The Influence of Salp Swarm Algorithm-Based Feature Selection on Network Anomaly Intrusion Detection

ALANOUĐ ALSALEH AND WOJDAN BINSAEEDAN
Department of Computer Science, College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 11432, Saudi Arabia
Corresponding author: Wojdan Binsaeedan (wmasaeedan@imamu.edu.sa)

ABSTRACT Network security plays a critical role in our lives because of the threats and attacks to which we are exposed, which are increasing daily; these attacks result in a need to develop various protection methods and techniques. Network intrusion detection systems (NIDSs) are a way to detect malicious network attacks. Many researchers have focused on developing NIDSs based on machine learning (ML) approaches to detect diverse attack variants. ML approaches can automatically discover the essential differences between normal and abnormal data by analysing the features of a large dataset. For this purpose, many features are typically extracted without discrimination, increasing the computational complexity. Then, by applying a feature selection method, a subset of features is selected from the whole feature set with the aim of improving the performance of ML-based detection methods. The salp swarm algorithm (SSA) is a nature-inspired optimization algorithm that has demonstrated efficiency in minimizing the processing challenges faced in performing optimization for feature selection problems. This research investigates the impact of the SSA on improving ML-based network anomaly detection using various ML classifiers, including the extreme gradient boosting (XGBoost) and Naïve Bayes (NB) algorithms. Experiments were conducted on standard datasets for comparison. Specifically, two datasets explicitly focused on network intrusion attacks were used: UNSW-NB15 and NSL-KDD. The experimental results show that the proposed method is more effective in improving the performance of anomaly NIDSs in terms of the F-measure, recall, detection rate, and false alarm rate on both datasets, outperforming state-of-the-art techniques recently proposed in the literature.

INDEX TERMS Network security, anomaly detection, swarm intelligence, salp swarm algorithm, feature selection, machine learning, XGBoost, Naïve Bayes.

I. INTRODUCTION
At present, computer network protection plays an essential role in protecting against both internal and external threats due to various gaps that attackers can exploit to break into and access networks in order to manipulate or steal sensitive information and cause considerable damage [1]. One of the ways to isolate and protect an environment from outside attacks is to use firewalls and traditional rule-based security protection techniques. However, to further increase the level of security protection, an additional system is needed to support traditional security techniques in protecting against various types of malicious attacks [2]. Moreover, advanced and sophisticated technologies are needed to examine and analyse enormous amounts of data from network infrastructure transactions. For this purpose, robust network intrusion detection systems (NIDSs) have been developed, which play a crucial role in ensuring network security and require the analysis of complex data [3], [4]. A NIDS protects a computer system or administrators when faced with various threats and attacks. Accordingly, when using a NIDS and its protection ability, one must ensure that it is up to date since there are many gaps in the network detection model to be determined, addressed, and filled [5]. NIDSs work by analysing and extracting the abnormal behaviours to be detected. After these suspicious behaviours are detected, alerts are sent to
notify interested parties of any abnormal behaviours that must be considered before sensitive information is accessed or data are manipulated or leaked. Two methods may be used by a NIDS. First, anomaly detection may be used to detect unknown attacks on the network or system, which are new attacks that have not been addressed in the past. Second, signature detection, also known as misuse detection, may be applied to detect abnormal behaviours based on prior knowledge, which must be defined in terms of rules or patterns for knowledge-based detection [3], [6]. Various techniques are applied in anomaly detection, such as machine learning (ML) techniques, to achieve a high detection rate and accurate results. ML-based detection plays an essential role in building effective models that rely on various algorithms and approaches to analyse big data consisting of network traffic flows in order to identify intrusions. One way to develop effective ML-based anomaly detection tools to reach a desired goal is to apply feature selection (FS) techniques [3]. FS is valuable for improving model performance to obtain accurate results. Therefore, one of the essential steps in developing an intelligent analytical tool is to establish data preprocessing procedures for finding relevant features that best reinforce the performance of predictive algorithms [4], [7]. FS plays a meaningful role in maximizing the performance of an ML model by disregarding redundant and irrelevant attributes that degrade the performance of the learning process and increase its complexity. FS can be expressed as a multiobjective optimization, which is NP-hard optimization problem because it concerns several (n) features, necessitating a large search space of size \(2^n\) to capture the various permutations of the features [7], [8]. Accordingly, numerous search methods can be utilized to detect the optimal subset of features. The first is to apply a greedy search method to traverse and assess all of the features; however, such a scheme is inherently time consuming. The second is to apply a random search method to randomly explore the domain, which has its own drawbacks and limitations. In particular, there is a chance of search stagnation, which may result in a very high time complexity, among other issues [8].

One way to address the drawbacks and gaps of the FS methods that have been previously proposed by researchers is to adopt the meta-heuristic paradigm. The meta-heuristic methods are global optimization approaches inspired by physical, biological, and animal activities. They can explore the search space both globally and locally when applied to FS problems. One class of meta-heuristic algorithms is based on the swarm intelligence (SI) approach [8]. This approach is inspired by the intelligence exhibited by swarms, herds, schools, or flocks of creatures in nature. SI algorithms have been widely used to address diverse optimization problems and reach suboptimal or optimal solutions. In particular, we investigate the influence of the salp swarm algorithm (SSA), one of the newest SI algorithms, to determine its effectiveness and its ability to address FS problems as an essential preprocessing step to enhance the performance of ML-based anomaly detection models [9]. Our contributions to the network security field are as follows. We propose an efficient network anomaly detection framework consisting of three phases. (i) In the first phase, to achieve a high classification accuracy and increase the detection rate while reducing the false alarm rate without using excessive computational resources. To accomplish these goals, we adopted the SSA-FS method to obtain the most relevant and accurate feature representation. According to our limited knowledge, no previous research has focused on the impact of SSA as an FS method on the network anomaly detection problem. (ii) In the second phase, we use two different classification algorithms, extreme gradient boosting (XGBoost) and Naïve Bayes (NB). Between these two algorithms, the NB algorithm shows weak performance, while the XGBoost algorithm achieves better results, thereby supporting our main goal: to determine the influence of SSA-FS and the extent of its ability to improve the performance of both robust and weak algorithms to support an effective anomaly NIDS. Based on FS, we investigate the effectiveness of the SSA and its impact on these two different algorithms. (iii) Corresponding to the last phase, i.e., model testing, we present an extensive experimental evaluation of the proposed method conducted using two datasets: UNSW-NB15 [10] and NSL-KDD [11]. In addition, the proposed method is compared with several state-of-the-art techniques.

