunang, Y. N., Nurmaini, S., Stiawan, D., Suprapto, B. Y. (2021). Attack classification of an intrusion detection system using deep learning and hyperparameter optimization. *Journal of Information Security and Applications, 58,* 102804. DOI 10.1016/j.jisa.2021.102804.
65. Ma, C., Du, X., Cao, L. (2019). Analysis of multi-types of flow features based on hybrid neural network for improving network anomaly detection. *IEEE Access, 7,* 148363–148380. DOI 10.1109/Access.6287639.
66. Yang, Y., Zheng, K., Wu, C., Niu, X., Yang, Y. (2019). Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks. *Applied Sciences, 9(2),* 238. DOI 10.3390/app9020238.
67. Pajouh, H. H., Dastghaibyfard, G., Hashemi, S. (2017). Two-tier network anomaly detection model: A machine learning approach. *Journal of Intelligent Information Systems, 48(1),* 61–74. DOI 10.1007/s10844-015-0388-x.

68. Pajouh, H. H., Javidan, R., Khayami, R., Dehghantanha, A., Choo, K. K. R. (2016). A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. *IEEE Transactions on Emerging Topics in Computing, 7(2),* 314–323. DOI 10.1109/TETC.6245516.
69. Jing, D., Chen, H. B. (2019). SVM based network intrusion detection for the UNSW-NB15 dataset. *2019 IEEE 13th International Conference on ASIC (ASICON)*, pp. 1–4. Chongqing, China, IEEE.
70. Bagui, S., Kalaimannan, E., Bagui, S., Nandi, D., Pinto, A. (2019). Using machine learning techniques to identify rare cyber-attacks on the UNSW-NB15 dataset. *Security and Privacy, 2(6),* e91. DOI 10.1002/spy2.91.
71. Manimurugan, S., Manimegalai, P., Valsalan, P., Krishnadas, J., Narmatha, C. (2020). Intrusion detection in cloud environment using hybrid genetic algorithm and back propagation neural network. *International Journal of Communication Systems, 35(16),* e4667.
72. Nasr, A. A., Ezz, M. M., Abdulmaged, M. Z. (2016). A learnable anomaly detection system using attributional rules. *International Journal of Computer Network and Information Security, 8(11),* 58–64. DOI 10.5815/ijcnis.2016.11.07.
73. Jyothsna, V., Prasad, V. R. (2016). FCAAIS: Anomaly based network intrusion detection through feature correlation analysis and association impact scale. *ICT Express, 2(3),* 103–116. DOI 10.1016/j.icte.2016.08.003.
74. Ikram, S. T., Cherukuri, A. K. (2016). Improving accuracy of intrusion detection model using PCA and optimized SVM. *Journal of Computing and Information Technology, 24(2),* 133–148. DOI 10.20532/cit.2016.1002701.
75. Javaid, A., Niyaz, Q., Sun, W., Alam, M. (2016). A deep learning approach for network intrusion detection system. *Eai Endorsed Transactions on Security and Safety, 3(9),* e2. DOI 10.4108/eai.24-5-2016.59124.
76. Aljawarneh, S., Yassein, M. B., Aljundi, M. (2019). An enhanced J48 classification algorithm for the anomaly intrusion detection systems. *Cluster Computing, 22(5),* 10549–10565. DOI 10.1007/s10586-017-1109-8.
77. Belouch, M., El Hadaj, S., Idhammad, M. (2017). A two-stage classifier approach using reptree algorithm for network intrusion detection. *International Journal of Advanced Computer Science and Applications, 8(6),* 389–394. DOI 10.14569/issn.2156-5570.
78. Wang, C. R., Xu, R. F., Lee, S. J., Lee, C. H. (2018). Network intrusion detection using equality constrained-optimization-based extreme learning machines. *Knowledge-Based Systems, 147,* 68–80. DOI 10.1016/j.knosys.2018.02.015.
79. Carrasco, R. S. M., Sicilia, M. A. (2018). Unsupervised intrusion detection through skip-gram models of network behavior. *Computers & Security, 78,* 187–197. DOI 10.1016/j.cose.2018.07.003.
80. Gauthama Raman, M. R., Somu, N., Jagarapu, S., Manghnani, T., Selvam, T. et al. (2020). An efficient intrusion detection technique based on support vector machine and improved binary gravitational search algorithm. *Artificial Intelligence Review, 53(5),* 3255–3286. DOI 10.1007/s10462-019-09762-z.
81. Tama, B. A., Rhee, K. H. (2019). An in-depth experimental study of anomaly detection using gradient boosted machine. *Neural Computing and Applications, 31(4),* 955–965. DOI 10.1007/s00521-017-3128-z.
82. Alrowaily, M., Alenezi, F., Lu, Z. (2019). Effectiveness of machine learning based intrusion detection systems. *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*, pp. 277–288. Cham: Springer.
83. Hosseini, S., Seilani, H. (2021). Anomaly process detection using negative selection algorithm and classification techniques. *Evolving Systems, 12(3),* 769–778. DOI 10.1007/s12530-019-09317-1.
84. Bindra, N., Sood, M. (2019). Detecting DDoS attacks using machine learning techniques and contemporary intrusion detection dataset. *Automatic Control and Computer Sciences, 53(5),* 419–428. DOI 10.3103/S0146411619050043.
85. Lee, J., Kim, J., Kim, I., Han, K. (2019). Cyber threat detection based on artificial neural networks using event profiles. *IEEE Access, 7,* 165607–165626. DOI 10.1109/Access.6287639.

86. Shukla, A. K. (2021). Detection of anomaly intrusion utilizing self-adaptive grasshopper optimization algorithm. *Neural Computing and Applications, 33(13),* 7541–7561. DOI 10.1007/s00521-020-05500-7.

87. Kaur, S., Singh, M. (2020). Hybrid intrusion detection and signature generation using deep recurrent neural networks. *Neural Computing & Applications, 32(12)*. DOI 10.1007/s00521-019-04187-9.

88. Elmasry, W., Akbulut, A., Zaim, A. H. (2020). Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic. *Computer Networks, 168,* 107042. DOI 10.1016/j.comnet.2019.107042.