Relative Network Entropy based Clustering Algorithm for Intrusion Detection

Quan Qian, Tianhong Wang, and Rui Zhan
(Corresponding author: Quan Qian)

School of Computer Engineering & Science, Shanghai University, Shanghai, 200072 China
State Key Laboratory of Information Security (Institute of Software, Chinese Academy of Sciences)
(Email: qqian@shu.edu.cn)

(Received Jan. 25, 2011; revised and accepted Oct. 17, 2011)

Abstract

Clustering, as a kind of data mining methods, with the characteristic of no supervising, quick modeling is widely used in intrusion detection. However, most of the traditional clustering algorithms use a single data point as a calculating unit, and the drawback exists in time wasting to calculate one data point after another when clustering, meanwhile, a single local change of data will significantly affect the clustering results. This paper proposes a novel clustering algorithm named EB-DBSCAN, a data mining algorithm based on relative network entropy. EB-DBSCAN use the batch data processing method which can cluster quickly, accurately and unsupervised for high-speed and massive network data stream with arbitrary shape. Experimental results show that EB-DBSCAN can achieve roughly the same average purity and average precision as DBSCAN. Moreover, concerning the number of clusters and execution time, EB-DBSCAN performs much better than DBSCAN, making both performance increased by an average of 1.5 times and 190 times more, which shows a prosperous potentiality for high speed network traffic analysis.

Keywords: Data clustering, intrusion detection, relative network entropy

1 Introduction

With the widespread use of the Internet, a variety of attacks, such as Denial of Service(DoS), Spam, Probing and so forth., emerge one after another every day. Intrusion detection system (IDS) as an effective method to protect against malicious attacks has been widely used and attracts many research interests [7]. Nowadays, intrusion detection researches are mainly focused on the following aspects: Adapting IDS to the high-speed and real-time network environments; distributed intrusion detection; innovative detection method, such as data mining, machine learning, collaborative research of intrusion detection and other security technologies; intrusion response technologies, etc. Among them, intrusion detection technology based on data mining is an attractive research direction. Two typical intrusion detection systems using data mining are MADAM ID [9,10], and ADAM [15], both of them use the audit data to detect the system abnormal behavior. Concerning data mining methods, those can be divided into the following categories: association rules, frequent sequences, clustering method, classification method and etc. Among them, clustering methods, such as K-means, CURE [3], etc., are widely used in intrusion detection because of the characteristic of their non-supervision and quick-building-model advantages.

In general, data mining algorithm can be divided into partitioning method, density method, hierarchical method, grid mapping method, and so forth [1, 6]. Among them, DBSCAN [2] based on density connectivity, overcoming the strict requirements of the clustering shape required by other algorithms, can be used to cluster data of any shape. The clustering algorithm EB-DBSCAN (Entropy-Based DBSCAN) proposed in this paper is improved based on DBSCAN algorithm. Considering that the time efficiency of DBSCAN is not very prominent, so in the process of clustering, the concept of "data window" is introduced to reduce the computation scale. Meanwhile, as most of the attack behavior is that abnormal data increase sharply within a certain period of time, so we use relative entropy, which aims to exclude an individual anomaly data packets (which can be considered as noise) interfering the entire detection system, and to make the efficiency of the algorithm significantly increased.

This paper is organized as follows: the second section describes the basic concepts of the network entropy, relative network entropy and their possible application in clustering. The third section describes the relative network entropy based DBSCAN algorithm, namely EB-DBSCAN. The forth part is the clustering results of the experiment. The last part draws a conclusion of the whole paper.

2 Network Entropy and Relative Network Entropy

2.1 Network Entropy

Entropy is a very important concept in information theory, which can be used to reflect the uncertainty of systems. From Shannon's theory, that information is the eliminating or reducing of people understanding the uncertainty of things [13]. He calls the degree of uncertainty as entropy. Supposing a discrete random variable X, which has $x_1, x_2, ..., x_n$, a total of n different values, the probability of x_i appears in the sample is defined as P(x_i), then the entropy of random variable X is:

$$\mathrm{H(P)} = -\sum_{i=1}^{n} P(x_i)\log P(x_i) \qquad (1)$$

Entropy value ranges between 0 and 1. If $H(P) \to 0$ (means close to 0), it indicates the lower level of uncertainty, and the higher similarity in the sample. On the other hand, if $H(P) \to 1$, it indicates the higher level of uncertainty, the lower similarity in the sample. For instance, in the real network environment, for a particular type of network attack, the data packets show a certain kind of characteristics. For example, DoS attacks, the data packets sent in a period of time are quite more similar in comparison to the normal network packets, which show smaller entropy, that is, the lower randomness. Another example is a network probing attack, which scans frequently a specific port in a certain period of time, so the destination ports will get smaller entropy compared with the random port selection of normal packets.

