Computer Science • 25(2) 2024 https://doi.org/10.7494/csci.2024.25.2.5891

Chiheb Eddine Ben Ncir
Mohamed Aymen Ben Haj Kacem
Mohammed Alattas

EXPLAINABLE SPARK-BASED PSO CLUSTERING FOR INTRUSION DETECTION

Abstract *Given the exponential growth of available data in large networks, the existence of rapid, transparent, and explainable intrusion detection systems has become of highly necessity to effectively discover attacks in such huge networks. To deal with this challenge, we propose a novel explainable intrusion detection system based on Spark, Particle Swarm Optimization (PSO) clustering, and eXplainable Artificial Intelligence (XAI) techniques. Spark is used as a parallel processing model for the effective processing of large-scale data, PSO is integrated to improve the quality of the intrusion detection system by avoiding sensitive initialization and premature convergence of the clustering algorithm and finally, XAI techniques are used to enhance interpretability and explainability of intrusion recommendations by providing both micro and macro explanations of detected intrusions. Experiments are conducted on large collections of real datasets to show the effectiveness of the proposed intrusion detection system in terms of explainability, scalability, and accuracy. The proposed system has shown high transparency in assisting security experts and decision-makers to understand and interpret attack behavior.*

Keywords Intrusion Detection System (IDS), Artificial Intelligence (AI), Explainable AI (XAI), Particle swarm optimization (PSO), Spark framework

Citation Computer Science 25(2) 2024: 1–27

1. Introduction

Emerging information technologies, such as cloud computing and control systems, have undergone rapid evolution in recent years. These systems often handle large volumes of information across diverse infrastructures and complex networks. Consequently, numerous malicious entities attempt to compromise these systems by exploiting communication networks. To secure computer systems and networks from unauthorized access and data breaches, Intrusion Detection Systems (IDS) are implemented. IDS can be broadly classified into three categories based on the detection model employed: signature-based IDS (S-IDS), anomaly-based IDS (A-IDS), and hybrid-based IDS (H-IDS). S-IDS relies on predefined rules and signatures to detect attacks, while A-IDS utilizes machine learning techniques to identify legitimate behaviors within a system. H-IDS combines both A-IDS and S-IDS approaches [2, 20].

In this research, our focus is on anomaly-based IDS that employ machine learning techniques [22, 36, 37]. One widely used technique is clustering, which groups similar data points, also known as clusters [26]. Various clustering methods have been proposed for intrusion detection systems, including partitional, hierarchical, and other types of clustering [13, 23, 38]. Among these methods, K-means-based clustering is the most commonly applied due to its linear time complexity [25]. However, it suffers from drawbacks such as sensitivity to initial cluster centers and convergence to local optima [6]. To address the sensitivity of initial cluster centers, several optimization techniques have been incorporated into the clustering approach [31]. One such technique is Particle Swarm Optimization (PSO) [17], which has been successfully used to address this drawback in clustering tasks [8, 27].

However, clustering-based intrusion detection methods face challenges when dealing with scalability issues in the analysis of large volumes of network traffic. To address this, several parallel clustering methods have been developed in the literature to handle large-scale data [9]. Many of these methods leverage the MapReduce framework [11] for data processing. However, MapReduce is not well-suited for iterative algorithms as it requires frequent disk reads and writes, leading to performance limitations. To overcome the limitations of MapReduce, the Spark framework [44] has been proposed for efficient processing of iterative algorithms. Spark is an in-memory parallel framework that has the ability to process big data using a cluster of machines. In comparison to the MapReduce framework, Spark demonstrates greater efficiency and provides a significant speed improvement of approximately 10 to 100 times faster for data processing tasks [16]. This makes it a more suitable choice for handling large-scale data in intrusion detection systems.

In addition to addressing scalability, it is crucial to consider the need for interpretability in clustering-based intrusion detection methods. Security experts require an understanding of the rationale behind identifying certain operations as intrusions. Unfortunately, existing clustering methods for intrusion detection often lack interpretability making it challenging to explain the obtained clusters. These methods typically function as "black-box" systems, providing a final organization of network

attacks without offering explanations for how the organization and attack detection were derived. However, explanations play an essential role in enabling security experts to transparently and effectively identify the correct attacks and implement appropriate strategies to secure the system. By incorporating interpretability into clustering-based intrusion detection methods, security experts can gain insights into the reasons behind classifying certain operations as security attacks and allow them to define effective security measures to protect the system.

To deal with all the discussed issues, we propose an Explainable Spark-based PSO Clustering for intrusion detection system (E-SPSO). The proposed system is based on the explainable artificial intelligence framework, named SHape Additive exPplanations (SHAP) [24], that allows explaining reasons behind the detected intrusions. Such explanations make the resulting prediction highly transparent for security experts. The proposed method performs parallel processing of intrusion detection tasks based on the Spark framework and integrates the PSO technique to improve the quality of obtained attack clusters. We will show in the next sections how the proposed intrusion detection system allows an enhance scalability, accuracy, and explainability of cyber-defense systems in large networks. The remainder of this paper is organized as follows: Section 2 discusses a survey of related works while Section 3 describes the backgrounds of this work including Particle Swarm Optimization (PSO), Spark framework, and Shapley Additive Explanations (SHAP). Then, Section 4 describes the proposed ESPSO-IDS system. Section 5 presents performed experiments and obtained empirical results. Finally, Section 6 summarises this work and discusses the future directions.

2. Related works

This section presents the related works of intrusion detection, explainable clustering, and explainable intrusion detection.

2.1. Intrusion detection

Several intrusion detection systems based on machine learning techniques were proposed in the literature [22, 37]. In this study, our focus is primarily on intrusion detection systems that utilize the clustering approach. Various clustering methods have been proposed specifically for intrusion detection systems [13, 19, 23, 38]. For instance, Li et al. [19] proposed a system that combines the K-means algorithm with PSO to create an effective intrusion detection system. This algorithm aims to benefit from the characteristics of PSO to overcome the premature convergence issue faced by the K-means algorithm. The proposed algorithm demonstrated relatively better results compared to the traditional K-means algorithm.

Indeed, Guan et al. [13] introduced a K-means-based clustering algorithm called Y-means specifically for intrusion detection. Y-means addresses the dependency of the K-means algorithm on the number of clusters. It aims to automatically partition

a dataset into a reasonable number of clusters by classifying instances into "normal" and "abnormal" clusters. This approach provides a way to differentiate between normal and potentially intrusive activities. Similarly, Liu et al. [23] proposed an intrusion detection method based on the genetic clustering algorithm. This method consists of two stages: nearest-neighbor clustering and genetic optimization. The approach operates under the assumption that intrusion activities are likely to appear as outliers among normal activities and can be grouped into separate clusters from the normal cluster. This proposed method allowed to automatically establish clusters and detect intruders by labeling normal and abnormal groups respectively.

Although existing clustering-based intrusion detection methods demonstrate good performance, they often face limitations when applied to large-scale networks. To address this challenge, the parallelization of clustering algorithms has gained significant attention due to its effectiveness in reducing runtime in large networks. Parallel algorithms exploit multiple processing nodes to achieve a speedup compared to running the sequential version of the algorithm on a single processor. Several parallel clustering methods have been proposed to tackle scalability challenges in intrusion detection [4, 14, 41]. For example, Al-Jarah et al. [4] proposed IDS-MRCPSO, a parallel intrusion detection system based on the MapReduce framework. The proposed system integrates a PSO technique in the clustering step to improve the quality of intrusion detection. The use of PSO helps to overcome the sensitivity problem associated with initial cluster centers. Wu et al. [41] proposed a parallel intrusion detection system using the MapReduce framework. They combined the differential evolution algorithm with the K-medoids clustering algorithm to improve convergence efficiency in large networks. Additionally, they introduced a dynamic Gemini population schema to further enhance the optimization of the clustering step by maintaining solution diversity and avoiding local optima. Peng et al. [33] proposed an intrusion detection system based on the Mini Batch K-means clustering algorithm and Principal Component Analysis. Firstly, they employed a pre-processing method to digitize strings and normalize the data. Secondly, they applied Principal Component Analysis (PCA) [1] to reduce the dimensionality of the processed data. Finally, they incorporated the Mini Batch K-means [35] algorithm for data clustering. Recently, Ben HajKacem et al. [14] proposed a Spark-based intrusion detection system that integrates PSO for large-scale networks. This system offers a favorable trade-off between scalability and accuracy. The use of PSO clustering has allowed solving the sensitivity issue related to initial cluster centers as well as premature convergence.

