Abstract—Network anomaly detection is a classically difficult research topic in intrusion detection. However, existing research has been solely focused on the detection algorithm. An important issue that has not been well studied so far is the selection of normal training data for network anomaly detection algorithm, which is highly related to the detection performance and computational complexity. Based on our previous proposed TCM-KNN (Transductive Confidence Machines for K-Nearest Neighbors) anomaly detection method, which can detect anomalies with high detection rate and low false positive rate, we develop an instance selection mechanism for TCM-KNN based on EFCM (Extended Fuzzy C-Means) clustering algorithm in this paper, aiming at limiting the size of training dataset, thus reducing the computational cost of TCM-KNN and boosting its detection performance. We report the experimental results over real network traffic. The results demonstrate the effectiveness of our methods presented in this paper by a series of experiments on the real network traffic collected in a research network backbone.

Keywords—Network security; anomaly detection; TCM-KNN; instance selection; EFCM

I. INTRODUCTION

Intrusion Detection System (IDS) plays vital role of detecting various kinds of attacks and it is valuable tools for the defense-in-depth of computer networks. Network IDS looks for known or potential malicious activities in network traffic and raise an alarm whenever a suspicious activity is detected.

Currently the two basic methods of detection are signature-based and anomaly-based [1, 2]. The signature-based method, also known as misuse detection, looks for a specific signature to match, signaling an intrusion. They can detect many or all known attack patterns, but they are of little use for as yet unknown attack methods. Most popular intrusion detection systems fall into this category. Snort is the most well-known signature-based open source software for network intrusion detection among them [22]. Another useful method to intrusion detection is called anomaly detection. Anomaly detection applied to intrusion detection and computer security has been an active area of research since it was originally proposed by Denning [3]. Anomaly detection algorithms have the advantage that they can detect new types of intrusions as deviations from normal usage. In this problem, given a set of normal data to train from, and given a new piece of test data, the goal of the intrusion detection algorithm is to determine whether the test data belong to “normal” or to an anomalous behavior. However, anomaly detection schemes suffer from a high rate of false alarms. This occurs primarily because previously unseen (yet legitimate) system behaviors are also recognized as anomalies, and hence flagged as potential intrusions. ADAM [23] is one example of the systems based on anomaly detection methods.

In this paper, according to our previous proposed unsupervised network anomaly detection method based on TCM-KNN (Transductive Confidence Machines for K-Nearest Neighbors) algorithm in [10], we emphasize the extended FCM (Fuzzy C-Means) based instance selection optimizations for our TCM-KNN. We design an instance selection mechanism to optimize TCM-KNN both in reducing its computational cost and in boosting its detection performance. We finally demonstrate the effectiveness of our methods presented in this paper by a series of experiments on the real network traffic collected in a research network backbone.

The rest of this paper is organized as follows. We outline the related work in network anomaly detection in Section II and introduce TCM-KNN (Transductive Confidence Machines for K-Nearest Neighbors) algorithm in Section III. Section IV details our EFCM (Extended FCM) based instance selection mechanism for TCM-KNN. Section V reports the relevant experiments and discussions. We conclude our work in Section VI.

II. RELATED WORK

In the past several years, a lot of anomaly detection methods attempt to build some kind of a model over the normal data and
then check to see how well new data fits into that model. In general, they can be classified into supervised and unsupervised anomaly detection techniques.

For supervised-based methods, a lot of machine learning-based techniques focus on building a system that improves its detection performance based on previous results. Several researchers have adopted a lot of machine learning ideas to create models for anomaly detection. Valdes et al. [4] developed an anomaly detection system that employed naïve Bayesian networks to perform intrusion detection on traffic bursts. Shyu et al. [5] proposed an anomaly detection scheme, where PCA (Principal Component Analysis) was used as an outlier detection scheme and was applied to reduce the dimensionality of the audit data and arrive at a classifier that is a function of the principal components. In general, they are effective while accompanied by a bit heavy computational cost. In addition, data mining-based techniques are also introduced for supervised anomaly detection since they can help improve the process of intrusion detection by adding a level of focus to anomaly detection. Dickerson et al. [6] developed the Fuzzy Intrusion Recognition Engine (FIRE) using fuzzy sets and fuzzy rules. FIRE uses simple data mining techniques to process the network input data and generate fuzzy sets for every observed feature. The fuzzy sets are then used to define fuzzy rules to detect individual attacks. In the domain of network anomaly detection, genetic algorithms have been used in a number of ways.

