Alexandria Engineering Journal (2022) 61, 2106–2115

Alexandria University

Alexandria Engineering Journal

www.elsevier.com/locate/aej
www.sciencedirect.com

Algorithm optimization and anomaly detection simulation based on extended Jarvis-Patrick clustering and outlier detection

Wei Wang [a,*], **Xiaohui Hu** [b], **Yao Du** [a]

[a] *College of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China*
[b] *Institute of Software Chinese Academy of Science, Beijing 100190, China*

Received 21 March 2021; revised 22 April 2021; accepted 1 August 2021
Available online 18 August 2021

KEYWORDS

Anomaly Detection;
Jarvis-Patrick Clustering;
Extended Shared Nearest Neighbor;
Outlier Detection

Abstract In this paper, the authors analyze the algorithm optimization and anomaly detection simulation based on extended jarvis-patrick clustering and outlier detection. We perform detection by using the jarvis-patrick graph-based clustering method. After that, to further improve the false alarm rate (FAR) of the algorithm, we use an extra outlier detection step combined with our proposed EJP to create a new anomaly detection method called LD-EJP. Using LD-EJP, the false alarm rate improved much (experiments show that the false alarm rate can reach 4.1% while the best JP clustering can reach is 7.4%). Then, we tested LD-EJP against two other anomaly detection methods using k-means and LGCCB, showing that our algorithm has a better detection rate and false alarm rate than these two clustering-based anomaly detection methods. In addition, the detection rate and false positives of the algorithm also have some room for improvement. In the labeling process, the proportion of anomaly clusters to normal clusters needs to be manually adjusted to find a better detection rate. In addition, the detection rate we chose can consume some of the detection rate gained in extended JP clustering to have the LD-EJP obtain a better FAR. Therefore, our future work contains finding or proposing another outlier detection algorithm with better performance than our LD-EJP method.

1. Introduction

Anomaly detection methods mainly use a series of modeling analyses for existing data, from which anomaly data can be identified and recorded. However, with the development of our modern society, extracting and identifying threats from higher dimension data and dealing with unknown attack types are problems that need to be solved urgently in the field [1]. Not only that, but algorithms used for anomaly detection also require precision and efficiency. In the field of anomaly detection, there are two widely accepted assumptions: one is that the majority of the data behaviors are normal behaviors and only a few behaviors are malicious and the other is that attack

* Corresponding author.
E-mail address: ww56050006ww@aliyun.com (W. Wang).
Peer review under responsibility of Faculty of Engineering, Alexandria University.

https://doi.org/10.1016/j.aej.2021.08.009

behaviors are significantly different from normal behaviors. Under the condition of the two assumptions, currently, many experts are interested in bringing many types of data mining techniques, such as clustering, regression analysis, neural networks, etc., into this research field [2–3].

As one of the most important data mining algorithms, clustering is a commonly used unsupervised algorithm for grouping objects into clusters, where objects having the same features can be gathered together and objects with no similarities can be separated into different clusters [4]. Due to the lack of labeled data in many cases, unsupervised learning algorithms represented by clustering have an advantage over supervised learning algorithms, as this kind of algorithm does not need to determine the training model in advance. However, the target data (including the known and the unknown types of anomaly data) are treated equally as input data that can be used directly for anomaly detection [5–6]. During recent years, many experts have adopted a few clustering methods for the anomaly detection field. *K*-means is a clustering method that was earlier used in the anomaly detection field but has its own limitations, such as cluster dependency and degeneracy. Y-means improves the *k*-means method by replacing the empty clusters generated by the method, removing the influence of outliers, and merging similar clusters to further optimize the clustering center [7]. In addition, many intrusion detection methods use *k*-means and other algorithms to improve the effectiveness of *k*-means algorithm, such as, which uses an ID3 decision tree cascade *k*-means method to reduce the problem of mandatory allocation and class dominance that may occur when *k* obtains certain values in *k*-means. Fuzzy-C means is another common clustering algorithm but has some problems when applied in the field of intrusion detection. For example, it is very sensitive to isolated points, its clustering results depend heavily on the cluster number *k* (similar to *k*-means), and its cluster centers must be determined in advance. From this point of view, graph-based clustering algorithms are very advantageous and they often do not need to consider the problem of clustering centers. These algorithms first abstract the data into points and determines whether two points belong to the same cluster by determining the similarity between the two points [8–9]. It proposes an anomaly detection approach using a graph-based algorithm by calculating the distance between the two points and setting the threshold to determine the similarity of these two points. In this paper, we propose a new graph-based approach for anomaly detection called Local Outlier Detection-based Extended Jarvis-Patrick Clustering (LD-EJP). The major contributions of the current work are two-fold. First, we perform detection using the Jarvis-Patrick Graph-based Clustering method on some datasets. Jarvis-Patrick Graph-based Clustering has the advantage of dealing with clusters of different size/shape and density and has good effects on high-dimensional data [10]. Additionally, it is good at discovering clusters of strongly correlated points when compared to other clustering approaches. After we finished a labeling process of all the data, however, the detection rate (DR) results obtained using this algorithm were not very satisfactory because the clustering results obtained using this algorithm are always fragmented and the number of small clusters and outliers is too large. Second, we propose an Extended Jarvis-Patrick Clustering Algorithm (EJP) by extending the conditions of shared *k*-nearest neighbors [11–12]. Then, we use the proposed EJP to experiment using the KDD Cup99 dataset, and the results show that our algorithm improved the detection rate when compared to the original JP clustering (the best detection rate of JP is only 78.7% while EJP can reach 97.10%). After that, to further improve the false alarm rate (FAR) of the algorithm, we use an extra outlier detection step combined with our proposed EJP to create a new anomaly detection method called LD-EJP. Using LD-EJP, the false alarm rate improved much (experiments show that the false alarm rate can reach 4.21% while the best JP clustering can reach is 7.47%). Then, we tested LD-EJP against two other anomaly detection methods using *k*-means and LGCCB, showing that our algorithm has a better detection rate and false alarm rate than these two clustering-based anomaly detection methods.

