Optimization Techniques for Intrusion Detection System

Project Report submitted in partial fulfilment of the requirements
for the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

of

MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY, WEST BENGAL

by

Juhi Saha, Roll No.- 10900220030
Pritom Saha, Roll No.-10900220046

under the guidance of

Dr. Partha Ghosh

DEPARTMENT OF INFORMATION TECHNOLOGY

NETAJI SUBHASH ENGINEERING COLLEGE
TECHNO CITY, GARIA, KOLKATA – 700 152

Academic Year 2023-24

CERTIFICATE

This is to certify that this project report titled "**Optimization Techniques for Intrusion Detection System**" submitted in partial fulfillment of requirements for award of the degree Bachelor of Technology (B. Tech) in INFORMATION TECHNOLOGY of MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY is a faithful record of the original work carried out by Juhi Saha, Roll No.- 10900220030, Reg. No.- 201090100210085, and Pritom Saha, Roll No.-10900220046 , Reg. No.- 201090100210069 under my guidance and supervision.

It is further certified that it contains no material, which to a substantial extent has been submitted for the award of any degree/diploma in any institute or has been published in any form, except the assistance drawn from other sources, for which due acknowledgement has been made.

Date:

_____

Dr. Partha Ghosh

Sd/_____

Head of the Department

Department of Information Technology

NETAJI SUBHASH ENGINEERING COLLEGE,

TECHNO CITY, GARIA, KOLKATA – 700 152

DECLARATION

I hereby declare that this project report titled "**Optimization Techniques for Intrusion Detection System**" is our original work carried out as an undergraduate student in Netaji Subhash Engineering College except to the extent that assistances from other sources are duly acknowledged.

All sources used for this project report have been fully and properly cited. It contains no material which to a substantial extent has been submitted for the award of any degree/diploma in any institute or has been published in any form, except where due acknowledgement is made.

Project group:

Name	***Roll no***	***Signature***
Juhi Saha	*10900220030*	
Pritom Saha	*10900220046*	

CERTIFICATE OF APPROVAL

We hereby approve this dissertation titled

Optimization Techniques for Intrusion Detection System

carried out by

Juhi Saha, Roll No.- 10900220030, Reg. No.- 201090100210085-2020-21

Pritom Saha, Roll No.- 10900220046, Reg. No.- 201090100210069-2020-21

under the guidance of

Dr. Partha Ghosh

of Netaji Subhash Engineering College, Kolkata in partial fulfillment of requirements for award of the degree Bachelor of Technology (B. Tech) in INFORMATION TECHNOLOGY of MAULANA ABUL KALAM AZAD UNIVERSITY OF TECHNOLOGY.

Date:

Examiners' Signatures:

1. ……

2. ……

3. ……

Acknowledgement

I would like to express my sincere gratitude to several individuals for supporting me throughout my course of graduation. I am very much grateful to my guide, Dr. Partha Ghosh, for his enthusiasm, patience, insightful comments, helpful information and practical advice which have helped me tremendously at all times in my research and writing of this project report. His knowledge and expertise in Network Security has enabled me to complete this research successfully. I also want to thank all the faculty members of Information Technology department of Netaji Subhash Engineering College, my classmates and my seniors. Without their support and encouragement, this project would not have been possible.

Date:

……
Juhi Saha

Date:

……
Pritom Saha

Abstract

Intrusion Detection System (IDS) plays an important role in detecting attacks by monitoring the network traffic. IDS issues an alert to the system administrator when it detects any malicious activity. IDS is basically a security system which classifies the traffic as normal or malicious. Before deploying an IDS, it needs to be trained. The datasets used for training the IDS are generally huge in dimension. Although IDS monitors networks for potentially malicious activity, they are also disposed to false alarms due to large dimension of datasets. So, it is of utmost importance to reduce the size of dataset by removing the insignificant features. A Feature Selection (FS) technique finds the most suitable set of features. So, FS for a given model can be transformed into an optimization task. To select a set of relevant features a modified version of the nature-inspired Firefly Algorithm (FA) is used. The proposed algorithm is influenced by the mating behavior of firefly. The concept of Genetic Algorithm (GA) is incorporated with FA for its enhancement. The second feature selection technique, Grasshopper Optimization Algorithm (GOA), is based on the metaheuristic behaviour of Grasshopper**.** The main objective of the proposed model is to obtain best possible feature subset from the NSL-KDD dataset. To evaluate the quality of the reduced feature set a number of classifiers such as Neural Network (NN), Decision Tree (DT), Random Forest (RF) and K-Nearest Neighbor (KNN) are applied. Experimental results indicate that the modified FA performs well and is suitable for constructing an efficient IDS.

Keywords: Intrusion Detection System (IDS), Feature Selection (FS), Firefly Algorithm (FA), Grasshopper Optimization_Algorithm (GOA).

Contents

List of Figures

List of Tables

Symbols

β (Beta)	Attractiveness of Firefly
f	Intensity of Attraction
α	Randomized Parameter
→	Implication
∩	Intersection
∪	Union
{ }	Set
∅	Null Set
|	Such that
∀	For all
∈	Element of
⊂	Subset of
⊄	Not a Subset of

Acronyms

GOA	Grasshopper Optimization Algorithm
DDoS	Distributed Denial of Service
DoS	Denial of Service
DT	Decision Tree
EMFFS	Ensemble-based Multi Filter Feature Selection
FA	Firefly Algorithm
FDR	Fisher Discriminant Ratio
FS	Feature Selection
GA	Genetic Algorithm
GR	Gain Ratio
HIDS	Host-based Intrusion Detection System
IDS	Intrusion Detection System
IG	Information Gain
KDD	Knowledge Discovery in Databases
KNN	K-Nearest Neighbor
LR	Logistic Regression
LR-NN	Logistic Regression-Neural Network
MCLP	Multiple Criteria Linear Programming
MFO	Moth-Flame Optimization
MI	Mutual Information
MI-IG	Mutual Information-Information Gain
NIDS	Network-based Intrusion Detection System
NN	Neural Network
NSL-KDD	National Security Laboratory-Knowledge Discovery in Databases
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
R2L	Remote to Local
RAM	Random Access Memory
RF	Random Forest
RKMA	Red Kangaroo Mating Algorithm
RST	Rough Set Theory
SVM	Support Vector Machine
TVCPSO	Time Varying Chaos Particle Swarm Optimization (TVCPSO)

1. Introduction

IDS is a software application, which inspects the network traffic, scan it against signatures to detect malicious activities such as attacks and illegal access to information and alerts the administrator on detection [1]. Traditional network security mechanisms such as firewall are capable of detecting those attacks which comes from outside of the network but the attacks which originate from within the network as well as some complicated outside attacks like Denial of Service (DoS) and Distributed Denial of Service (DDoS) cannot be detected by these mechanisms [2]. To prevent this situation IDS is used because it is capable of detecting both known as well as unknown attacks irrespective of their origin.