The paper [4] has put forward a method to use Maximum Entropy Model(MEM) to evaluate the uncertainty degree of some attributes, which can recognize the anomaly data from the normal one to some extent. However, that method cannot indicate each attribute's importance in a whole. Another problem in MEM is that two packets of different uncertainty degrees may have the same network entropy, so the network entropy cannot be used as an indicator of the dissimilarity between two different data packets. For this reason, paper [11] introduces the concept of relative entropy. In the next section, we will focus on relative network entropy and how we use relative network entropy for data packet distance calculation in EB-DBSCAN algorithm practically.

2.2 Relative Network Entropy

Euclidean distance formula is often used to calculate the distance between two data points in clustering algorithm. Supposing that there are two n-dimensional data points $P(x_1, x_2, ..., x_n)$, $Q(y_1, y_2, ..., y_n)$, the Euclidean distance formula is defined as follows:

$$Dist(P,Q) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (2)$$

Using the Euclidean distance formula to calculate the distance between two data points has the following four disadvantages: (1)Requiring continuous variables for calculation and discrete variables need to be continuation; (2) If there exists some features with particular greater or smaller value, it will bring great influence on the overall results, those features need to be normalized; (3)Using each single data packet as computational unit to evaluate the network condition will cause the IDS very sensitive, for example the high false alarm rate of the IDS. (4)Using a single data packet as computational unit will bring large amount of calculation, high complexity and poor real-time performance.

Relative Network Entropy, or called the Kullback-Leibler (KL) divergence [11] reflects the degree of similarity between two data packets in network, which can be used as an evaluation factor to measure the distance between two data packets. Its advantages include: (1)Can reflect the similarity among the arriving data in a certain time window of arriving data; (2) Using batch data processing method, can reflect the real network status in a period of time, avoid the frequent alarming issues of the current IDS to some extent.

In the next section, when calculating the distance between two data points, relative network entropy can be used as an indicator to measure the distance between two data points. And the relative network entropy is defined as follows:

Supposing there are two data packets *P* and *Q* with the same feature space $CF1(\omega_1, \omega_2, ..., \omega_n)$, $CF2(\omega_1, \omega_2, ..., \omega_n)$, for any one-dimension-property ω_i has $x_1, x_2, ..., x_n \in \omega_i$, with *m* possible values, then for a given dimension P_i and Q_i, the relative network entropy $D(P_i \| Q_i)$ is defined as:

$$D(P_i \| Q_i) = \sum_{k=1}^{m} P(x_k)\log\frac{P(x_k)}{Q(x_k)} \qquad (3)$$

In Equation (3), P_i and Q_i both stand for *i-th* dimensional property, x_k stands for the *k-th* possible value of the *i-th* dimensional property. As for *n*-dimensional property *P* and *Q*, the relative network entropy is defined as:

$$D(P \| Q) = \frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{m} P_j(x_i)\log\frac{P_j(x_i)}{Q_j(x_i)} \qquad (4)$$

What need to be noted are:

- if $\exists k$, satisfy with $x_k \in Q_i$ and $x_k \notin P_i$, then we have $P(x_k)\log\frac{P(x_k)}{Q(x_k)} = 0$
- if $\exists k$, satisfy with $x_k \notin Q_i$ and $x_k \in P_i$, then set $Q(x_k) = 0.001 \approx 0$

3 EB-DBSCAN Clustering Algorithm

3.1 DBSCAN Algorithm

DBSCAN algorithm uses the density connectivity to build a cluster with arbitrary shape. The core ideas of the algorithm are as follows:

The object that has lots of other objects aggregated around contains the number of objects larger than the given minimum number (*MinPts*) in its given radius(*EPS*). In such a condition, we usually call an object as a point. First of all, get a point from the object set. If the point is a core point (that is its neighborhood with radius EPS contains points not less than the minimum number *Minpts*), then find all the density reachable points extended from that core point and make them with the same category. If the point is not a core point, classified the point as a noise temporarily.

