Hindawi
Journal of Computer Networks and Communications
Volume 2019, Article ID 4708201, 13 pages
https://doi.org/10.1155/2019/4708201

Research Article

Multistage System-Based Machine Learning Techniques for Intrusion Detection in WiFi Network

Vu Viet Thang[1] **and F. F. Pashchenko**[2,3]

[1]*Moscow Institute of Physics and Technology (State University), Moscow, Russia*
[2]*The Department of Information and Communication Technologies, MIPT (State University), Moscow, Russia*
[3]*Trapeznikov Institute of Control Sciences, Russian Academy of Sciences, Moscow, Russia*

Correspondence should be addressed to Vu Viet Thang; vv.tkhang@phystech.edu

Received 6 December 2018; Accepted 9 April 2019; Published 28 April 2019

Guest Editor: Arash H. Lashkari

The aim of machine learning is to develop algorithms that can learn from data and solve specific problems in some context as human do. This paper presents some machine learning models applied to the intrusion detection system in WiFi network. Firstly, we present an incremental semisupervised clustering based on a graph. Incremental clustering or one-pass clustering is very useful when we work with data stream or dynamic data. In fact, for traditional clustering such as K-means, Fuzzy C-Means, DBSCAN, etc., many versions of incremental clustering have been developed. However, to the best of our knowledge, there is no incremental semisupervised clustering in the literature. Secondly, by combining a K-means algorithm and a measure of local density score, we propose a fast outlier detection algorithm, named FLDS. The complexity of FLDS is $O(n^{1.5})$ while the results obtained are comparable with the algorithm LOF. Thirdly, we introduce a multistage system-based machine learning techniques for mining the intrusion detection data applied for the 802.11 WiFi network. Finally, experiments conducted on some data sets extracted from the 802.11 networks and UCI data sets show the effectiveness of our new proposed methods.

1. Introduction

Machine learning is a central problem in artificial intelligence. The purpose of machine learning is concerned with the development of algorithms and techniques that allow computers to *learn*. There are some principal kinds of machine learning such as supervised learning, unsupervised learning, and semisupervised learning. The application of machine learning techniques is very varied, for example, fault detection in bank data, transaction data, and intrusion detection system in networking, bioinformatics, natural language processing, image analysis, etc. [1]. Additionally, machine learning is very useful in cases in which human expertise does exist (robot in the Mars, in the sea, etc.), solution change in time (networking, surveillance), or solution needs to be adapted to particular cases. This paper focuses on developing machine learning techniques for intrusion detection systems in WiFi network.

Intrusion detection system (IDS) is one of the most emerging tasks in the network connectivity. Each year, there are lots of network attacks in the world; consequently, the cost for solving these problems is very big, and was reported to be about 500 billion USD in 2017. This problem is a challenge not only for government/organizations but also for individuals in daily lives. To protect the computer network system, in general, some methods can be used such as firewalls, data encryption, or user authentication. The firmware is one technique to protect the system, but nowadays, the external mechanisms have emerged and quickly become popular. One important method for data mining in intrusion detection problem proposed in the literature is to use machine learning techniques [2–8]. The IDS has monitored directly the network transactions where each transaction is either normal or malicious. The aim of IDS is to detect and alert network administrators when it detects a transaction that

is an attack. In some case, the IDS can even immediately block the connection.

Generally, data mining task in IDS must detect two kinds of attack including known attacks and outlier (anomaly) attacks. For the known attacks, we can use a (semi-)supervised learning method such as neural network, support vector machine, random forest, decision tree, and naïve Bayes, to mention a few, to construct a classifier from data training (labeled normal/attacks connection) [4–7, 9]. The classifier trained is used for detecting new connections, and the supervised learning model is illustrated in Figure 1. With the outlier attacks in which we do not know its labels, the trained classifier cannot detect them. In this case, we have to use another kind of machine learning called unsupervised outliers detection such as LOF [10], ODIN [11], and so on. The outliers detection process can be realized offline for some periods of time defined by users/experts. The general schema for outlier detection is presented in Figure 2, and this is the unsupervised learning model. The aim of this schema is to detect outliers in a period of time. For example of IDS systems, the users can set a period of time from u to v for capturing the data, then the data will be transformed by the preprocessing step, and finally, we can use an outlier detection method to detect attacks from the observed data.

The contributions of our paper are as follows:

(i) We propose an incremental semisupervised graph-based clustering. To the best of our knowledge, this is the first incremental semisupervised clustering algorithm. The preliminary work is presented in [12].

(ii) We introduce a fast outliers detection method based on local density score and K-means clustering algorithm. The preliminary work is introduced in [13].

(iii) We propose a multistage system-based machine learning techniques which can boost the accuracy of the intrusion detection process for the 802.11 WiFi data set.

(iv) The experiments carefully conducted on data set extracted from Aegean WiFi Intrusion Dataset (AWID) show the effectiveness of our proposed algorithms [14]. The AWID is a publicly available collection of sets of data which contain real traces of both the normal and intrusive 802.11 traffic. Up to date, AWID is one of the standard data sets to evaluate the capacity of IDS systems.

This paper is organized as follows. Section 2 presents the related work. Section 3 introduces the new incremental semisupervised clustering method and a new fast outlier detection algorithm. Section 4 presents experiments for the proposed algorithms and proposes a hybrid framework applied for the AWID data set. Finally, Section 5 concludes the paper and presents some direction for further research studies.

2. Incremental Clustering and Outlier Detection

2.1. Incremental Clustering. Clustering is the task of partitioning a data set into k clusters in which the points in the same cluster are similar and the points in different clusters are dissimilar. The context of incremental clustering is as follows: given some current clusters, the incremental clustering is one-pass clustering kind which aims to identify cluster label for incremental data points. Incremental clustering is very useful for data stream or dynamic data (data warehouse). In general, the incremental clustering is combined with two processes of insertion and deletion. Given a set of clusters, the insertion step aims to identify the labels of a new data point based on the current clusters. In some cases, some new clusters will be created or the new data points will be integrated with the current clusters. With the deletion process, if we want to remove one or some data points, we need to reform the clusters because some clusters may be affected by these operations. For each kind of clustering, there are some incremental clustering algorithms proposed in the literature such as Incremental K-means [15], IncrementalDBSCAN [16], or Incremental graph clustering [17]. The key idea of these algorithms is that we need to identify the situation for each kind of algorithm for the insertion step and deletion step. The incremental clustering addresses the problem of identifying the label for a new data object or updating clusters when we remove points in the current clusters. This problem is very meaningful when we tackle with the big data in which the data set is too big to fit into the available memory. For each kind of clustering, there are some versions of incremental clustering proposed in the literature.

In [16], the Incremental density-based clustering (IncrementalDBSCAN) is introduced. Based on the notion of density-based clustering, the IncrementalDBSCAN can efficiently add and delete points for the current clusters. The adding process of a new point has some cases; for example, the new point can be noise, the new point will be added in a cluster, and the new point can merge some clusters. For the deletion process, the point can be a noise point and the point can split to some clusters or not affect the current clusters. Some cases of the insertion process and deletion process of IncrementalDBSCAN are shown in Figure 3.