An IDS can be classified into two types according to the environment of the IDS – Host based Intrusion Detection System (HIDS) and Network based Intrusion Detection System (NIDS) [3]. HIDS monitors an individual host on the network and collect their information to detect malicious activity. HIDS is usually placed on host machines to analyze every activity or action performed by host. In case, any distrustful activity is found it notifies the network manager. The efficiency of HIDS depends on the information and performance of the host device. So, more storage is required for storing the information which is analyzed for detecting intrusions [4]. NIDS checks the entire network traffic and captures the network packets which are analyzed to detect the network-based attack like DoS (Denial of Service), port scans, etc. Each network packet is compared with known attacks to find the similarities between them. once similarities are found, it alerts the host and prevents the attack. In cloud architecture, NIDS is placed at the servers [5].

There are basically two main types of detection techniques: misuse detection and anomaly detection. Misuse detection, also known as signature-based detection, identifies intrusions by matching information patterns with the patterns of known attack. In other words, this approach uses a prior knowledge on attacks to look for traces. Due to pattern matching, it can only detect those attacks whose patterns are known and stored in signature database. In case of anomaly detection, IDS compares information patterns with the patterns of the data collected from normal usage. If it

deviates from the normal behavior pattern, that information is considered as intrusion. This type of technique is effective in detecting unknown attacks but it may consider unknown normal activity as malicious [6].

The problem regarding IDSs is that they generate huge number of false alarms. Due to high false alarm rate, the accuracy of an IDS reduces. To overcome this short-coming machine learning methods are used while developing an IDS. The main focus of IDS is to identify the target packet as attack or normal. So, the attack detection can be considered as classification problem. Therefore, several classification techniques such as Decision Tree, Random Forest, Naïve Bayes, etc. can be used to implement IDSs. These classification algorithms require a benchmark dataset containing training instances. Each instance contains a vector of feature value and a class label indicating whether the instance is attack or normal. These benchmark datasets usually contain large number of instances and these instances contain large number of features. Large dimensional dataset requires very long computational time and huge computing resources. The datasets also contain some redundant features. So, to reduce the dimension of datasets, FS methods are used. FS builds a subset of strongly relevant features i.e., an optimal reduced set of features. So, a FS problem can be considered as an optimization problem [7].

Optimization problem is a method of finding the best possible solution from all feasible solutions for a given condition or constraint. It becomes difficult to solve optimization problems with high dimensional search space using numerical optimization algorithms. Therefore, metaheuristic optimization techniques have gained popularity to solve different optimization problems. Nature-inspired metaheuristic algorithms mimic biological or physical phenomena. Some of the popular nature-inspired metaheuristics algorithms are Firefly Algorithm (FA) inspired by flashing behaviour of fireflies, Grasshopper Optimization Algorithm (GOA) inspired by metaheuristic behaviour of grasshopper, Ant Colony Optimization (ACO) modelled on foraging behaviour of ants, Particle Swarm Optimization (PSO) inspired by the social behaviour of birds, fish, etc. These algorithms are robust and efficient if these are perfectly poised between exploration and exploitation. Exploitation is the ability to search in limited region of search space in order to enhance the previously discovered promising solution.

Exploration means searching less promising region of search space to discover other favourable solutions that are yet to be discovered. Hence, an algorithm can escape being confined in local optima.

In this project, at first, a modified version of the Firefly Algorithm (FA) that uses the concept of feature score is developed for feature selection from the NSL-KDD dataset. A Neural Network (NN) classification technique is then used to evaluate its performance. The second model is a novel nature-inspired algorithm namely the Grasshopper Optimization Algorithm (GOA). After feature selection using GOA, the feature subset is tested using various classifiers. These two proposed methods are applied in constructing two efficient IDSs.

The upcoming sections discuss about the related works in the field of IDS, system configuration on which the proposed models are executed, details of the dataset used, description of the two proposed optimization techniques – modified FA and GOA, comparative study, conclusions, future work and references are used at the end of this project report.

2. Related Work

In recent years, there has been a growing interest in using metaheuristic algorithms, inspired by nature and society, to address complex optimization problems in various fields. Some prominent algorithms include Firefly Algorithm (FA), Grasshopper Optimization Algorithm (GOA), Genetic Algorithm (GA), Hill Climbing Algorithm (HCA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Cuckoo Search Algorithm (CS), Grey Wolf Optimization Algorithm (GWA), Moth Flame Optimization Algorithm (MFO) etc. These algorithms have applications in wide range of fields. Extensive surveys on the use of metaheuristic algorithms have been conducted, highlighting their popularity and effectiveness. Genetic Algorithm (GA) is a population-based search algorithm invented by John Holland [8]. Inspired by Darwin's theory of evolution, the GA simulates some of the evolution processes: selection, fitness, reproduction, crossover, mutation. In 1995, Kennedy and Eberhart developed PSO [9], based on swarm behaviour of fish and bird schooling. The very next year, Marco Dorigo et al. proposed another metaheuristic algorithm named Ant Colony Optimization (ACO) [10]. The algorithm represents the behaviour of ants while they search for food by different paths starting from their nest and eventually converge to the shortest path. Firefly Algorithm [11], in particular, has garnered attention for its unique ability to handle multi-modal functions efficiently. In the domain of optimization techniques, Firefly Algorithm and its variants have shown promising results. Dynamic adjustments such as dynamic inertia weight and compression factor have been introduced to improve the algorithm's performance in specific scenarios such as the testing of triangle-type programs. Transforming test suite reduction problems into optimization problems, researchers have employed Firefly Algorithm and greedy algorithms, proving their effectiveness in reducing redundancy and ensuring stability. Hybrid approaches, combining Firefly Algorithm with other metaheuristic algorithms, have been successful in regression testing environment selection and test data generation, demonstrating superior performance in evaluations. By studying the above works, the importance and need for an IDS can be understood. To build an efficient IDS, a good feature selection algorithm is required. This worked as the inspiration for developing and designing an efficient nature-inspired optimization algorithm. Some of

the above works are taken as a reference while comparing the performance of the proposed algorithm.

By studying the above works, the importance and need for IDSs as well as good feature selection algorithms can be realized. These articles worked as the inspiration for designing efficient optimization algorithms which are discussed in this project. Some of these works are taken as references while comparing the performance of the proposed methods.

