An Alert Aggregation Algorithm Based on K-means and Genetic Algorithm

Xianguang Lu1,*, Xuehui Du1 and Wenjuan Wang1

1Zhengzhou Information Science and Technology Institute, Zhengzhou, 450001, China

*Corresponding Author: hzcxlxg@163.com

Abstract. The Intrusion Detection System often produces a large number of alerts, in which 90% are useless. This makes it difficult for security administrators to identify real attack alerts. Using clustering algorithms, such as K-means, DBSCAN, can efficiently cluster similar alerts, thus greatly reducing the number of alerts that need to be processed. However, the original clustering algorithm has some shortcomings, for example, the K-means has great dependence on the initial value selection, and it is easy to fall into local optimum. Therefore, this paper proposes a new alert aggregation method based on the genetic algorithm and K-means algorithm. We use the Darpa99 dataset to test the performance of our algorithm. And the experimental results show that the algorithm can obtain good aggregation results.

1. Introduction

Intrusion Detection System (IDS) is an important tool to protect the security of computer networks. In order to protect important resources better, people usually deploy multiple IDSs. However, these IDSs will produce a large number of false alerts, which will make it difficult for security managers to identify useful information.

The purpose of alert aggregation is to aggregate similar alerts in the original alert. For a large number of alerts produced by low level IDSs, alert aggregation can reduce the number of alerts effectively. The aggregated alert is more representative, which can help the security administrator to analyse the causality between the alert. But there are some defects in the existing alert aggregation methods, such as overdependence on expert knowledge and low degree of aggregation. Therefore, it is a hot topic to design an efficient method which is less dependent on expert knowledge.

K-means algorithm is a common clustering algorithm, which has the advantages of simple, easy implementation, linear and so on. But there are some shortcomings: (1) users need to specify the number of clusters K in advance [1]; (2) the clustering effect depends on the choice of the initial center, and the clustering results of different initial centers are often different; (3) it might fall into local optimum. Aiming at these shortcomings, this paper proposes a new clustering algorithm based on the Genclust algorithm proposed by Rahman [2] firstly, and applied this algorithm to alarm aggregation.

The remaining part of this paper is arranged as follows: The second chapter mainly introduces the research status of alert aggregation. The third chapter mainly introduces the improved clustering algorithm. The fourth chapter analyses the algorithm through vertical and horizontal comparative experiments. Finally, the fifth part briefly summarizes the work and points out the direction of future work.
2. Related Work

Valdes et al. [3] proposed a method of aggregation by calculating the similarity of alert. They defined the degree of similarity between each attack class in advance, and defined the method of similarity calculation for each alert attribute (such as time and IP address, etc.), then calculated the overall similarity by the formula. If the similarity exceeds a certain threshold, they are aggregated. Debar and Wespi [4] proposed an aggregation and correlation components (ACC) alert correlation component, and used predefined rules to do correlation analysis. The definition of the ACC rule took into account the two relationships of alerts: duplicate and consequence, that is, they considered both the similarity and the causality. Lin et al. [5] proposed an improved multiresource heterogeneous alert aggregation method, which divided the alert aggregation features into attack mode constraints and time interval constraints, and different attack modes had different time interval constraints. The author used the relative mean variance of time interval proposed in [6] as the dynamic updating coefficient of time interval, and set the average value of the time interval to the threshold of the initial time interval, thus making the time interval threshold adaptive network attack the environment change. Siraj [7] proposed a method that uses Fuzzy Cognitive Modelling for alert aggregation, and used a multilevel clustering method to adapt to inaccurate matching.

In recent years, researchers have begun to use artificial intelligence (AI) technology to solve the problems of intrusion detection technology. In [8], Wang et al. put forward a framework called FCANN. First, they used the fuzzy C mean clustering algorithm to cluster the data, and then classified each cluster by ANN. The framework integrates unsupervised learning and supervised learning, shortens modelling time and improves detection rate. Gogoi et al. [9] put forward a MLH-IDS framework, which included three levels: Supervision layer, unsupervised layer and outlier layer. The supervision layer is used to detect DoS and Probe attacks. The unsupervised layer is used to detect normal data, while outliers are used to distinguish R2L and U2 attacks. Hybrid multilevel frameworks make full use of different machine learning algorithms, which are more flexible and have better performance. In [10], Zhu proposed a method that automatically extracts attack strategies from a large number of Intrusion Alerts without relying on a specific prior knowledge, based on multilayer perceptron and support vector machine, and used an alert correlation matrix to store the correlation between any two types of alerts.