Eskin and Portnoy, et al. proposed a series of clustering-based techniques such as fixed-width clustering, KNN score, one-class SVM, to fulfill anomaly detection tasks. Learning from the previous literatures, we found that the fixed-width clustering, KNN score and one-class SVM proposed by authors in [8] were demonstrated superior to those above supervised anomaly detection techniques, especially the performance of one-class SVM algorithm behaves the best both in detection true positive rate (TP) and false positive rate (FP). However, its FP is not as satisfactory as we expected (10% for FP when its TP reaches 98% [8]). Recently, Zhang and Zulkernine applied random forests algorithm to unsupervised anomaly detection. They reported that their detection rate is 95% and the false rate is 1% when evaluated over the KDD Cup 1999 dataset [9].

In our previous work, we introduced an efficient supervised anomaly detection technique based on TCM-KNN algorithm, which performed well both using “attack free” or “noisy” dataset (the dataset contains about 1% to 1.5% attack data) for training [10]. In this paper, we focus on optimizing it as a lightweight on-line anomaly detection candidate by means of an extended FCM-based instance selection mechanism, thus reducing its computational cost on the premise of ensuring its high detection performance.

III. TCM-KNN ALGORITHM

A. Introduction to TCM-KNN

TCM-KNN (Transductive Confidence Machines for K-Nearest Neighbors) is a novel algorithm combing TCM [11, 12, 13] and KNN algorithm effectively. In the process of adopting KNN algorithm, we denote the sorted sequence (in ascending order) of the distances of point i from the other points with the same classification y as \( D_i \). In this paper, we use Euclidean distance to calculate the distances between points.

To apply TCM-KNN to anomaly detection, we assign to every point a measure called the individual strangeness measure. This measure defines the strangeness of the point in relation to the rest of the points. In our case the strangeness measure for a point \( i \) belonging to a normal class is defined as:

\[
\alpha_i = \sum_{j=1}^{k} D_{ij}
\]  

where \( k \) is the number of neighbors used. Thus, our measure for strangeness is the ratio of the sum of the \( k \) distances from the point \( i \) to the other points with the same classification (normal class) as \( D_i \). Also, \( D_{ij} \) will stand for the \( j \)th shortest distance in this sequence. The strangeness measure for a point \( i \) belonging to a normal class is defined as:

\[
\alpha_i = \sum_{j=1}^{k} D_{ij}
\]  

The \( \alpha \) computed for an anomaly (a point that does not belong to any of the classes) will be the ratio between two large numbers (the distances from the point in question to those in any of the classes are large). In some cases, this ratio will be small enough to be comparable to the \( \alpha \) values for points already in the class, leading to false negatives. In Equation (2), we denote the sorted sequence (in ascending order) of the distances of point \( i \) from the other points with the same classification (normal class) as \( D_i \). Also, \( D_{ij} \) will stand for the \( j \)th shortest distance in this sequence. \( k \) is the number of neighbors used. Thus, our measure for strangeness is the ratio of the sum of the \( k \) nearest distances from the same class to the sum of the \( k \) nearest distances from all other classes. This is a natural measure to use, as the strangeness of a point increases when the distance from the points of the same class becomes bigger or when the distance from the other classes becomes smaller [10].