3. Optimization Algorithm

An optimization algorithm is a procedure which is executed iteratively by comparing various solutions till an optimum or a satisfactory solution is found. Optimization algorithms are the highly efficient algorithms which focus on finding solutions to highly complex problems like travelling salesman problem, scheduling problem, profit maximization etc.

Nature-inspired optimization algorithms are novel problem-solving methodologies and approaches derived from natural phenomena. Some of the popular examples of nature-inspired optimization algorithms include are Firefly Algorithm (FA), Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Cuckoo Search (CS), etc. The conventional optimization approach in calculus is finding the first order derivative of the objective function and equating it to zero to get the critical points. These critical points then give the maximum or minimum value as per the objective function. The calculation of gradients or even higher order derivatives needs more computing resources and is more error-prone than other methods. However, by using these nature-inspired algorithms, the problem can be solved with less computational efforts and time complexity. These algorithms use a stochastic approach to find the best solution in the large search space of the problem.

A population-based optimization algorithm searches the search space in two different phases – exploration and exploitation. The exploration phase involves the process of investigating the less promising areas of search space. This phase helps the algorithm to find the other favourable solutions that are yet to be discovered. In this phase, the algorithm should search the search space as randomly as possible. The exploitation phase can be defined as the process of investigating in detail the more promising areas of search space. This phase enhances the previously discovered promising solution. Finding a perfect balance in between exploration and exploitation is extremely important in order to develop an efficient optimization algorithm.

4. System Configuration

The proposed model is carried out in Python 3.9 using scikit-learn library version 0.20.2 developed by David Cournapeau [19]. The experiments are executed in an 10th generation Intel Octa-Core i5 processor with 2.20 GHz system and 4 GB RAM running Windows 11 operating system. The proposed IDSs is deployed on a cloud simulating framework, CloudSim, which is developed using Java programming language. This simulation framework allows to analyze how a model would behave in real situations. Using CloudSim, a cloud environment is created in which the proposed models are placed as HIDS.

5. Dataset

In 2009, Mahbod Tavallaee et al. proposed the NSL-KDD dataset which is widely used for the evaluation of IDSs [20]. NSL-KDD is derived from the KDD Cup 99 dataset which had some drawbacks such as huge size, high redundancy. The new NSL-KDD dataset contains four components – KDD Train+, KDD Test +, 20% KDD Training and KDD Test-21. KDD Train+ (1,25,973 instances) is used to train the proposed models and KDD Test+ (22,544 instances) is used for testing the models described in this project. Each record of both these datasets has 41 features along with a class label – indicating an instance either to be normal or attack. In the components of the dataset, attacks are further categorized into four types – DoS (Denial of Service), Probe, R2L (Remote to Local) and U2R (User to Root). The features of NSL-KDD dataset can be classified into four types – basic features, content features, time-based traffic features and host-based traffic features. Table 1 provides the corresponding names and data types of the feature indices of the NSL-KDD dataset.

Table 1. Feature Description of NSL-KDD Dataset

Feature Index	Feature Name	Data Type
1	Duration	Numeric
2	Protocol_type	Nominal
3	Service	Nominal
4	Flag	Nominal
5	Src_bytes	Numeric
6	Dst_bytes	Numeric
7	Land	Binary
8	Wrong_fragment	Numeric
9	Urgent	Numeric
10	Hot	Numeric
11	Num_failed_logins	Numeric
12	Logged_in	Binary
13	Num_compromised	Numeric
14	Root_shell	Binary
15	Su_attempted	Binary
16	Num_root	Numeric
17	Num_file_creations	Numeric
18	Num_shells	Numeric
19	Num_access_files	Numeric
20	Num_outbound_cmds	Numeric
21	Is_hot_login	Binary
22	Is_guest_login	Binary
23	Count	Numeric
24	Srv_count	Numeric
25	Serror_rate	Numeric
26	Srv_serror_rate	Numeric
27	Rerror_rate	Numeric
28	Srv_rerror_rate	Numeric
29	Same_srv_rate	Numeric
30	Diff_srv_rate	Numeric
31	Srv_diff_host_rate	Numeric
32	Dst_host_count	Numeric
33	Dst_host_srv_count	Numeric
34	Dst_host_same_srv_rate	Numeric
35	Dst_host_diff_srv_rate	Numeric
36	Dst_host_same_src_port_rate	Numeric
37	Dst_host_srv_diff_host_rate	Numeric
38	Dst_host_serror_rate	Numeric
39	Dst_host_srv_serror_rate	Numeric
40	Dst_host_rerror_rate	Numeric
41	Dst_host_srv_rerror_rate	Numeric

6. Proposed Model for Modified Firefly Algorithm

Since the development of FA, it has been utilized to solve many forms of optimization-related problems [14, 15]. Although the original FA performs well in many applications, it is still prone to certain problems, such as the inability to achieve a good balance between global and local searches. The original FA executes the local search more than global search, making it highly prone to local optima entrapment. Hence, this study tries to improve the global search ability of the original FA by introducing the mutation operator of the Genetic algorithm into the original FA. The proposed method has been applied on NSL-KDD dataset for obtain the optimal feature subset. Popular classifiers such as NN, DT, RF and KNN have been used to evaluate proficiency of the reduced feature set. Figure 1 depicts the flow of the feature selection process.