The remainder of this paper is organized as follows. In the next section, we will introduce the general process of the JP clustering algorithm and then put the algorithm directly into the real dataset using the results to explain its shortcomings when applied to anomaly detection. In Section 3, we describe the Extended JP Clustering algorithm that we proposed. Section 4 describes our LD-EJP method in detail and shows our experimental results when compared to other clustering results. Finally, we summarize our contributions and point out our future work in this field.

2. Jarvis-Patrick clustering algorithm.

The Jarvis-Patrick clustering algorithm, proposed by R.A. Jarvis and Edward A. Patrick, is a kind of bottom-up clustering algorithm. The algorithm is shown in Algorithm 1. The main idea of the algorithm is to first find the *k*-nearest neighbor of each data point and second, find all the points in the graph that have a shared *k*-nearest neighbor relation [13]. If two points have a shared *k*-nearest neighbor relation, these two points can have a line between them, so finally we will obtain a series of subgraphs (clusters) with points in each of these subgraphs having a shared *k*-nearest neighbor relationship between them. Fig. 1 shows an example of a shared *k*-nearest neighbor relation between points. We can see that when $k = 3$, P_1 and P_2 have the same three nearest neighbors, namely, points A, B, and C. Therefore, P_1 and P_2 have a relationship of shared *k*-nearest neighbors when $k = 3$, and for the Jarvis-Patrick Clustering Algorithm, P_1 and P_2 belong to the same cluster.

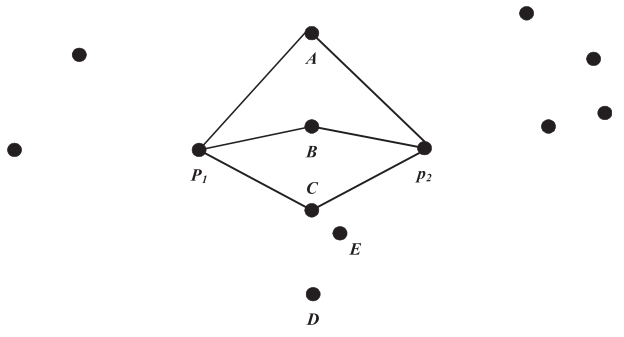

Fig. 1 Shared *k*-nearest neighbor relation of P_1 and P_2 when $k = 3$.

Algorithm 1 (*Jarvis-Patrick Clustering Algorithm*).

```
1. for i ← 0 to DATASET.size(){
2.   for j ← i + 1 to DATASET.size(){
3.      calculateSNN(DATASET[0], DATASET[j]);
4.      }
5.   }
6. Sort the result clusters in ascending order by their size and
accumulate their size at the same time;
7. if accumulated value ≥ α% DATASET.size(){
8. label the rest of the points as anomaly data; others are normal
data;
9.}
```

The algorithm can cluster objects (points) by considering the surrounding environment of each object; it has the ability to deal with clusters of different sizes, shapes, and densities and has a good effect on high dimensional data. However, after we applied the algorithm to some real datasets, we found that it still has some shortcomings:

)1. Because the conditions of the algorithm itself are relatively harsh (*iff* two points have a shared nearest neighbor relationship can they have an edge between them) when compared to other graph-based clustering algorithms, the results of the clustering seem fragmented. That is to say, too many small-sized clusters appear in the clustering results. According to the two assumptions of intrusion detection, only a small part of the data will be labeled as anomaly points; thus, too many small clusters will have a negative influence on the accuracy of the labeling process;
)2. When using the *JP* clustering algorithm is applied to the real-world intrusion detection dataset, its detection rate and false alarm rate are not satisfactory.