The remainder of this paper is organized as follows: In Section II, we present related works to explore the existing solutions for FS in network anomaly detection. A brief overview of the SSA is provided in Section III. This overview is followed by Section IV, which introduces the design of the presented framework in detail. An evaluation and discussion, including comparisons of the proposed method with state-of-the-art methods, are given in Section V. Finally, conclusions and future work are presented in Section VI.

II. RELATED WORK

A number of network anomaly detection models, designed with many techniques and methods to improve their detection performance, have been widely researched in many studies. In this paper, we limit our review to approaches using the FS method known as dimensionality reduction as a general preprocessing method for high-dimensional data visualization, modelling, and analysis. In [6], a hybrid FS method comprising three different algorithms (particle swarm optimization (PSO), the ant colony algorithm (ACA), and a genetic algorithm (GA) based on a reduced error pruning tree (REPT) was proposed. Moreover, a two-level classifier (rotation forest and bagging) was used, with a majority vote determining the final decision. The authors of [2] examined the capabilities of XGBoost in NIDSs compared to other ML algorithms, using the information gain (IG) as the basis for FS. In [4], two phases of detection were applied; the first phase was FS using a wrapper method, a support vector machine (SVM), and a GA with multiparent crossover and multiparent mutation (MGA), and
the second phase was detection using an artificial neural network (ANN). To achieve more accurate results from the ANN, the authors integrated a hybrid gravitational search and PSO (HGS-PSO-ANN). The work presented in [12] focused on a NIDS utilizing a two-stage classification approach that addressed imbalanced data by filtering the dataset into majority and minority malicious classes. The IG method was used for FS, and random forests (RFs) were used as classifiers on the majority and minority classes individually to produce different models for detecting suspicious attacks.

The authors of [13] focused on applying the Ant Tree Miner (ATM) classification algorithm and the IG-based FS method to detect malicious attacks. ATM builds decision trees (DTs) using ant colony optimization, rather than the traditional C4.5 or Classification and Regression Trees (CART) method. In [14], the focus was on the ability of FS to support efficient network intrusion detection utilizing different ML classifiers. Both wrapper- and filter-based FS methods (IG, gain ratio, symmetrical uncertainty, Relief-F, One-R, and chi-square) were used, and the average was considered. In this method, the evaluators’ results were scaled to choose the most relevant features and examine the performance in the subsequent classification stage (J48 and NB) for detecting intrusion attacks. In [15], two different procedures involving misuse detection for known attacks were implemented by applying some generated rules and anomaly detection methods. For anomaly detection, both fuzzy C-means clustering and correlation-based FS were used to divide the training dataset and choose the most relevant features. Moreover, a hybrid classification technique based on the artificial bee colony and artificial fish swarm algorithms was used to detect network intrusion attacks. In [16], a geometric area analysis based on trapezoidal area estimation was proposed for network intrusion detection. In [17], two methods for network intrusion detection were proposed based on misuse and anomaly detection. In the anomaly detection method, three different classifiers (ANN, J48, and NB) were applied, and the IG method was used for FS. In [18], the researchers implemented five classification algorithms and examined their accuracy performance in combination with XGBoost-based FS. They found that the DT algorithm yielded the most accurate results in detecting network attacks. In [19], the researchers applied an association rule mining algorithm as an FS method. They used two classification algorithms (expectation maximization (EM) clustering and NB) to explore the model performance. In [20], a two-stage technique was applied for network intrusion detection. In the first stage, an FS method was applied using recursive feature elimination (RFE) and RFs. The second stage included five classification algorithms. Based on the results, the SVM yielded the best performance. The authors of [21] investigated the capabilities of an ANN as a supervised learning algorithm for NIDSs. For the FS procedure, the IG was used to choose the most relevant features to improve the model performance. In [22], the researchers applied a two-stage NIDS. The first stage was an FS method using the attribute ratio (AR), and the second was a classification stage using both K-means clustering and the XGBoost algorithm. In [23], the researchers applied four SI algorithms as FS methods to detect network intrusion attacks: the firefly optimization algorithm (FOA), PSO, a GA, and the Grey Wolf Optimizer (GWO). Moreover, two classifiers (SVM and J48) were used. In [24], to improve the model performance, the researchers used an FS method to choose the most relevant features to increase the detection rate of a NIDS by applying ant colony optimization. For the classification process, the k-nearest neighbours (KNN) method was used. In [25], fuzzy entropy-based ant colony optimization was proposed as an FS method. To test the model performance, four different individual classifiers (random tree, J48, JRip, and RF) were used. In [26], the researchers presented a hybrid algorithm to control and set the optimal parameters of classifiers and choose relevant features of a dataset using time-varying chaos PSO (TVCPSO). They used the concept of chaos to perform PSO faster while searching for the optimal features and avoiding trapping in local minima.

As the classifiers, they applied multicriteria linear programming (MCLP) and an SVM. In [27], the performance of NIDSs using two FS algorithms, namely, PSO and a GA, was investigated. Additionally, to examine the capabilities of the FS techniques, the authors used four different classification algorithms: rule induction, KNN, NB, and DTs. In [28], a primary NIDS was applied in the network layer to analyse traffic by adopting a cloud network node (CNN). At the same time, a secondary intrusion detection system (IDS) was placed in each tenant virtual machine (TVM), and the authors relied on RFE as an FS method and a chi sequence. During training on the dataset, an RF was used to determine whether the traffic contained malicious attacks. In [29], a significant NIDS algorithm based on anomaly detection with an SVM was used. Two algorithms were applied: binary-based PSO (BPSO) and standard-based PSO (SPSO). The objective of BPSO, as an FS method, was to select the most relevant network features. In contrast, the aim of the SPSO algorithm was to improve the performance of the SVM by adjusting its control parameters. In [30], the authors wished to enhance the detection rate of a NIDS by means of FS. Discretized differential evolution (DDE) was used as the FS method, and C4.5 DTs were used for classification. In [31], the authors proposed a model for anomaly intrusion detection based on the use of the mutation cuckoo fuzzy (MCF) method for selecting the best subset of features and a multiverse optimizer-ANN (MVO-ANN) for classification. The results showed the potential of the proposed model to enhance the intrusion detection efficiency and performance. Table 1 summarizes some of the new and current research on network intrusion detection using various FS methods for comparison with the anomaly-based NIDS using SSA-FS and both the XGBoost and NB classification algorithms that is proposed in this paper.
III. SALP SWARM ALGORITHM

The importance and effectiveness of SI have been proven in many applications and research. In the global optimization framework, SI addresses the problem of identifying and improving the suboptimal solutions for a problem from among a group of alternative solutions, among which is the optimal solution [9]. One of the newest SI algorithms is the SSA, which is a meta-heuristic algorithm introduced by Mirjalili (2017) [9] that is currently being tested and analyzed to determine its efficiency and ability to solve many problems. Motivated by the foraging behavior of salps in deep oceans, it has proven useful in finding global optima for optimization problems. The principle of the SSA is to rely on fitness values (optimal solutions). It depends on studying solutions through their fitness values and chains of search agents to reach suboptimal or optimal solutions.