3. IGenClust: an Improved Clustering Algorithm

In view of the shortcomings of the K-means algorithm, there are many works that have improved the original method [11] [12] [13] [14]. Combining genetic algorithm and K-means algorithm, on the one hand, do not require the user to enter the exact number of cluster [15], and can better solve the local optimum problem due to the improper selection of the initial center [16]; on the other hand, can speed up the convergence of genetic algorithm. Rahman proposed a clustering technique called GenClust in [2]. In GenClust, the authors introduced new fitness function and gene rearrangement technology, and achieved good clustering results. This method requires the user to enter multiple clustering radius in advance, but in the actual operation, the good clustering radius may not appear in the initial set provided by the user, so there may not be genes in the initial population that represent the best cluster center. Moreover, as an improved genetic algorithm, this algorithm doesn’t solve the inherent premature problem of genetic algorithm, which has a certain influence on the result of clustering. This paper proposed an improved GenClust algorithm, IGenClust, which overcomes the premature problem of the GenClust algorithm by introducing the population entropy, and avoids the poor clustering result caused by the improper selection of the radius by improving the selection of the initial clustering center.

3.1. Main components

The IGenClust algorithm mainly consists of 8 main components:

- Component 1: Standardized data set and distance calculation;
- Component 2: Candidate population;
- Component 3: Chromosome selection;
- Component 4: Crossover;
- Component 5: Twin Removal;
- Component 6: Elitism;
- Component 7: Mutation;
- Component 8: K-means algorithm.

3.2. Detailed description

3.2.1. Standardized data set and distance calculation

3.2.1.1. Standardized data sets. Assume that the original data set has \( n \) records, that is \( R=\{R_1,R_2,R_3,\ldots,R_n\} \), the record \( R_k \) has \( m \) attributes, that is \( R_{i,1},R_{i,2},R_{i,3},\ldots,R_{i,m} \). That is to say, \( R_{ij} \) represents the \( j \)th property of the \( i \)th record, it can be numeric or categorized. We normalize the numeric attributes of the record in order to avoid the scale or measurement unit problem of distance calculation in subsequent calculation, for example, the range of the attribute \( p \) is \([1,u]\), \( R_{ip} \) should be normalized to \( R\_ip^*=\frac{(R\_ip-1)}{(u-1)} \).

3.2.1.2. Distance calculation. The calculation of distance between records is usually based on Euclidean distance between attributes, but this method can only calculate the distance between numerical attributes. Giggins designed a method to calculate the similarity of categorical attributes in [17], and Rahman applied it to GenClust. For the record \( R_a \) and \( R_b \), the distance of categorical attribute \( j \) can be expressed as \( \text{Dist}(R\_a\_j,R\_b\_j) = 1-\text{Sim}(R\_a\_j,R\_b\_j) \). The total distance between record \( R_a \) and \( R_b \) is:

\[
\text{Dist}(R\_a,R\_b) = \frac{\sum_{i=1}^{N} \text{abs}(R\_a\_i,R\_b\_i) + \sum_{j=N+1}^{m} \text{dist}^c(R\_a\_j,R\_b\_j)}{N+C}
\]

User should sort the attributes of records firstly, and the first \( N \) attributes are numerical attributes after normalization, the latter \( C \) attributes are categorized.

The above method can be used as a general purpose method for dealing with general problems, but they should be adjusted appropriately when dealing with specific problems. In this paper, our method is mainly used to solve the problem of alert aggregation, so the above method can be adjusted as follows:

1. Source IP, Destination IP: The similarity of the IP address is calculated by converting the IP address to the 32-bit binary string and comparing them. Assuming that the number of two strings from the left to the right is \( n \), the similarity is \( \text{Sim}_b(ip_a,ip_b)=n/32 \). For example, for two addresses: \( ip_1 = 172.16.112.100 \), \( ip_2 = 172.16.112.192 \). First, convert the two addresses 10101100001000001100001100100 and 101011000001000011100011000000, it is easy to see that \( n = 24 \), so \( \text{Sim}_b(ip_1,ip_2) = 24/32 = 0.75 \). The range of similarity between IP is \([0,1]\).