The following experiment can explain the above situation. Here, we use the two indicators for the evaluation of anomaly detection: detection rate (DR) and false alarm rate (FAR). The detection rate is the percentage of abnormal points detected by the algorithm (i.e., how many abnormal points can be detected by the algorithm) and the false alarm rate is the percentage of normal points mistaken as abnormal points (i.e., how many normal points will be misjudged as abnormal points by the algorithm). We applied this algorithm to the KDD Cup 1999 dataset. The KDD Cup 1999 dataset is a version of the 1998 DARPA Intrusion Detection Assessment dataset, which was written and managed by MIT Lincoln Labs, contains a variety of intrusions simulated in a military network environment and is an authoritative test dataset for the current intrusion detection field. It includes more than 4,000,000 records and contains 22 different types of attacks. As the size of the data is relatively large, in this experiment, we randomly extract 1000, 2000, and 5000 data points from the dataset for testing. After accomplishment of the clustering, according to the two assumptions mentioned above, we usually refer to the actual size of each cluster when labeling the clustering results. Points in the larger cluster tend to be labeled as normal points, while the others are labeled as anomaly points. It should be noted that according to the two assumptions of anomaly detection we mentioned earlier, the value of α should be consistent with the number points that are labeled as normal being much larger than the number of anomalous points after clustering [14–15]. Therefore, we take the values $\alpha = 0.8$ and 0.9 in these 2 cases. Table 1 shows the best results using JP when extracting 1000, 2000, and 5000 data points at both $\alpha = 0.8$ and 0.9; we can see that 78.7% is the best DR of all these situations. From the table, we can see that the highest detection rate is only 78.7%, which is not very satisfactory when compared to other anomaly detection methods (which have detection rates above 90%). Here, we use Fig. 2 to show the detection rate and false alarm rate of our results when testing with 2000 extracted data points (with this extracted data we obtain the best detection rate), and Fig. 3 to show the number of elements in each cluster *before labeling* with different k values. Fig. 2 shows that the maximum value of the detection rate is 78.7% when $k = 20$, and we can see that if we choose a small k, such as 10, 20, or 30, the detection rate is better than with larger k values, but the best detection rate is 78.7%. Combined with Fig. 3, we find that when the value of k is small (e.g., when $k = 10$), the number of points in each cluster is relatively average, meaning that the difference in size between large clusters and small clusters is not very obvious. This is not good for us because too many small clusters do not meet the two assumptions we mentioned before, thus having a negative influence on our data labeling. With an increase in the k value, the detection rate did not increase but rather decreased. In addition to the detection rate, the false alarm rate is also not very satisfactory. The false alarm rate is generally under 15% when the detection rate is high and the detection rate is generally not good (30–40%) when the false alarm rate is lower (5–9%), as shown in Fig. 2. A reasonable explanation for this is that as the size of the large clusters grows, too many anomaly data points are merged into large clusters (as seen from Fig. 3, when $k > 40$ and the largest cluster is over 800 points, which is 4 times more than the largest one when $k = 20$), and so many anomaly data points can be misjudged as normal at the time of labeling, thereby lowering the detection rate. The same problems occur when testing with more than 2000 data points from the dataset. Too many small clusters can scatter anomalies to each of these small clusters, so once α in Algorithm 1 is given, two problems can arise: one is that α is too high, resulting in a slightly larger cluster labeled as normal and leading to a lower detection rate; the other is that α is too low, resulting in many small clusters being recognized as outliers, making false positives not satisfactory. No matter which problem mentioned above occurs, we cannot avoid the problem that anomaly points are scattered to each small cluster, thus affecting the detection rate[16]. Additionally, Algorithm 1 has a limited maximum detection rate and a high false alarm rate. To solve this problem, we should be able to

Table 1 Best results using JP when extracting 1000, 2000, and 5000 data points.

DR	FAR	α	Data Selected
76.80%	11.44%	0.8	1000
78.70%	9.12%	0.8	2000
74.40%	13.09%	0.8	5000

Fig. 2 Detection rate and false alarm rate of different k for the original JP clustering when extracting 2000 data points ($\alpha = 0.8$).