The algorithm works by dividing the swarm into two parts: the leader and a group of follower salps, with the leader placed at the head of the chain and the follower salps moving behind it as a body to form a simple chain [8]. Figure 1 shows an illustration of the SSA. The position of each salp is defined in an \( n \)-dimensional search space, where \( n \) is the number of problem variables. The positions of all salps are stored in a two-dimensional matrix called \( x \). A food source called \( F \) in the search space is the swarm’s target. To update the position of the leader, the following equation is introduced:

\[
x^1_j = \begin{cases} 
F_j + c_1((u_b - l_b)c_2 + l_b) & c_3 \geq 0.5 \\
F_j - c_1((u_b - l_b)c_2 + l_b) & c_3 < 0.5
\end{cases}
\]

where \( x^1_j \) shows the position of the first salp (leader); \( F_j \) is the position of the food source in the \( j \)-th dimension; \( u_b \) and \( l_b \) are the upper and lower bounds, respectively, of the \( j \)-th dimension; and \( c_1, c_2, \) and \( c_3 \) are random numbers in the range of [0, 1]. Equation (1) shows that only the position of the leader is updated with regard to the food source.

The coefficient \( c_1 \) is the main parameter in the SSA because it balances exploration and exploitation; it is defined as follows:

\[
c_1 = 2e^{-\left(\frac{l}{L}\right)^2}
\]

where \( l \) is the current iteration and \( L \) is the maximum number of iterations. Moreover, the position of each follower salp is updated as follows:

\[
x^i_j = \frac{1}{2}(x^i_j + x^{i-1}_j)
\]

where \( i > 2 \) and \( x^i_j \) is the position of the \( i \)-th follower salp in the \( j \)-th dimension. The SSA has been applied in various fields of research to study the effectiveness of this algorithm and its capabilities. The results of these studies show the remarkable superiority and high performance of the SSA; it outperforms the most recent meta-heuristic algorithms. Several stochastic operators are combined in the SSA, allowing it to better avoid local solutions in multimodal search landscapes. This also enables the SSA to perform well on both small and large datasets [32], which is what has made it one of the most effective current algorithms. It is a new and modern algorithm that is suitable for application in many areas and for...
diverse problems, such as binary optimization, forecasting, the optimization of virtual machine placement, and chemical descriptor selection [8]; hence, it is a rich topic of study. To investigate its capabilities in the field of network anomaly detection, we examine its efficiency and performance in attaining an optimal result and increasing a model’s accuracy and rate of detection. The final pseudocode of the SSA is presented in Algorithm 1.

Algorithm 1: Salp Swarm Algorithm (SSA)

Input: Problem, Parameters
Output: $F_{\text{best}}$
1: InitializationParameters(Parameters)
2: Population ← InitializationSalp(Problem)
3: while exit criterion not met do
4: EvaluateSalp(Population)
5: $F_{\text{best}} ← \text{FittestSalp}(\text{Population})$
6: Update($c_{i}$); // using Equation (2)
7: for $i = 1$ to $N$ do
8: if ($i == 1$) then
9: UpdatePositionLeaderSalp(); // using Equation (1)
10: else
11: UpdatePositionFollowerSalp(); // using Equation (3)
12: end if
13: UpdateSalps(); // utilizing the legal limits
14: end for
15: end while
16: return $F_{\text{best}}$

IV. DESIGN OF A NETWORK ANOMALY DETECTION FRAMEWORK BASED ON THE SALP SWARM ALGORITHM

In this section, we present an overview of the proposed structural model. Next, we explain the FS method based on the SSA along with the preprocessing procedures and the architecture of the FS model.

A. FRAMEWORK FOR NETWORK ANOMALY DETECTION

The proposed framework for network anomaly detection is presented in Figure 2. The structure comprises three different phases. The first phase is dataset preprocessing, including dataset cleaning, transformation, and scaling as well as FS. This phase involves reducing the dimensionality of the feature space and increasing the algorithm speed. In addition, it can eliminate redundant, irrelevant, or noisy data, enhance the data quality, and increase the accuracy of the anomaly detection model. These goals are accomplished by using the SSA as an FS procedure. The second phase is classifier modelling, for which we consider two different classification algorithms: XGBoost and NB. Third, the objective of the model testing phase is to test the ability of the model to achieve accurate results in decision making for anomaly detection by applying various evaluation metrics to obtain the final binary classifier results and decide whether the data correspond to normal behaviour or a malicious attack. Furthermore, we present a set of tests conducted for an overall comparison with various existing techniques. Seven performance measures are applied to evaluate the test results: the accuracy, recall, precision, F-measure, false positive rate, false negative rate, and true negative rate.

B. PREPROCESSING

The workflow of the proposed SSA-based FS method is shown in Figure 3. Step 1, preprocessing, is a significant initial phase that improves the quality of the data to assist in the extraction of meaningful information. Therefore, the preprocessing step is essential for the construction of ML models. Data preprocessing encompasses various procedures, including cleaning, organization, transformation, scaling, and FS.

1) CLEANING AND TRANSFORMATION

Data cleaning, in general, eliminates unnecessary and redundant values, such as duplicate or irrelevant records or features. For instance, in this study, the ID numbers are removed from both datasets. We also remove the columns corresponding to the attack category and label from the UNSW-NB15 dataset [10] and those corresponding to the class and difficulty level from NSL-KDD [11] because they are used as target features. Additionally, we remove null cells from the datasets. Regarding the transformation procedure, the datasets considered in this study include categorical data, namely, variables that identify information as belonging to specific categories. We convert these categorical features into numerical features since most ML models and scalers can most efficiently work with numerical values. For instance, we transform the proto, service, and state features from the UNSW-NB15 dataset and the prototype, service, and flag features from NSL-KDD into numerical values using the LabelEncoder() class from the scikit-learn library.