Provided with the definition of strangeness, we will use Equation (2) to compute the p-value as follows:

\[
p(\alpha_{new}) = \frac{\# \{ i : \alpha_i \geq \alpha_{new} \}}{n + 1}
\]  

In Equation (2), \( \# \) denotes the cardinality of the set, which is usually computed as the number of elements in finite set. \( \alpha_{new} \) is the strangeness value for the test point (the test points are processed one at a time), is a valid randomness test in the iid (the training as well as new (unlabeled) points are independently and identically distributed) case. The proof takes advantage of the fact that since our distribution is iid, all permutations of a sequence have the same probability of occurring. If we have a
sequence \( \{\alpha_1, \alpha_2, ..., \alpha_n\} \) and a new element \( \alpha_{n+1} \), then \( \alpha_{n+1} \) can take any place in the new (sorted) sequence with the same probability, as all permutations of the new sequence are equiprobable. Thus, the probability that \( \alpha_{n+1} \) is among the \( j \) largest occurs with probability of at most \( \frac{j}{n+1} \).

![Figure 1. TCM-KNN algorithm for anomaly detection](image)

In general classification cases, using the \( \alpha \) values, we can compute a p-value for the new point for the normal class. We call the p-value \( p \). This gives us a way of testing the fitness of point \( \gamma \) for each class \( y \) with a confidence of at least \( \delta = 1 - \tau \). Selecting a confidence level \( \delta \) (usually 95%), we test if \( p \leq \tau \), in which case, we can declare the point an anomaly. Otherwise, we declare it’s normal. The process of our new simplified TCM-KNN algorithm for anomaly detection is depicted in Figure 1 [10].

### Complexity Analysis

Since the method requires finding \( k \) nearest neighbors on each class, we need \( O(n) \) distance computations, per each point to be diagnosed, where \( n \) is the number of data points in the normal dataset. Hence, to diagnose \( s \) points, the complexity would be \( O(ns) \). Moreover, to find out the \( k \) nearest neighbors for the normal dataset, we require \( O(n^2) \) comparisons. We observe that this step is done off-line and only once, before the detection of anomalies starts. However, if \( n \) is very large, the off-line computation may still be too costly.

### IV. FCM-BASED INSTANCE SELECTION FOR TCM-KNN

#### A. FCM Principles

Fuzzy C-Means (FCM) algorithm is a clustering technique that is separated from hard k-means that employs hard partitioning [14, 15]. The FCM employs fuzzy partitioning such that a data point can belong to all groups with different membership grades between 0 and 1. FCM is an iterative algorithm. The aim of FCM is to find cluster centers (centroids) that minimize a dissimilarity function. To accommodate the introduction of fuzzy partitioning, the membership matrix \( U \) is randomly initialized according to Equation (3):

\[
\sum_{j=1}^{c} u_{ij} = 1, \forall j = 1, ..., n
\]

The dissimilarity function used in FCM is given as Equation (4):

\[
J(U,c_1,c_2, ..., c_n) = \sum_{i=1}^{n} \sum_{j=1}^{c} u_{ij}^{m} d_{ij}^{2}
\]

where, \( u_{ij} \) is between 0 and 1, \( x_j \) is the \( j \)th data point in the dataset, \( c_i \) is the centroid of cluster \( i \), \( d_{ij} \) is the Euclidian distance between \( i \)th centroid and \( j \)th data point; \( m \in [1, \infty) \) is a weighting exponent.

To reach a minimum of dissimilarity function there are two conditions. These are given in Equation (5) and Equation (6).

\[
c_i = \frac{\sum_{j=1}^{n} u_{ij}^{m} x_j}{\sum_{j=1}^{n} u_{ij}^{m}}
\]

\[
u_{ij} = \frac{1}{\sum_{k=1}^{c} \left( \frac{d_{ik}}{d_{ij}} \right)^{2/(m-1)}}
\]

Detailed algorithm of fuzzy c-means proposed by Bezdek in 1973 can be found in [16].