2. Source port, Destination port: The maximum value of port is 65535, so the similarity of ports can be normalized \( \text{Sim}_p(port_a, port_b) = 1-\left|\frac{port_a - port_b}{65535}\right| \). For example, for ports 23 and 25625, \( \text{Sim}_p(port_a, port_b) = 1-\left|\frac{23-25625}{65535}\right| = 0.61 \). The range of similarity between port is \([0,1]\).

3. Time: The similarity of time is mainly calculated by the time interval and threshold. We set a maximum threshold \( T_{max} \) and a minimum threshold \( T_{min} \), then the similarity of time, \( t_1,t_2 \), can be expressed as:
If $|t_1 - t_2| < T_{\text{min}}$, then $\text{Sim}_\text{a} = 1$;
If $|t_1 - t_2| > T_{\text{max}}$, then $\text{Sim}_\text{a} = 0$;
If $T_{\text{min}} < |t_1 - t_2| < T_{\text{max}}$, then $\text{Sim}_\text{a} = (T_{\text{max}} - |t_1 - t_2|)/ (T_{\text{max}} - T_{\text{min}})$.

The range of similarity between time is $[0,1]$.

(4) Type: If the type of the two alert is the same, then $\text{Sim}_\text{type} = 1$, otherwise, $\text{Sim}_\text{type} = 0$.

To sum up, the overall similarity of the alert is:

$$\text{Sim}(R_i, R_j) = \frac{\text{Sim}_{\text{sp}} + \text{Sim}_{\text{sp}} + \text{Sim}_{\text{size}} + \text{Sim}_{\text{type}} + \text{Sim}_{\text{size}} + \text{Sim}_{\text{type}}}{6}$$  \hspace{1cm} (2)

Distance is: $\text{Dist}(R_i, R_j) = 1 - \text{Sim}(R_i, R_j)$.

Because the ranges of similarity of each attribute are $[0,1]$, the range of $\text{Sim}(R_i, R_j)$ and $\text{Dist}(R_i, R_j)$ are $[0,1]$.

3.2.2. Candidate population. The function of this component is to generate the candidate population of the genetic algorithm. The chromosome $CR_j$ is composed of $K$ genes, that is, $CR_j = (G_{j1}, G_{j2}, G_{j3}, ..., G_{jk})$, and the $K$ is the number of clusters in the K-means algorithm. Each of the genes in the chromosome represents the cluster center in the K-means algorithm, which means that the genes with the highest fitness in the final output of the genetic algorithm correspond to the optimal initial cluster center of the K-means algorithm.

In our method, the candidate population consists of three parts, each of which generates $W$ chromosomes in different ways: (1) given $W$ radius $r$; (2) given $W$ cluster number $k$; (3) randomly select $W$ cluster numbers $K$. Finally, the candidate population contains $3W$ chromosomes.

(1) In the first part, users should provide $W$ radius $r$, $r_1, r_2, ..., r_W$. In the original data set, we first calculate the distance between any two points: $\text{Dist}(R_i, R_j)$, and then use the given radius $r_j$ to calculate the density of each point: $\text{Density}(R_i) = |\{R_j : \text{Dist}(R_i, R_j) \leq r_j, \forall j\}|$. The record $R_i$ with the highest density ($\text{Density}(R_i) > \text{Density}(R_j), \forall j$) is set as the first gene $G_{j1}$, and all the records within the distance $r_j$ ($|\{R_j : \text{Dist}(R_i, R_j) \leq r_j, \forall j\}|$) were delete from the data set. As long as our record has a density greater than the user defined threshold $T$, we continue to select the genes. For a given $r_j$, we get multiple genes $G_{j1}$ to form chromosomes $CR_j$. As we often can’t obtain the radius of optimal cluster in real life, provide multiple $r$ can adapt to the change of the radius, but also increased the genetic diversity, which can help us to find the optimal solution finally.