In [15], a single-pass incremental clustering for large data set based on K-means is introduced (named GenIC). GenIC updates each center with each new data point and merges clusters only at the end of a generation (i.e., window of data). By a generalized incremental algorithm, GenIC algorithm can move a center in the list of centers using a weighted sum of the existing center and the new point presented. The idea of GenIC is to divide the stream of data into chunks or windows as is common with streaming algorithms. We view each chunk of n data points as a generation and think of the "fitness" of a center as being measured by the number of points assigned to it. In general, the fittest centers survive to the next generation, but occasionally new centers are selected and old centers are killed off. The GenIC is compared with K-means and shown the effectiveness in running time and less affected by the choice of initial centers than K-means. In [18], a version of Incremental K-means clustering is also proposed. In the

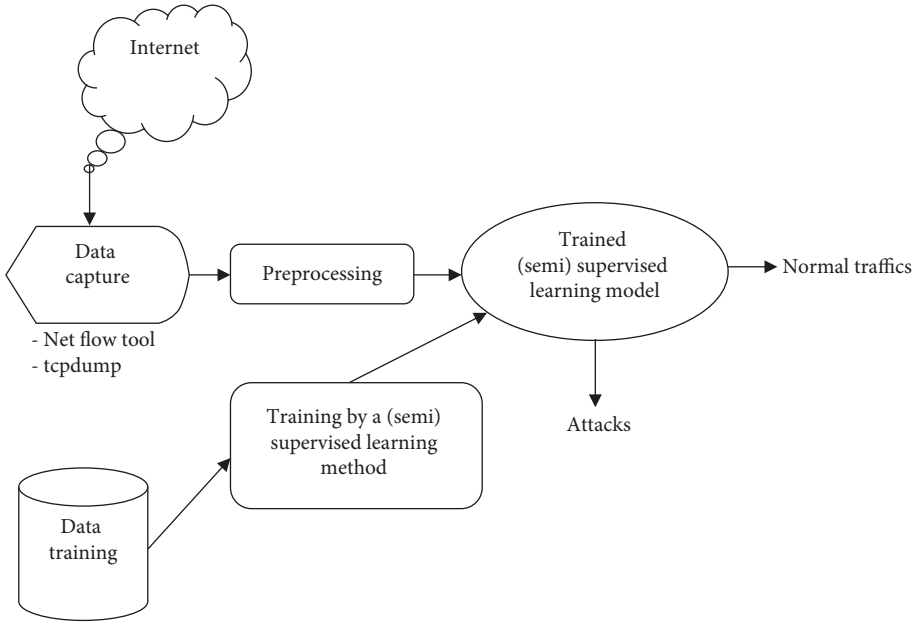


Figure 1: A general model for misuse detection in IDS.

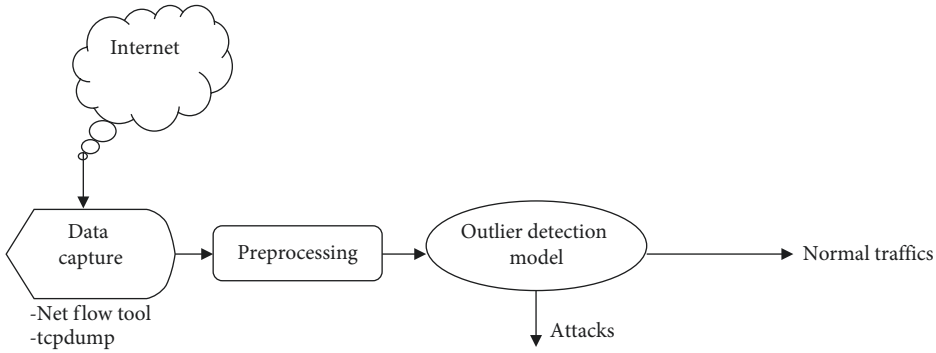


Figure 2: A general model for outlier detection in IDS.

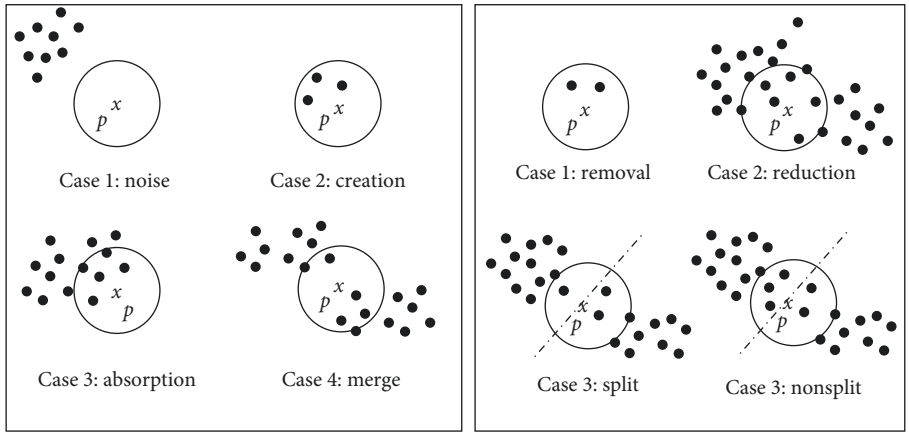


Figure 3: Insertion cases (a) and deletion cases (b) of IncrementalDBSCAN [16].

algorithm, clusters are built incrementally by adding one cluster center at a time. In [19], a novel two-phase static single-pass algorithm as well as a dynamic two-phase single-pass algorithm based on Fuzzy C-means have been presented and are showing high utility. The idea behind the multistage methods reported in the paper is that an estimate of the partition matrix and the location of the cluster centers can be obtained by clustering a sample of the data. A small sample is expected to produce a fast yet less reliable estimation of the cluster centers. This leads to a

multistage approach, which involves several stages of sampling (with replacement) of the data and estimating the membership matrix for the next stage. The experiments conducted show the effectiveness of the proposed method. In [17], Chandrasekhar et al. propose an incremental local density clustering scheme for finding dense subgraphs in streaming data, i.e., when data arrive incrementally (ILDC). The incremental clustering scheme captures redundancy in the streaming data source, by finding dense subgraphs, which correspond to salient objects and scenes. The ILDC process performs greedy operations like cluster expansion, cluster addition and cluster merging based on the similarity between clusters defined. The ILDC shows the effectiveness when using in image-retrieval applications. In [20], an incremental semisupervised ensemble clustering algorithm has successfully presented, named ISSCE. ISSCE uses constraints to update incremental members. The authors develop an incremental ensemble member selection process based on a global objective function and a local objective function to remove the redundant ensemble members. The experiment results show the improvement of ISSCE over traditional semisupervised clustering ensemble approaches or conventional cluster ensemble methods on six real-world datasets from UCI machine learning repository and 12 real-world data sets of cancer gene expression profiles. In the context of classification, we need to find the label for a new data object by using a classifier trained by data training. The problem of identifying the label for a new object in incremental clustering can be seen similar to classification context.

2.2. Outlier Detection Problem. Outlier (anomaly) detection is one of the important problems of machine learning and data mining. As mentioned in [21], outliers detection is the problem of finding patterns in data that do not conform to expected behavior. The applications of outlier detection can be found in many applications such as intrusion detection, credit fraud detection, video surveillance, weather prediction, discovery of criminal activities of electronic commerce, etc. [9, 21]. There are some kinds of outliers including point outliers, contextual outliers, and collective outliers. In this paper, we focus on point outliers detection that can be applied in a variety of applications. For a data set consisting of points, a point will be called outlier if it is different from a large number of the rest of the points. To detect outliers, there are some principal methods in the literature such as classification methods, nearest neighbor methods, clustering methods, statistical methods, distance-based methods, etc.