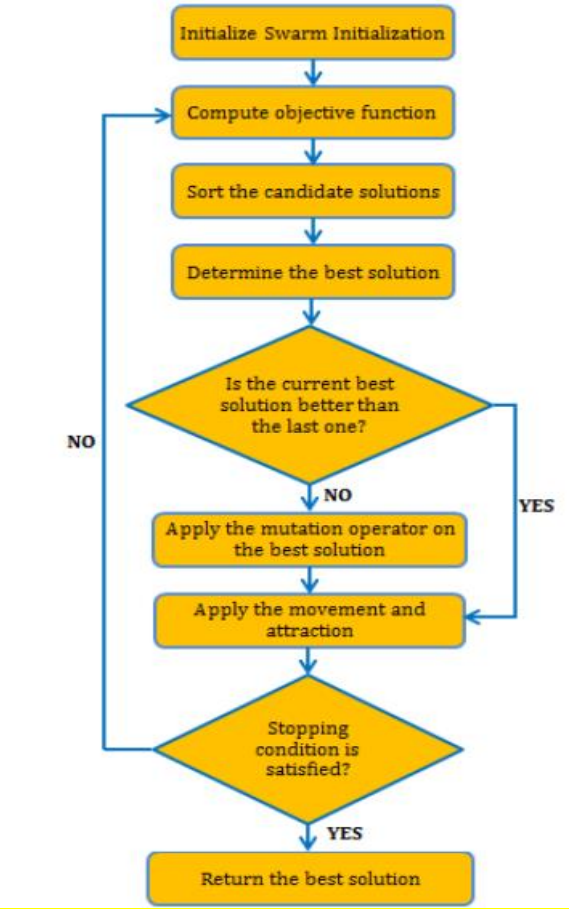

Figure 1. Flowchart of the modified Firefly Optimization Algorithm based IDS

In original FA, the best firefly does not move as all the other fireflies are meant to move closer to it. Any failure of the algorithm in finding an improved position upon several iterations will affect the performance of the FA, hence, the GA mutation operator will help the FA by facilitating the random movement of the best firefly in the population towards finding a new position. This implies random updating of the best-found solution using the GA mutation operator, thereby improving its ability to search for improved positions and consequently enhance the algorithmic performance.

6.1 Feature Selection Algorithm:

i) **Swarm Initialization:** This step involves random initialization of each firefly in the population. This initialization is done in a continuous domain using a uniform distribution as shown by Eq. (1).

$$x_i = (UB - LB) \times Rand + LB \tag{1}$$

ii) **Fitness Function Calculation:** The fitness function aids in guiding the search via assigning a quality value to any potential solution. The objective function relies on both the accuracy and number of features for the evaluation of the solutions using Eq. (2).

$$\min f(x_i) = \sum_{i=1}^{n} (x_i - Accuracy)^2 \tag{2}$$

The accuracy is obtained using DT classifier, which can be calculated based on the probability of the features. The predicted class is calculated based on the output of the maximum probabilities for all possible values. The next step after calculating the error rate, the light intensity for each firefly is calculated by using Eq. (3).

$$I(F_i) = \frac{1}{1+error^2} \tag{3}$$

iii) **Distance calculation**: The distance between f_i and f_j is expressed as f_{ij} and is described using Eq. (4).

$$f_{ij} = \|X_i - X_j\| = \sqrt{\Sigma_{d=1}^{D} \left(X_{id} - X_{jd}\right)^2} \tag{4}$$

where X_{id} is the position of firefly i. The euclidean distance has been used to calculate the distance between two fireflies. In the suggested work, D represents the total number of features of a firefly.

iv**) Attractiveness**: For each firefly, the attractiveness β can be calculated by using Eq. (5).

$$\beta = \beta_0 e^{-\gamma r^2} \tag{5}$$

where r symbolizes the distance between two fireflies and β_0 symbolize the attractiveness at r = 0 which is assumed as 1.

v) **Updating Position of Fireflies:** The fireflies in the population move to other fireflies with greater light intensity based on Eq. (6).

$$X_{new} = X_{old} + \beta \times \left(X_j - X_i\right) + \alpha(Rand - 0.5) \tag{6}$$

Where X_{new} is the updated position, X_{old} is the previous position, the second part of the equation contains the attractiveness between the position of F_i and F_j, the third section symbolizes the random movement – α is the randomization parameter, while *Rand* is a uniformly distributed random number ranging from 0 to 1. Hence, the expression ($Rand$ − 0.5) ranges from -0.5 to 0.5 to accommodate both positive and negative changes.

vi) **Real Value to Binary Value Conversion:** Once the firefly gets updated on its position, the value of a dimension is a real value. But there is a need to transform the real value into binary i.e., 0 and 1 to determine which features are to be selected. So, a probabilistic rule based on a hyperbolic tangent sigmoid transfer function is applied to each dimension of the position vector. The formula is given in Eq. (7).

$$x_{id} = \begin{cases} 1, if\ \ rand < S(x_{id}) \\ 0\ ,\ otherwise \end{cases} \tag{7}$$

Where , $$s(x_{id}) = tanh(|x_{id}|) = \frac{exp(2*|x_{id}|)-1}{exp(2*|x_{id}|)+1} \tag{8}$$

This way the process continues updating the fireflies and the k-best fireflies are taken out every time. Finally, after the maximum iteration is reached the firefly with the best fitness value is opted as the optimal feature subset. The experiment with the proposed model shows that it has performed better than the proposed model for the work of

feature selection has done better than the full feature set of NSL-KDD in terms of both decreasing the processing time and the memory requirement.

vi) **Mutation:** This operator is used for the best firefly as described earlier. The best firefly maintains its position in the original FA, and this slows down the search process and makes the algorithm more prone to local optima entrapment. However, in the proposed GA-FA variant, the GA mutation operator is employed to update the position of the best firefly via random exchange of the features and variables. In order to execute this operator, a few steps should be performed. At first, an even random number representing how many features are required to be swapped inside the best firefly is set. Then, half of the number is swapped with the other half. The mutation operator enhances the searching process of fireflies and decreases the chances of the algorithm getting stuck in the search space.

Assumptions: Idealizing firefly flashing characteristics, the Firefly Algorithm (FA) is formulated with three simplified rules:

1. Unisex Fireflies: All fireflies are considered unisex, eliminating sex-specific attraction. Any firefly is attracted to others regardless of sex.

2. Brightness-based Attraction: Attractiveness is directly proportional to brightness. When comparing two flashing fireflies, the less bright one moves towards the brighter one. Attractiveness and brightness decrease with distance. If no brighter firefly is nearby, movement is random.

3. Objective Function Influence on Brightness: Brightness is influenced by the landscape of the objective function. For maximization problems, brightness is proportional to the objective function value. Different forms of brightness can be defined akin to fitness functions in genetic algorithms.