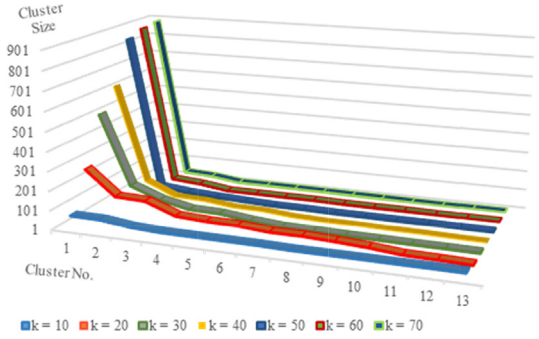

Fig. 3 Distribution of cluster size for the original JP clustering results before labeling when extracting 2000 data points ($\alpha = 0.8$).

control the growth of cluster size based on the original *JP* clustering algorithm appropriately, so that large clusters will not grow too fast and clustering results appear without too many small clusters, which is more convenient for obtaining a reasonable labeling result, thus improving the detection rate. Therefore, in the next section, we will introduce an improved *JP* clustering algorithm that can improve these defects.

3. An extended JP clustering algorithm

From the previous section, we can infer that there are some limitations in detection using the JP clustering algorithm directly for the KDD Cup99 data, so we consider improving the JP clustering algorithm. It can be seen from the experiment in the previous section that the JP algorithm has a high detection rate when the value of k is small, but the number of small clusters and outliers is too large, so we consider extending the conditions of shared k-nearest neighbors[17–18].

3.1. Algorithm description

Before explaining our algorithm, we first present some definitions.

Definition 1 (*Extended Shared K-Nearest Neighbors*). *For a directed graph G, for any two points w_p and w_q in G, if w_p is the k-nearest neighbor of w_q and w_q and it is the $(k + m)$-nearest neighbor of w_p (where m is an arbitrary integer), then there exists an edge $e_{pq} = <w_p, w_q>$ between w_p and w_q, which means w_p and w_q are extended shared k-nearest neighbor relations.*

Definition 2 (*Extended Shared K-Nearest Neighbor Clustering Graph*). *A directed graph G consisting of a set of nodes $W = \{w_1, w_2, \ldots, w_n\}$ and an edge set $E = \{e_1, e_2, \ldots, e_m\}$ is called an extended shared k-nearest neighbor clustering graph, where $e_i = <w_p, w_q>$, $w_p, w_q \in W$, $i = 1, 2, 3, \cdots, n$. There exists an edge e_i if w_p and w_q are extended shared k-nearest neighbor relations.*

According to the definitions above, we extend the concept of shared k-nearest neighbors to the *JP* clustering algorithm; then, after the clustering process, we are able to obtain an extended shared k-nearest neighbor clustering subgraph(s). Fig. 4 shows an example of two points P_3 and P_4 of the extended shared k-nearest neighbor relationship. Here, we can see that when $k = 4$ and $m = 1$, P_3 has 4 nearest neighbors A, B, C, and D, while P_4 has 4 nearest neighbors A, B, C, and E. If we use the extended shared k-nearest neighbor relationship and set $m = 1$, then P_4's $(k + m)$-nearest neighbors are A, B, C, D, and E, so P_3 and P_4 can be in one cluster when $k = 4$. From Fig. 4 we can also see that under the same k-value, the proposed extended shared k-nearest neighbor relationship can merge part of the clusters that were originally split (and sometimes can split some of JP's clusters if m is lower than k) by the JP cluster, to reduce the number of clusters in the final clustering result. All steps of the extended JP clustering algorithm (EJP) are shown below. In these steps, we first construct every cluster (subgraph) by setting edges between each point, then we accumulate each cluster's size in ascending order. If the total accumulated value is more than α%, the clusters we accumulated already are labeled as normal data while others are labeled as anomaly data.

Algorithm 2 (*Extended JP Clustering Algorithm*).

```
1. for i ← 0 to DATASET.size(){
2.   for j ← i + 1 to DATASET.size(){
3.     calculateExtendSNN(DATASET[0], DATASET[j]);
4.     }
5.   }
6. Sort the result clusters in ascending order by their size, and
accumulate their size at the same time;
7. if accumulated value ≥ α% DATASET.size(){
8. label the rest points as anomaly data; others are normal data;
9.}
```

3.2. Experiments

We still use the KDD CUP99 dataset for our experiment. It is worth noting that the KDD CUP99 dataset has a total of 41 attributes, of which approximately half of the attribute types are continuous and the remainder are discrete. Different attribute types can have large differences; for example, text-type