Although many existing intrusion detection systems based on clustering techniques are effective in analyzing large amounts of data, they often fail to provide security experts with a way to understand and interpret the results they obtain. These methods often act as "black-boxes" that build clusters without offering any explanations for the underlying reasoning behind their formation. This lack of transparency makes it challenging to interpret the detected intrusions, particularly for non-domain experts, and significantly reduces user trust.

2.2. Explainable clustering

Explainable clustering, a sub-field of eXplainable Artificial Intelligence (XAI) [28], aims to address the issue of explainability by assisting decision-makers in interpreting the resulting clusters and providing insightful explanations. The demand for XAI has grown in recent years driven by the widespread use of machine learning models in critical domains that require explanations for their decision-making processes. Two main approaches to XAI methods have been proposed [3]: intrinsic XAI and post-hoc XAI. Intrinsic XAI approaches focus on explaining the structure and functioning of the model itself, but are limited to specific types of models. On the other hand, post-hoc approaches explain the final decisions of the model by analyzing the set of input data and can be applied to any model. Another classification of XAI techniques involves global (macro) and local (micro) explanations [7, 21]. Global techniques aim to explain the general structure of the models by analyzing all of their decisions, whereas local techniques aim to provide explanations for individual decisions at the item level.

Explainable clustering methods follow a two-step process that utilizes XAI techniques to provide explanations for the clusters [5, 10, 29]. The first step focuses on assigning labels to the clusters, while the second step involves using these labels as target variables in a classification task. Explanations are then generated based on the resulting classification model. For instance, Morichetta et al. [29] proposed the EXPLAIN-IT method, which employs a supervised XAI technique to interpret clustering results. Initially, the authors cluster the input data using algorithms such as K-means or DBSCAN. Subsequently, a classifier is trained on the input data, employing the obtained cluster labels as the target variable. Finally, the classifier is explained using existing XAI models like LIME [34], which is commonly used to generate interpretations for individual predictions made by any classifier. Similarly, Horel et al. [15] also introduced a two-step method to explain the resulting clusters. First, a classifier is trained to assign cluster labels. Then, the Single Feature Introduction Test (SFIT) is applied to identify statistically significant features that characterize each cluster.

2.3. Explainable intrusion detection

Several works related to explainable intrusion detection systems were proposed in the literature [18, 39, 43]. Neupane et al. [32] conducted a study on existing explainable intrusion detection systems, which are primarily based on SHAP [24] and LIME [34] methodologies. Wang et al. [39] proposed a framework that utilizes SHAP to generate explanations for intrusion detection systems (IDS). This framework provides both local and global explanations to enhance the interpretation of IDS. The local explanation focuses on interpreting individual instances based on input features, whereas the global explanation reveals relationships between feature values and different attack types. Furthermore, Younisse et al. [43] also proposed an explainable intrusion detection system by combining deep neural networks with interpretable model predictions. Their system utilizes SHAP to provide both local and global explainability

for improving the interpretation of IDS. In addition, in the context of Internet of Things (IoT) networks, Keshk et al. [18] proposed an explainable intrusion detection system. They developed an IDS using a Long Short-Term Memory (LSTM) model for cyber-attack identification and utilized the SPIP (S: Shapley Additive exPlanations, P: Permutation Feature Importance, I: Individual Conditional Expectation, P: Partial Dependence Plot) technique to produce explanations for the model's decisions. The proposed system achieved high detection accuracy, efficient processing time, and improved interpretability compared to other IDS systems.

Despite the significant efforts to enhance the transparency and explainability of IDS, there are still several limitations and open challenges that need to be addressed. One major challenge is generating explanations in large-scale IDS systems. The vast amount of data in such systems poses difficulties in maintaining scalability while ensuring IDS accuracy. Developing techniques that can handle the volume and velocity of data in real-time while still providing meaningful and interpretable explanations, remains an interesting and complex challenge. Additionally, most of the existing explainable intrusion detection systems are built for IDS based on supervised classification approaches. These systems require labeled data for training and rely on predefined attack types. However, in real-world scenarios, there may be unknown or novel attack types that have not been labeled or encountered before. Therefore, exploring the explainability of IDS based on unsupervised approaches is an important area that has yet to be extensively studied in the literature.

3. Background

3.1. Particle Swarm Optimization

Particle Swarm Optimization (PSO) was originally proposed by Kennedy and Eberhart in 1995 [17]. PSO is inspired by the behavior of birds in a flock. It mimics their social interactions while searching for food. The algorithm has been widely applied to solve various optimization problems. In PSO, a population of particles forms a swarm. Each particle represents a potential solution to the optimization problem. At time t, each particle P_i is characterized by its current position $x_i(t)$ in the search space, its velocity $v_i(t)$, and its personal best position $pbestP_i(t)$ along with the corresponding fitness value $pbestF_i(t)$. The personal best position of a particle represents the best solution it has encountered and is defined as follows:

$$pbestP_i(t+1) = \begin{cases} pbestP_i(t) \; if \; f(pbestP_i(t)) <= f(x_i(t+1)) \\ x_i(t+1) \; if \; f(pbestP_i(t)) > f(x_i(t+1)) \end{cases} \tag{1}$$

The global best position represents the best fitness value of any particle and is defined as follows:

$$gbestP(t+1) = min\ (f(y), f(gbestP(t))) \tag{2}$$

where $y \in \{pbestP_0(t), ..., pbestP_S(t)\}$. The particle position and velocity are updated using the following formula:

$$x_i(t+1) \leftarrow x_i(t) + v_i(t) \tag{3}$$

$$v_i(t+1) \leftarrow wv_i(t) + c_1r_1(pbestP_i(t) - x_i(t)) + c_2r_2(gbestP(t) - x_i(t)) \tag{4}$$

where w is the inertia weight, $x_i(t)$ is the position of the particle P_i at time t, $v_i(t)$ is the velocity of the particle P_i at time t, c_1 and c_2 are two acceleration coefficients, and r_1 and r_2 are two random values in the range $[0, 1]$. The main algorithm of PSO is shown in Algorithm 1. The algorithm begins by creating an initial population of particles from the input dataset R. Then, it enters a loop until the convergence criteria are met. Within each iteration, the fitness value of each particle is calculated. The personal best position of each particle is updated using Equation (1). The global best position is updated using Equation (2). The velocities and positions of the particles are then updated using Equations (3) and (4), respectively. The algorithm continues iterating until the convergence criteria are reached.

Algorithm 1 PSO main algorithm

1: **Input:** Input dataset R
2: **Output:** Particle information
3: Create an initial population of particles from R.
4: **while** Convergence not reached **do**
5: Calculate the fitness value of particles.
6: Update the personal best position of each particle using Equation (1).
7: Update the global best position using Equation (2).
8: Update the velocities and positions using Equations (3) and (4), respectively.
9: **end while**

3.2. Spark framework

MapReduce [11] is a parallel programming framework based on the *map* and *reduce* phases. Each phase involves input and output $< key/value >$ pairs. During the map phase, map functions are executed in parallel to process each *key* and *value* pair and lead to the generation of a collection of intermediate $< key'/value' >$ pairs. Then, the shuffle phase compiles a list of all intermediate values associated with a particular intermediate key. After that in the reduce phase, the reduce function merges all intermediate values that belong to the same intermediate key. The data flow of the MapReduce framework is depicted in Figure 1. Input and output data for MapReduce are stored in a distributed file system accessible from the cluster of machines. Hadoop has implemented the MapReduce framework [40] and provides a distributed file system called Hadoop Distributed File System (HDFS) for data storage on the machines. Despite its high performance, the MapReduce framework is not suitable for iterative algorithms. The need to read and write data from disks in each iteration can significantly decrease the algorithm's efficiency.