(2) The number of alert is very large, so it’s not easy to determine the radius. That is to say, the radius provided by users may not contain the optimal radius, and the convergence rate is slower. As pointed out above, when calculating the distance between alert, the similarity of type is calculated simply according to same or not, so the effect of difference of the type is larger than that of other attributes, therefore, the $K$ may be close to the total number of alert data types. For the alert, it is easier to determine the $K$ based on experience than to determine the radius of each cluster. So in the second part, we use the $W$ $K$ values provided by users, and use the radius $\text{Dist}(R_i, R_j)$ which calculated in (1) to get the tightness of each point,

$$\text{Tight}(R_i) = \frac{1}{\sum_{j\neq i} \text{Dist}(R_i, R_j)}$$  \hspace{1cm} (3)

and the probability of selecting each point is:
and select K points as genes to form chromosomes. The advantage of such deterministic process is that the selected chromosomes are basically optimal and will speed up the speed of convergence.

(3) After deterministic selection of the two sets of chromosomes, we have obtained most of the useful individual genes, but may loss some individuals in remote clusters. So, we use the method of random selection in the third part. We randomly generate an integer in the range of \([2, \sqrt{n}]\) as the number of genes in the chromosome, and repeat W times to get W chromosomes. Although the overall fitness of chromosomes selected by random process is relatively low, chromosomes still have some useful genes.

### 3.2.3. Chromosome selection

Before introducing selection, we first introduce the fitness function used in GenClust algorithm. The fitness function used in GenClust is not based on the sum of squares of errors in the general K-means algorithm, but is based on the tightness and separation of clusters. The degree of closeness is the average distance from other points to cluster centers in clusters,

\[
\text{Comp}_j = \frac{\sum_{a \in C_j} \text{Dist}(R_j, m_j)}{|C_j|} \tag{5}
\]

Where \(|C_j|\) is the total number of records in \(C_j\); the degree of separation is the minimum value of the distance between the cluster centers and other clusters, \(\text{Seq}_j = \min_{a \in C_j} \text{Dist}(m_j, m_a)\). Thus, the fitness function is constructed as follows:

\[
\text{Fitness} = \sum_{a \in C_j} (\text{Seq}_j - \text{Comp}_j) \tag{6}
\]

In order to use highly adaptable chromosomes in subsequent work, we need to select W chromosomes from the candidate population by roulette, that is, the probability of each chromosome being selected is \(P(CR_j) = \frac{\text{fitness}(CR_j)}{\sum_{a \in 3W} \text{fitness}(CR_k)}\). Finally, we get the initial population containing W chromosomes.

### 3.2.4. Crossover

Our cross operation basically follows the idea of GenClust algorithm. When selecting the parent chromosomes, we sort the chromosomes according to the fitness firstly, then select the most adaptable chromosome \(CR_p\) (delete it after selection from the cluster), finally select another chromosome \(CR_q\) in the chromosomes from the remaining chromosomes by roulette technique, and \(CR_q\) is also deleted after being selected.

The offspring generated after single point crossover of chromosomes get some genes on the left part of one parent and the right part of another parent, but these genes may be similar to each other and can’t be extended to the whole space, so we should rearrange the gene before single point crossover on the chromosome. Set \(CR_p\) as a reference chromosome, \(CR_q\) as a target chromosome, the general idea of gene rearrangement is as follows:

In all genes of the target chromosome, the gene with the smallest distance from the first gene of the reference chromosome is placed in the first place of the target chromosome rearrangement version. In the remaining genes of the target chromosome, the gene with the smallest distance from the second genes of the reference chromosome is placed in the second place of the rearrangement version. If the length of the target chromosome (that is, the number of genes) is equal to or less than the length of the reference chromosome, calculate the best gene in each position in sequence according to the above method, until all the genes are rearranged on the target chromosome. If the length of the target
chromosome is larger than that of the reference chromosome, the rearrangement is divided into two stages. In the first stage, the first X gene of the target chromosome (x is the length of the reference chromosome) is rearranged as above. In the second stage, calculate the distances between the remaining genes of the target chromosome and the genes of the rearranged chromosomes and insert them into the back of their nearest genes.

After the recombination, we perform the traditional single point crossover, that is, the left part of one parent chromosome is connected to the right part of another parent of the parent chromosome to form an offspring chromosome, and the remaining parts are combined together to form another offspring chromosome.

3.2.5. Twin removal. After generating new offspring chromosomes, we need to remove duplicate genes from chromosomes. If the distance of gene \( G_i \) and \( G_j \), \( \text{dist}(G_i,G_j) < \delta \), \( \delta \) is a very small positive number, we think that these two genes are duplicate. When the number of genes in the chromosome is more than 2, we regard these genes as duplicate, when the number of genes is equal to 2, one of these genes is randomly mutated until these genes are no longer duplicate.