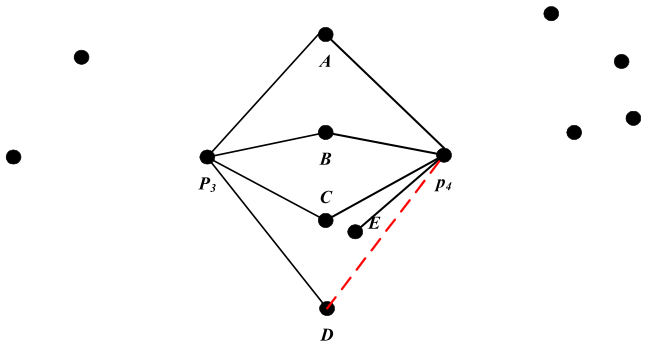

Fig. 4 Extended shared k-nearest neighbor relation of P_3 and P_4 when $k = 4$, $m = 1$.

data and the data of the numeric type cannot be put together to calculate directly. In addition, some attributes used to represent a certain type of data (such as time) can be measured in different units. Therefore, it is necessary to normalize the attribute values of the data before using an algorithm. The normalization method is shown as below:

For continuous attributes We use the following formula to normalize continuous attributes: first, we calculate the average value m_j and the average absolute offset S_j of each attribute value,

$$m_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}$$

$$S_j = \frac{1}{n}\sum_{i=1}^{n} |x_{ij} - m_j|$$

where n represents the total number of data elements in the dataset and x_{ij} represents the jth attribute value of the attribute x of the ith data. Therefore, the normalized value z_{ij} that corresponding to the attribute x_{ij} is:

$$z_{ij} = \frac{x_{ij} - m_j}{S_j}$$

For discrete attributes The KDD CUP99 dataset has nine kinds of discrete attributes, of which there are six kinds of attributes that can only take value 1 or 0, so these six attributes do not require normalization. For the other three attributes (*protocol-type, service,* and *flag*), we use the following methods for normalization. First, for a discrete attribute A, we use $A(a_1, a_2, \cdots, a_n)$ to denote the set of all possible values. Then, $A(a_1, a_2, \cdots, a_n)$ is mapped to the $|A|$ coordinates in the feature space ($|A|$ represents the number of elements in A). For $a_i (i = 1, 2, \cdots, n)$ of the attribute A, given a value, the corresponding coordinate value is $\frac{1}{|A|}$, and the other $|A - 1|$ coordinates are zero. In addition, in some cases after we extracted data from the dataset, we found that there are too many anomaly data points, which cannot match the two assumptions in the anomaly detection field we mentioned[19–21]. Therefore, in this situation, we replace some anomaly data with normal data extracted randomly from the dataset if needed.

Then, we use these data for the experiment. We randomly extract 1000, 2000, and 5000 data points from the KDD Cup99 dataset to test our extended Jarvis-Patrick clustering algorithm. It should be noted that 1. when $m = 0$, the EJP algorithm is the JP algorithm itself; 2. according to the two assumptions of anomaly detection we mentioned earlier, the value of α should be consistent with the number of points labeled as normal being much larger than the number of anomalous points after clustering. Therefore, we take the values $\alpha = 0.8$ and 0.9 for these two cases as before. After the clustering and labeling processes, we recorded the number of elements in each cluster for different values of k to explain the effect of our algorithm. Here, we selected the same three groups of data extracted from the dataset[22]. Table 2 shows the maximum detection rate (with the best false alarm rate) after experimenting with these three sets of data. It should be noted that, for Table 2, Table 3, and Fig. 9, we count the value of $(k + m)$ rather than k because m may be negative and the use of positive integers can facilitate the study, making our statistics less error-prone. Therefore, during the process of the experiment, we keep our original data record using the value of $(k + m)$. We see that the detection rate has been greatly improved over that of Fig. 2 (using the original JP algorithm with a detection rate of up to 78.2%). The best results have a detection rate as high as 97.1% when extracting 5000 data points. In the meantime, in the case of the highest detection rate, $m = 0$ (original JP). Not only can the values of k and m in Table 2 reach a higher detection rate than the original JP, but for other k values, the detection rate of EJP can be higher than the original JP. Figs. 5 and 6 show two examples (using the same 2000 data points as in Figs. 3 and 4) to explain the change in detection rate when we select different values of m for the given value of k. As seen from Fig. 5, in the case of $k = 20$, when m takes a different value (here from −10–80, when the value of $(k + m)$ is from 10 to 100), the maximum detection rate is when $m = 20$ or 30 (and detection rate is up to 95%), which is better than when $m = 0$ (original JP Clustering itself). When $m = 20$ (or $m = 30$ or 40), the largest cluster contains more points than the largest one when $m = 0$. This situation can also be observed when k is other large integers. Fig. 6 shows that when $k = 70$ and m = -50 or −10, the detection rate reaches the largest value (95.7%), while the detection rate is only more than 60% when $m = 0$. Fig. 7 shows the size of clusters with the same $k = 20$ and different m-values after we use the EJP algorithm. At $m = 20$ (where the maximum detection rate is taken), the difference in size between large clusters and small clusters (maximum size is 395 and minimum is 1, and when $m = 0$, the largest cluster size is 252 and the smallest cluster size is 1) has been greatly improved. This shows that our EJP algorithm can improve the detection rate by increasing the cluster size difference. Similarly, when k is a large value, for example, $k = 70$ (Fig. 8), and when m is −10 (detection rate 95.7%), the difference between the largest cluster and the smallest cluster (largest cluster size = 312, smallest size = 1) is reduced compared to $m = 0$ (indicating that the difference between the largest cluster and the smallest cluster is too large or the total number of clusters is too small when $m = 0$, thus causing the detection rate to not be satisfactory). Moreover, when m = -50, the detection rate is also high (94.4%). Although the size difference of clusters is not significant compared to $m = 0$, the total number of clusters increases, which shows that EJP may split some clusters made by the JP algorithms to achieve high detection rates.