For the classification-based outliers detection, we have two categories: multiclass and one-class anomalies detection methods. In multiclass classification techniques, we assume that the training data contain labeled points of all normal classes. The learner using a supervised learning model trains a model using the labeled data. The classifier can distinguish between each normal class and the rest of the class. A test point will be called outlier if it does not belong to any normal class. In one-class outliers detection methods, we assume that the number of normal class is only one. The classifier learns a model that can detect the boundary of the normal class. If a test point does not fall in the boundary, it will be called outliers. Although many techniques have been done, however, the main disadvantage of these methods based on the availability of accurate labels for normal classes which is not easy to apply for real applications.

For the nearest neighbor-based outlier detection methods, we use the assumption as follows: normal points belong to the dense regions, while outliers belong to the sparse regions. The most famous method of this kind is the LOF algorithm. The idea of LOF is based on the local density evaluation score for points. Each point will be assigned a score which is the ratio of the average local density of the k-nearest neighbors of the point and the local density of the data point itself. Many variants of LOF can be cited here such as COF [22], ODIN [11], LOCI [23], etc. The main drawback of the method is the $O(n^2)$ complexity required.

For the clustering-based outliers detection techniques, the idea here is using clustering methods to group data into clusters. The points do not belong to any clusters called outliers. Some clustering methods can be detected outliers such as DBSCAN [24], SNN [25], etc. In fact, the purpose of clustering is finding clusters, so the outliers are just the product of the clustering process and hence are not carefully optimized. One more reason that can be made here is the complexity of clustering techniques required $O(n^2)$.

In the statistical outliers detection methods, these methods are based on the assumption as follows: normal data points occur in high-probability regions of a stochastic model, while anomalies occur in the low-probability regions of the stochastic model. Some methods have been done for the kind of outliers detections. In general, statistical methods fit a statistical model (Gaussian distribution, the mixture of parametric statistical distribution, etc.) to the given data and then apply a statistical inference test to determine if an unseen instance belongs to this model or not. The key limitation of these methods is the assumption about the distribution of data points. This assumption is not true, especially when the dimension of data is high [21].

In the distance-based outliers detection methods, a point is considered as outlier if it does not have enough pct% points in the data set that distance from this point is smaller than the threshold value $d_{\min}$ [26].

3. Proposed Method

3.1. Semisupervised Graph-Based Clustering. In recent years, semisupervised clustering is an important research topic that is illustrated by a number of studies introduced [27]. The purpose of semisupervised clustering is to integrate side information for improving the clustering performances. Generally, there are two kinds of side information including constraints and seeds. Given a data set X, constraints involve must-link and cannot-link in which the must-link constraint

(ML) between two observations $x \in X$ and $y \in X$ means that x and y should be in the same cluster, and the cannot-link constraint (CL) means that x and y should not be in the same cluster. With seeds, a small set of labeled data (called seeds) $S \in X$ will be provided for semisupervised clustering algorithms. In fact, this side information is available or can be collected from users [28–31]. We can cite here the work of semisupervised clustering for K-means [32], hierarchical clustering [33], graph-based clustering [34, 35], spectral clustering [36, 37], density-based clustering [38], etc. While many semisupervised clustering algorithms are introduced, to the best of our knowledge, there are no incremental semisupervised clustering algorithms in the literature.

Our new incremental clustering introduced in the next section is based on the work of semisupervised graph-based clustering using seeds (SSGC). We choose the SSGC algorithm because SSGC algorithm has several advantages such as SSGC use only one parameter and SSGC can detect clusters in varied density regions of data [35]. SSGC includes two steps as the following description (see Algorithm 1):

Step 1. Given a k-nearest neighbor graph presenting a data set X, this step uses a loop in which at each step, all edges which have the weight less than threshold θ will be removed. The value of θ is initialized by 0 at the first step and incremented by 1 after each step. This loop will stop when each connected component has at most one kind of seeds. The main clusters are identified by propagating label in each connected component that contains seeds.

Step 2. The remaining points (graph nodes) that do not belong to any main clusters will be divided into two kinds: points that have edges which relate to one or more clusters and other points which are isolated points. In the first case, points will be assigned to the cluster with the largest related weight. For the isolated points, we can either remove them as outliers or label them.

We note that, in SSGC, the weight $\omega(x_i, x_j)$ of the edge (the similarity) between two points x_i and x_j in the k-nearest neighbor graph is equal to the number of points that the two points share, as the following equation:

$$\omega\left(x_i, x_j\right) = \left|\mathrm{NN}\left(x_i\right) \cap \mathrm{NN}\left(x_j\right)\right|, \quad (1)$$

where $\mathrm{NN}(\cdot)$ is the set of k-nearest neighbors of the specified point.

SSGC is efficient when compared with the semi-supervised density-based clustering in detecting clusters for batch data; however, it is not adapted for data stream or data warehousing environment where many updates (insertion/deletion) occur.

3.2. Incremental Graph-Based Clustering Using Seeds. In this section, we propose IncrementalSSGC, based on the SSGC algorithm. In the IncrementalSSGC, the seeds will be used to train a k-nearest neighbor graph to construct connected components and identify the value of θ as in SSGC algorithm. Like other incremental clustering algorithms, two procedures must be developed, including insertion and deletion.

Algorithm 2 shows the insertion step of IncrementalSSGC for a new data point x_{new}. At first, the list of edges between x_{new} and the current clusters is created, and all edges with weight smaller than θ will be removed. If the list is empty, it is illustrated that x_{new} is an outlier with the current situation, and hence, x_{new} will be added in a temporary list Lo. In the case of existing edges between x_{new} and some connected components, we need to remove some edges until x_{new} connects to components with one kind of label. Finally, the label of x_{new} will be identified by the label of its connected components. In Step 10, x_{new} and its related edges will be added to L; some edges between x_{t} and x_{l} will also be recalculated if x_{new} appears in the nearest neighbors list of x_t or x_l. In Step 12, after some insertion steps, we can examine the points in Lo.

Algorithm 3 presents the detailed steps of the deletion process. When we want to remove a point x_{del} from the current clusters, we simply remove x_{del} and all edges related with x_{del} in the graph. Step 2 of the algorithm shows the updating process. In this step, we need to update all edges affected by x_{del}. It means that all edges between x_i and x_j must be updated if x_{del} appears in the commune list of the nearest neighbors. Finally, Step 3 is simply to remove all edges that have weight less than θ.

3.2.1. The Complexity Analysis. Now, we will analyse the complexity of IncrementalSSGC. Given a data set with n object, we recall that the complexity of SSGC is $O(k \times n^2)$, in which k is the number of nearest neighbors. Assuming that we have the current clusters including n objects, we will analyse the complexity of the insertion and deletion process of IncrementalSSGC at step $(n + 1)$ as follows.

For the insertion process which aims to identify the cluster label for a new data point x_{new}, in Step 1, to create the list of edges between x_{new} and the current clusters, the complexity is $O(n \times k)$. In Steps 2, 6, and 7, the complexity is just $O(k)$. In Step 10, some edges between x_t and x_l will also be recalculated if x_{new} appears in the nearest neighbors list of x_t or x_l; in fact, the number of such edges is also small. So for the insertion of a new point, the complexity is $O(n \times k)$.