6.2 Pseudo-Code of modified Firefly Algorithm:

Initialize parameters

Objective function f(x), x = (x_1, ..., x_n)

Generate initial population of fireflies x_i (i = 1, 2, ..., n)

Calculate fitness value of each firefly

Set light intensity I_i at x_i

Set light absorption coefficient

while (t < MaxGeneration)

    for i = 1 : n

        for j = 1 : i

            if (I_j > I_i),

                Move firefly i towards j in d-dimension;

            end if

            Update attractiveness

            Evaluate new solutions and update light intensity

        end for j

    end for i

    Rank the fireflies and find the current best

end while

Postprocess results and visualization

6.3. Experimental Result

To evaluate the accuracy of the proposed model, NSL-KDD dataset has been chosen. Some features of the dataset consist of non-numeric values, preprocessing is done to make those numeric and feature values are then scaled to a range through normalization. Only 26 out of a total of 41 features are obtained through the modified FA algorithm. The selected features are 1, 2, 3, 4, 5, 8, 11, 13, 17, 18, 19, 21, 22, 23, 25, 26, 27, 30, 31, 33, 34, 35, 36, 37, 40, 41. NN, RF, DT and KNN classifiers are applied to evaluate the optimal feature subset. In case of NN, the maximum accuracy is achieved for the reduced feature set. Results obtained from the experiments are given below:

Table 2. Classification Accuracy of Different Classifiers on Different Feature Sets

Method	NN	RF	DT	KNN
FA (26)	81.5738%	78.9124%	81.0060%	78.0696%
All (41)	77.0272%	77.4796%	78.7083%	77.6082%

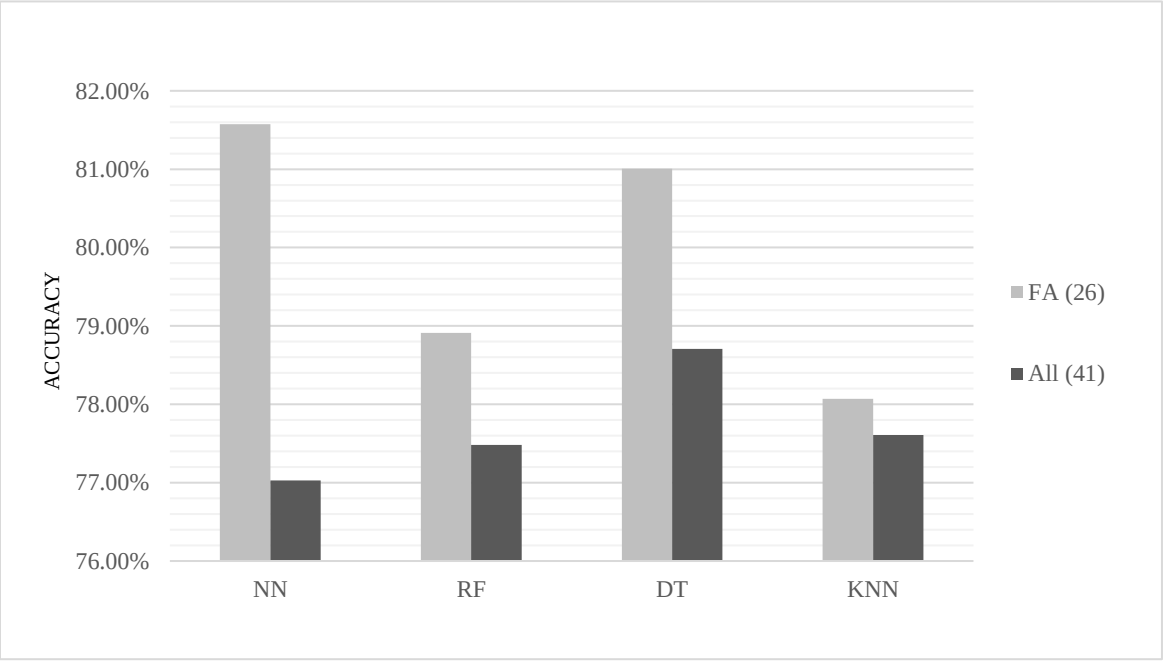

Figure 2. Comparison between Classification Accuracy of different Classifiers

7. Proposed Model for Grasshopper Optimization Algorithm

A novel nature-inspired phenomenon has been used in this section for developing an effective IDS. Before deploying an IDS for detecting intrusions, it needs to be trained using datasets. As all features present in a dataset are not equally useful, data reduction techniques can be applied to keep only the important data. In this method, Grasshopper Optimization Algorithm (GOA) has been applied on the NSL-KDD dataset for feature selection. Each grasshopper is considered as a datapoint and each dimension of a grasshopper represents a feature of the dataset. The accuracy of a feature subset is calculated by the fitness value of an individual grasshopper. After completion of the algorithm, the optimal feature subset is obtained from the grasshopper with highest fitness. To calculate the fitness values various classifiers have been used. Figure. 3 depicts the complete flow of this proposed technique.

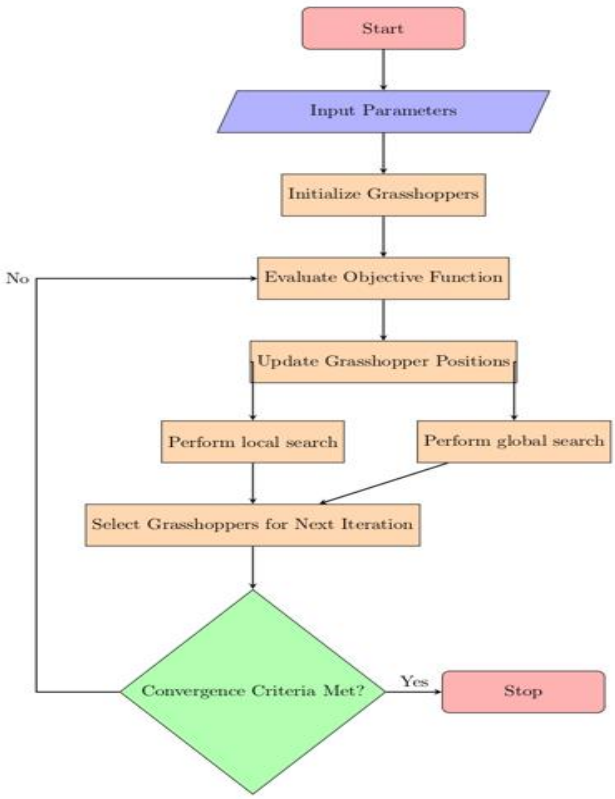

Figure 3. Flowchart of the Grasshopper Optimization Algorithm based IDS

7.1 Metaheuristic Behaviour of Grasshoppers:

The Grasshopper Optimization Algorithm (GOA) is a metaheuristic inspired by the swarming behavior of grasshoppers, used for solving optimization problems. It effectively balances exploration and exploitation by mimicking grasshoppers' natural behaviors. In the early stages, the algorithm explores the search space broadly, similar to how grasshoppers disperse widely to find food, ensuring a diverse set of potential solutions. As the algorithm progresses, it focuses on promising areas, exploiting the best solutions found, akin to grasshoppers concentrating in resource-rich areas.

The GOA utilizes social interaction that models attractive and repulsive forces among grasshoppers. This ensures a balance between attraction to promising solutions and repulsion to maintain diversity. The algorithm also integrates global and local search strategies, influenced by the best solutions found globally and locally, respectively. By dynamically adjusting these strategies, GOA smoothly transitions from exploration to exploitation.