Then, we show the effect of our algorithm on the false alarm rate. Of the three best-performing k and m values in Table 2, no $m = 0$ appears, and the false alarm rate was

Table 2 Best results using EJP compared to JP when extracting 1000, 2000, and 5000 data points.

Algorithm	DR	FAR	α	Data Selected	k	m
EJP	96.80%	11.40%	0.8	1000	24	20
JP	76.9%	10.12%			21	
EJP	96.50%	16.12%	0.8	2000	22	17
JP	78.2%	13.84%			30	
EJP	97.10%	18.09%	0.8	5000	16	32
JP	67.4%	7.47%			39	

Table 3 Best results using EJP when extracting 1000, 2000, and 5000 data points.

k	m	DR	FAR	α	Data Selected
24	20	94.50%	4.21%	0.8	1000
22	17	95.60%	4.22%	0.8	2000
16	32	96.40%	4.25%	0.8	5000

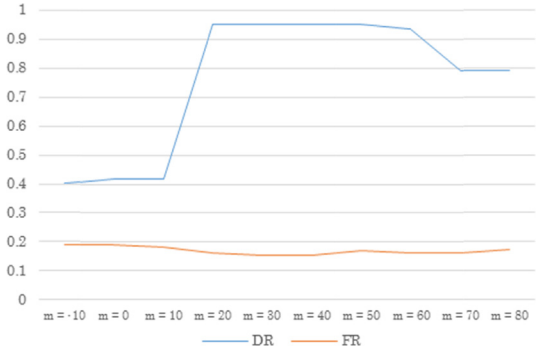

Fig. 5 Detection rate and false alarm rate results when k = 20 and α = 0.8.

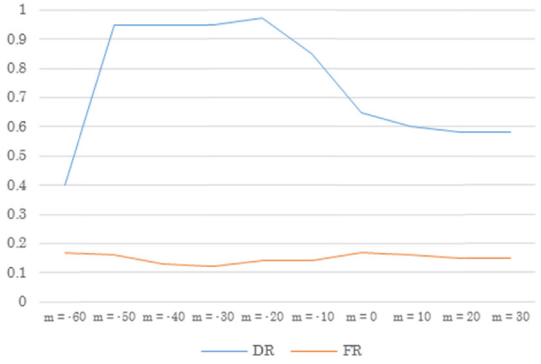

Fig. 6 Detection rate and false alarm rate results when k = 70 and α = 0.8.

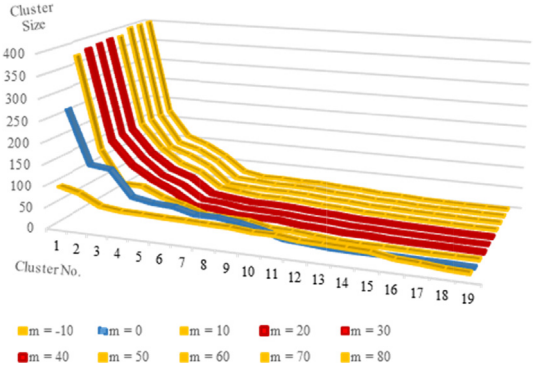

Fig. 7 Distribution of cluster size in EJP clustering results before labeling (k = 20, α = 0.8).