Several different and improved algorithms are proposed on the basis of the DBSCAN algorithm. As the DBSCAN algorithm need to find the density reachable points by repetitive region query, increases the time complexity heavily. So Zhou, etc. proposed a fast clustering method based on density called FDBSCAN [16], whose core idea is to select objects from the neighborhood of the core point as representative seed objects, and to reduce the number of candidate objects. Wang etc. focused on FDBSCAN defects, if some objects can only be achievable through the ignored core object *P*, when the cluster *C* including object *P* has extended, then those objects will not be included in the cluster *C*. Some objects have been lost. So they proposed an improved fast clustering algorithm IF-DBSCAN [14], which can effectively resolve the problem of object loss. Paper [5] proposed PDBSCAN algorithm using MPI to design parallel programs, which can effectively utilize system resources and improve the algorithm parallelization to reduce the algorithm execution time. Although, there have been some extension research on DBSCAN related area, those methods also exits some limitations and the improvement of time complexity are quite limited. So in this paper we propose EB-DBSCAN algorithm especially for the network environment, which can effectively reduce algorithm's time complexity.

3.2 Experimental Algorithm Description

In this paper, considering the advantage that DBSCAN can find clusters with arbitrary shape and the relative network entropy can reflect characteristics of general chaos in a data block, we improve the DBSCAN algorithm in combination with the relative network entropy as the computing model. EB-DBSCAN algorithm first reads data from data sets in every "*WindowSize*" as a data-processing point, retaining the attribute distribution of each dimension, and then continues to find the direct density reachable points using relative network entropy as an indicator to measure the distance between two data points. Finally, as to those noise points, the processing principle is that the points that have smaller relative network entropy value have the higher similarity, so that we compare each noise point with normal points and classify each noise point into the cluster that the closest normal point belongs to. EB-DBSCAN algorithm has a great prospect in clustering of high-speed and massive data stream of arbitrary shape.

The paper [15] also points out that in dealing with the attribute, different attributes contribute to the relative entropy differently. So we can set each dimension attributes dimension with different weight in order to get optimal clustering results. The framework of the specific algorithm is described as follows.

EB-DBSCAN algorithm is mainly constructed on the basis of DBSCAN algorithm with the concept of the data window, reducing the size of data computation scale. So with reference to the time complexity of DBSCAN [2], EB-DBSCAN's time complexity is $O[(n/m)]^2$ (*n* stands for the size of data and *m* stands for the size of data window).

Algorithm 1: *EB-DBSCAN* (*EPS, Minpts*)

```
//EPS is radius，Minpts is the given minimum number of
  the neighbor points
1: Begin
2: Read data. Create data sets (SetOfPoints) according to
   the size of data window("WindowSize"), that is, every
   "WindowSize "data regards as a processing unit.
3: Initialize the default cluster ID of all points in
   SetOfPoints equals to 0; the current status of all points
   marked as unclassified;
4: Set the temporary variable ClusterID=1;
5: Read the point called P in SetOfPoints one after another;
   if P has not been classified, then goto (6);
      if all data points in SetOfPoints have been processed,
        then
           goto (7); else continue to read and process the
           data point P in SetOfPoints.
6: if call the function ExpandCluster successfully,
   then
      ClusterID++; goto (5)
7: Noisy points processing: Comparing the noisy points in
   SetOfPoints with the classified points, and put each
   noisy point into the cluster that the closest non-noisy
   point belongs to.
8: End
```

Algorithm 2: *ExpandCluster* (*SetOfPoints, Point P, ClusterID, EPS, Minpts* **)**

```
//SetOfPoints is the set of data point, P is the current
  processing point; ClusterID is the current cluster ID;
  EPS is radius; Minpts is the given minimum number of
  points.
1: Begin
2: Obtain the direct density reachable set of points from
   point P (called set L);
3: if the size of L is lower than Minpts,
   then
      mark point P as noise temporarily and return failure;
   else
      goto (4);
4: Set the cluster ID of the points in L to ClusterID;
5: Process the point called PT in L one after another;
   if all the points have been processed,
     then
       goto (9);
6: Get the direct density reachable set of points from point
   PT(called set Pt_Neighbour);
   if the size of Pt_Negighbour is greater than Minpts,
     then
       goto(7),
   else goto (5);
7: Process the point called PP in Pt_Neighbour one after
   another;
   if all the points have been processed,
     then
      goto (5)
8: if the status of PP is unclassified or regarded as a noise,
    then
```

```
        set the cluster ID of $PP$ with $ClusterID$;
    **if** the status of $PP$ is *unclassified*,
      **then**
         add $PP$ into $L$; goto(7);
9: Expansion success, Return.
10: End
```