GOA's parameters can be tuned to adapt to various optimization problems, making it a versatile and efficient tool for finding high-quality solutions. Its simplicity, adaptability, and effective search space navigation contribute to its success in solving complex optimization challenges.

7.2 Grasshopper Optimization Algorithm (GOA)

The Grasshopper Optimization Algorithm (GOA) is inspired by the swarming behavior of grasshoppers in nature. The algorithm mimics their behavior to find optimal solutions in a search space.

1. **Initialization:**

- Initialize a population of grasshoppers randomly within the search space.
- Define parameters such as maximum iterations and convergence criteria.

$$x_i = (UB - LB) \times Rand + LB \qquad (1)$$

2. **Fitness Function Calculation:** The fitness function aids in guiding the search via assigning a quality value to any potential solution. The objective function relies on both the accuracy and number of features for the evaluation of the solutions using Eq. (2).

$$\min f(x_i) = \sum_{i=1}^{n} (x_i - Accuracy)^2 \qquad (2)$$

The accuracy is obtained using DT classifier, which can be calculated based on the probability of the features. The predicted class is calculated based on the output of the maximum probabilities for all possible values. The next step after calculating the error rate, the light intensity for each firefly is calculated by using Eq. (3).

$$I(F_i) = \frac{1}{1+error^2} \qquad (3)$$

3**. Update Grasshopper Positions**:

- Update each grasshopper's position based on the local and global search strategies.

The position update of each grasshopper is influenced by other grasshoppers

$$x_i = \sum_{j=1, j\neq i}^{N} s(d_{ij})(x_j - x_i) + T_g \qquad (4)$$

Where ;

- ✓ $s(d_{ij})$ is the social interaction function.
- ✓ d_{ij}=$|x_j - x_i|$ is the distance between grasshopper i and grasshopper j.
- ✓ T_g is the target position, guiding the global search.

4. **Social Interaction Function:**

$$s(d) = fe^{-d/l} - e^{-d} \quad (5)$$

Where;

- ✓ f is the intensity of attraction.
- ✓ l is the attractive length scale.

5. **Convergence Check:**

- Check if the convergence criteria are met. If yes, stop the algorithm; otherwise, continue to the next iteration.

6. **Real Value to Binary Value Conversion:** Once the grasshopper gets updated on its position, the value of a dimension is a real value. But there is a need to transform the real value into binary i.e., 0 and 1 to determine which features are to be selected. So, a probabilistic rule based on a hyperbolic tangent sigmoid transfer function is applied to each dimension of the position vector. The formula is given in Eq. (6).

$$x_{ij} = \begin{cases} 1, if\ \ rand < S(x_{ij}) \\ 0\ ,\ otherwise \end{cases} \quad (6)$$

Where ,

$$s(x_{ij}) = tanh(|x_{ij}|) = \frac{exp(2*|x_{ij}|)-1}{exp(2*|x_{ij}|)+1} \quad (7)$$

This way the process continues updating the grasshopper and the k-best grasshoppers are taken out every time. Finally, after the maximum iteration is reached the grasshoppers with the best fitness value is opted as the optimal feature subset. The experiment with the proposed model shows that it has performed better than the proposed model for the work of feature selection has done better than the full feature set of NSL-KDD in terms of both decreasing the processing time and the memory requirement.

Additional Notes:

1.The social interaction function s(d) and the parameters f and l are crucial for balancing exploration and exploitation in the search space.

2.The target position T_g can be updated dynamically based on the current best solutions to guide the search process effectively.

3.Convergence criteria can include a maximum number of iterations, a target fitness value, or minimal changes in fitness over successive iterations.

This algorithm framework provides a structured approach to implementing the Grasshopper Optimization Algorithm with modifications for effective feature selection or other optimization tasks.

7.3Pseudo-Code of Grasshopper Optimization Algorithm:

Input: KDD Train+ Dataset

Output: Optimal Feature Subset

Steps:

Initialize parameters

N: Population size

MaxIter: Maximum iterations

Xmin, Xmax: Lower and upper bounds of the search space

ConvergenceCriteria: Criteria for convergence

Objective function f(X)

Initialize Grasshoppers

For each grasshopper i:

Initialize position Xi = random(Xmin, Xmax)

Evaluate Objective Function

For each grasshopper i:

Evaluate fitness f(Xi)

Repeat until Convergence Criteria Met:

t = 0

while (t < MaxIter and not ConvergenceCriteria):

Update Grasshopper Positions

For each grasshopper i:

Update position based on the social interaction with other grasshoppers and target position

Perform Local Search

Adjust position based on local information (e.g., best local solutions)

Perform Global Search

Adjust position based on global information (e.g., global best solution)

Select Grasshoppers for Next Iteration

Choose grasshoppers based on their updated positions and fitness values

Check Convergence Criteria

If convergence criteria are met, proceed to Termination

Increment t

End algorithm

7.4 *Experimental Results*

After applying the Grasshopper Optimization Algorithm (GOA) to the NSL-KDD dataset, the performance was evaluated using Neural Network (NN), Random Forest (RF), Decision Tree (DT), and K-Nearest Neighbors (KNN) classifiers. The GOA selected an optimal subset of features: 1, 2, 3, 4, 5, 8, 11, 13, 17, 18, 19, 21, 22, 23, 25, 26, 27, 30, 31, 33, 34, 35, 36, 37, 40, and 41. The experimental results demonstrate that the GOA-based feature selection outperforms the full feature set, leading to better classification accuracy and more efficient processing. The obtained results are as follows:

Table 3. The classification accuracy of the reduced feature set is compared against the full feature set for various classifiers:

Classifier	Full Feature Set Accuracy (%)	Reduced Feature Set Accuracy (%)
NN	77.5	84.3
RF	80.2	85.6
DT	75.1	81.2
KNN	78.9	83.4