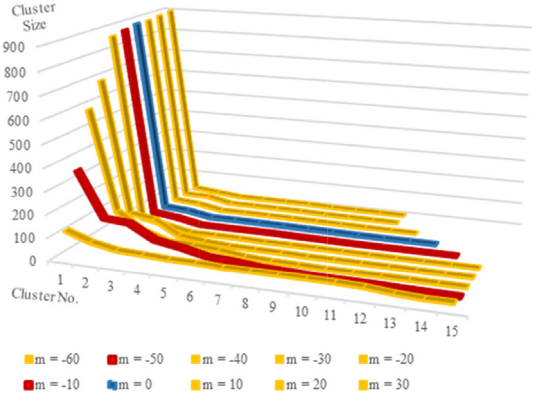

Fig. 8 Distribution of cluster size in EJP clustering results before labeling (k = 70, α = 0.8).

greater than 7.47%. In Fig. 5, $m = 40$ can make false alarm rate achieve the best rate, better than that when $m = 0$; when $k = 70$, as shown in Fig. 6, the false alarm rate reaches the minimum value when $m = -30$, while for $m = 0$ the false alarm rate is over 18%. However, comparing Table 1 to Table 2, we can see that the false alarm rate of the JP algorithm has the possibility to be lower than EJP for the best detection rate

and false alarm rate. However, in general, both the JP algorithm and the EJP algorithm have many false alarm rates higher than 10% while maintaining a high detection rate. This is because when labeling the data, we judged too many points as "noise" ("noise" points are actually points of normal data) in the anomaly clusters [23–24]. In addition, we used $\alpha = 0.8$ in the experiments above to facilitate a comparison with the algorithm proposed in Section 4. In fact, when $\alpha \geq 0.9$, this algorithm's false alarm rate is also not very low while maintaining the higher detection rate. We do not draw any figures showing the result when $\alpha \geq 0.9$ because, using the same data, the results in detection rate are not as good as when $\alpha = 0.8$. Therefore, in the next section, we will introduce a method based on extended JP clustering called *LD-EJP*. This method uses an outlier detection algorithm based on local distance after EJP clustering. The experimental results show that the results of false alarm rate can be improved while detection rate performance can fluctuate in a small range [25].

Fig. 9 shows the detection rate we obtained using the same three sets of data above 79%, which is higher than the maximum detection rate that the JP algorithm can achieve using the same extracted data. Earlier we discussed that the EJP algorithm has the ability to bring out better detection and false alarm rates than the JP algorithm; however, it does not indicate the rates at which k and m for the EJP can obtain the better detection rate and false alarm rate. Fig. 9 is the sum of k and $(k + m)$ for all detection rates higher than 79% (the maximum detection rate achievable by the JP algorithm in the same case, is shown in Table 1) using the same three sets of data, The abscissa represents k and the ordinate represents $(k + m)$. When $\alpha = 0.8$, the values of (k, m) that can achieve $\alpha > 79\%$ are far more than $\alpha = 0.9$, and when $\alpha = 0.9$, a high detection rate cannot be obtained. When $\alpha = 0.8$, taking the values of k and m in the three cases, extracting 1000, 2000, and 5000 data points, we can see that when k is in the range of (14, 21) or (23, 25) there are basically no restrictions on the value of m. If we consider the interval (13, 30) for k, we suggest taking the value of $(k + m)$ within the range (60,100). In general, we suggest that for $\alpha = 0.8$, the value of k is a smaller integer with a range of approximately (15, 30) and the value of $(k + m)$ is (60, 100).

4. LD-EJP algorithm

4.1. Algorithm description

As seen from Section 3, we need to remove points that actually represent normal data as much as possible from small clusters and outliers to improve the false alarm rate. To solve this problem, we use an outlier detection algorithm to determine each of the target points [26–27]. That is, the more likely one point is an isolated point, the more likely it is an anomaly point. Here, we choose one outlier detection algorithm called the local outlier coefficient (LOC) algorithm. This algorithm arranges the target points according to their LOC values and takes the top n points of the largest value as the isolated ones.

Definition 3 (*(k-distance and k-distance neighbor)*). *For any positive integer k and dataset D, the k-distance of object p, denoted ask − dist(p), is defined as the distance dist(p, o)between the point p and the point o ∈ D, which satisfies:*

(1). *at least k points* $q \in D(q \neq p)$, *so thatdist*$(p, q) \leq dist(p, o)$;
(2). *at most (k-1) points* $q \in D(q \neq p)$, *so that* $dist(p, q) < dist(p, o)$;
(3). *Given the k-distance of p, the k-distance neighbor* $N_k(p)$ *of p contains every point whose distance from p is not greater than the k-distance:*

$$N_k(p) = \{q | d(p, q) < k - dist(p), q \neq p\}$$

Definition 4 (*The sum of the local distances of point p*). *The sum of the local distances of point p (denoted as* lds_k*) is defined as the sum of the distance between point p and its nearest k neighbors, as follows:*