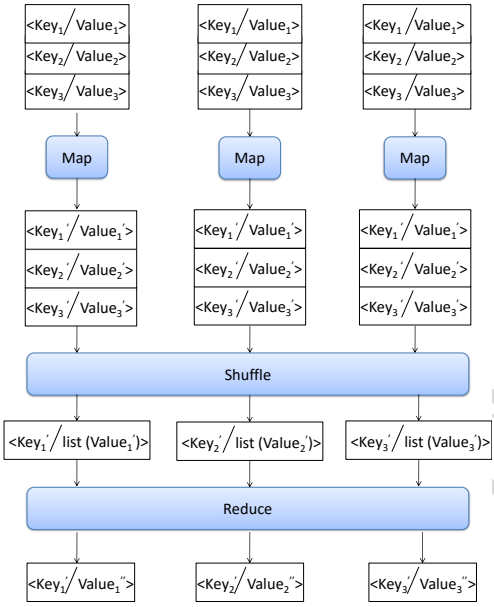


Figure 1. Data flow of MapReduce framework

Spark [44] is a parallel framework and one of the most widely used Big Data parallel processing frameworks. The high efficiency of Spark is due to resilient distributed datasets (RDDs) allowing to perform in-memory processing. RDDs can be stored in memory and used in multiple consecutive operations. Spark is integrated with Hadoop allowing it to read data from Hadoop Distributed File System (HDFS). Moreover, Spark provides a set of in-memory operators that enable faster data processing on distributed environments compared to standard MapReduce. The Spark framework supports two types of operators that can be applied to RDDs: transformations and actions. Transformations are used to apply a function to all elements of an RDD and return new RDDs. Actions, on the other hand, either return a value to the program or write the computation result to an external storage. This work employed the following Spark functions:

- `mapToPair(func)`: Creates a new RDD by applying the function `func` to all elements of the RDD.
- `flatMapToPair(func)`: Creates a new RDD by applying the function `func` to each element of the RDD and merging the results.
- `groupByKey(nums)`: Groups and distributes the values for each key in the RDD into a single sequence.
- `partitionBy(Partitioner)`: Creates a copy of the RDD partitioned using the specified partitioner.

- `filter(condition)`: Creates a new RDD that stores only the elements satisfying a given condition.
- `collect()`: Creates an array that stores all the elements in a particular RDD.
- `saveAsTextFile("...")`: Saves the elements of the RDD in a text file at the specified file path directory.

By utilizing these functions, Spark enables efficient and flexible data processing in distributed environments.

3.3. SHAP (SHapley Additive exPLanations)

SHAP is a XAI technique which is used to analyze predictions made by machine learning models [24]. It is based on game theory and provides explanations by detecting how each feature contributes to the accuracy of the predictions. SHAP also provides the most important features and their impact on model prediction. It deals with the Shapley values to evaluate each feature's impact on the machine learning prediction model. Shapley value is calculated as the (weighted) average of marginal contributions. It is defined by the impact of feature value on the prediction overall potential feature coalitions. Shapley value for an instance x is computed as follows:

$$\phi_{j_r}(x) = \sum_{S \subseteq \{j_1 \cdots j_m\} \setminus \{j_r\}} \frac{\mid S \mid ! \times (m - \mid S \mid -1)!}{m!} \times \delta_{j_r} \tag{5}$$

with

$$\delta_{j_r} = [f_{S \cup j_r}(x) - f_S(x)] \tag{6}$$

where $\phi_{j_r}(x)$ represent the Shapley value for feature value with the index $j_r \in [1..m]$, S is a subset of the features employed in the prediction model., $|S|$ is the carnality of S, m is the number of features, $f_{S \cup j_r}(x)$ and $f_S(x)$ are the prediction function for the set of feature values in S with and without including the feature j_r respectively. The Shapley value $\phi_{j_r}(x)$ quantifies how much the feature j_r influences the prediction model, either positively or negatively. To this end, the model is trained with and without including this feature and then predictions from the two models are compared for all subset features $S \subset \{j_1, ..., j_m\} \setminus \{j_r\}$. A large positive value for $\phi_{j_r}(x)$ indicates that the feature j_r has a significant positive influence on the prediction. However, a large negative value for $\phi_{j_r}(x)$ shows that the feature j_r has a significant negative contribution on the prediction.

4. Proposed explainable Spark-based PSO clustering method for intrusion detection

In order to simultaneously solve the issues of large amounts of network traffic data and the complexity of explaining the built intrusion detection model, we propose a new design of an explainable Spark-based PSO clustering approach for intrusion detection.

As shown in Figure 2, E-SPSO consists of three main phases: *data detector modeling phase*, *data labeling phase* and *model explaining phase*.

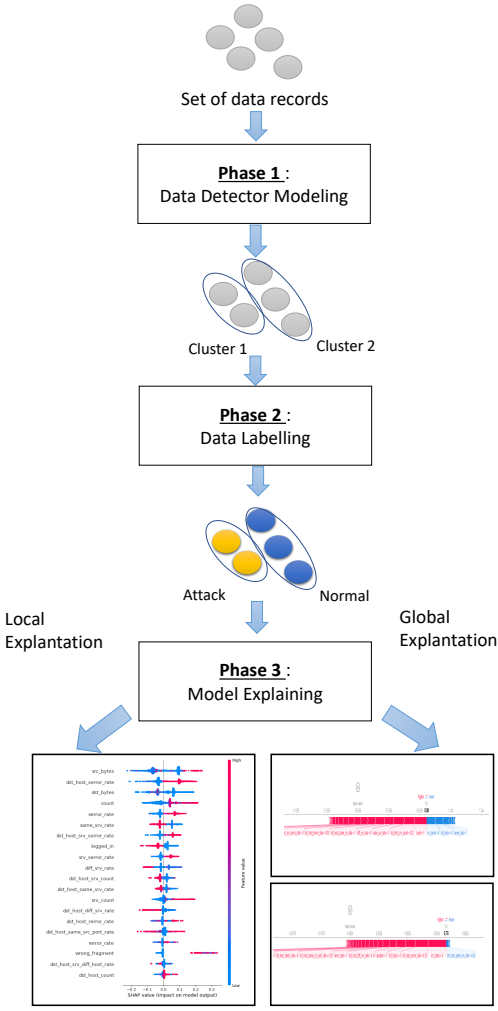


Figure 2. The main phases of the proposed explainable Spark-based PSO clustering approach for intrusion detection. Phase 1: Data detector modeling, phase 2: Data labeling and phase 3: Model explaining

In the first phase, we propose to exploit a Spark-based PSO clustering method [30] to guide the intrusion detection model by generating global best centers of obtained clusters. The use of particle swarm optimization for the clustering task is a very efficient way since particle swarm optimization avoids the sensitivity problem of initial cluster centers as well as premature convergence. In the second phase, we assign a

cluster label to testing data by computing distances between the testing data and the final global best centers. In the third phase, we propose to explain the built intrusion detection model by exploiting XAI SHAP method capabilities. In fact, the transparency and the easy interpretation of the intrusion detection model are almost as important as the classification accuracy. Hence, an intelligent SHAP-based process is designed having as outputs detailed explanations of the global structure of the intrusion detection model as well as local explanations regarding the assignment of each connection to any specific class.

4.1. Phase 1: Data detector modeling

The data detector modeling phase consists of applying the parallel particle swarm optimization clustering method through the Spark framework. The use of particle swarm optimization in the clustering task is an efficient solution to avoid the sensitivity problem of initial cluster centers. The proposed method is a partitioning clustering type that uses representatives, called also centers, to model the clusters. The initial centers are selected randomly from the input data, then the centers are updated based on the swarm particle velocities until the convergence is achieved. The best cluster centers are then used in the data labeling phase by computing the average minimum distances between the data and the cluster centers.

It is important to note that the proposed method stores for each particle the following information: *position vector*, *velocity vector* and *fitness value*. The particle information is updated in each iteration using the information from the previous iteration. The proposed method is composed of three MapReduce jobs namely, *Data assignment and fitness computation*, *Personal and global best update* and *Position and velocity update*.