4 Experimental Results and Analysis

4.1 Data Pre-processing and Evaluation Index

Experimental machine's CPU is Intel Xeon 3330(2.66GHz), 2G memory size and use Linux Centos 5.4 as the operating system with java for programming.

The selected data set is 10% of the kdd-cup99 [8], Which is the well known public competition data set for IDS so far, approximately including 49 million records. Each Kdd-cup99 data contains 41 attributes and the experiment selects the most critical nine attributes: *duration*, *protocol_type*, *service*, *flag*, *src_bytes*, *dst_bytes*, *count*, *srv_count and dst_host_count*. For details of feature selection can refer to [12]. Kdd-cup99 totally includes four categories of attacks: DoS, U2R, R2L, PROBE and a normal data labeled NORMAL. Table 1 shows the details of the 4 different attack categories. The experiments randomly select four attacks' data and the normal data sets to bulid four data sets: DataSet1, DataSet2, DataSet3, DataSet4. In kdd-cup99, due to the proportion of NORMAL, DoS, PROBE is larger than U2R and R2L, so we put more NORMAL, DOS, PROBE than U2R and R2L in constructing the data set and we especially increase the data confusion in DataSet4 by injecting a single data attack type in a data window. Table 2 shows the distribution of different data sets.

Table 1: The Attack details of KDD-CUP99 data

Attack Type	***Attack Name***
DoS	back land neptune pod smurf teardrop
U2R	buffer_overflow loadmodule perl rootkit
R2L	ftp_write guess_passwd imap multihop phf spy warezclient warezmaster
PROBE	ipsweep nmap protsweep satan

Table 2: Distribution details of 4 data sets

	Dataset1	***Dataset2***	***Dataset3***	***Dataset4***
NORMAL	5471	13094	32595	41179
DOS	1904	2899	12459	49689
U2R	639	1470	1471	2036
R2L	707	1154	1198	1713
PROBE	1274	1381	2272	4378
Total Records	10000	20000	50000	100000

As to network attacks, a single abnormal packet is hard to be accurately identified as an attack, only if the sum of abnormal packets reaching such a certain number that can be identified as an attack. So this paper uses the concept of the data window, while the parameter "*WindowSize*" is used to describe the size of a data window. Since EB-DBSCAN is demonstrated by comparing with the DBSCAN, DBSCAN algorithm uses the Euclidean distance formula requiring continuous data values, so for discrete attributes, it is necessary to convert these discrete attributes to continuous ones.

The method we adopt to convert discrete values is encoding. For Example, in KDDCUP99 data, the protocol-type attribute has three values: *tcp*, *udp* and *icmp*, encoded as 001, 010 and 100. For other discrete attributes, we adopt the similar approach, respectively.

Meanwhile, for an attribute if exists great difference between the maximum and minimum values that will lead to the deviation of results, it's necessary to standardize the data. Specific method is as follows:

Calculating the average absolute deviation S_f:

$$S_f = \frac{1}{N}\sum_{i=1}^{n}\left|X_{if} - m_f\right|;$$

$$m_f = \frac{1}{N}\sum_{i=1}^{n} X_{if} \;;$$

$$X_{if} \in (X_{1f}, X_{2f}, \ldots X_{nf}) \qquad (5)$$

Where mf stands for the average value of certain attribute; X_{if} stands for the *i-th* value of all attribute.

Standardization for the continuous attributes is:

$$Z_{if} = \frac{X_{if} - m_f}{S_f} \qquad (6)$$

Because the clustering results may directly affect the results of intrusion detection, so it is necessary to evaluate the quality of the clustering.

According to the EB-DBSCAN algorithm in this experiment, the criteria of clustering evaluation we use are the following four: clustering time, number of clusters, the average cluster purity and the average clustering precision.