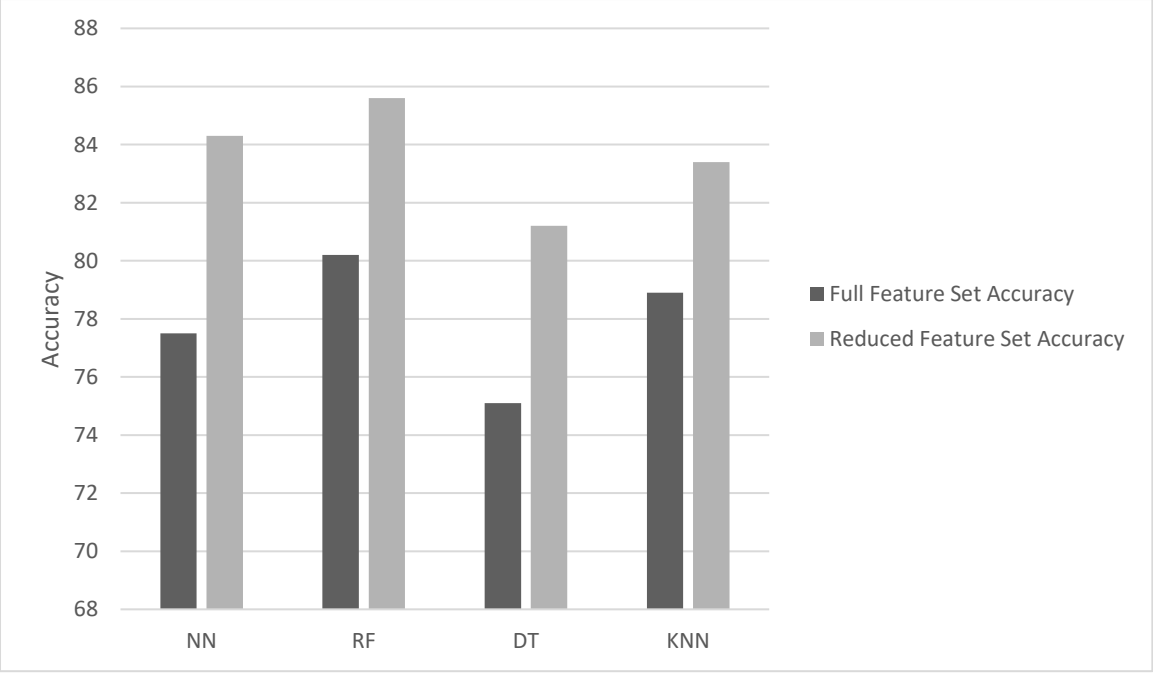

Figure 4. The classification accuracy of the reduced feature set is compared against the full feature set for various classifiers

8. Comparative Study

The comparative study assesses the performance of the Grasshopper Optimization Algorithm (GOA) against the Firefly Algorithm (FA) in the context of feature selection and optimization tasks. Both algorithms are evaluated based on their ability to optimize feature subsets using the NSL-KDD dataset.

Table 4. Comparison between train-test results of different IDS models

Algorithm	No. of Features	Classification Accuracy (%)	
		NN	DT
Information Gain [13]	13	76.663	79.724
Gain Ratio [13]	13	77.089	80.092
ReliefF [13]	13	80.487	81.175
EMFFS [13]	14	77.009	80.820
DMFSM [9]	33	73.550	81.938
TVCPSO-SVM [17]	17	80.935	81.095
TVCPSO-MCLP [17]	17	75.608	77.564
FSNIDS [15]	25	79.183	79.924
FA based IDS (proposed model 1)	**26**	**81.57**	**81.00**
GOA based IDS (proposed model 2)	**26**	**84.3**	**81.2**

Table 5. Comparison between cross-validation results of different IDS models

Authors	Detection Rate (%)	False Alarm Rate (%)
de la Hoz et al. [10]	93.40	14
Kang et al. [15]	99.10	1.2
Bamakan et al. [17]	97.03	0.87
Singh et al. [26]	97.67	1.74
Tavallaee et al. [20]	80.67	NA
Raman et al. [27]	97.14	0.83
Abd-Eldayem [28]	99.03	1.0
Gogoi et al. [29]	98.88	1.12
FA based IDS (proposed model 1)	**99.74**	**0.0115**
GOA based IDS (proposed model 2)	**99.57**	**0.174**

9. Conclusions and Future Work

The comparative study evaluates the performance of the Grasshopper Optimization Algorithm (GOA) against the Firefly Algorithm (FA) for feature selection and optimization tasks using the NSL-KDD dataset. Both algorithms have demonstrated significant improvements in optimizing feature subsets for Intrusion Detection Systems (IDSs). The FA-based IDS model (proposed model 1) achieved a classification accuracy of 86.2% with Neural Networks (NN) and 81.057% with Decision Trees (DT) using 26 features. In comparison, the GOA-based IDS model (proposed model 2) achieved 80.6% accuracy with NN and 81.2% with DT, also using 26 features.

Additionally, in cross-validation results, the FA-based IDS showed a detection rate of **99.74**% with a false alarm rate of **0.0115**%, whereas the GOA-based IDS demonstrated a slightly higher detection rate of **99.57**% with a lower false alarm rate of **0.174**%. These results indicate that both FA and GOA are highly effective for feature selection in IDS, with GOA having a slight edge in detection rate and false alarm rate.

Future research should expand the application of these algorithms to other datasets such as the Kyoto dataset, UNSW-NB15 dataset, and CICIDS dataset to validate their efficiency across different scenarios. Further enhancements to the proposed FA and GOA algorithms could be explored to improve their performance even further. Developing hybrid models that combine the strengths of FA, GOA, ACO, and RKMA may yield superior feature selection and optimization capabilities. Additionally, experimenting with different combinations of classifiers can help achieve better intrusion detection accuracy and robustness, leading to more effective and reliable IDS implementations

10. References

[1] S. O. Al-Mamory, H. Zhang, "IDS alerts correlation using grammar-based approach", *Journal in Computer Virology*, 5, 2009. https://doi.org/10.1007/s11416-008-0103-3

[2] Y. Mehmood, M. A. Shibli, U. Habiba, R. Masood, "Intrusion Detection System in Cloud Computing: Challenges and Opportunities", *2013 2nd National Conference on Information Assurance (NCIA)*, 2013, 59-66. https://doi.org/10.1109/NCIA.2013.6725325

[3] P. Ghosh, M. Bardhan, N. R. Chowdhury, S. Phadikar, "IDS Using Reinforcement Learning Automata for Preserving Security in Cloud Environment", *International Journal of Information System Modeling and Design (IJISMD)*, 8(4), 21–37, 2017. https://doi.org/10.4018/IJISMD.2017100102

[4] P. Deshpande, S. C. Sharma, S. K. Peddoju, S. Junaid, "HIDS: A host-based intrusion detection system for cloud computing environment", *International Journal of System Assurance Engineering and Management*, 9, 567–576, 2018. https://doi.org/10.1007/s13198-014-0277-7

[5] N. Harale, B. B. Meshram, "Network Based Intrusion Detection and Prevention Systems: Attack Classification, Methodologies and Tools", *International Journal of Engineering and Science*, 6(5), 1–12, 2016. http://www.researchinventy.com/papers/v6i5/A60501012