4.1.1. Data assignment and fitness computation

In the first MapReduce job, E-SPSO starts by initializing particle information. The positions of particles are randomly selected from the input data as initial cluster centers. Then, the data is divided into chunks where each chunk is assigned to a map function. The particle information are then transferred to all chunks. The map function first assigns each data record to the nearest cluster center in each particle by computing distances. Then, the map function generates a key-value pair as output where the key represents the couple $< particleID, centerID >$ and the value represents the minimum distance between a single data and the centerID in a particleID. Once all data are assigned to the nearest cluster, a reduce function is applied to compute the fitness value by merging data from different map functions. The fitness value is computed as follows:

$$Fitness = \frac{\sum_{j=1}^{k} \sum_{i=1}^{|C_j|} dis(d_i, C_j)}{k} \tag{7}$$

with $dis(d_i, C_j)$ represents the distance between a data record d_i and the cluster center C_j. $|C_j|$ represents the number of data records assigned to the center C_j, and

k represents the total number of clusters. The reduce function in the MapReduce job generates key-value pairs as output where the key represents the particle ID, and the value represents the fitness value.

Let $D = \{d_1, d_2, ..., d_n\}$ be the set of data records. Let $P(t) = \{P_1(t), P_2(t), ..., P_S(t)\}$ be the set of particle information, where $P_c(t) = \{x_c(t), v_c(t), pbestP_c(t), pbestF_c(t)\}$ represents the information of particle c in iteration t. Here, $x_c(t)$ is the position, $v_c(t)$ is the velocity, $pbestP_c(t)$ is the best position, and $pbestF_c(t)$ is the best fitness of particle c. Let $F = \{F_1, F_2, ..., F_S\}$ be the set of fitness values, where F_c represents the fitness value of particle c. The main steps of Data assignment and fitness computation MapReduce job are described in Algorithm 2.

Algorithm 2 Data assignment and fitness computation MapReduce job

Input: Input dataset D
Output: Fitness values F
$P(t) \leftarrow$ Initialize particle information from D
Divide the data D into m RDD $D = \{D^1 ... D^m\}$
% *Map Phase*
Let D^p be an RDD assigned to map task p.
for each $d_i \in D^p$ **do**
  for each $P_c \in P(t)$ **do**
    $x_c(t) \leftarrow$ Extract positions from $P_c(t)$
    Assign each data point to its nearest cluster centroid by computing distances.
    Let mindis the minimum computed distance.
    Let CentroidID the index of the cluster centroid where the data point r_i is assigned.
    Let ParticleID the index of the particle P_c.
  end for
  Emit (key: ParticleID, CenterID/value: mindis)
end for
% *Reduce Phase*
for each $P_i \in P(t)$ **do**
  Calculate fitness value F_i using Equation (7).
  Emit (key: ParticleID /value: F_i)
end for

4.1.2. Pbest and gbest update

Once all particle fitness values are computed, they are automatically distributed to RDD collections. Given that the computation of pbest (personal best) and gbest (global best) is not an expensive operation, they are computed locally without using the parallel framework. Let $pbestF(t) = \{pbestF_1(t), pbestF_2(t), ..., pbestF_S(t)\}$ be the set of personal best fitness values where $pbestF_i(t)$ represents the pbest fitness of particle i at iteration t. Let $pbestP(t) = \{pbestP_1(t), pbestP_2(t), ..., pbestP_S(t)\}$ be the set of personal best positions where $pbestP_i(t)$ represents the pbest position of particle i at iteration t. Let $gbestP$ be the position of the best particle. The main steps of the pbest and gbest update MapReduce job are described in Algorithm 3.

Algorithm 3 pbest and gbest update MapReduce job

Input: F, $pbestF(t)$, $pbestP(t)$
Output: $pbestF(t+1)$, $pbestP(t+1)$, $gbestP$
$gbestP \leftarrow \emptyset$
for each $P_i(t) \in P(t)$ **do**
  $pbestF_i(t+1) \leftarrow \emptyset$
  $pbestP_i(t+1) \leftarrow \emptyset$
  if $(pbestF_i(t) \leq F_i)$ **then**
    $pbestF_i(t+1) \leftarrow pbestF_i(t)$
    $pbestP_i(t+1) \leftarrow pbestP_i(t)$
  else
    $pbestF_i(t+1) \leftarrow F_i$
    $pbestP_i(t+1) \leftarrow x_i(t+1)$
  end if
end for
Let i^* is the index of a particle having the best fitness value.
$gbestP \leftarrow x_{i^*}(t)$

4.1.3. Position and Velocity update

During the MapReduce job, the E-SPSO algorithm begins by assigning particle information to different map functions. Each map function then performs velocity and position updates using Equations (3) and (4). The reduce function groups all the intermediate key-value pairs computed by the map functions. After the reduce phase is completed, the particle information is distributed among RDD collections which are stored in memory for the next iteration. Let $x(t) = \{x_1(t), x_2(t), ..., x_S(t)\}$ represent the set of position values, where $x_i(t)$ denotes the position of particle i at iteration t. Similarly, let $v(t) = \{v_1(t), v_2(t), ..., v_S(t)\}$ denote the set of velocity values where $v_i(t)$ represents the velocity of particle i at iteration t. The main steps of the Position and Velocity Update MapReduce job are described in Algorithm 4.

Algorithm 4 Position and Velocity update MapReduce job

Input: $gbestP$, $P(t)$
Output: $P(t+1)$
% *Map Phase*
Divide the data $P(t)$ into m RDD $D = \{P^1 ... P^m\}$ Let $P^p(t)$ be an RDD assigned to a map task p.
$x_i(t+1) \leftarrow \emptyset$
$v_i(t+1) \leftarrow \emptyset$
Update the new position value $x_i(t+1)$ using Equation 4
Update the new velocity value $v_i(t+1)$ using Equation 3
Emit(key: 1/value: $P_i(t+1)$)
% *Reduce Phase*
Group outputs from the different map functions and update the new particle information $P(t+1)$
Emit $(P(t+1))$

4.2. Phase 2: Data labeling

Once the data detector modeling phase is completed, the global best cluster centers are extracted from the final particle's information. In this phase, the detection model is evaluated by computing distances between the testing data and the global best centers. To accomplish this, the testing data is assigned to their nearest clusters based on the computed distances.

The next step involves the cluster labeling process where the correct labels are predicted for the clusters generated during the testing data assignment. Cluster labeling is performed by determining the maximum intersection percentage between the true labels of the testing data and the assigned clusters generated during the testing data assignment phase. The main steps of the labeling phase are described in Algorithm 5.

Algorithm 5 Data labelling phase

1: **Input:** Testing data T, Final Particle information P
2: **Output:** Labelled data
3: Let $C()$ the k centers extracted from the final particle P.
4: **for** each $t_i \in T$ **do**
5: Compute distances between t_i and C.
6: Assign t_i to its nearest center.
7: **end for**
8: Apply the cluster labeling.

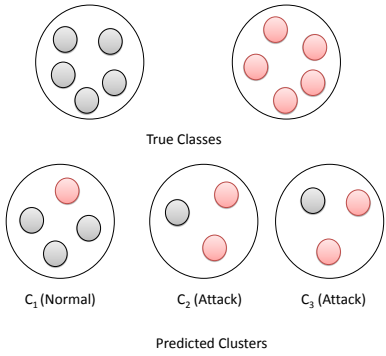


Figure 3. An illustrative example of the clusters labeling process

Figure 3 provides an illustrative example to aid in understanding the cluster labeling process. Let's examine the labeling process for each cluster: Cluster C_1 consists of $\frac{3}{4}$ normal connections and $\frac{1}{4}$ attack connections. As a result, it is labeled as "normal connections"; Cluster C_2 comprises $\frac{1}{3}$ normal records and $\frac{2}{3}$ attack records, leading to its labeling as "attack cluster"; Similarly, cluster C_3 contains $\frac{1}{3}$ normal

connections and $\frac{2}{3}$ attack connections, resulting in its classification as an "attack cluster".

4.3. Phase 3: Model explaining

This phase aims to build detailed explanations on the built classes resulting from the previous phase. These explanations allow cybersecurity experts to better understand and interpret the resulting classification of the connections in terms of local and global levels. Local explanations have the objective to explain the reasons behind the prediction of each connection to any class in terms of feature values while global explanations try to explain important feature values at the level of classes. This phase is based on explaining predictions using the SHAP method capabilities. First, the resulting classes are used as the label class variables of the explainable process. Then, local and global explanations are generated by applying the SHAP technique.