For the deletion process, the complexity of Step 1 is $O(k)$. In Steps 2 and 3, the number of edges updated is the number of edges that received x_{del} as commune points, and the value of commune points depends on the data set. Let q be the average value of v deletion processes; in fact, q is determined by perfoming experiments. So, the complexity of a deletion process is $O(q \times n \times k)$.

In summary, with the analysis of the insertion and deletion process above, we can see that it is very useful for data set that we usually need to update. In the next section, we also present the running time of both SSGC and IncrementalSSGC for some data sets extracted from intrusion detection problem.

Input: $\mathcal{X}$, number of neighbors k, a set of seeds $\mathcal{S}$
Output: A set of detected clusters/outliers
PROCESS:
(1) Constructing the k-NN graph of $\mathcal{X}$
(2) $\theta = 0$
(3) **repeat**
(4) Constructing the connected components using the threshold θ
(5) $\theta = \theta + 1$
(6) **until** the *cut condition* is satisfied
(7) Propagating the labels to form the *principal clusters*
(8) Constructing the final clusters

Algorithm 1: The algorithm SSGC [35].

Input: a new data object x_{new}; a set of current clusters C, list containing edges for each point of current clusters L, θ (threshold), and number of nearest neighbors (NN) k.
Output: label for x_{new}
Process:
(1) Create the k-nearest neighbors list of edges (LE) between x_{new} and all current clusters
(2) Delete all $(u, v) \in \text{LE}$: $\text{weight}(u, v) < \theta$
(3) **if** (LE = $\varnothing$) **then**
(4) x_{new} is added in a temporary list Lo
(5) **else**
(6) **If** x_{new} related to two or more components with different label then
(7) Delete edges in LE with ascending order of weight until x_{new} connecting with components with at most one kind of label
(8) **end if**
(9) Get label for x_{new} and its connected points (if any) by propagating
(10) Update list L: adding edges relating to x_{new} to L; some edges between x_t and x_l will also be recalculated if x_{new} appears in the nearest neighbors list of x_t or x_l.
(11) **end if**
(12) Examine points in Lo

Algorithm 2: IncrementalSSGC: insertion process.

Input: an object x_{del} in a component will be deleted; a set of current clusters C, list of edges for each point of current clusters L, θ (threshold)
Output: the updated C, the updated L,
Process:
(1) Delete x_{del} and all edges related to x_{del} in L
(2) Update all weights $(k, l) \in C$: $x_{\text{del}} \in \text{NN}(k) \cap \text{NN}(l)$
(3) Delete all updated (at Step 2) $(k, l) \in L$: $\text{weight}(t, l) < \theta$

Algorithm 3: IncrementalSSGC: deletion process.

3.3. A Fast Outlier Detection Method. Given a k-nearest neighbors graph (k-NNG), the local density score *LDS* of a vertex $u \in k$-NNG is defined as follows [39]:

$$\text{LDS}(u) = \frac{\sum_{q \in \text{NN}(u)} \omega(u, q)}{k}, \tag{2}$$

in which ω is calculated as in equation (1), and k is the number of nearest neighbors used. The LDS is used as an indicator of the density of the region of a vertex u. The LDS value is in the interval of $[0, k-1]$; the larger the LDS of u, the denser the region that u belongs to, and vice versa. So, we can apply the way of LDS's calculation to identify outliers. To detect outlier by this method, we have to use a parameter as the threshold: the point which has LDS value smaller than the threshold can be seen as an outlier and vice versa. Similar to LOF, the method has required $O(n^2)$ of complexity.

To reduce the running time of the method, we propose a Fast outlier detection method based on Local Density Score, called FLDS. The basic idea of the algorithm FLDS is to use divide-and-conquer strategy. Given a data set X to find outliers, first, the input data set will be split into k clusters using K-means algorithm. Next, k-nearest neighbor graphs will be used for each cluster and identify outlier on each local cluster. The outliers found in all clusters will be recalculated on the whole data set. The idea of divide-and-conquer strategies by using the K-means in the preprocessing step has been successfully applied in solving some problems such as fast spectral clustering problem [40] and fast minimum spanning tree problem [41] and in the efficient and effective shape-based clustering paper [42]. The FLDS algorithm is described in Algorithm 4.

The FLDS algorithm is an outlier's detection method based on K-means and local density score using graph. The complexity of FLDS is $O(n \times k) + O(k^2) + O(t \times n)$; in which the value of k may be used up to $n^{0.5}$ [41, 42]; $t \ll n$ is evaluated approximately equal to k; so the complexity of the FLDS is $O(n^{1.5})$.

4. Experiment Results

This section aims to evaluate the effectiveness of our proposed algorithms. We will show the results of the IncrementalSSGC, the results of FLDS, and the results when using our methods for a hybrid framework for intrusion detection problem. The IncrementalSSGC will be compared with the IncrementalDBSCAN, while the FLDS will be compared with the LOF.

The data sets used in the experiments are mostly extracted from the Aegean WiFi Intrusion Dataset (AWID) [14]. AWID is a publicly available collection of sets of data in an easily distributed format, which contain real traces of both the normal and intrusive 802.11 traffic. In the AWID, many kinds of attacks have been introduced, but they also fall into four main categories including flooding, injection, and impersonation. The AWID has 156 attributes, we use 35 attributes extracted by an artificial neural network, as presented in [8]. We also use some supplement data sets that come from UCI [43] and data sets with different size, shape, and density and contain noise points as well as special artifacts [44] in this experiment.

4.1. Experiment Setup

4.1.1. Data Sets for Incremental Clustering Algorithms. To show the effectiveness of the IncrementalSSGC, two aspects will be examined including the running time and accuracy. 5 UCI data sets and 3 data sets extracted from AWID will be used for testing IncrementalSSGC and IncrementalDBSCAN. The details of these data sets are presented in Table 1.

To evaluate clustering results, the Rand Index is used. Given a data set X with n points for clustering, P_1 is an array containing the true labels, P_2 is an array containing the results of a clustering algorithm, the Rand Index (RI) is calculated as follows:

$$\mathrm{RI} = \frac{a+b}{(n!/(2!(n-2)!))}, \tag{3}$$

in which a/b is the number of pairs that are in the same/different clusters in both partitions P_1 and P_2. The bigger the Rand Index, the better the result.

4.1.2. Data Sets for FLDS and LOF. We used 5 data sets extracted from AWDI and four 2D data sets including DS1 (10000 points), DS2 (8000 points), DS3 (8000 points), and DS4 (8000 points) [44] for FLDS and LOF. These 2D data sets have clusters of different size, shape, and orientation, as well as random noise points and special artifacts. The details of these AWID data sets are presented in Table 2.

To compare LOF and FLDS for AWID data sets, we use the ROC measure that has two factors including False Positive (False Alarm) Rate (FPR) and False Negative (Miss Detection) Rate (FNR). The detail of these factors is shown in the following equations:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}}, \tag{4}$$

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}}, \tag{5}$$

in which True Positive (TP) is the number of attacks correctly classified as attack; True Negative (TN) is the number of normal correctly detected as normal; False Positive (FP) is the number of normal falsely classified as attacks, namely false alarm; and False Negative (FN) is the number of attacks falsely detected as normal.