2) FEATURE SCALING

Input variables frequently have various units; accordingly, they may also have various scales. To avoid any influence...
from the selection of particular measurement units, we use a feature scaling method to limit the ranges of the variables to allow them to be compared on a common basis. To this end, we apply a min-max scaler to transform the data to within the range of [0,1]. The min-max normalization method is expressed as

\[ X'_{ij} = \frac{X_{ij} - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \]  

(4)

where \( X_{\text{min}} \) and \( X_{\text{max}} \) are the minimum and maximum values, respectively, of the \( j \)-th feature. Notably, the XGBoost algorithm does not require any scaling because it requires only the selection of cut-off points for each feature at which to split nodes, and the splitting process is not sensitive to scaling [33].

3) FEATURE SELECTION
FS is a discrete problem, and all solutions are restricted to binary values. Therefore, we must convert the SSA into a binary version with [0,1] values. The meaning of these values is that ‘1’ indicates a selected feature, while ‘0’ indicates a feature that is not selected. Equation (5) is used to map continuous values to their binary versions:

\[ Z_{mn} = \begin{cases} 
1 & \text{if } X_{mn} > 0.5 \\
0 & \text{otherwise}
\end{cases} \] 

(5)

\( Z_{mn} \) can be regarded as a discrete form of the solution vector \( X \), where \( X_{mn} \) represents the continuous position of search agent \( m \) in \( n \) dimensions. As our next step, we will define a fitness function, and we will use two different classifiers and select between them based on which achieves the greater accuracy. The first is an XGBoost classifier, and the second is an NB classifier. The goal is to assess all search agents’ performance and achieve a high (maximal) classification accuracy. The fitness function given in Equation (6) is used in both classifiers for the SSA-FS problem.

\[ \text{FitnessFunction} = \frac{TP + TN}{TP + TN + FP + FN} \] 

(6)

This fitness function is calculated based on four essential retrieval parameters: \( TP \), the number of true positives, is the number of malicious connections correctly classified as malicious by the classifier. \( TN \), the number of true negatives, is the number of normal connections that are correctly classified as normal by the classifier. \( FP \), the number of false positives, is the number of normal connections that are misclassified as malicious by the classifier. \( FN \), the number of false negatives, is the number of malicious connections that are misclassified as normal by the classifier. In our architecture, we implement the SSA as a wrapper FS method using two different classifiers, i.e., XGBoost and NB, as the basis of the fitness function. The goal of using two different classification algorithms is to evaluate the quality of each selected subset of features and determine the effectiveness of the SSA using different fitness function classifiers to achieve improvement. The workflow of the proposed SSA-based FS method is presented in Figure 3. As input, the model takes the original dataset. In Step 1, we apply preprocessing, which represents our main contribution to the application of the SSA as an FS method, to choose valuable features. The results are obtained from the SSA based on either the XGBoost classifier or the NB classifier, and the final output is assessed. The SSA is applied after splitting the dataset into training and test sets. In Step 2, we randomly initialize the position of each individual salp. Each of these individuals represents a solution to the FS problem. In Step 3, we perform the fitness calculation by selecting the features corresponding to the solution to be evaluated and removing the non-selected
features from the training set. The fitness is calculated as shown in Equation (6). Then, we update the positions of the leader and followers to exploit and explore the search space using Equations (1) and (3); to achieve a balance between exploration and exploitation in the search, we use Equation (2). These steps are then performed iteratively until the maximum number of iterations is reached, at which time the list of the best chosen features is returned to prepare for the attack detection phase.

C. CLASSIFICATION ALGORITHMS

1) EXTREME GRADIENT BOOSTING

XGBoost is a variant of the gradient boosting machine (GBM) model that was introduced by Dr. Chen Tianqi of the University of Washington in 2016, and it has been widely used in academia and industry [33]. It has shown high efficiency and performance in prediction tasks. It has also outperformed many ML algorithms in various competitions; therefore, many authors have researched and tested XGBoost on many datasets to assess its efficiency and its ability to solve problems. Accordingly, it has become a popular research topic in the ML field [2], [34]. One of the advantages of the XGBoost model is that it relies on additive training to progressively optimize the objective function by considering the optimization result from the previous step [1], [5]. The $t$-th objective function of the model is shown in Equation (7):

$$\text{obj}^{(t)} = \sum_{i=1}^{n} l(y_i, y_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + \text{constant} \quad \text{(7)}$$

where $l$ denotes the loss term in the $t$-th round (the prediction of the $i$-th instance in the $t$-th round for $f_t$, which must be added to minimize the presented objective), constant is a constant term, and $\Omega$ is the regularization term of the model, which is expressed as

$$\Omega(f_t) = \gamma \cdot T_f + \lambda \frac{1}{2} \sum_{j=1}^{T} w^2_j \quad \text{(8)}$$

In this formula, both $\gamma$ and $\lambda$ are custom parameters of XGBoost—the larger these two values are, the easier it is to structure the tree and efficiently avoid overfitting—and $x^*_j$ represents the optimal weight solution for leaf $j$ [1].

2) NAIVE BAYES

The NB model is classified as a Bayesian probability model. Probabilities are considered in the NB model, with the final result leading to various related evidence variables [35]. An NB classifier aims to provide output that can distinguish between two different classes (i.e., normal (0) and malicious (1)). This goal is accomplished by applying the maximum a posteriori (MAP) function, which is expressed as

$$P(C|\mathbf{x}) \arg \max_{\gamma \in \{1,2,3,4, \ldots, N\}} P(C_{\mathbf{x}}) \prod_{j=1}^{N} P(I_j|C_{\mathbf{x}}) \quad \text{(9)}$$

Here, $C$ denotes the labelled classes, $I$ corresponds to all observations, $w$ is the class index, $P(C|I)$ is the class probability for a specific observation, and $P(C_{\mathbf{x}}) \prod_{j=1}^{N} P(I_j|C_{\mathbf{x}})$ is the product of all conditional probabilities, for which the aim is to obtain the maximum result [22].

V. EXPERIMENTAL SETTINGS

In this section, we present the two datasets and the experimental settings along with the experimental results obtained when applying the SSA as a FS method. The specifications of the computing environment used for the experiments discussed in this paper are as follows: the Microsoft Windows 10 operating system (64-bit), an Intel Core i7 8th Gen processor, Python version 3.6.7 in Google Colab Pro, and 27.4 GB of RAM and GPU memory.

A. ANOMALY DETECTION DATASETS

We examine two publicly available anomaly detection datasets that have been used extensively in previous studies.