In the experiment, we define the cluster category is the same as the attack category that most objects belong to. For example, supposing for cluster *C*, most objects in *C* belong to category U2R, then we can say cluster *C* belongs to U2R.

(1) The average purity is defined as: in each cluster, the ratio percentage of the number of records with the maximum category and the total number of records of all categories. Compute each cluster's percentage and sum of them, and then divide by total number of clusters, the result is the average purity. The computation equation is as follows:

$$AvgPurity = \frac{\sum_{i=1}^{N}\frac{|C_i^m|}{|C_i|}}{N} * 100\% \qquad (7)$$

Where $|C_i^m|$ stands for the size of category *m* in cluster *i*, $|C_i|$ the size of the cluster *i*, and *N* the number of clusters.

(2) The average precision is defined as: For a particular cluster, the cluster's category is the category that most objects in it belong to. For those clusters with the same category, sum the records with the maximum category, and then divide by the total records of the category. Compute each category's ratio and sum of the percentage, and then divide by the total number of categories; we can get the average precision of the clustering. The computation equation is:

$$Avg\,Precision = \frac{\sum_{i=1}^{K} \frac{\sum_{j=1}^{N} |T_i^j|}{|T_i|}}{K} * 100\% \qquad (8)$$

Where $|T_i^j|$ stands for the size of category i in cluster j, $|T_i|$ the size of category i, N the number of clusters and K the number of categories.

4.2 Experiment of Average Purity and Average Precision

In the following experiment, we manually set *EPS*, *Minpts* and *WindowSize*. The parameters are shown in Table 3.

Table 3: Parameters of DBSCAN and EB-DBSCAN Algorithms

Dataset	**EB-DBSCAN**			**DBSCAN**	
	EPS	***Minpts***	***WindowSize***	***EPS***	***Minpts***
DataSet1	2.192	4	50	1.410	4
DataSet2	1.530				
DataSet3	1.550				
DataSet4	1.350				

Figure 1 shows the average purity comparison of DBSCAN and EB-DBSCAN using three data sets, DataSet1, DataSet2, and DataSet3. Figure 2 shows the average precision through averaging different categories using 3 data sets, DataSet1, DataSet2 and DataSet3. From Figure 1 and Figure 2, EB-DBSCAN algorithm has a bit lower average purity than DBSCAN algorithm, but almost equivalent. The reason lies in DBSCAN algorithm using each data packet as a data point, while the EB-DBSCAN algorithm using "*WindowSize*" data packets as a data point, mixing the different attack categories of data in one data window, thus resulting in lower purity. As to the average precision, EB-DBSCAN and DBSCAN are quite equivalent, but for some categories, such as U2R, EB-DBSCAN's precision even outweighs DBSCAN. Anyway, all categories' average precision is above 80%.

Figure 3 shows different categories' clustering precision using the EB-DBSCAN algorithm with four data sets. On the whole, NORMAL, DoS and PORBE's clustering precision is relatively high, owing to these 3 categories accounting for the majority of the testing data set, while U2R and R2L have only a small part and more dispersed, it's more likely to cause misclassification and lead to low precision for these 2 classes.