Concerning local explanations, we calculate for each record and each feature the Shapley value regarding the assigned class. These Shapley values measure how much each feature, for each record, contributes to the final prediction. These local explanations support cybersecurity experts to better understand the reasons behind assigning a connection to any specific normal or attack class. Experts can also analyze the positive and negative contribution of each feature of each connection on the predicted class label. Concerning global explanations, we calculate further explanations regarding the importance and contributions of the features when building each class. We use local Shapley values of each data record as a "single unit" to build an overall score for each class. The global explanation is performed as follows :

$$G_j = \frac{1}{n} \sum_{i=1}^{n} |\phi_j(x_i)| \tag{8}$$

where $\phi_j(x_i)$ is the Shapley value of the feature j for the record x_i, G_j refers to the overall Shapley value of the feature j and n is the total number of records in the dataset. Global shapley values are then sorted in decreasing order to generate the most important feature of the model.

5. Experiments and results

5.1. Dataset description

In order to evaluate the performance of the proposed method, we used a Big intrusion detection dataset[1] which is well suited for intrusion detection problems. This dataset contains a standard set of data records which includes a wide variety of normal and attack connections in a military network environment. Each record in the dataset represents a connection between two IP addresses. The data is classified into normal traffic and four kinds of attacks namely, denial of service (DOS), probe (PROB), remote to local (R2L) and user to root (U2R). Each connection in the dataset is

described by 3 categorical and 38 numerical features for a total of 41 features. The detailed descriptions of the features are given in [12]. The training dataset contains $4,898,431$ data records collected during seven weeks of network traffic while the testing dataset contains $311,029$ records collected during two weeks.

In order to evaluate the scalability of the proposed method, we extract 4 different data samples from the whole training data set. To simplify the names of the data samples, we will use the notations Train20, Train40, Train80, and Train100 to denote an extracted data set that stores 20%, 40%, 80%, and 100% of the whole training data set. Statistics of these datasets are summarized in Table 1.

Table 1
Summary of the data samples

Dataset	Number of records	Normal	Attack
Train20	979,686	194,556	785,130
Train40	1,959,372	389,112	1,570,260
Train80	3,918,745	778,225	3,140,520
Train100	4,898,431	972,781	3,925,650

5.2. Data pre-processing

A set of pre-processing techniques was applied to the training and testing datasets. Firstly, records with missing values were eliminated as distances cannot be computed for these records. Additionally, columns containing categorical features were removed to retain only the numerical features. Subsequently, a min-max normalization within the range of $[0, 1]$ is applied to the obtained dataset. This normalization step helps address bias issues arising from features that exhibit significant variation between their minimum and maximum values. The normalization is performed using the following equation:

$$x_{ij_{new}} = \frac{x_{ij} - x_{j_{min}}}{x_{j_{max}} - x_{j_{min}}} \tag{9}$$

where x_{ij} is the value of record i for feature j, $x_{ij_{new}}$ is the normalized value of record i for feature j, $x_{j_{min}}$ is the minimum value of feature j and $x_{j_{max}}$ is the maximum value of feature j.

5.3. Evaluation measures

In order to evaluate the scalability of the proposed method, we use the Speedup measure [42] which consists of fixing the dataset size and varying the number of machines. The Speedup measure is defined as follows:

$$Speedup = \frac{T_1}{T_m}, \tag{10}$$

[1] https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

where T_1 is the running time of processing data on 1 machine and T_m is the running time of processing data on m machines.

In order to evaluate the quality of the proposed method, we use true positives, true negatives, false positives, and false negatives. A true positive (TP) indicates that the intrusion detection system detects precisely a particular attack that has occurred. A true negative (TN) indicates that the intrusion detection system has not made a mistake in detecting a normal connection. A false positive (FP) indicates that a particular attack has been detected by the intrusion detection system but that attack did not actually occur. A false negative (FN) indicates that the intrusion detection system is unable to detect the intrusion after a particular attack has occurred. We use in this paper the True Positive Rate (TPR) and False Positive Rate (FPR) that are defined in Equation (11) and Equation (13) respectively.

$$TPR = \frac{TP}{TP + FN} \tag{11}$$

$$FPR = \frac{FP}{FP + TN} \tag{12}$$

Furthermore, we use the Area Under Curve (AUC) measure [46] to combine the TPR and FPR which is considered a good indicator of these rates. The AUC can be defined as follows:

$$AUC = \frac{(1 - FPR) \times (1 + TPR)}{2} + \frac{FPR \times TPR}{2} \tag{13}$$

A greater value of these measures indicates better quality results.

5.4. Evaluation of the clustering quality

We evaluate the accuracy of the proposed method (E-SPSO) compared to four existing methods: K-means [25], PSO [27], MRKM [45] and MRPSO [4]. We use in the experiments the following parameters: the number of particles equal to 10, the number of iterations equal to 50, the inertia weight equal to 0.72 and the acceleration coefficients equal to 1.49. Table 2 reports the TPR, FPR and AUC values obtained by the proposed method using different training data sample sizes. The obtained results show that the proposed E-SPSO gives nearly the same results as the existing MRPSO. In addition, we observed that TPR value of E-SPSO using the whole training data (i.e Train100) reaches its best value compared to those obtained in smaller training datasets. Furthermore, Table 2 shows that E-SPSO obtains the lowest FPR for Train100 dataset.

Table 2
Comparison of the accuracy of E-SPSO with existing methods

Dataset	Method	TPR	FPR	AUC
Train20	K-means	0.789	0.214	0.758
	PSO	0.879	0.015	0.927
	MRKM	0.789	0.214	0.758
	MRPSO	0.903	0.038	0.933
	E-SPSO	0.848	0.096	0.875
Train40	K-means	0.876	0.155	0.733
	PSO	0.841	0.019	0.901
	MRKM	0.876	0.155	0.733
	MRPSO	0.911	0.021	0.945
	E-SPSO	.856	0.085	0.902
Train80	K-means	0.798	0.087	0.828
	PSO	0.899	0.113	0.914
	MRKM	0.798	0.087	0.828
	MRPSO	0.935	0.013	0.961
	E-SPSO	0.879	0.068	0.944
Train100	K-means	0.871	0.149	0.719
	PSO	0.939	0.013	0.963
	MRKM	0.871	0.149	0.719
	MRPSO	0.939	0.013	0.963
	E-SPSO	0.883	0.059	0.905

For instance, the E-SPSO has a high TPR value (0.883) for Train100 while it is equal to 0.848 for Train20 dataset. In addition, E-SPSO has a low FPR of 0.059 for Train100 compared to the value of 0.096 for Train20 dataset. The obtained high scores of TPR and the low values of FPR show that the proposed method can effectively distinguish between normal and attack data records. Better scores are shown as the size of the training dataset increases.

5.5. Evaluation of the scalability

We firstly evaluated the running time of the proposed method compared to the existing methods. Figure 4 shows the obtained running times for the 4 training data samples. The obtained results show that the proposed method is faster than existing methods. For instance, the E-SPSO is faster by a factor of 3.72 and 1.04 than PSO and MRPSO respectively for Train100 dataset. This result can be explained by the effectiveness of the parallel framework Spark when processing large-scale data. Hence, we can conclude that the parallel PSO clustering method based on Spark framework can be a good solution to handle large intrusion detection problems.

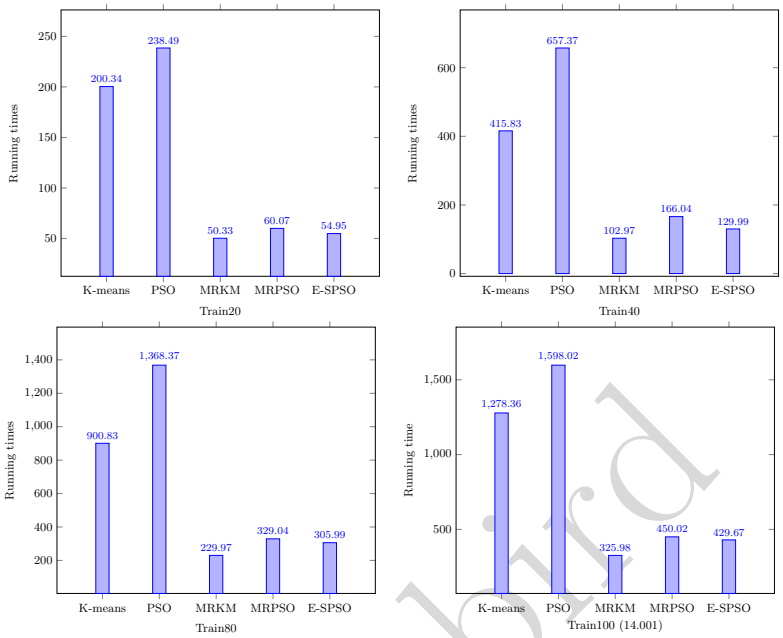


Figure 4. Comparison of the running times of E-SPSO with existing methods.