The NSL-KDD was presented in 2009 [11]. It is considered an improved version of KDDCUP99, produced by removing all redundant records from the training set and all duplicate records from the test set. It contains a normal class and four attack classes: R2L, U2R, probe, and DoS. We consider the training (KDDTrain) and test (KDDTest) subsets, which consist of 125,973 and 22,544 normal and malicious data instances, respectively. The class distribution is shown in Table 2, along with a breakdown of the traffic record distribution. Typically, almost the half of the records correspond to normal traffic, and the U2R and R2L attacks are shallow. This dataset includes 41 features, as listed in Table 4 [36].

The UNSW-NB15 dataset was developed and released in 2015 by the Australian Center for Cybersecurity (ACCS) [10]. Many datasets have weaknesses in the detection process, which has become a challenge for cybersecurity research groups. Additionally, many researchers in this field have created datasets that are more efficient and accurate in discovering and detecting particular types of attacks, especially at the network level. In contrast, UNSW-NB15 is a more comprehensive resource for evaluating existing network intrusion detection methods more reliably. It contains a normal class and nine types of attacks: worms, shellcode, reconnaissance, generic, exploits, DoS, backdoors, analysis, and fuzzers. We consider the training (UNSW-NB15train) and test (UNSW-NB15test) subsets, which consist of 175,341 and 82,332 instances, respectively. The class distribution is shown in Table 3, along with a breakdown of the traffic record distribution. Typically, almost the half of the records correspond to normal traffic, and the rest correspond to malicious attacks with different prevalences. The traffic record features represent information about the traffic that can be provided as input to a NIDS. This dataset includes 42 features, as listed in Table 5 [10].
TABLE 2. Class distribution in the NSL-KDD dataset.

| Class Name | Training | Test |
|------------|----------|------|
| Normal     | 67345    | 9771 |
| DoS        | 45927    | 7458 |
| Probe      | 11656    | 2421 |
| U2R        | 52       | 200  |
| R2L        | 995      | 2754 |
| **Total Records** | **125973** | **22544** |

TABLE 3. Class distribution in the UNSW-NB15 dataset.

| Class Name | Training | Test |
|------------|----------|------|
| Normal     | 56000    | 37000|
| Analysis   | 2000     | 677  |
| Backdoors  | 1746     | 583  |
| DoS        | 12264    | 4089 |
| Exploits   | 33393    | 11132|
| Fuzzers    | 18184    | 6062 |
| Generic    | 40000    | 18871|
| Reconnaissance | 10491  | 3496 |
| Shellcode  | 1133     | 378  |
| Worms      | 130      | 44   |
| **Total Records** | **175341** | **82332** |

B. EXPERIMENTAL RESULTS AND DISCUSSION

Our experiments were designed to examine the performance of the proposed network anomaly detection method using the XGBoost and NB algorithms as ML techniques. We employed the whole feature space of both datasets, UNSW-NB15 and NSL-KDD, for the binary classification problem. To estimate the performance of the proposed model in network anomaly detection and examine the effectiveness and impact of the SSA as an FS method, the algorithm was run ten times to obtain the average result. Recall that our aims in developing the proposed model were to reduce detection errors and increase the model accuracy. The evaluation metrics used for experimental comparison with other detection methods were the false negative rate (FNR), true negative rate (TNR), false positive rate (FPR), precision, F-measure, recall and accuracy; these metrics are calculated based on four essential retrieval parameters, namely, the numbers of true negatives (TN), false positives (FP), true positives (TP), and false negatives (FN).

1) True positive rate, or recall: The percentage of malicious actions correctly classified as attacks.

\[
\text{Recall} = \frac{TP}{TP + FN} \quad (10)
\]

2) FNR: The rate of incorrect classification such that a malicious attack is classified as normal.

\[
\text{FNR} = \frac{FN}{TP + FN} \quad (11)
\]

3) TNR: The percentage of normal traffic instances that are correctly classified as belonging to the normal class.

\[
\text{TNR} = \frac{TN}{TN + FP} \quad (12)
\]

4) FPR: The rate of incorrect classification such that normal traffic is classified as a malicious attack.

\[
\text{FPR} = \frac{FP}{FP + TN} \quad (13)
\]

5) F-measure: The statistical function of this metric is to act as a measure of the overall accuracy of a model on a dataset. The F-measure is a way of combining the precision and recall of the model.

\[
\text{F-measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (14)
\]

6) Precision: This metric reflects the quality of the accuracy since it focuses on how close two or more measurements are to each other, regardless of whether the measurements themselves are accurate.

\[
\text{Precision} = \frac{TP}{TP + FP} \quad (15)
\]

7) Accuracy: The degree to which the obtained measurement result corresponds to the correct value; it indicates how close a measured value is to a standard or known value.

\[
\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (16)
\]

We applied the features chosen through FS to enhance the model performance. We assessed the results and the effectiveness before and after applying the SSA as an FS method. Table 6 shows the accuracy results of the XGBoost and NB algorithms on the two datasets before the application of the SSA. Moreover, when we applied SSA-based FS, we noted that this algorithm has many essential parameters that have a great impact on the model efficiency. After an initial test study, suitable values were empirically selected based on both datasets for the following parameters: the number of search agents, the number of iterations, and the lower bound (lb) and upper bound (ub). Specifically, we found that twenty search agents, a maximum of 50 iterations, an lb of $-5$ and a ub of 5 led to satisfactory results, whereas other sets of values required much longer run times and/or yielded worse results.

The many parameters of the SSA that were tested are reported in Table 7, and the effective parameter values selected based on these tests to obtain the final experimental results for both algorithms (SSA-XGBoost and SSA-NB) are listed in Table 8. In the experiments, out of the 41 features in the NSL-KDD dataset, 20 features were selected by SSA-XGBoost, and 13 features were selected by SSA-NB. Meanwhile, for the UNSW-NB15 dataset, 23 of the
42 features were selected by SSA-XGBoost, whereas with SSA-NB, 14 features were selected via the proposed SSA-FS technique, as presented in Table 10. These features were selected in accordance with their importance as evaluated by the objective function based on each classifier, taking advantage of the ability of the SSA to explore and exploit the search space to find the combinations of features that could lead to the most accurate results, thus improving the model performance. Table 9 shows the improvements achieved via SSA-FS when the SSA-NB and SSA-XGBoost algorithms were utilized on the two different datasets. In addition, these experiments show that our proposed anomaly-based NIDS approach improves the classification performance with an acceptable time cost. On the NSL-KDD dataset, SSA-XGBoost consumed 1.77 seconds for model training and 0.03 seconds for testing, whereas SSA-NB required a training time of 0.02 seconds and a testing time of 0.008 seconds.