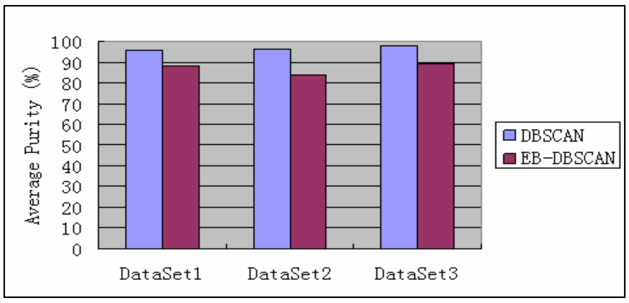

Figure 1: Average purity comparison between DBSCAN and EB-DBSCAN algorithms

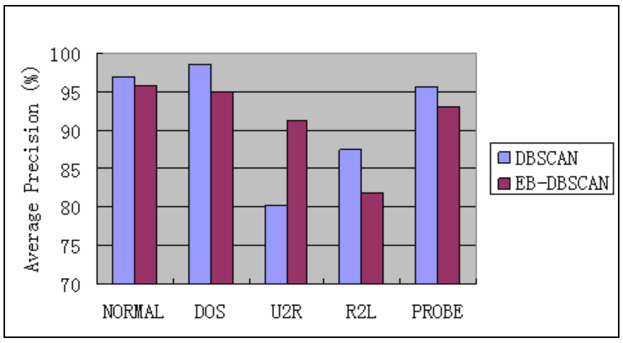

Figure2: Average precision comparison between DBSCAN and EB-DBSCAN algorithms

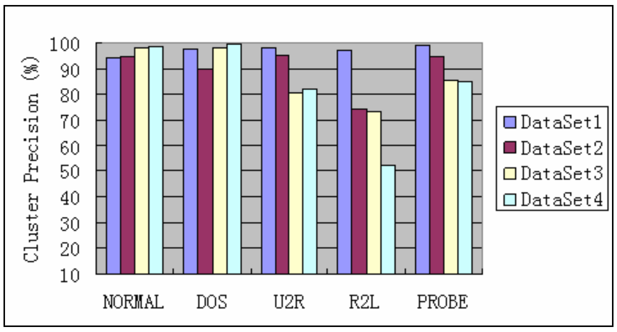

Figure3: Detailed cluster precision comparison of EB-DBSCAN algorithm among 4 data sets

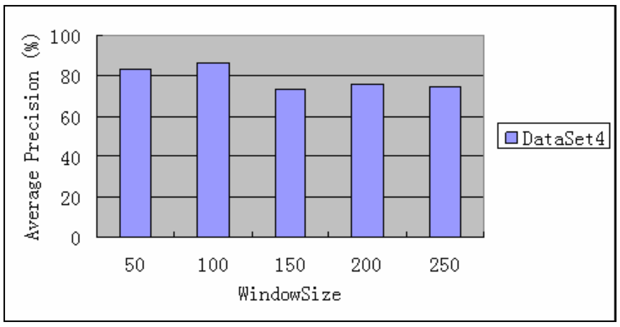

Figure4: Different average precision of EB-DBSCAN with DataSet4 using different *WindowSize*

In order to obtain a higher precision, parameter selection is very critical for the EB-DBSCAN algorithm, "*WindowSize*" is an important parameter that affect the clustering results. We select DataSet4 and set *Minpts* = 4, *EPS* = 1.350 to reveal how the different *WindowSize* will affect all categories' average precision. The experimental results show in Figure 4. It is obvious from Figure 4, with the increase of *WindowSize*, the average precision of each cluster decreases. The reason is with the increase of the data window, more different attack category samples comes in, which results in the increase of entropy in the data window. This also demonstrates that the size of window is a direct factor affecting the average clustering precision. So, how to select optimal parameters or adaptive parameters acquisition is a meaningful research direction in the future.

4.3 Experiment of Number of Clusters and Time Performance

Table 4 and 5 show the number of clusters and clustering execution time under the condition of ensuring optimal average purity and average precision with the parameters set according to Table 2. From Table 4 and 5, it is obvious that EB-DBSCAN algorithm's time complexity and number of clusters are both better than DBSCAN and the execution time improves more than 100 times. The main reason is EB - DBSCAN algorithm uses batch data processing for data entry, rather than single data processing. So, the size of the data processing is effectively reduced and it can greatly reduce the time complexity, which has a very wide application prospects in quick-building detecting models for high-speed, huge amount of stream data.

Table 4: The Execution time of DBSCAN AND EB-DBSCAN(millisecond)

	DBSCAN	*EB-DBSCAN*	*Ratio*
DataSet1	319428	1208	264.43
DataSet2	1360902	5640	241.29
DataSet3	9105231	45459	200.30
DataSet4	>9105231	139249	>65.39

Table 5: Number of clusters of DBSCAN AND EB-DBSCAN

	DBSCAN	*EB-DBSCAN*	*Percentage (%)*
DataSet1	35	23	65.71
DataSet2	58	30	51.72
DataSet3	118	61	51.69
DataSet4	>118	72	<61.02

5 Conclusion and Future Work

Based on the traditional DBSCAN clustering algorithm, relative network entropy based clustering algorithm EB-DBSCAN is posed in this paper. From the experiments, the average precision and the average purity of EB-DBSCAN and DBSCAN are generally equivalent. But for some small amount, scattered attacks data, such as U2R and R2L, EB-DBSCAN algorithm performs a little poorer than DBSCAN algorithm. The main reason is DBSCAN algorithm making each data packet as a processing point, while EB-DBSCAN uses batch data processing, making packets of “*WindowSize*” as a processing point. For time performance and number of clusters, EB-DBSCAN is far better than DBSCAN, in particular, the clustering time improves more than 192 times in average, which makes the algorithm’s bright prospects in stream clustering and processing of high-speed, massive data of arbitrary shape. In future, the adaptive acquisition of parameters, such as, *EPS*, *Minpts* and *WindowSize* should be studied further. Moreover, applying the network entropy and relative network entropy method to the real network environment to evaluate the practical effectiveness is also very significant in future work.