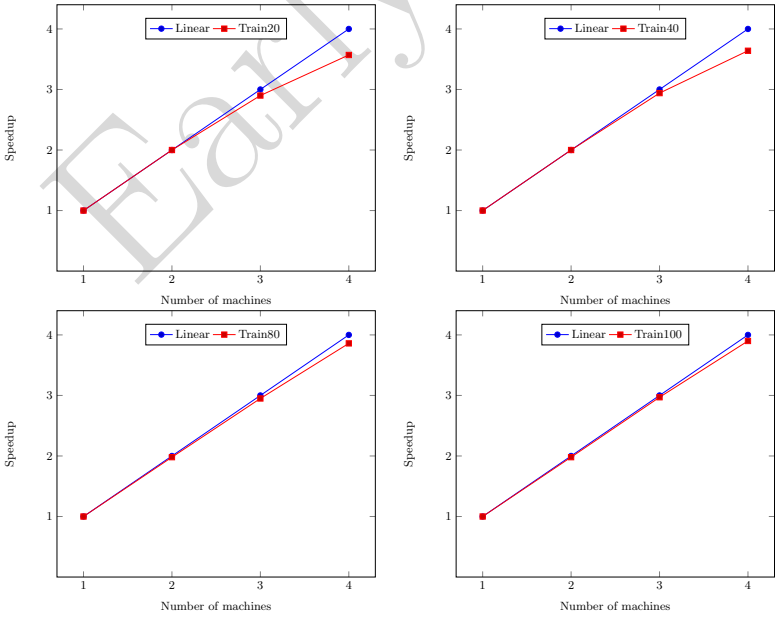


Figure 5. Evaluation of Speedup results using an increasing number of machines

After that, we evaluated the scalability of the proposed method using an increasing number of machines. Figure 5 shows the Speedup results using different training data sizes with different numbers of machines. This figure shows good Speedup results as the data size increases. For example, the Speedup value when running E-SPSO using 4 machines for Train20 is 3.57 while it is 3.90 for Train100 data. In addition, the proposed method shows approximately a linear speedup when the number of machines increases. This can be explained by the benefits of the in-memory processing of the Spark framework which can significantly reduce the network cost when the number of machines increases.

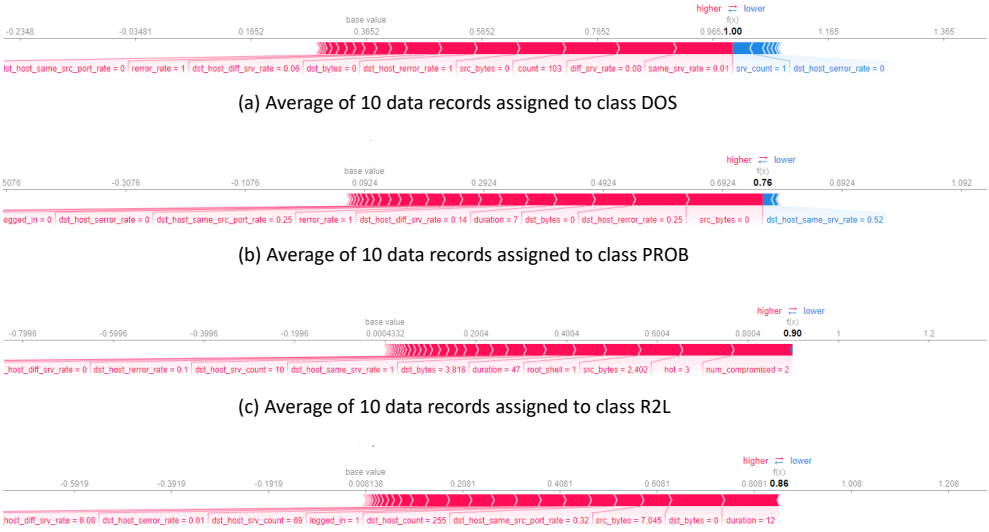
5.6. Evaluation of the local explanation

To provide local explanations, we utilized a force plot diagram to visually illustrate the contribution of each feature in the decision-making process of the model. The length of each feature's representation along the horizontal axis indicates its importance in the decision-making process. The colors red and blue are used to indicate the influence of a particular feature on the decision to assign records to a specific class. The red color signifies that a feature with a specific value may boost the assignment to that class. On the other hand, the blue color indicates that a feature with a specific value boosts the assignment to the other classes. In Figure 6, instead of analyzing a single record for local feature importance, we chose to randomly select 10 data records from each class. This approach provides a more significant analysis of the feature contributions as it considers a broader range of records for each class.

Figure 6(a) presents the force plot diagram for an average of 10 data records that were correctly classified by the model as a DOS attack. This figure shows that the feature *same_srv_rate* plays a significant role in determining the assignment of records to the DOS attack class, with a value of 0.01. This lower value of *same_srv_rate* increases the probability of assigning the record to the DOS attack class. Furthermore, the features *diff_srv_rate* with a value of 0.08 and *count* with a value of 113 also result in a higher probability of assigning the data record to the DOS attack class.

Besides, Figure 6(b) focuses on an average of 10 data records classified as PROB attacks. The figure highlights that features such as *src_bytes = 0*, *dst_host_rerror_rate = 0.25*, and *dst_bytes = 0* contribute significantly to the decision of classifying a record as a PROB attack. Notably, the feature *dst_host_same_srv = 0.52* reduces the probability of assigning a record to the PROB attack class. Moving on to Figure 6(c) that provides a local explanation for 10 data records that were correctly classified as R2L attacks, the features *num_compromised=2*, *hot = 3*, and *src_bytes = 2.42* play a significant role in the decision to classify a record as an R2L attack. Higher values of these features greatly increase the likelihood of the record being classified as an R2L attack. Concerning Figure 6(d) that presents a local explanation for 10 data records that were correctly assigned to the U2R attack class, the feature *duration* with a value of 12 contributes positively to the final classification, indicating that

a longer duration increases the likelihood of the connection being assigned to the U2R attack class. Additionally, features such as *dst_bytes = 0* and *src_bytes = 7.045* also positively contribute to this decision.



(a) Average of 10 data records assigned to class DOS

(b) Average of 10 data records assigned to class PROB

(c) Average of 10 data records assigned to class R2L

(d) Average of 10 data records assigned to class U2R

Figure 6. Force plot diagram illustrating local feature contribution in the decision of assigning records to each attack class, based on an average of 10 data records per class. The varying colors, red and blue, indicate whether a feature increases or decreases the probability of assigning records to a particular attack class

5.7. Evaluation of the global explanation

The global explanation of E-SPSO results aims to provide detailed insights into the key features that played a crucial role in the classification of attack classes. To build such explanations, we used the SHAP feature-summary-plot as shown in Figure 7. Each dot in the figure represents a data record from the used dataset. The dot's vertical position represents the feature while the horizontal position represents the impact of that feature value on the model's classes (local to each class). The color of each dot indicates the value of the corresponding feature for that record, with red representing high values, purple representing medium values, and blue representing low values.

Figure 7(a) illustrates that high values of the *src_bytes* feature increase the probability of the connection being classified as a DOS attack by 20% to 30%. Conversely, low values of the *dst_bytes* feature increase the probability of the model identifying the records as DOS attacks. Figure 7(b) demonstrates the global explanation of PROB attacks. Low values of the *src_bytes* feature increase the probability of a prediction

as a PROB attack by 1% to 30%, while large values of the *dst_host_same_srv* feature increase the probability of the connection being classified as a PROB attack by 10% to 30%.