| Feature number | Feature | Feature number | Feature |
|----------------|---------|----------------|---------|
| f1             | duration| f22            | is_guest_login |
| f2             | proto_type| f23            | count |
| f3             | service| f24            | srv_count |
| f4             | flag| f25            | srverror_rate |
| f5             | src_bytes| f26            | srverror_rate |
| f6             | dst_bytes| f27            | rerror_rate |
| f7             | land| f28            | srv_rerror_rate |
| f8             | wrong_fragment| f29            | same_srv_rate |
| f9             | urgent| f30            | diff_srv_rate |
| f10            | hot| f31            | srv_diff_host_rate |
| f11            | num_failed_logins| f32            | dst_host_count |
| f12            | logged_in| f33            | dst_host_srv_count |
| f13            | num_compromised| f34            | dst_host_same_srv_rate |
| f14            | root_shell| f35            | dst_host_diff_srv_rate |
| f15            | su_attempted| f36            | dst_host_same_src_port_rate |
| f16            | num_root| f37            | dst_host_srv_diff_host_rate |
| f17            | num_file_creations| f38            | dst_host_serrorrate |
| f18            | num_shells| f39            | dst_host_srv_serror_rate |
| f19            | num_access_files| f40            | dst_host_rerror_rate |
| f20            | num_outbound_cmds| f41            | dst_host_srv_rerror_rate |
| f21            | is_host_login| |

| Parameter | Definition | Values |
|-----------|------------|--------|
| upper bound | ub, upper bound of variable n | 10 | 1 | 0 | 1 | 0 | 1 | 1 | 2 | 1 | 0 | 1 | 5 | 1 | 5 | 6 | 5 | 5 | 6 | 4 |
| lower bound | lb, lower bound of variable n | 0 | 1 | -100 | 0 | 1 | 0 | 1 | -100 | 0 | 1 | 0 | 1 | -5 | 5 | 6 | 5 | 5 | 6 | 4 |
| SearchAgents_no | Number of salps | 30-20 | 30 | 40-50 | 60-70 | 80-90 | 100 |
| dimension | Number of variables | Number of features |
| max_iteration | Maximum number of generations | 30-40 | 50-60 | 70-80 | 90-100 | 150-200 |
| fobj | YourobjectFunction | Fitness function in terms of the selected classifier |

| Parameter | Value |
|-----------|-------|
| upper bound | 5 |
| lower bound | -5 |
| SearchAgents_no | 20 |
| dimension | Number of features |
| max_iteration | 50 |
| fobj | Fitness function in terms of the selected classifier |

| Method | Accuracy | F-measure | Recall | Precision | FPR | PSNR | TNR |
|--------|----------|-----------|--------|-----------|-----|------|-----|
| SSA-XGBoost | 99.0% | 99.0% | 99.1% | 98.0% | 02.1% | 99.6% | 98.0% |
| SSA-NB | 94.0% | 94.0% | 97.2% | 90.5% | 08.4% | 04.2% | 91.5% |

| Method | Accuracy | F-measure | Recall | Precision | FPR | PSNR | TNR |
|--------|----------|-----------|--------|-----------|-----|------|-----|
| SSA-XGBoost | 93.0% | 94.4% | 96.0% | 93.0% | 06.9% | 07.7% | 93.0% |
| SSA-NB | 85.0% | 87.5% | 99.0% | 82.0% | 18.0% | 03.8% | 82.0% |
Similarly, for the UNSW-NB15 dataset, SSA-XGBoost required 4.28 seconds for training and 0.057 seconds for testing, and the corresponding time costs for SSA-NB were 0.050 seconds and 0.01 seconds, respectively. Overall, from our examination of the proposed SSA-based FS method with different algorithms and datasets, we conclude that enhanced performance can indeed be achieved. While some features are highly effective for model training, the inclusion of noisy, redundant, and irrelevant feature dimensions can render learning algorithms very slow and even degrade the performance on learning tasks; therefore, the ability to select the most relevant features from the datasets is beneficial for increasing the detection rate and decreasing the false alarm rate of a NIDS. In particular, one of the critical factors that can affect model performance is data imbalance. In general, the characteristic representations of malicious attack behaviours in datasets used for intrusion detection are usually strongly imbalanced since certain attacks tend to occur more often than others. Therefore, these datasets must be balanced by means of appropriate preprocessing methods before ML algorithms are applied to enable the development of an effective NIDS [12], [37].

C. COMPARISON AND DISCUSSION

To further evaluate the results of the proposed method in comparison with similar well-established methods, we consider recent research in which FS methods have been applied in experiments and use the same datasets. In this way, we can achieve sufficiently fair comparisons to illustrate the impact of our proposed method and compare it with previous studies to prove its effectiveness in detecting network attacks. Table 11 shows that for the UNSW-N15 dataset, the proposed method (SSA-XGBoost and SSA-NB) outperforms the other techniques considered for comparison and achieves better results, especially in terms of the FNR and FPR. Note that the results are worse when the FNR is higher, as this means that the model cannot detect or recognize malicious attacks; therefore, the higher values seen for the other models compare unfavourably with the FNR of 0.38% achieved by SSA-NB. Similarly, the FPR represents the rate of false alarms, where normal traffic is detected as malicious, confusing the system administrators. According to the results obtained, SSA-XGBoost outperforms the other methods in terms of this metric, achieving the lowest false alarm rate of 6.9%. Regarding the F-measure and recall, as shown in Table 11, our method again noticeably outperforms the methods presented in previous studies; SSA-XGBoost reaches an F-measure of 94.4%, and the SSA-NB algorithm achieves the highest recall of 99.0%. In contrast, the previous work on SVM-PSO [23] exhibits the highest results in terms of the TNR 97.4% and precision 96.3%; however, this method yields lower results in terms of the accuracy, F-measure, and