#### B. EFCM: An Extended FCM-based Instance Selection

Taking into account of the training dataset for our TCM-KNN anomaly detection algorithm, it only includes normal data (attack free dataset) [10]. In this case, we claim empirically by a series of experiments and the complexity analysis in Section III, that the size of normal data is huge and a lot of them are abundant and irrelevant to the detection performance of TCM-KNN, thus it unnecessarily increases the computational cost of TCM-KNN.

Therefore, we should remove the irrelevant normal data from the training dataset by using our implemented EFCM (Extend FCM) algorithm. Firstly, we categorize the normal training dataset into three classes (see Figure 2): notable data, obscure data and redundant data. We refer to the notable data as what belong to and represent a series of clusters and they are most centric to a cluster. Those data represent the most classical normal network traffic and network user patterns, which can be utilized to effectively detect the network anomalies. The obscure data denotes those that belong to any cluster with very small (or under a very small constant) membership grades calculated by EFCM. In fact, they represent the most effective patterns that can be used to distinguish normal from abnormal since they closely lie on the right boundary of normal and abnormal points. The redundant data then includes the remaining data that belong to none of the above two classes in the training dataset, which are essentially those data that lie on the boundaries of clusters. Although these points near the boundaries are most effective in classification problems, they are not necessarily useful for our network anomaly detection application, because these points lie
boundaries between two “normal” clusters and thus make no any positive contribution to anomaly detection, instead they increase the computational costs of current anomaly detection algorithms. Secondly, we choose the notable data, the whole obscure data to form the new training dataset, thus removing all the redundant data (Step 5 to Step 8 to be discussed later). In the process of selecting the notable data, we deliberately choose the first $K$ data points with the $K$ highest membership grades (in common sense, their membership grades often exceed 0.8) for each cluster, which would represent the most notable normal activities in the training dataset. As for the selection of obscure data, we choose these data points the difference of whose membership grades for any two different clusters is less than 0.5, and these data points represent the most obscure normal (although it is obscure between normal and abnormal, it is still normal) activities in the training dataset.

The more detailed steps of the instance selection algorithm based on EFCM are depicted as follows:

**Step 1.** Randomly initialize the membership matrix $(U)$ that has constraints in Equation (3).

**Step 2.** Calculate centroids $(c_i)$ by using Equation (5).

**Step 3.** Compute dissimilarity between centroids and data points using Equation (4). Stop if its improvement over previous iteration is below a threshold.

**Step 4.** Compute a new $U$ using Equation (6). Go to Step 2.

**Step 5.** Choose the data points the difference of whose membership grades for any two different clusters is less than 0.5, and then add them to subset $STR$ and delete them from the original dataset $TR$.

**Step 6.** Rank the matrix $(U)$ for each formed cluster in descending order.

**Step 7.** Select the batch of data points with the first $k$ highest membership grades and the membership grades must be bigger than 0.8 for each cluster in the training dataset $TR$, and delete them from $TR$. If the satisfactory data points for each cluster are less than $k$, add the satisfactory ones to the new training dataset $STR$.

**Step 8.** Output $STR$ as the final reduced training subset.

Since our TCM-KNN algorithm is trained with the most notable and obscure normal data, it can make the best use of them to distinguish the abnormal from the normal with high detection rate and low false positive rate, as well as with the cheap computational cost. The next section will give the corresponding experiments to prove.

It is worth noting here that when using EFCM for clustering as illustrated in Step 1 to Step 4, two parameters should be seriously considered for they directly influence the clustering effectiveness: the weighting exponent $m$ and the number of clusters $c$. Since literature [16] has proved that the best value for $m$ belongs to $[1.5, 2.5]$ and the most researchers use $m=2$, we will use $m=2$ for our experiments in this paper in terms of experiences.