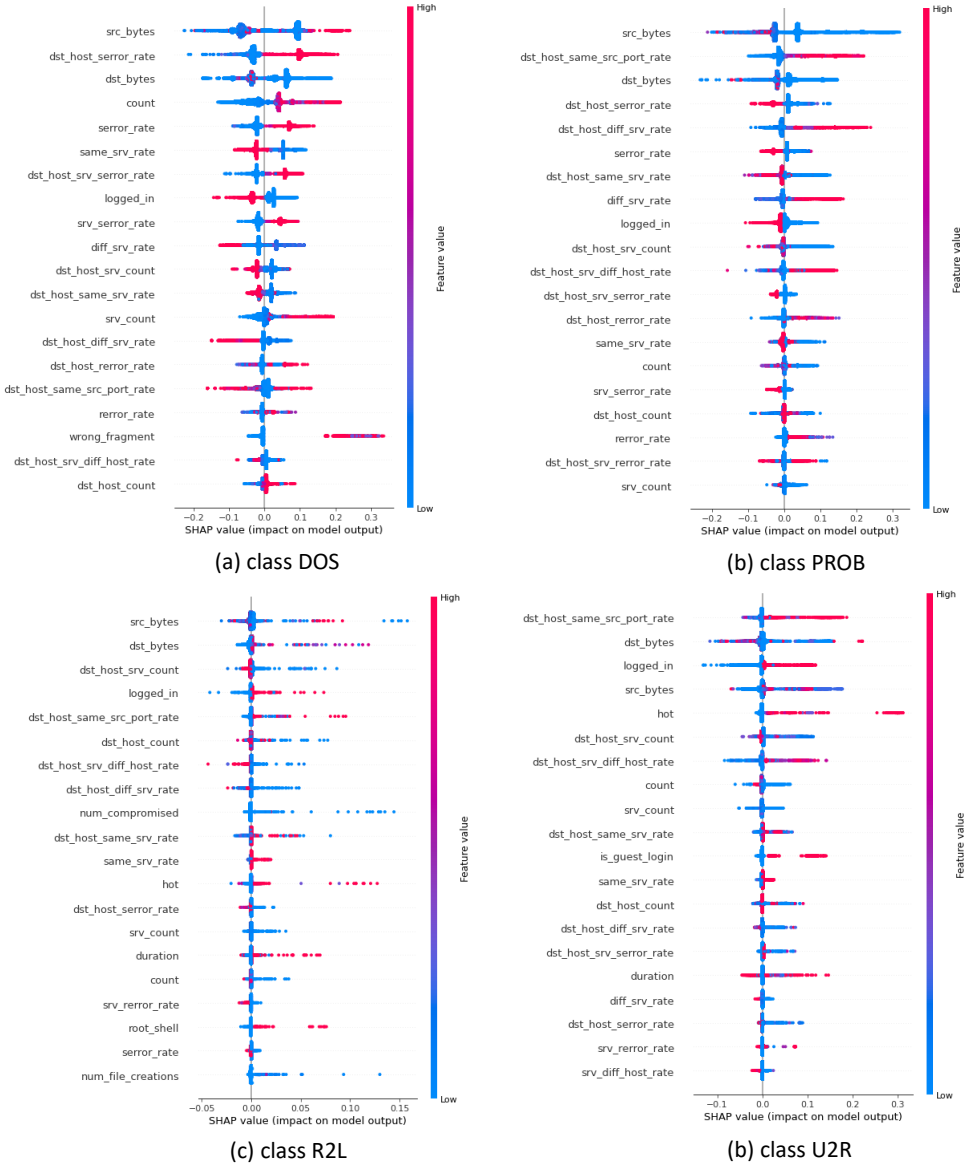


Figure 7. Top 20 important features that were crucial in building each attack class.

Then, Figure 7(c) provides the global explanation of R2L attacks. Low values of the *dst_srv_count* feature increase the probability of a prediction as an R2L attack by 1% to 30%. Additionally, the figure shows that large values of the *logged_in* feature can increase the probability of the connection being classified as an R2L attack by up to 30%. The last sub-figure, Figure 7(d), illustrates the global explanation of U2R attacks. Low values of the *dst_srv_count* feature can increase the probability of the records being considered as U2R attacks by up to 30%. Moreover, large values of the *dst_host_same_src* and *logged_in* features can increase the probability of the connection being classified as a U2R attack by up to 20%.

6. Conclusion

We proposed in this work an explainable Spark-based PSO clustering for an effective intrusion detection system that deals with the issues of scalability, accuracy, and explainability of intrusion detection. The proposed system uses parallel clustering and explainable artificial intelligence capabilities to build local micro and macro explanations. It includes three independent phases: data detector modeling, data labeling, and model explaining. The first phase aims to build clusters based on Spark-based PSO clustering, the second phase assigns an attack label to each build group while the third phase is devoted to the generation of local and global explanations. Empirical experiments performed on real-world data from the military environment have shown a significant improvement of the scalability, accuracy, and explainability of intrusion detection.

The output of the proposed system relies on the quality of defined features. The initial selection of features on such applications has a large impact on model outputs and also on the explainability of results. It would be interesting to include a pre-step of feature selection to extract the most important features when building attack clusters. Besides, we considered in this work a pre-configured number of clusters (known in advance). However, in real-life intrusion detection problems, it would be interesting to integrate automatic techniques for estimating the number of existing attack classes. Another interesting future direction to improve this work is to integrate other XAI techniques such as LIME which may improve the model interpretability.

Acknowledgements

This work was funded by the University of Jeddah, Jeddah, Saudi Arabia, under grant No. (UJ-23-DR-101). Therefore, the authors thank the University of Jeddah for its technical and financial support

References

[1] Abdi H., Williams L.J.: Principal component analysis, *Wiley interdisciplinary reviews: computational statistics*, vol. 2(4), pp. 433–459, 2010. doi: 10.1002/wics.101.

[2] Ahmad Z., Shahid Khan A., Wai Shiang C., Abdullah J., Ahmad F.: Network intrusion detection system: A systematic study of machine learning and deep learning approaches, *Transactions on Emerging Telecommunications Technologies*, vol. 32(1), p. e4150, 2021.

[3] Ali S., Abuhmed T., El-Sappagh S., Muhammad K., Alonso-Moral J.M., Confalonieri R., Guidotti R., Del Ser J., Díaz-Rodríguez N., Herrera F.: Explainable Artificial Intelligence (XAI): What we know and what is left to attain Trustworthy Artificial Intelligence, *Information Fusion*, p. 101805, 2023. doi: 10.1016/j.inffus.2023.101805.

[4] Aljarah I., Ludwig S.A.: Towards a scalable intrusion detection system based on parallel pso clustering using mapreduce. In: *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, pp. 169–170, 2013. doi: 10.1145/2464576.2464661.

[5] Bandyapadhyay S., Fomin F., Golovach P.A., Lochet W., Purohit N., Simonov K.: How to Find a Good Explanation for Clustering? In: *AAAI-2022*, vol. 36(4), 2022. doi: 10.1609/aaai.v36i4.20306.

[6] Bradley P.S., Fayyad U.M.: Refining initial points for k-means clustering. In: *ICML*, vol. 98, pp. 91–99, Citeseer, 1998.

[7] Carvalho D.V., Pereira E.M., Cardoso J.S.: Machine learning interpretability: A survey on methods and metrics, *Electronics*, vol. 8(8), p. 832, 2019. doi: 10.3390/electronics8080832.

[8] Cura T.: A particle swarm optimization approach to clustering, *Expert Systems with Applications*, vol. 39(1), pp. 1582–1588, 2012. doi: 10.1016/j.eswa.2011.07.123.

[9] Dafir Z., Lamari Y., Slaoui S.C.: A survey on parallel clustering algorithms for big data, *Artificial Intelligence Review*, vol. 54, pp. 2411–2443, 2021.

[10] Dasgupta S., Nave Frost M.M., Rashtchian C.: Explainable k-Means and k-Medians Clustering. In: *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[11] Dean J., Ghemawat S.: MapReduce: simplified data processing on large clusters, *Communications of the ACM*, vol. 51(1), pp. 107–113, 2008. doi: 10.1145/1327452.1327492.

[12] Dhanabal L., Shantharajah S.: A study on NSL-KDD dataset for intrusion detection system based on classification algorithms, *International journal of advanced research in computer and communication engineering*, vol. 4(6), pp. 446–452, 2015.

[13] Guan Y., Ghorbani A.A., Belacel N.: Y-means: A clustering method for intrusion detection. In: *CCECE 2003-Canadian Conference on Electrical and Computer Engineering. Toward a Caring and Humane Technology (Cat. No. 03CH37436)*, vol. 2, pp. 1083–1086, IEEE, 2003.