3.2.6. Elitism. The elitism operation is to store the highest fitness chromosomes of each iteration as \( CR_e \), and compare it with the worst (the best) chromosomes of the offspring. If the fitness of \( CR_e \) is larger than that of the worst chromosomes of offspring, the worst chromosome is replaced by \( CR_e \). If the fitness of \( CR_e \) is smaller than that of the best chromosomes of offspring \( CR'_e \), then \( CR_e = CR'_e \).

3.2.7. Mutation. Mutation is an operation to change one attribute of an individual into other values within the range of the value of the attribute. Its purpose is to increase the diversity of chromosomes in the population. If the mutation probability is too small, it is difficult to produce a new individual. If the mutation probability is too large, the genetic algorithm will become a pure random search algorithm.

Srinivas and Patnaik proposed an adaptive genetic algorithm (AGA) [14]. When the fitness is high, the mutation probability is low; when the fitness is low, the mutation probability is high, that is to say, the probability is adjusted adaptively according to the fitness. Although the adaptive genetic algorithm has better probability of the individual in each generation, it does not solve the inherent precarious problem of the genetic algorithm. The premature problem is that when the algorithm has not yet searched the global optimal solution, the fitness of the offspring can no longer exceed the parent generation, that is, there are almost no differences between individuals. The problem of precocity is essentially due to the lack of species diversity, which leads to the failure of genetic algorithm to jump out of the local optimal solution. In other words, when two parents are exactly the same or very similar (similar to inbreeding), crossover will be difficult to produce new individuals, which will prevent the algorithm from searching for new solution space, resulting in the loss of population diversity.

In order to solve the premature problem, we introduce the population entropy in this components.

1) The definition of population entropy: the total number of chromosomes in the population is \( N \), and the similar chromosomes in the population can be divided into \( M \) class (\( M < N \)). The number of individuals in each class is \( n_i \), then the entropy of the population is defined as:

\[
S = \sum_{i=1}^{M} \frac{n_i}{N} \log_2 \frac{n_i}{N}
\]  

(7)

2) The calculation method for population entropy: first, determine which chromosomes belong to the same class, the number of the genes in the same class of chromosome is the same (the same species), and the chromosomes with small difference of genes belong to the same class. On the other hand, if the number of genes is different or the difference between genes is large, the chromosomes do not belong to the same class.

The method of judging whether genes are similar follows the method of attribute similarity mentioned earlier. After finding similar points, we can get the chromosomes belonging to the same
class and then get the population entropy.

3) The relationship between population entropy and population diversity: if the population has a high diversity, the probability of each individual appears is similar, the population is in the active evolutionary selection stage, and the information entropy of the population is large.

Therefore, when the diversity is low and the population entropy is small, it is necessary to increase mutation probability and increase individual diversity. Suppose the value of entropy is $S$, and the mutation probability is

$$P = \begin{cases} \frac{k_{1} e^{-\frac{f_{\text{max}} f}{f_{\text{max}} f_{\text{avg}}}}} {f > f_{\text{avg}}} \\ k_{1} e^{-\frac{f}{f_{\text{avg}}}} & f \leq f_{\text{avg}} \end{cases}$$

Where $f_{\text{avg}}$ represents the average fitness of the current population, $f_{\text{max}}$ represents the highest fitness of the current population, and $f$ represents the fitness of the current chromosome. Because the range of $S$ is $[0, \log n]$, $0 \leq k_{1} \leq \frac{1}{n}$.

Increasing the mutation probability can increase the diversity of the population, but it is not conducive to the convergence of the population. It will add additional work if the mutation is performed every iteration, so the mutation is performed every 10 generations in this article.

3.2.8. K-means algorithm. We use the genes in the best chromosome as the initial seed of the K-means algorithm, which will produce a high quality cluster solution. In this component, we use the improved distance calculation method mentioned earlier to calculate distance.

4. Experiment

Weka (Waikato Environment for Knowledge Analysis) is an open source machine learning and data mining software based on the Java environment. It is one of the most complete data mining tools in the present day. In order to show the experimental results better, this paper writes the algorithm based on the interface document of Weka, and adds the IGenClust algorithm to the cluster module of Weka, then carries out clustering analysis on the Darpa1999 dataset.