| Method           | Accuracy | F-measure | Recall | Precision | FPR  | FNR  | TNR  | #SF |
|------------------|----------|-----------|--------|-----------|------|------|------|-----|
| SSA-XGBoost      | 93.0%    | 94.4%     | 96.0%  | 93.0%     | 06.9%| 07.7%| 93.0%| 23  |
| SSA-NB           | 85.0%    | 87.5%     | 99.0%  | 82.0%     | 18.0%| 03.8%| 82.0%| 14  |
| Two-level ensemble [6] | 91.2%   | N/A       | 91.3%  | 91.6%     | 89.0%| N/A  | N/A  | 19  |
| J48-GA [23]      | 86.8%    | 86.8%     | 96.7%  | 78.8%     | 21.1%| 33.0%| 78.8%| 23  |
| J48-GWO [23]     | 85.6%    | 85.4%     | 93.7%  | 78.5%     | 20.9%| 62.0%| 79.0%| 20  |
| J48-FPA [23]     | 86.0%    | 86.1%     | 96.5%  | 77.7%     | 22.5%| 34.1%| 77.4%| 21  |
| SVM-PSO [23]     | 89.1%    | 87.1%     | 79.5%  | 96.3%     | 25.9%| 20.4%| 97.4%| 25  |
| SVM-GA [23]      | 86.3%    | 86.5%     | 96.9%  | 78.0%     | 22.2%| 30.3%| 77.7%| 23  |
| RF-IG [12]       | 85.7%    | N/A       | N/A    | N/A       | 15.6%| N/A  | N/A  | N/A |
| LR-XGBoost [18]  | 77.6%    | 82.2%     | 93.7%  | 73.1%     | N/A  | N/A  | N/A  | 19  |
| ANN-XGBoost [18] | 84.3%    | 87.4%     | 98.5%  | 78.5%     | N/A  | N/A  | N/A  | 19  |
| kNN-XGBoost [18] | 84.4%    | 87.0%     | 95.0%  | 80.3%     | N/A  | N/A  | N/A  | 19  |
| NB-RFE&RF [20]   | 60.7%    | 82.4%     | N/A    | N/A       | N/A  | N/A  | N/A  | 5   |
| XGBoost-IG [2]   | 75.8%    | N/A       | N/A    | N/A       | N/A  | N/A  | N/A  | 23  |

N/A= Not available
#SF= Number of selected features
recall. Moreover, this model shows a high FPR 25.9% and the worst result in terms of the FNR 20.4%, meaning that it can only reliably identify normal traffic and cannot detect malicious attacks effectively and reasonably. Additionally, our model shows some advantages in reducing the number of selected features while achieving increased detection performance. Table 12 presents the results obtained on the NSL-KDD dataset. Here again, the authors of other existing approaches, including those of [4], [6], [22], [26], [30] and [29], have used the same datasets but applied different FS techniques. Our proposed method with SSA-XGBoost yields an accuracy of 99.0%, which is substantially higher than the accuracy values reported in previous studies. Similarly, when we consider the other evaluation metrics, our model generally outperforms the others. However, the SVM-SPSO-BPSO technique [29] performs as well as or better than our approach in terms of various metrics, although the differences are not large considering the efforts expended by these previous researchers to obtain their results. They used SPSO to control the parameters of the SVM to improve the classifier model while also performing a second round of processing utilizing BPSO as an FS method. Even with all of these improvements adopted in [29], our model was able to achieve competitive performance with only the application of SSA-based FS, and it outperformed the methods presented in other previous studies.

Moreover, we must consider the ability of our proposed method to outperform the other methods in terms of the FNR and recall in particular. In terms of these two metrics, it reaches values of 00.6% and 99.1%, respectively, indicating the highest detection rate and an enhanced detection ability. Tables 11 and 12 show that the proposed approach is highly competitive and effective for solving network anomaly detection problems. Through this performance analysis, we have demonstrated that the proposed model exhibits superior performance and a greater ability to achieve increased accuracy while reducing the false alarm rate than state-of-the-art techniques.

VI. CONCLUSION AND FUTURE WORK

Network anomaly detection has been recognized as a robust tool for protecting networks from various types of malicious attacks. This paper has proposed a useful network detection framework by examining two different algorithms, XGBoost and NB, as a basis for applying the SSA as an FS technique. The detection rate can be affected by the features considered during data preprocessing and model training, which are recognized as two fundamental determinants of the final detection capability and efficiency; therefore, in the proposed method, the SSA is applied to select the most relevant and optimal features to improve the model performance. Accordingly, two classification algorithms, SSA-XGBoost and SSA-NB, were used to build a robust network anomaly detection model. We used two primary datasets, namely, NSL-KDD and UNSW-NB15, to examine the model performance. Our proposed network detection model achieved high and robust performance, with a high accuracy, a high detection rate, and a low false alarm rate, demonstrating the effectiveness of applying the SSA for FS. Moreover, our method showed either absolute superiority or highly competitive performance compared to other recently proposed techniques for addressing network anomaly detection problems. In future work, we anticipate further improving the SSA to increase the detection rate by using different methodologies for different datasets while overcoming the challenges presented by data imbalance to enhance the model performance. Furthermore, we will explore further comparisons with other techniques, such as feature extraction methods.

REFERENCES

[1] H. Jiang, Z. He, G. Ye, and H. Zhang, “Network intrusion detection based on PSO-xgboost model,” IEEE Access, vol. 8, pp. 58392–58401, 2020.

[2] A. Husain, A. Salem, C. Jim, and G. Dimitoglou, “Development of an efficient network intrusion detection model using extreme gradient boosting (XGBoost) on the UNSW-NB15 dataset,” in Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT), Dec. 2019, pp. 1–7.
B. Ingre and A. Yadav, “Building an efficient intrusion detection system based on feature selection and ensemble classifier,” *Comput. Netw.*, vol. 174, Jun. 2020, Art. no. 107247.

S. Hosseini and B. M. H. Zade, “New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN,” *Comput. Netw.*, vol. 173, May 2020, Art. no. 107168.

J. Gao, S. Chai, B. Zhang, and Y. Xia, “Research on network intrusion detection based on incremental extreme learning machine and adaptive principal component analysis,” *Energies*, vol. 12, no. 7, p. 1223, Mar. 2019.

B. A. Tama, M. Comuzzi, and K. Rhee, “TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system,” *IEEE Access*, vol. 7, May 2019, pp. 94497–94507, 2019.

R. A. Ibrahim, A. A. Ewees, D. Oliva, M. Abd Elaziz, and S. Lu, “Improved salp swarm algorithm based on particle swarm optimization for feature selection,” *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 8, pp. 3153–3169, Aug. 2019.

I. Aljahr, M. Habib, H. Faris, N. Al-Madi, A. A. Heidari, M. Mafarja, M. A. Elaziz, and S. Mirjalili, “A dynamic locality multi-objective salp swarm algorithm for feature selection,” *Comput. Ind. Eng.*, vol. 147, Sep. 2020, Art. no. 106628.