As for the second parameter $c$, pal and Bezdek had proved the Xie-Beni index provided the best response over a wide range of choices (such as Bezdek index, Fukuyama-Sugeno index, etc.) for the number of clusters in 1995 [17]. The Xie-Beni index is defined as

$$S = \frac{\sum_{i=1}^{C} \sum_{j=1}^{N} \mu_{ij}^m ||V_i - X_j||^2}{n \times \min_{i,j} ||V_i - V_j||^2}$$

where $n$ is the number of data points in training dataset, $c$ is the number of clusters, $u_{ij}$ denotes the membership grades of data point $i$ belonging to cluster $j$. $||V_i - X_j||^2$ represents the distance between data point $i$ and the centroid of cluster $j$, $||V_i - V_j||^2$ is the distance between the centroids of two clusters. Determining the number of cluster can be described as the following steps:

**Step 1.** Assume: $c \in [2, n^{1/3}]$, which is based on the previous researches and experiences [17].

**Step 2.** Calculate the corresponding value of $S$ for each $c (2 \leq c \leq n^{1/3})$.

**Step 3.** Select the $c$ value as the final number of cluster when the smallest $S$ appears.

### V. Experimental Results

#### A. Dataset Description and Preprocessing

We performed a series of experiments on a real research network in China. The reason not selecting the classic KDD Cup 1999 intrusion dataset as our dataset in this paper is that it was criticized by many researchers for its several inherent problems [18, 19, 20], therefore, it is no longer quite suitable for evaluating our proposed method and giving the most convincing experimental results.

For our real network environment, we selected a Gigabit Ethernet link between a large ISP and a college. The experimental environment contains both standard network services like web traffic, ftp traffic, peer-to-peer (P2P) application traffic, online games as well as live audio and video streams, therefore, the more it is like the real network environment, the more convincing results we may claim. That is to say, we want to demonstrate the effectiveness of our anomaly detection method is independent of the experimental environment. We setup a web server located in the college environment.
running apache http service (version 2.2) on Linux platform (Red Hat Enterprise Edition 4, kernel 2.6.9-42.0.2.EL). We conducted many experiments over several days during busy hours and with background traffic generated from more than 5000 hosts of the college.

In the experiments, the attackers can access the victim web server and they launched many fashionable attacks collected from the following sources: the SecurityFocus archives http://www.SecurityFocus.com/; the Open Source Vulnerability Database http://www.osvdb.org/. These attacks include SQL injection, DNS cache poisoning, DDoS (Distributed Denial-of-Service) attacks, etc. Moreover, in terms of its most serious impact on web server, some well-known DDoS attacks were extensively launched by using a series of DDoS tools such as Stacheldraht [24] and TFN2K [25]. We performed many flooding attacks with spoofed IP’s like SYN Floods, UDP and ICMP attacks [26, 28]. The environment and dataset was also applied to network anomaly detection scenario by us in [29] and [30].

Moreover, we placed the detection engine in a monitor host to periodically collect the web server status info, as well as determine if the web server encounters anomalies by using various classic network anomaly detection algorithms.

In the controlled real network environment as described earlier, we periodically collected and did some processing (manually removed the attack data in the training period to ensure its cleanness), then formed a final dataset containing 86,400 normal data points as our training dataset for network anomaly detection algorithms, which was collected for each 5 seconds from Monday (Sep 15th, 2008) 00:00 PM to Friday (Sep 19th, 2008) 24:00 AM, and they are all normal status before the web server encounters various attacks discussed earlier in a monitor environment to ensure the quality of training data. The dataset contains a series of five tuples (one-way delay, request/response delay, packet loss, overall transaction duration, delay variation). In other words, each point (feature vector) in the training dataset is 5-tuple. Moreover, we also collected 24 hours’ (from 00:00 PM to 24:00 AM on Sep 22nd, 2008) statistical data points (the time slot for sampling is also 5 seconds) after launching many attacks in this period, thus we got 17,280 five tuples that contain the representatives of the anomalous (about three hours’ data points representing the abnormal status, the number is 2,160) and normal statistical status (about 21 hours’ data points reflecting the normal status, the total is 15,120) of the web server experienced by the end users. We used it as a separate test dataset to verify the detection performance of our anomaly detection engine as well as the effectiveness of them after adopting the instance selection mechanisms. Before beginning our experiments, we preprocessed the dataset. In order to avoid one value will dominate another for the numerical data, we normalized the dataset by replacing each attribute value with its distance to the mean of all the values for that attribute in the instance space.