We compare the improved algorithm with the SimpleKMeans, GenClust and other algorithms in the same operating environment according to two aspects, error and execution time. The experimental operating environment is Inter CPU 2.40 GHz and the memory is 8 GB, the software environment is as follows: the operating system is Windows10 Microsoft, the experimental program is written in Java and the development environment is Visual Studio Microsoft 2015. The development tools used are Myeclipse10 and Weka3.6.15.

4.1. Preparation

Firstly, we convert the original alert information for the Darpa data set generated by IDS into .arff format files that can be identified by Weka, and then import Weka3.6 engineering file into Myeclipse10 for the compilation of the algorithm.

4.2. Implementation

The experiment is divided into two stages. The first stage is divided into 4 parts to compare different algorithms:

- The first part uses SimpleKMeans algorithm that comes with Weka.
- The second part composes the algorithm which is suitable for distance comparison of alert information, and add it to SimpleKMeans algorithm to form ImproveKMeans algorithm.
- The third part uses the original GenClust algorithm.
- The fourth part uses our IGenClust algorithm.

The second stage is divided into two parts:
The first part is repeated experiments to measure the stability of the algorithm. The second part is to test the performance trend of the IGenClust algorithm under different data sets.

4.2.1. Vertical comparison. The test data used by different algorithms are 2534 records extracted from the alert collected from the Darpa99 dataset for week 5, day 3.

(1) In the first part, the maximum iteration is set to 500 times, and the generated clusters are set to 10. The result is shown as figure 1 (a).

SimpleKMeans constantly judged whether the polymerization was complete, so the experiment ended up only 5 times. The speed of SimpleKMeans operation is very fast, but this algorithm uses Euclidean distance as fitness function. The reference value as a longitudinal comparison is smaller.

(2) In the second part, the ImproveKMeans algorithm does not make any changes except the use of our own distance calculation function to replace the Euclidean function. The maximum iteration is set to 500 times, and the generated clusters are set to 10. The result is shown as figure 1 (b).

The smaller fitness shows that the degree of aggregation is poor.

(3) In the third part, we use the GenClust algorithm with our own distance function and the maximum iteration 500 times. The result is shown as figure 1 (c).

The fitness indicates that the algorithm can get better clustering results than SimpleKMeans.

(4) In the fourth part, we use the IGenClust algorithm and set the maximum iteration 500 times. The result is shown as figure 1 (d).

We can see that although the IGenClust algorithm is slightly worse than the GenClust algorithm in the execution time for the reason of the population entropy, the overall fitness has been improved.
The results are summarized as shown in Table 1. From this table, we can compare the differences between algorithms more intuitively.

**Table 1. Comparison of execution time and fitness of different algorithms.**

|                | SimpleKMeans | ImproveKMeans | GenClust | IGenClust |
|----------------|--------------|---------------|----------|-----------|
| **Fitness**    | \            | 0.119         | 0.943    | 1.540     |
| **Time (ms)**  | 0.17         | 5.69          | 175.89   | 205.37    |

4.2.2. **Horizontal comparison.** (1) Part one: In the first part, we have done many experiments on 2534 records to verify the stability of the algorithm. The results of the experiment are as follows:

![Figure 2](image)

**Figure 2.** Different fitness in multiple experiments.

From the figure 2, we can see that there is little change in the fitness of the algorithm under repeated tests, which shows that the algorithm is relatively stable.

(2) Part two: In the second part, we used five sets of different datasets, the numbers were 2534, 3813, 4857, 6861, 8192, respectively. The transformation curves of fitness and time are as follows:

![Figure 3](image)  ![Figure 4](image)

**Figure 3.** Different fitness under different data sets.  
**Figure 4.** Different time under different data sets.

As shown in the figure 3 and figure 4, the number of experimental data has little effect on fitness and time, and higher fitness often means more time.

4.3. **Conclusion**

Through the above two stage experiments, we can see that our IGenClust algorithm sacrificed a certain amount of execution time, but improved a certain aggregation effect. In theory, the algorithm can solve the problem of precocious algorithm, and it is relatively stable.