To combine FPR and FNR values, we calculate the Half Total Error Rate (HTER) that is similar to the evaluation method used in [11], defined as follows:

$$\mathrm{HTER} = \frac{\mathrm{FPR} + \mathrm{FNR}}{2}. \tag{6}$$

4.2. Clustering Results. We note that there is no incremental semisupervised clustering algorithm in the literature. So we compare the performance obtained by our algorithm and the IncrementalDBSCAN algorithm. IncrementalDBSCAN can be seen as the state of the art among Incremental clustering proposed. The algorithm can detect clusters with different size and shape with noises. Because both SSGC and IncrementalSSGC produce the same results, we just show the results for IncrementalSSGC and IncrementalDBSCAN. The results are shown in Figure 4.

We can see from the figure that the IncrementalSSGC obtains better results compared with the IncrementalDBSCAN. It can be explained by the fact that the IncrementalDBSCAN cannot detect clusters with different densities as mentioned in the paper . . .*we assumed that the parameter values Eps and MinPts of DBSCAN do not change significantly when inserting and deleting objects*. . ..

Input: a data set X with n points, the number of nearest neighbors k, number of clusters nc, *theta*,
Output: outliers of X,
Process:
(1) Using K-means to split X into nc clusters
(2) Using LDS algorithm on each separate cluster to obtain local outliers (using the threshold *theta*)
(3) The local outliers obtained in Step 2 will be recalculated LDS's value across the data set

Algorithm 4: Algorithm FLDS.

Table 1: Main characteristics for clustering evaluation.

ID	Data	#Normal + impers.	#Attributes	#Clusters
1	Iris	150	4	3
3	Wine	178	13	3
2	*E coli*	336	8	8
4	Breast	569	30	2
5	Yeast	1484	8	10
6	AWID1	5000	35	2
7	AWID2	8000	35	2
8	AWID3	12000	35	2

Table 2: Main characteristics for FLDS and LOF.

ID	Data	#Objects	Categories
1	O-AWID1	3030	Impers., flooding, injections
3	O-AWID2	5030	Impers., normal, flooding
2	O-AWID3	7040	Flooding, normal, impers.
4	O-AWID4	10040	Normal, impers., injection, flooding
5	O-AWID5	15050	Normal, flooding, injection, and impers.

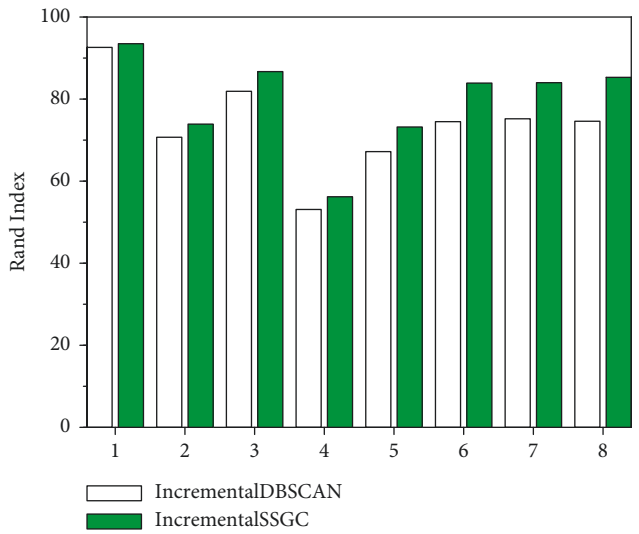


Figure 4: Clustering results obtained by IncrementalDBSCAN and IncrementalSSGC for 8 data set of Table 1, respectively.

This assumption means that the IncrementalDBSCAN cannot work well with the data set having different densities. In contrary to IncrementalDBSCAN, the algorithm IncrementalSSGC does not depend on the density of the data because the similarity measure used is based on shared nearest neighbors.

4.2.1. Running Time Comparison. Figure 5 presents the running time for IncrementalSSGC and IncrementalDBSCAN for three AWID data sets. We can see the running time of both algorithms is similar. It can be explained by the fact that both algorithms use k-nearest neighbor to find clusters for each step of incremental. We also present the running time of the SSGC algorithm for reference purpose. From this experiment, we can see advantages of the incremental clustering algorithms.

4.3. The Results of FLDS and LOF. Table 3 presents the results obtained by FLDS and LOF for 5 AWID data sets. We can see that the results of FLDS are comparable with the algorithm LOF. The parameters used for both methods are shown in Table 4. For some 2D data sets, Figure 6 presents the results obtained by FLDS and LOF. Intuitively, the outliers detected by both methods are mostly similar. We can explain the results by the fact that the strategy for evaluating a point is outlier or not based on local density score.

Figures 7 and 8 illustrate the running time comparison between FLDS and LOF. With 4 data sets mentioned above, it can be seen from the figure that the calculation time of FLDS is about 12 times faster than the running time of LOF. This is the significant improvement compared with LOF. It can be explained by the fact that the complexity of FLDS is just $O(n^{1.5})$ compared with $O(n^2)$ of LOF.

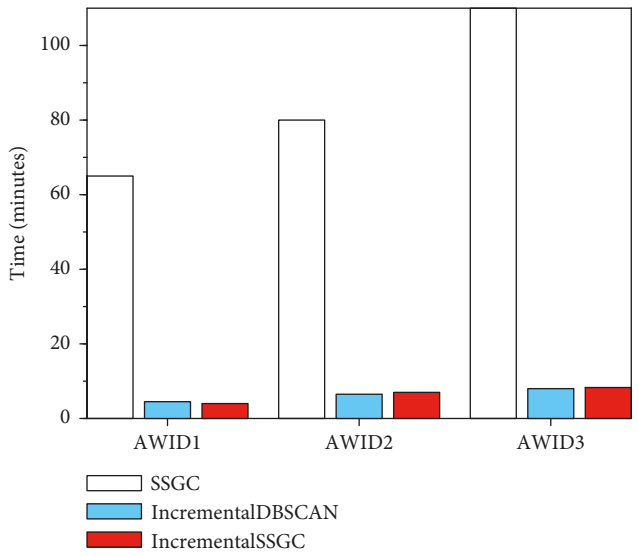


Figure 5: Running time comparison between IncrementalSSGC and IncrementalDBSCAN.

Table 3: The HTER measure of LOF and FLDS (the smaller, the better) for some extracted AWID data sets.