Acknowledgments

This work is supported by Shanghai Leading Academic Discipline Project(J50103), the Innovation Project of Shanghai Municipal Education Committee (09YZ05), Doctoral Fund of Ministry of Education of China for Youth Teachers (20093108120016), National Science Foundation of China (61003248).

References

[1] P. Berkhin, “Survey of clustering data mining techniques,” Technical Report, Accrue Software, SanJose, CA, 2002.

[2] M. Ester, H. Kriegel, J. Sander and X. W. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” Proceedings of the 1996 2nd Int’l Conf on Knowledge Discovery and Data Mining. Portland : AAAI Press, pp. 226-231, 1996.

[3] S. Guha, R. Rastogi, K. Shim, “Cure: An Efficient Clustering Algorithm for Large Database.,” Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Seattle, Washington, pp. 73-84, 1998.

[4] Y. Gu, A. M. Callum, and D. Towsley, Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation, Technical Report. Department of Computer Science, Umass, Amherst, 2005.

[5] Z. S. He, “Data-partitioning-based parellel DBSCAN algorithm,” Mini- Micro Systems, vol. 27, no. 1, pp.114-116, 2006.

[6] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data clustering: a survey,” ACM Computer Survey, vol. 31, pp. 264-323, 1999.

[7] P. Kabiri and A. Ghorbani, “Research on intrusion detection and response: a survey,” International Journal of Network Security, vol. 1, no. 2, pp. 84-102, 2005.

[8] KDD99 Cup Dataset. (1999-01-02)
http://kdd.ics.uci.edu/databases/kd-dcup99/kddcup99.html

[9] W. Lee, S. J. Stolfo, and K. W. Mok, “Mining in a data-flow environment: experience in network intrusion detection,” Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining(KDD'99), pp114-124, 1999.

[10] W. Lee and S. J. Stolfo, “A framework for constructing features and models for intrusion

detection systems," ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 227-261, 2000.

[11] G. Nychis, "An Empirical Evaluation of Entropy-based Anomaly Detection," Information Networking Institute Carnegie Mellon University Pittsburgh, Master thesis, 2007.

[12] I. V. Onut and A. A. Ghorbani, "A feature classification scheme for network intrusion detection," International Journal of Network Security, vol. 5, no. 1, pp. 1-15, 2007.

[13] C. E. Shannon and W. Weaver, "The Mathematical Theory of Communication," University of Illinois Press, 1949.

[14] G. Z. Wang and G. L. Wang, "Improved fast DBSCAN algorithm," Journal of Computer Applications. vol. 29, no. 9, pp. 2505-2508, 2009.

[15] N. N. Wu, "Audit data analysis and mining", Ph.D. Thesis, George Mason University, USA, 2001.

[16] S. G. Zhou, "A fast density-based clustering algorithm," Journal of Computer Research & Development. vol. 37, no. 11, pp. 1287-1292, 2000.

Quan Qian is an associate professor in Shanghai University, China. His main research interests concerns computer network and network security, especially in cloud computing, IoT and wide scale distributed network environments. He received his computer science Ph.D. degree from University of Science and Technology of China (USTC) in 2003 and conducted postdoc research in USTC from 2003 to 2005. After that, he joined Shanghai University and now he is the lab director of network and multimedia.

Tian-hong Wang received BS degree in computer science from Shanghai University in 2009. He is currently working toward a master degree in the school of computer science, Shanghai University. His research interests include computer and network security, distributed data storage.

Rui Zhang received her B.E. and Ph.D. degree from Department of Electronic Engineering & Information Science, University of Science and Technology of China, in 2003 and 2008, respectively. After graduation, she works in School of Computer Engineering and Science, Shanghai University. Her main research interests include computer networks, network coding for wireless networks and wireless communication, etc.