B. Comparison Results

The effectiveness of our TCM-KNN in network anomaly detection compared to the previous techniques has been thoroughly analyzed and compared them in [10] over classic KDD Cup 1999 intrusion dataset [18, 19, 20, 21], we found TCM-KNN performed well both in true positive rates and false positive rates in comparison with the best three algorithms: one-class SVM, fixed-width clustering and KNN score. Therefore, in this paper, we will not discuss them in detail in term of the space limitation of this paper.

C. Experimental Results after Instance Selection

1) Optimization results for detection performance

To verify the effectiveness and efficiency of EFCM for anomaly detection, we designed the following experiments. We first employed EFCM on the training dataset (86,400 normal data points). In the process of instance selection, we also determined the number of cluster c=22 using Xie-Beni index, determined the weighting exponent m for EFCM as 2 and selected the data points with the top 400 highest membership grades for each cluster by a lot of experiments (i.e., K = 400). We consequently selected 22*400+538=9,338, where 538 is the number of obscure data points among them.

Figure 3 depicts the comparison results of the training time (for training) and the detection time (for test) before and after using EFCM instance selection methods for TCM-KNN. Table I list the relevant detection rate and false positive rate (see the 2nd and the 3rd column) in these two cases, all the performances results were obtained in the context of varying the confidence measure as our threshold and selected the average performances in comparison. We can clear see that although the training dataset is greatly reduced (the reduction rate is about 90%), TCM-KNN still retains good detection performance, which greatly attributes to the effective EFCM-based instance selection

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Detection rate</th>
<th>False positive rate</th>
<th>Memory usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCM-KNN</td>
<td>99.40%</td>
<td>1.28%</td>
<td>1,688KB</td>
</tr>
<tr>
<td>TCM-KNN+EFCM</td>
<td>99.48%</td>
<td>1.03%</td>
<td>183KB</td>
</tr>
</tbody>
</table>

Figure 3 depicts the comparison results of the training time (for training) and the detection time (for test) before and after using EFCM instance selection methods for TCM-KNN. Table I list the relevant detection rate and false positive rate (see the 2nd and the 3rd column) in these two cases, all the performances results were obtained in the context of varying the confidence measure as our threshold and selected the average performances in comparison. We can clear see that although the training dataset is greatly reduced (the reduction rate is about 90%), TCM-KNN still retains good detection performance, which greatly attributes to the effective EFCM-based instance selection.
proposed in this paper. The most important thing we found is that both the training time and detection time is reduced to a large extent. Moreover, the last column in Table I clearly showed that the memory usage for TCM-KNN was greatly reduced (from 1,688 KB to 183KB, the reduction rate was about 90%) because of EFCM instance selection mechanism. That is, we could yield computational effective TCM-KNN algorithm by utilizing EFCM instance selection scheme, which may be very significative for real-time online network anomaly detection scenario.

2) Impact of K value selection on detection performance

In addition, we investigated on the principles of selecting the K value of EFCM and the effect of K on the detection performance using a lot of experiments. Table II illustrated the results when we used the same 86,400 data points for training dataset as used in the last experiment. We found that when we just selected about the same number of data points from each of the 22 clusters as that of obscure data points, the most ideal detection performance would be reached.