Methods	O-AWID1	O-AWID2	O-AWID3	O-AWID4	O-AWID5
FLDS	0.13	0.12	0.10	0.11	0.06
LOF	0.23	0.11	0.11	0.09	0.09

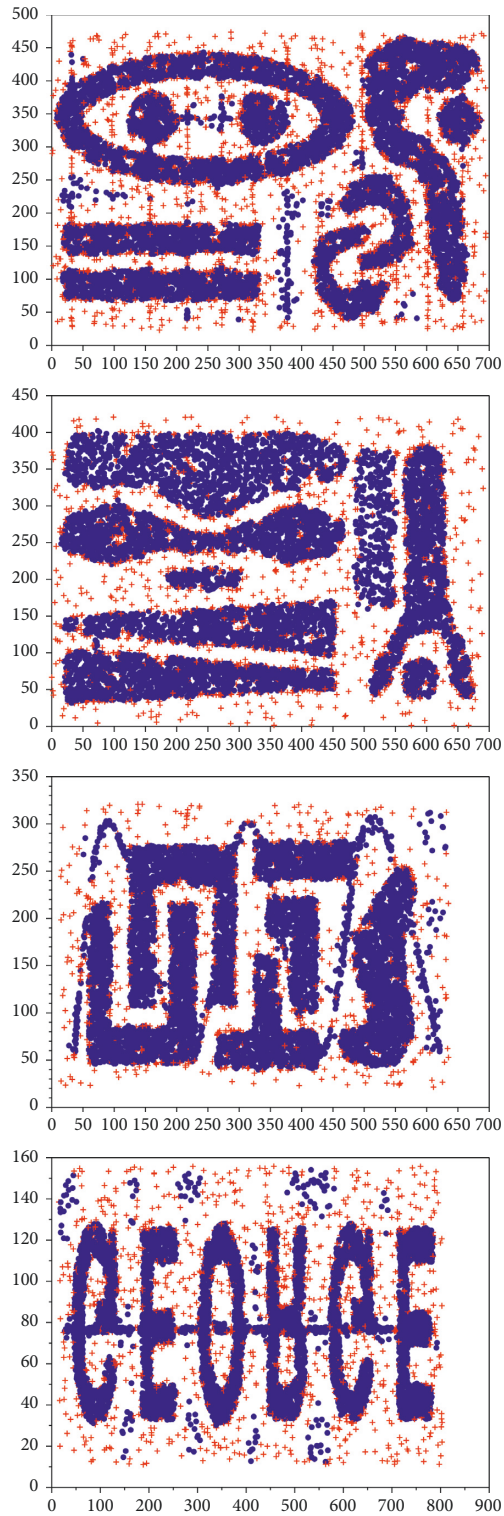
Table 4: The parameters used in data sets.

Methods	O-AWID1	O-AWID2	O-AWID3	O-AWID4	O-AWID5
FLDS (k, nc, θ)	(25, 30, 6)	(25, 30, 6)	(25, 30, 6)	(25, 45, 6)	(25, 45, 6)
LOF (MinPts, η)	(27, 1.2)	(27, 1.2)	(25, 1.2)	(25, 1.2)	(27, 1.2)

k: the number of nearest neighbors; nc: number of cluster used; θ: the threshold.

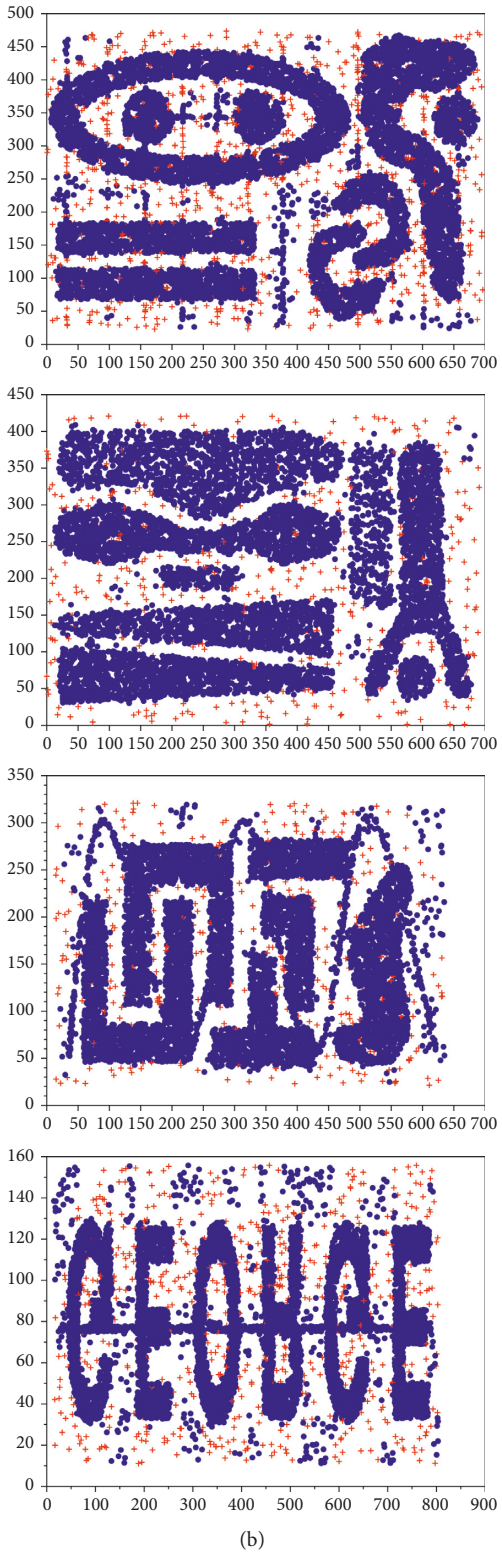
4.4. A Framework for Intrusion Detection in 802.11 Networks. In this section, we propose a multistage system-based machine learning techniques applied for the AWDI data set. The detail of our system is presented in Figure 9. Three components are used for intrusion detection task: a supervised learning model (J48, Bayes, random forest, support vector machine, neural network, etc.) trained by labeled data set, and this model can be seen as misuse detection component; an outlier detection method (LOF, FLDS, etc.) is optionally used to detect new attacks in some periods of time; additionally, for the AWID data sets as presented above, it is very difficult to detect impersonation attacks, so we use an Incremental clustering algorithm (IncrementalDBSCAN, IncrementalSSGC, etc.) for further finding this kind of attack.

In this experiment, we use J48 for the misuse detection process and IncrementalSSGC for the detecting impersonation attacks. In the outliers detection step, we propose to use FLDS or LOF, and the results have been presented in the subsection above. Because the outliers detection step can be realized offline for some periods of time, we just show the results obtained by combining J48 and IncrementalSSGC. The confusion matrix of



(a)

Figure 6: Continued.



(b)

Figure 6: Results of LOF (a) and FLDS (b) on some 2D data sets: the outliers marked as red plus.

these results is illustrated in Table 5. The total accuracy obtained 98.9% compared with 96.26% in the paper [14]. We can explain the results obtained by IncrementalSSGC by the fact that the algorithm used the distance based on shared nearest

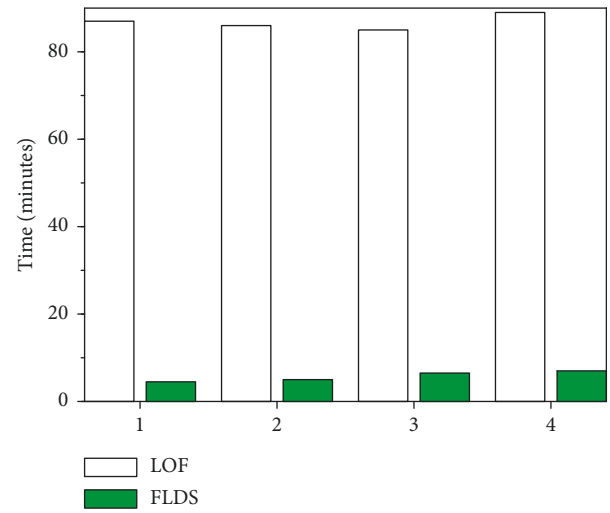


Figure 7: Running time comparison between FLDS and LOF for four 2D data sets.