S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, “Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems,” *Adv. Eng. Softw.*, vol. 114, pp. 163–191, Dec. 2017.

N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in *Proc. Mil. Commun. Inf. Syst. Conf. (MICIS)*, Nov. 2015, pp. 1–6.

M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.

2. W. Zong, Y.-W. Chow, and W. Sasilo, “A two-stage classifier approach for network intrusion detection,” in *Information Security Practice and Experience*, C. Su and H. Kikuchi, Eds. Cham, Switzerland: Springer, 2018, pp. 329–340.

F. H. Botes, L. Leenen, and R. De La Harpe, “Ant colony induced decision trees for intrusion detection,” in *Proc. 16th Eur. Conf. Cyber Warfare Secure. (ACPS)*, 2017, pp. 53–62.

H. M. Anwer, M. Farouk, and A. Abdel-Hamid, “A framework for efficient network anomaly intrusion detection with features selection,” in *Proc. 9th Int. Conf. Inf. Commun. Syst. (ICICS)*, Apr. 2018, pp. 157–162.

V. Hajsalem and S. Babaie, “A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection,” *Comput. Netw.*, vol. 136, pp. 37–50, May 2018.

N. Moustafa, J. Slay, and G. Creich, “Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks,” *IEEE Trans. Big Data*, vol. 5, no. 4, pp. 481–494, Dec. 2019.

A. S. Varal and S. K. Wagh, “Misuse and anomaly intrusion detection system using ensemble learning model,” in *Proc. Int. Conf. Recent Innov. Electr. Electron. Commun. Eng. (ICRIEEC)*, Jul. 2018, pp. 1722–1727.

S. M. Kasongo and Y. Sun, “Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset,” *J. Big Data*, vol. 7, no. 1, pp. 1–20, Dec. 2020.

N. Moustafa and J. Slay, “The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems,” in *Proc. 4th Int. Workshop Building Anal. Datasets Gathering Expert. Returns Secur. (BADGERS)*, Nov. 2015, pp. 25–31.

S. Meftah, T. Rachidi, and N. Assen, “Network based intrusion detection using the UNSW-NB15 dataset,” *Int. J. Comput. Digit. Syst.*, vol. 8, no. 5, pp. 478–487, 2019.

B. Ingre and A. Yadav, “Performance analysis of NSL-KDD dataset using ANN,” in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst.*, Jan. 2015, pp. 92–96.

A. Pattawaro and C. Polprasert, “Anomaly-based network intrusion detection system through feature selection and hybrid machine learning technique,” in *Proc. 16th Int. Conf. ICT Knowl. Eng. (ICT&KE)*, Nov. 2018, pp. 1–6.

O. Almoman, “A feature selection model for network intrusion detection system based on PSO, GWO, FFA and GA algorithms,” *Symmetry*, vol. 12, no. 6, p. 1046, Jun. 2020.

M. H. Aghdam and P. Kabiri, “Feature selection for intrusion detection system using ant colony optimization,” *Int. J. Netw. Secur.*, vol. 18, no. 3, pp. 420–432, May 2016.

P. R. K. Varma, S. S. Kumari, and V. V. Kumar, “Feature selection using relative fuzzy entropy and ant colony optimization applied to real-time intrusion detection system,” *Procedia Comput. Sci.*, vol. 85, pp. 503–510, Dec. 2016.

S. M. H. Babamakan, H. Wang, T. Yingjie, and Y. Shi, “An effective intrusion detection framework based on MCLP/SVM optimized by time-varying chaos particle swarm optimization,” *Neurocomputing*, vol. 199, pp. 90–102, Jul. 2016.

I. Syarif, “Feature selection of network intrusion data using genetic algorithm and particle swarm optimization,” *EMITTER Int. J. Eng. Technol.*, vol. 4, no. 2, pp. 277–290, Dec. 2016.

K. K. Ghanshala, P. Mishra, R. C. Joshi, and S. Sharma, “BNID: A behavior-based network intrusion detection at network-layer in cloud environment,” in *Proc. 1st Int. Conf. Secure Cyber Comput. Commun. (ICSCCC)*, Dec. 2018, pp. 100–105.

M. M. Sakr, M. A. Tawfeeq, and A. B. El-Sisi, “Network intrusion detection system based PSO-SVM for cloud computing,” *Int. J. Comput. Inf. Netw. Secur.*, vol. 10, no. 3, p. 22, 2019.

E. Popoola and A. O. Adewumi, “Efficient feature selection technique for network intrusion detection system using discrete differential evolution and decision,” *IJ Netw. Secur.*, vol. 19, no. 5, pp. 660–669, 2017.

S. Sarvari, N. F. M. Sani, Z. M. Hanapi, and M. T. Abdullah, “An efficient anomaly intrusion detection method with feature selection and evolutionary neural network,” *IEEE Access*, vol. 8, pp. 70651–70663, 2020.

H. Faris, M. M. Mafarja, A. A. Heidari, I. Aljahr, A. M. Al-Zoubi, S. Mirjalili, and H. Fujita, “An efficient binary salp swarm algorithm with crossover scheme for feature selection problems,” *Knowl.-Based Syst.*, vol. 154, pp. 43–67, Aug. 2018.

T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.

S. S. Dhaliwal, N. Abdullal-Al, and R. Abbas, “Effective intrusion detection system using XGBoost,” *Information*, vol. 9, no. 7, p. 149, Jul. 2018.

M. Panda and R. R. Patra, “Network intrusion detection using Naive Bayes,” *Int. J. Comput. Sci. Netw. Secur.*, vol. 7, no. 12, pp. 258–263, 2007.

A. Karani, “An anomaly-based intrusion detection system in presence of benign outliers with visualization capabilities,” *Expert Syst. Appl.*, vol. 108, pp. 36–60, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417418302768

M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” *Comput. Secur.*, vol. 86, pp. 147–167, Sep. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016740481930118X

**ALANOUD ALSALEH** received the B.Sc. degree in computer sciences from the Department of Computer Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Saudi Arabia, where she is currently pursuing the master’s degree. Her research interests include network security, intrusion detection, web security, and artificial intelligence.

**WOJDAN BINSAESDAN** received the B.Sc. degree from Imam Mohammad Ibn Saud Islamic University (IMSIU), Saudi Arabia, the M.Sc. degree from King Saud University, and the Ph.D. degree from the Department of Computer Science, King Saud University. She is currently an Assistant Professor with the Department of Computer Science, College of Computer and Information Sciences, IMSIU. Her research interests include artificial intelligence, cybersecurity, and digital forensics.