5. **Summary and future work**

For the large number of similar alerts generated by Intrusion Detection Systems, more and more experts apply the clustering algorithm to alert aggregation to enhance the aggregation effect. The
GenClust algorithm proposed by Rahman has a high degree of aggregation and good scalability. Based on this algorithm, this paper proposes an improved algorithm, IGenClust, and applies it to alert aggregation. Experiments show that the IGenClust algorithm can not only improve the premature problem of GenClust algorithm theoretically, but also improve the degree of aggregation to a certain extent. However, although the number of alerts after aggregation has become smaller, the causality between alerts is not well represented. Therefore, in order to provide more intuitive alert information for the security managers, further association analysis of the aggregated alert should be carried out to reconstruct the real attack scene.

Acknowledgments
This research is supported by the National High-tech R&D Program of China under grant No.2015AA016006, and the Natural National Key Basic Research Program of China under grant No.2016YFB0501901.

References
[1] Lewis R, Krawiec M, Confer E, et al. 2014 Data clustering: 50 years beyond K-means, Pattern Recognition Letters, Vol. 31, No. 8, pp.651-666
[2] Rahman MA, Islam MZ 2014 A hybrid clustering technique combining a novel genetic algorithm with K-Means, Knowledge-Based Systems, Vol. 71, No. 71, pp. 345-365
[3] Valdes, A, & Skinner, K 2000 Adaptive, Model-Based Monitoring for Cyber Attack Detection, International Workshop on Recent Advances in Intrusion Detection, Vol.1907, pp.80-92
[4] Wespi, A 2001 Aggregation and Correlation of Intrusion-Detection Alerts, International Symposium on Recent Advances in Intrusion Detection, Vol.2212, pp.85-103
[5] Lin H, Zhijie W, Xiaofang H, et al. 2014 An improved multisource heterogeneous alarm aggregation scheme, Application Research of Computers, Vol.31, No. 2, pp.579-582
[6] Gong J, Haibin M, Yong D, et al. 2005 Redundant elimination of intrusion events with multi feature association, Journal of Southeast University (Natural Science), Vol.35, No. 3, pp. 366-371
[7] Siraj A, Vaughn R 2005 Multi-level alert clustering for intrusion detection sensor data, Fuzzy Information Processing Society, pp.748-753
[8] Wang, G, Hao J, Ma J, Huang L 2010 A new approach to intrusion detection using artificial neural networks and fuzzy clustering, Expert Systems with Applications, Vol.37, No.9, pp. 6225-6232
[9] Gogoi, P, Bhattacharyya DK, Borah B, Kalita JK 2014 MLH-IDS: a multi-level hybrid intrusion detection method, Computer Journal, Vol.57, No.4, pp. 602-623
[10] Zhu B, Ghorbani A 2006 Alert correlation for extracting attack strategies, International Journal of Network Security, Vol.3, No.3, pp. 244-258
[11] Rahman M, Islam M 2012 CRUDAW: a novel fuzzy technique for clustering records following user defined attribute weights, Tenth Australasian Data Mining Conference, pp.27-41
[12] Yu S, Chu, SW, Wang CM, Yan YK, Chang TC 2017 Two improved k-means algorithms, Applied Soft Computing
[13] Srinivas M, Patnaik LM 2002 Adaptive probabilities of crossover and mutation in genetic algorithms, IEEE Transactions on Systems Man & Cybernetics, Vol.24, No.4, pp.656-667.
[14] Jie L, Liu W, Sun Z, Teng S 2017 Hybrid fuzzy clustering methods based on improved self-adaptive cellular genetic algorithm and optimal-selection-based fuzzy c-means, Neurocomputing, No.249, pp.140-156.
[15] Hruschka ER, Campello RJGB, Freitas AA 2009 A survey of evolutionary algorithms for clustering, IEEE Transactions on Systems Man & Cybernetics Part C Applications & Reviews, Vol.39, No.2, pp.133-155
[16] Pakhira MK, Bandyopadhyay S, Maulik U 2005 A study of some fuzzy cluster validity indices, genetic clustering and application to pixel classification, Fuzzy Sets & Systems, Vol.155, No. 2, pp.191-214
[17] Gogins H, Brankovic L 2012 VICUS: a noise addition technique for categorical data, Tenth Australasian Data Mining Conference, Vol.134, pp.139-148