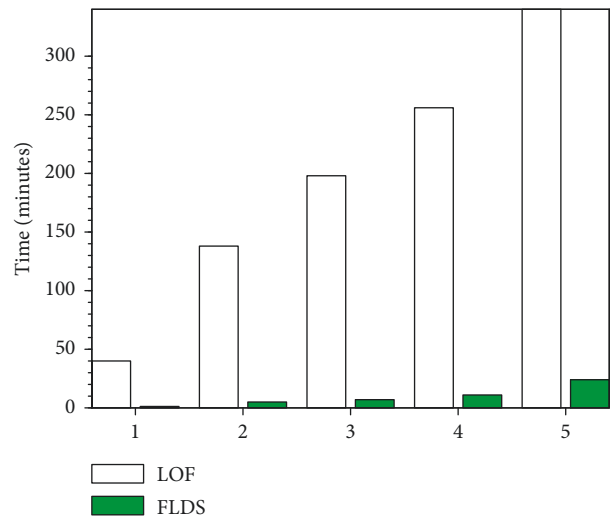


Figure 8: Running time comparison between FLDS and LOF for five AWID data sets.

neighbors, which overcome the limit of transitional distance measures such as Euclidean or Minskovki distance, and the shared nearest neighbors measure does not depend on the density of data. This proposed system is generally called hybrid method which is one of the best strategies in developing Intrusion Detection Systems [7, 9] in which there is no single classifier that can exactly detect all kinds of classes.

We also note that for real applications, whenever an attack appears, the system needs to immediately produce a warning. The multistage system-based machine learning techniques provide a solution for users for constructing the real IDS/IPS system that is one of the most important problems in the network security.

5. Conclusion

This paper introduces an incremental semisupervised graph-based clustering and a fast outlier detection

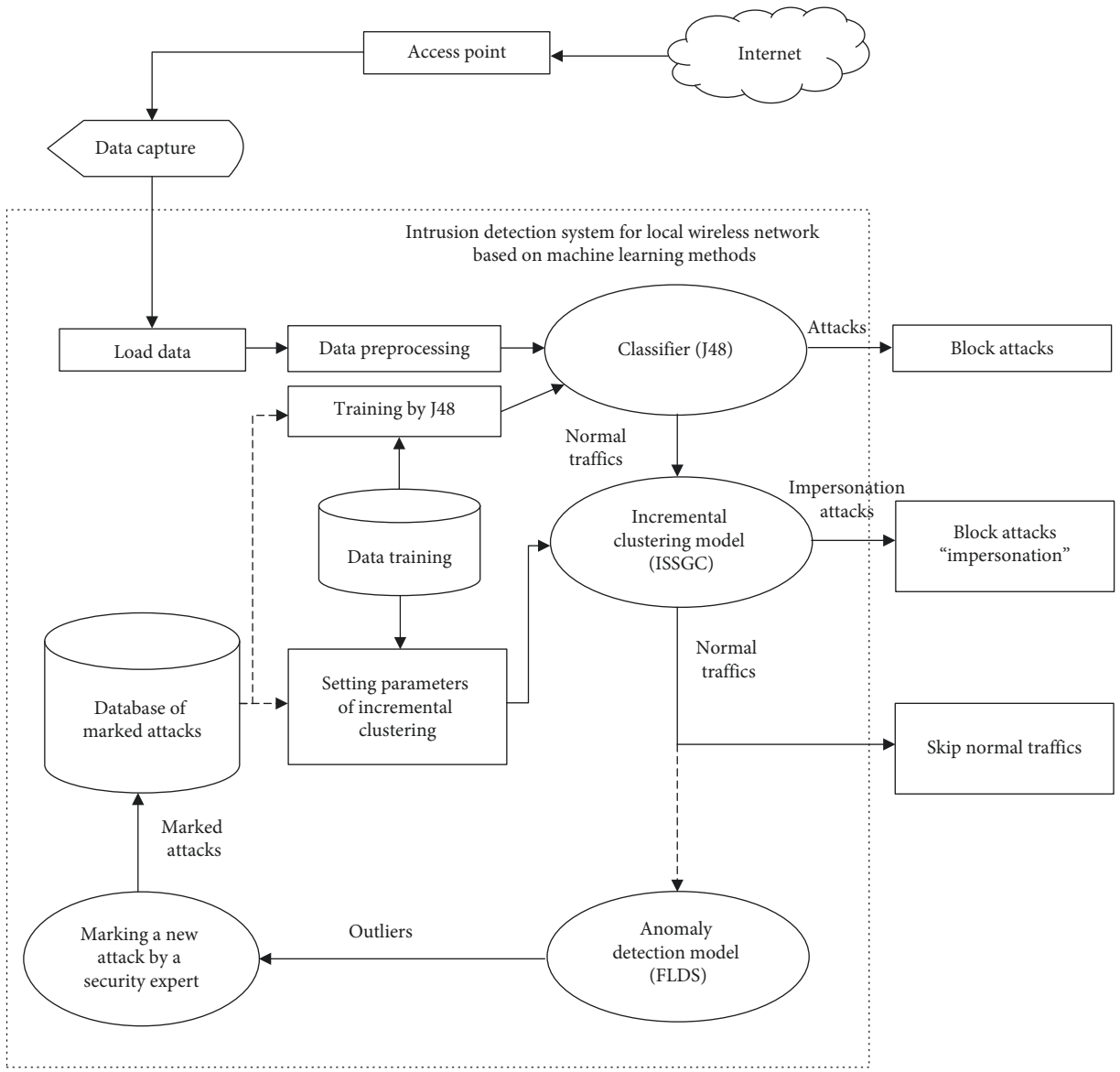


Figure 9: A new framework for intrusion detection in 802.11 networks.

Table 5: Confusion matrix for AWID data set using J48 and IncrementalSSGC in proposed framework.

Normal	Flooding	Impersonation	Injection	Classification
530588	116	6	75	Normal
2553	5544	0	0	Flooding
2	0	16680	0	Injection
3297	148	0	16364	Impersonation

method. Both methods can be used in a hybrid framework for the intrusion detection problem of WiFi data sets (AWID). Our proposed multistage system-based machine learning techniques provide a solution to guideline for constructing the real IDS/IPS system that is one of the most important problems in the network security. Experiments conducted on the extracted data sets from the AWID and UCI show the effectiveness of our proposed methods. In the near future, we will continue to develop other kinds of machine learning methods for intrusion detection problem and test for other experimental setup.

Data Availability

The data used to support the findings of this study can be downloaded from the AWID repository (http://icsdweb.aegean.gr/awid/download.html).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*, MIT press, Cambridge, MA, USA, 2012.

[2] B. Ahmad, W. Jian, and Z. Anwar Ali, "Role of machine learning and data mining in internet security: standing state with future directions," *Journal of Computer Networks and Communications*, vol. 2018, Article ID 6383145, 10 pages, 2018.

[3] T. Bakhshi and B. Ghita, "On internet traffic classification: a two-phased machine learning approach," *Journal of Computer Networks and Communications*, vol. 2016, Article ID 2048302, 21 pages, 2016.

[4] B. Luo and J. Xia, "A novel intrusion detection system based on feature generation with visualization strategy," *Expert Systems with Applications*, vol. 41, no. 9, pp. 4139–4147, 2014.

[5] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: an intrusion detection system based on combining cluster centers and nearest neighbors," *Knowledge-Based Systems*, vol. 78, pp. 13–21, 2015.