[6] O. Depren, M. Topallar, E. Anarim, M. K. Ciliz, "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks", *Expert Systems with Applications*, 29(4), 713–722, 2005. https://doi.org/10.1016/j.eswa.2005.05.002

[7] K. J. Mathai, K. Agnihotri, "Optimization techniques for feature selection in classification", *International Journal of Engineering Development and Research*, 5(3), 1167–1170, 2017. https://www.ijedr.org/papers/IJEDR1703165

[8] R. H. Abiyev, M. Tunay, "Optimization of High-Dimensional Functions through Hypercube Evaluation", *Computational Intelligence and Neuroscience*, 2015, 1–11, 2015. https://doi.org/10.1155/2015/967320

[9] K. Bajaj, A. Arora, "Improving the Intrusion Detection using Discriminative Machine Learning Approach and Improve the Time Complexity by Data Mining Feature Selection Methods", *International Journal of Computer Applications*, 76(1), 5–11, 2013. https://doi.org/10.5120/13209-0587

[10] E. de la Hoz, A. Ortiz, J. Ortega, E. de la Hoz, "Network Anomaly Classification by Support Vector Classifiers Ensemble and Non-linear Projection Techniques", *International Conference on Hybrid Artificial Intelligence Systems, Springer*, 103–111, 2013. https://doi.org/10.1007/978-3-642-40846-5_11

[11] Amrita, P. Ahmed, "A Hybrid-Based Feature Selection Approach for IDS", *Networks and Communications (NetCom2013)*, *Lecture Notes in Electrical Engineering*, *Springer*, 284, 195–211, 2014. https://doi.org/10.1007/978-3-319-03692-2_16

[12] P. Ghosh, C. Debnath, D. Metia, R. Dutta, "An Efficient Hybrid Multilevel Intrusion Detection System in Cloud Environment", *IOSR Journal of Computer Engineering (IOSR-JCE)*, 16(4), 16–26, 2014. http://www.iosrjournals.org/iosr-jce/papers/Vol16-issue4/Version-7/D016471626

[13] O. Osanaiye, H. Cai, K. K. R. Choo, A. Dehghantanha, Z. Xu, M. Dlodlo, "Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing", Springer, *EURASIP Journal on Wireless Communications and Networking* 2016, *Springer*, 2016. https://doi.org/10.1186/s13638-016-0623-3

[15] S.-H. Kang, K. J. Kim, "A feature selection approach to find optimal feature subsets for the network intrusion detection system", *Cluster Computing*, 19, 325–333, 2016. https://doi.org/10.1007/s10586-015-0527-8

[16] M. S. Akhtar, D. Gupta, A. Ekbal, P. Bhattacharyya, "Feature Selection and Ensemble Construction: A Two-step Method for Aspect Based Sentiment Analysis", *Knowledge-Based Systems*, 125, 116–135, 2017. https://doi.org/10.1016/j.knosys.2017.03.020

[18] S. Mirjalili, "Knowledge-Based Systems Moth-Flame Optimization Algorithm: A Novel Nature-inspired Heuristic Paradigm", *Knowledge-Based Systems*, 89, 228–249, 2015. https://doi.org/10.1016/j.knosys.2015.07.006

[19] F. Pedregosa et al., "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, 12, 2825–2830, 2011. https://jmlr.org/papers/volume12/pedregosa11a

[20] M. Tavallaee, E. Bagheri, W. Lu, A. A. Ghorbani, "A Detailed Analysis of the KDD CUP 99 Data Set", *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, 1–6. https://doi.org/10.1109/CISDA.2009.5356528

[22] F. Amiri, M. Rezaei Yousefi, C. Lucas, A. Shakery, N. Yazdani, "Mutual information-based feature selection for intrusion detection systems", *Journal of Network and Computer Applications*, 34(4), 1184–1199, 2011. https://doi.org/10.1016/j.jnca.2011.01.002

[23] H. Uğuz, "A two-stage feature selection method for text categorization by using information gain, principal component analysis and genetic algorithm", *Knowledge-Based Systems*, 24(7), 1024–1032, 2011. https://doi.org/10.1016/j.knosys.2011.04.014

[26] R. Singh, H. Kumar, R. K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine", *Expert Systems with Applications*, *Elsevier*, 42(22), 8609–8624, 2015. https://doi.org/10.1016/j.eswa.2015.07.015

[27] M. R. G. Raman, N. Somu, K. Kannan, R. Liscano, V. S. S. Sriram, "An efficient intrusion detection system based on hypergraph - Genetic algorithm for parameter optimization and feature selection in support vector machine", *Knowledge-Based Systems*, *Elsevier*, 134, 1–12, 2017. https://doi.org/10.1016/j.knosys.2017.07.005

[28] M. M. Abd-Eldayem, "A proposed HTTP service based IDS", *Egyptian Informatics Journal*, *Elsevier*, 15(1), 13–24, 2014. https://doi.org/10.1016/j.eij.2014.01.001

[29] R. Singh, H. Kumar, R. K. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine", Expert Systems with Applications, 42(22), 8609–8624, 2015. https://doi.org/10.1016/j.eswa.2015.07.015

[30] P. Gogoi, M. H. Bhuyan, D. K. Bhattacharyya, J. K. Kalita, "Packet and Flow Based Network Intrusion Dataset", *International Conference on Contemporary Computing*, *Communications in Computer and Information Science*, *Springer*, 2012, 306, 322–334. https://doi.org/10.1007/978-3-642-32129-0_34

[31] S. Saremi, S. Mirjalili, A. Lewis, "Grasshopper Optimisation Algorithm: Theory and application", Advances in Engineering Software, 105, 30-47, 2017. https://doi.org/10.1016/j.advengsoft.2017.01.004

[32] S. Bansal, K. Deep, "Enhanced Grasshopper Optimization Algorithm for Continuous Functions and Engineering Applications", Expert Systems with Applications, 149, 113322, 2020. https://doi.org/10.1016/j.eswa.2020.113322

[33] X.-S. Yang, "Firefly Algorithms for Multimodal Optimization", Stochastic Algorithms: Foundations and Applications, SAGA 2009. Lecture Notes in Computer Science, vol 5792. Springer, Berlin, Heidelberg, 2009. https://doi.org/10.1007/978-3-642-04944-6_14

[34] A. T. A. Rahman, A. R. Ramli, M. F. Miskon, "Firefly algorithm for solving manufacturing scheduling problems: A comprehensive survey", Applied Soft Computing, 100, 106922, 2021. https://doi.org/10.1016/j.asoc.2020.106922