<table>
<thead>
<tr>
<th>K value</th>
<th>Detection rate</th>
<th>False positive rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>85.40%</td>
<td>9.28%</td>
</tr>
<tr>
<td>200</td>
<td>88.34%</td>
<td>9.03%</td>
</tr>
<tr>
<td>300</td>
<td>95.74%</td>
<td>5.67%</td>
</tr>
<tr>
<td>400</td>
<td>99.48%</td>
<td>1.03%</td>
</tr>
<tr>
<td>500</td>
<td>99.48%</td>
<td>1.03%</td>
</tr>
<tr>
<td>1000</td>
<td>99.48%</td>
<td>1.03%</td>
</tr>
<tr>
<td>1500</td>
<td>99.48%</td>
<td>1.03%</td>
</tr>
<tr>
<td>2000</td>
<td>99.48%</td>
<td>1.03%</td>
</tr>
<tr>
<td>3000</td>
<td>99.48%</td>
<td>1.19%</td>
</tr>
<tr>
<td>4000</td>
<td>99.40%</td>
<td>1.28%</td>
</tr>
</tbody>
</table>

In more detail, since we got 538 obscure data points and 22 clusters, therefore, finally we chose 400 as our K value for instance selection parameter (500, 600 are also effective while 400 is the smallest among them). The results depicted in Table II clearly demonstrated the rationality. When K is under 400, the detection performance is not ideal enough. With the increase of K, the performance becomes better, while when K is between 500 (a little less than 538) and 2000, the change of performance is not so obvious and it reaches a steady state. However, when we continue to increase the K value until it reaches 3000 and 4000, the detection performance is changing and at that time it is almost the same as the initial value when we don’t use the instance selection mechanism, since the data points we choose is almost the whole points in the training dataset when we increase K to 4000. That is, if we increase K after 400, we cannot obtain obviously better detection performance, on the contrary, we would get too more data points, which thus eventually increase the computational cost of network anomaly detection algorithm. It is worth noting that if the number of data points in any cluster is less than that of the obscure data points, selecting all the data points in that cluster is reasonable for the selection process.

D. Discussions

After employing EFCM instance selection mechanism aiming at optimizing TCM-KNN, we found its detection performances became even better (better detection rate and false positive rate). Moreover, the size of training dataset was reduced almost 90%, and the training time (the time for calculating strangeness and p-values as presented in Section III for each data points) is reduced from 0.4636s to 0.1045s (the reduction rate is about 77.5%), the detection time was reduced from 0.4573s to 0.1047s (the reduction rate is also about 77.1%). Since TCM-KNN algorithm judges the data point one at a time with the training dataset, it is rather meaningful to greatly reduce the training dataset, thus reducing the training time and detection time, and the experimental results is very optimistic. Therefore, based on these results, we argue that the proposed methods in this paper make TCM-KNN be a good lightweight and near real-time anomaly detection method.

In addition, we also explored the impact of instance selection scheme on the other classical network anomaly detection algorithms stated earlier (including one-class SVM, KNN score and fixed-width clustering). The experimental results have demonstrated:

a) After instance selection, all the algorithms showed better detection performance than themselves when not adopting the optimization;

b) TCM-KNN also performed much better than the others did after adopting instance selection method presented in this paper, which was consistent to the previous results before utilizing the instance selection for optimization. In terms of the limitation of this paper, the detailed information could be found in our previous technical report [27].

VI. CONCLUSIONS AND FUTURE WORK

In this paper, to boost the detection performance and availability of our previously proposed TCM-KNN algorithm, we introduce an extended FCM-based instance selection mechanism - EFCM, to optimize it as a good candidate for on-line anomaly detection. The experimental results demonstrate it could be optimized greatly thereby be a good candidate for real-time anomaly detection in practice.

In the near future, we would like to use the more effective instance selection mechanism in our web server anomaly detection engine [29, 30] for better network anomaly detection performance.
ACKNOWLEDGMENT

This work has been supported by the National Basic Research Program of China (973 Program) under Grant No. 2007CB311100, the National High Technology Research and Development Program of China (863 Program) under Grant No.2007AA01Z446, and the National Natural Science Foundation of China under Grant No.60873217.

We authors wish to sincerely thank the anonymous reviewers of IEEE Globecom 2009 for their valuable comments on this paper.

REFERENCES