[6] C.-F. Tsai and C.-Y. Lin, "A triangle area based nearest neighbors approach to intrusion detection," *Pattern Recognition*, vol. 43, no. 1, pp. 222–229, 2010.

[7] F. Kuang, S. Zhang, Z. Jin, and W. Xu, "A novel SVM by combining kernel principal component analysis and improved chaotic particle swarm optimization for intrusion detection," *Soft Computing*, vol. 19, no. 5, pp. 1187–1199, 2015.

[8] M. E. Aminanto and K. Kim, "Detecting impersonation attack in WiFi networks using deep learning approach," in *Information Security Applications*, D. Choi and S. Guilley, Eds., Vol. 10144, Springer, Berlin, Germany, 2016.

[9] A. A. Aburomman and M. B.I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Applied Soft Computing*, vol. 38, pp. 360–372, 2016.

[10] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 93–104, Dallas, TX, USA, May 2000.

[11] V. Hautamäki, I. Kärkkäinen, and P. Fränti, "Outlier detection using *k*-nearest neighbour graph," in *Proceedings of the 17th International Conference on Pattern Recognition*, pp. 430–433, Cambridge, MA, USA, August 2004.

[12] V. Thang and F. F. Pashchenko, "A new incremental semi-supervised graph based clustering," in *Proceedings of the IEEE International Conference on Engineering and Telecommunication*, Moscow, Russia, March 2018.

[13] V. Thang, D. V. Pantiukhin, and A. N. Nazarov, "FLDS: fast outlier detection based on local density score," in *Proceedings of the IEEE International Conference on Engineering and Telecommunication*, pp. 137–141, Moscow, Russia, November 2016.

[14] C. Kolias, G. Kambourakis, A. Stavrou, and S. Gritzalis, "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 184–208, 2016.

[15] Ch. Gupta and R. Grossman, "GenIc:a single pass generalized incremental algorithm for clustering," in *Proceedings of the Fourth SIAM International Conference on Data Mining*, pp. 147–153, Lake Buena Vista, FL, USA, April 2004.

[16] M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, and X. Xu, "Incremental clustering for mining in a data warehousing environment," in *Proceedings of the International Conference on Very Large Data Bases*, pp. 323–333, New York, NY, USA, August 1998.

[17] V. Chandrasekhar, C. Tan, M. Wu, L. Li, X. Li, and J.-H. Lim, "Incremental graph clustering for efficient retrieval from streaming egocentric video data," in *Proceedings of the International Conference on Pattern Recognition*, pp. 2631–2636, Stockholm, Sweden, August 2014.

[18] A. M. Bagirov, J. Ugon, and D. Webb, "Fast modified global K-means algorithm for incremental cluster construction," *Pattern Recognition*, vol. 44, no. 4, pp. 866–876, 2011.

[19] A. Bryant, D. E. Tamir, N. D. Rishe, and K. Abraham, "Dynamic incremental fuzzy C-means clustering," in *Proceedings of the Sixth International Conference on Pervasive Patterns and Applications*, Venice, Italy, May 2014.

[20] Z. Yu, P. Luo, J. You et al., "Incremental semi-supervised clustering ensemble for high dimensional data clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 3, pp. 701–714, 2016.

[21] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: a survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[22] J. Tang, Z. Chen, A. Fu, and D. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Proceedings of the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, Taipei, Taiwan, May 2002.

[23] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos, "Loci: fast outlier detection using the local correlation integral," in *Proceedings of the 19th International Conference on Data Engineering*, pp. 315–326, Bangalore, India, March 2003.

[24] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 226–231, Portland, OR, USA, August 1996.

[25] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *Proceedings of the SIAM International Conference on Data Mining*, pp. 47–58, San Francisco, CA, USA, May 2003.

[26] E. M. Knorr, R. T. Ng, and V. Tucakov, "Distance-based outliers: algorithms and applications," *The International Journal on Very Large Data Bases*, vol. 8, no. 3-4, pp. 237–253, 2000.

[27] S. Basu, I. Davidson, and K. L. Wagstaff, "Constrained clustering: advances in algorithms, theory, and applications," in *Chapman and Hall/CRC Data Mining and Knowledge Discovery Series*, , CRC Press, Boca Raton, FL, USA, 1st edition, 2008.

[28] A. A. Abin, "Clustering with side information: further efforts to improve efficiency," *Pattern Recognition Letters*, vol. 84, pp. 252–258, 2016.

[29] Y. Shi, C. Otto, and A. K. Jain, "Face clustering: representation and pairwise constraints," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1626–1640, 2018.

[30] A. A. Abin and B. Hamid, "Active constrained fuzzy clustering: a multiple kernels learning approach," *Pattern Recognition*, vol. 48, no. 3, pp. 953–967, 2015.

[31] S. Xiong, J. Azimi, and X. Z. Fern, "Active learning of constraints for semi-supervised clustering," *IEEE Transactions on*

Knowledge and Data Engineering, vol. 26, no. 1, pp. 43–54, 2014.

[32] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl, "Constrained K-means clustering with background knowledge," in *Proceedings of the International Conference on Machine Learning (ICML)*, pp. 577–584, Williamstown, MA, USA, June 2001.

[33] I. Davidson and S. S. Ravi, "Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results," *Data Mining and Knowledge Discovery*, vol. 18, no. 2, pp. 257–282, 2009.

[34] B. Kulis, S. Basu, I. Dhillon, and R. Mooney, "Semi-supervised graph clustering: a kernel approach," *Machine Learning*, vol. 74, no. 1, pp. 1–22, 2009.

[35] V.-V. Vu, "An efficient semi-supervised graph based clustering," *Intelligent Data Analysis*, vol. 22, no. 2, pp. 297–307, 2018.

[36] X. Wang, B. Qian, and I. Davidson, "On constrained spectral clustering and its applications," *Data Mining and Knowledge Discovery*, vol. 28, no. 1, pp. 1–30, 2014.

[37] D. Mavroeidis, "Accelerating spectral clustering with partial supervision," *Data Mining and Knowledge Discovery*, vol. 21, no. 2, pp. 241–258, 2010.

[38] L. Lelis and J. Sander, "Semi-supervised density-based clustering," in *Proceeding of IEEE International Conference on Data Mining*, pp. 842–847, Miami, FL, USA, December 2009.

[39] D.-D. Le and S. Satoh, "Unsupervised face annotation by mining the web," in *Proceedings of the 8th IEEE International Conference on Data Mining*, pp. 383–392, Pisa, Italy, December 2008.

[40] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *Proceedings of the Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 907–916, New York, NY, USA, June 2009.

[41] C. Zhong, M. Malinen, D. Miao, and P. Fränti, "A fast minimum spanning tree algorithm based on K-means," *Information Sciences*, vol. 295, pp. 1–17, 2015.

[42] V. Chaoji, M. Al Hasan, S. Salem, and M. J. Zaki, "SPARCL: an effective and efficient algorithm for mining arbitrary shape-based clusters," *Knowledge and Information Systems*, vol. 21, no. 2, pp. 201–229, 2009.

[43] A. Asuncion and D. J. Newman, *UCI Machine Learning Repository*, American Statistical Association, Boston, MA, USA, 2015, http://archive.ics.uci.edu/ml/index.php.

[44] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, vol. 32, no. 8, pp. 68–75, 1999.