[14] HajKacem M.A.B., Moslah M., Essoussi N.: Spark Based Intrusion Detection System Using Practical Swarm Optimization Clustering. In: *Artificial Intelligence and Blockchain for Future Cybersecurity Applications*, pp. 197–216, Springer, 2021. doi: 10.1007/978-3-030-74575-2_11.

[15] Horel E., Giesecke K., Storchan V., Chittar N.: Explainable clustering and application to wealth management compliance. In: *Proceedings of the First ACM International Conference on AI in Finance*, pp. 1–6, 2020. doi: 10.1145/3383455.3422530.

[16] Javed Awan M., Mohd Rahim M.S., Nobanee H., Yasin A., Khalaf O.I.: A big data approach to black friday sales, *MJ Awan, M Shafry, H Nobanee, A Yasin, OI Khalaf et al," A big data approach to black friday sales," Intelligent Automation & Soft Computing*, vol. 27(3), pp. 785–797, 2021. doi: 10.32604/iasc.2021.014216.

[17] Kennedy J., Eberhart R.: Particle swarm optimization. In: *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995. doi: 10.1007/978-0-387-30164-8_630.

[18] Keshk M., Koroniotis N., Pham N., Moustafa N., Turnbull B., Zomaya A.Y.: An explainable deep learning-enabled intrusion detection framework in IoT networks, *Information Sciences*, vol. 639, p. 119000, 2023. doi: 10.1016/j.ins.2023.119000.

[19] Li Z., Li Y., Xu L.: Anomaly intrusion detection method based on k-means clustering algorithm with particle swarm optimization. In: *2011 international conference of information technology, computer engineering and management sciences*, vol. 2, pp. 157–161, IEEE, 2011. doi: 10.1109/icm.2011.184.

[20] Liao H.J., Lin C.H.R., Lin Y.C., Tung K.Y.: Intrusion detection system: A comprehensive review, *Journal of Network and Computer Applications*, vol. 36(1), pp. 16–24, 2013. doi: 10.1016/j.jnca.2012.09.004.

[21] Linardatos P., Papastefanopoulos V., Kotsiantis S.: Explainable ai: A review of machine learning interpretability methods, *Entropy*, vol. 23(1), p. 18, 2020. doi: 10.3390/e23010018.

[22] Liu H., Lang B.: Machine learning and deep learning methods for intrusion detection systems: A survey, *applied sciences*, vol. 9(20), p. 4396, 2019. doi: 10.3390/app9204396.

[23] Liu Y., Chen K., Liao X., Zhang W.: A genetic clustering method for intrusion detection, *Pattern Recognition*, vol. 37(5), pp. 927–942, 2004. doi: 10.1016/j.patcog.2003.09.011.

[24] Lundberg S.M., Lee S.I.: A unified approach to interpreting model predictions, *Advances in neural information processing systems*, vol. 30, 2017.

[25] MacQueen J.: Some methods for classification and analysis of multivariate observations. In: *Proc. 5th Berkeley Symposium on Math., Stat., and Prob*, p. 281, 1965.

[26] Madhulatha T.S.: An overview on clustering methods, *arXiv preprint arXiv:12051117*, 2012. doi: 10.9790/3021-0204719725.

[27] Van der Merwe D., Engelbrecht A.P.: Data clustering using particle swarm optimization. In: *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, vol. 1, pp. 215–220, IEEE, 2003.

[28] Moore J.D., Swartout W.R.: *Explanation in expert systemss: A survey*, Tech. rep., University of Southern California Marina del Rey Information Sciences Inst, 1988.

[29] Morichetta A., Casas P., Mellia M.: EXPLAIN-IT: Towards Explainable AI for Unsupervised Network Traffic Analysis. p. 22–28, Big-DAMA '19, Association for Computing Machinery, New York, NY, USA, 2019. doi: 10.1145/3359992.3366639.

[30] Moslah M., HajKacem M.A.B., Essoussi N.: Spark-based design of clustering using particle swarm optimization, *Clustering Methods for Big Data Analytics: Techniques, Toolboxes and Applications*, pp. 91–113, 2019.

[31] Nanda S.J., Panda G.: A survey on nature inspired metaheuristic algorithms for partitional clustering, *Swarm and Evolutionary computation*, vol. 16, pp. 1–18, 2014. doi: 10.1016/j.swevo.2013.11.003.

[32] Neupane S., Ables J., Anderson W., Mittal S., Rahimi S., Banicescu I., Seale M.: Explainable intrusion detection systems (x-ids): A survey of current methods, challenges, and opportunities, *IEEE Access*, vol. 10, pp. 112392–112415, 2022. doi: 10.1109/access.2022.3216617.

[33] Peng K., Leung V.C., Huang Q.: Clustering approach based on mini batch kmeans for intrusion detection system over big data, *IEEE Access*, vol. 6, pp. 11897–11906, 2018. doi: 10.1109/access.2018.2810267.

[34] Ribeiro M.T., Singh S., Guestrin C.: " Why should i trust you?" Explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016. doi: 10.1145/2939672.2939778.

[35] Sculley D.: Web-scale k-means clustering. In: *Proceedings of the 19th international conference on World wide web*, pp. 1177–1178, 2010. doi: 10.1145/1772690.1772862.

[36] Shinde P.P., Shah S.: A review of machine learning and deep learning applications. In: *2018 Fourth international conference on computing communication control and automation (ICCUBEA)*, pp. 1–6, IEEE, 2018. doi: 10.1109/iccubea.2018.8697857.

[37] Tsai C.F., Hsu Y.F., Lin C.Y., Lin W.Y.: Intrusion detection by machine learning: A review, *expert systems with applications*, vol. 36(10), pp. 11994–12000, 2009. doi: 10.1016/j.eswa.2009.05.029.

[38] Wang G., Hao J., Ma J., Huang L.: A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering, *Expert systems with applications*, vol. 37(9), pp. 6225–6232, 2010. doi: 10.1016/j.eswa.2010.02.102.

[39] Wang M., Zheng K., Yang Y., Wang X.: An explainable machine learning framework for intrusion detection systems, *IEEE Access*, vol. 8, pp. 73127–73141, 2020. doi: 10.1109/access.2020.2988359.

[40] White T.: *Hadoop: The definitive guide*, " O'Reilly Media, Inc.", 2012.

[41] Wu W., Xu S.: Intrusion detection based on dynamic gemini population DE-K-mediods clustering on hadoop platform, *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 35(01), p. 2150001, 2021.

[42] Xu Y., Qu W., Li Z., Min G., Li K., Liu Z.: Efficient k-Means++ approximation with MapReduce, *IEEE Transactions on parallel and distributed systems*, vol. 25(12), pp. 3135–3144, 2014.

[43] Younisse R., Ahmad A., Abu Al-Haija Q.: Explaining Intrusion Detection-Based Convolutional Neural Networks Using Shapley Additive Explanations (SHAP), *Big Data and Cognitive Computing*, vol. 6(4), p. 126, 2022. doi: 10.3390/bdcc6040126.

[44] Zaharia M., Xin R.S., Wendell P., Das T., Armbrust M., Dave A., Meng X., Rosen J., Venkataraman S., Franklin M.J., *et al.*: Apache spark: a unified engine for big data processing, *Communications of the ACM*, vol. 59(11), pp. 56–65, 2016. doi: 10.1145/2934664.

[45] Zhao W., Ma H., He Q.: Parallel k-means clustering based on mapreduce. In: *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*, pp. 674–679, Springer, 2009. doi: 10.1007/978-3-642-10665-1_71.

[46] Zhu W., Zeng N., Wang N., *et al.*: Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations, *NESUG proceedings: health care and life sciences, Baltimore, Maryland*, vol. 19, p. 67, 2010.

Affiliations

Chiheb Eddine Ben Ncir
University of Jeddah, College of Business, Saudi Arabia, cbenncir@uj.edu.sa

Mohamed Aymen Ben Haj Kacem
University of Tunis, LARODEC Laboratory, Tunisia, medaymenhajkacem@gmail.com

Mohammed Alattas
University of Jeddah, College of Business, Saudi Arabia, mialatas@uj.edu.sa

Received: 11.11.2023
Revised: 22.01.2024
Accepted: 22.01.2024