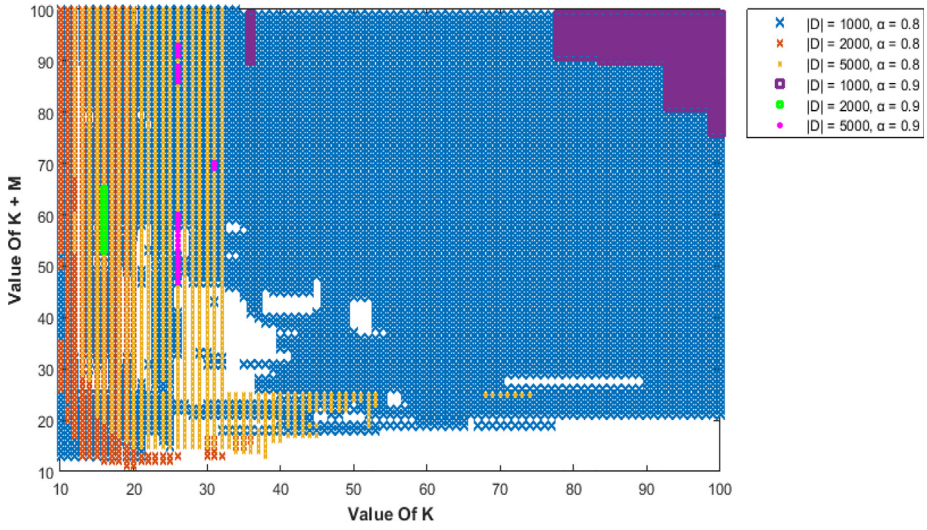

Fig. 9 Relationship of k and m in the extracted 3 sets of data using EJP when $\alpha = 0.8$ and 0.9.

$$lds_k(p) = \sum_{o \in N_k(p)} dist(p, o)$$

where dist (p, o) is the actual distance between the point p and its nearest k points.

Definition 5 (*Local Outlier Coefficient for point p*). *The local outlier coefficient of point p, LOC (p), is defined as the ratio of $lds_k(p)$ to the sum of all the local distances of the k-distance neighbors of p. The formula is as follows:*

$$LOC_k(p) = \frac{lds_k(p)}{\frac{\sum_{o \in N_k(p)} lds_k(o)}{|N_k(p)|}}$$

If the LOC value of a point is low, indicating that the distribution of the neighbors around the point is dense, then it is less likely to be an isolated (anomaly) point; otherwise, it is more likely to be an isolated point, for the distribution of its neighbors is sparser [28].

When computing k-distance and k-distance neighbor, it is necessary to compute the k nearest neighbors of each target point, which was accomplished before by the above-proposed extended JP clustering. Therefore, using this kind of outlier detection can greatly improve computational efficiency. The intrusion detection algorithm in paper uses outlier detection based on the k-distance, but it does not have this advantage. Moreover, the local deviation coefficient-based outlier detection method used in is not very intuitive to understand and will make points that do not belong to any k-nearest neighbors participate in the calculation, which increases the difficulty in the calculation. In contrast, the LOC algorithm is much simpler, easier to understand, and more suitable when combined with our extended JP clustering.

Thus, to combine the LOC algorithm with our proposed JP clustering algorithm, we proposed a new anomaly detection algorithm LD-EJP; the algorithm steps are shown below:

Algorithm 3 (*Local Outlier Detection-based Extended Jarvis-Patrick Clustering*).

```
1. for i ← 0 to DATASET.size(){
2.   for j ← i + 1 to DATASET.size(){
3.     calculateExtendSNN(DATASET[0], DATASET[j]);
4.     }
5.   }
6. Sort the result clusters in ascending order by their size, and
accumulate their size at the same time;
7. if accumulated value ≥ α% DATASET.size(){
8. label the rest points as anomaly data; others are normal data;
9.}
10. Outlier detection on the labeled results above;
```

4.2. Experiments

We apply the entire LD-EJP algorithm to the KDD Cup99 dataset for experimentation using the same randomly selected data points from the previous section to be tested and used for comparison to the classical k-means algorithm and a graph-based clustering detection called LGCCB.

Table 3 shows the DR and FAR of our algorithm. We find that when $\alpha = 0.8$ and when we choose top 25% of the points with high LOC values (so that we actually choose 5% of the entire target dataset as anomaly labeled points), the method can reach its best performance. Compared to Table 2, it can be seen that although this algorithm consumes some of the detection rate obtained by the extended JP algorithm we proposed in Section 3, the false alarm rate has indeed been greatly improved (while maintaining a high detection rate, the false alarm rate is 4% up and down). This is because the LOC values of the anomaly points are relatively high compared to the normal points.

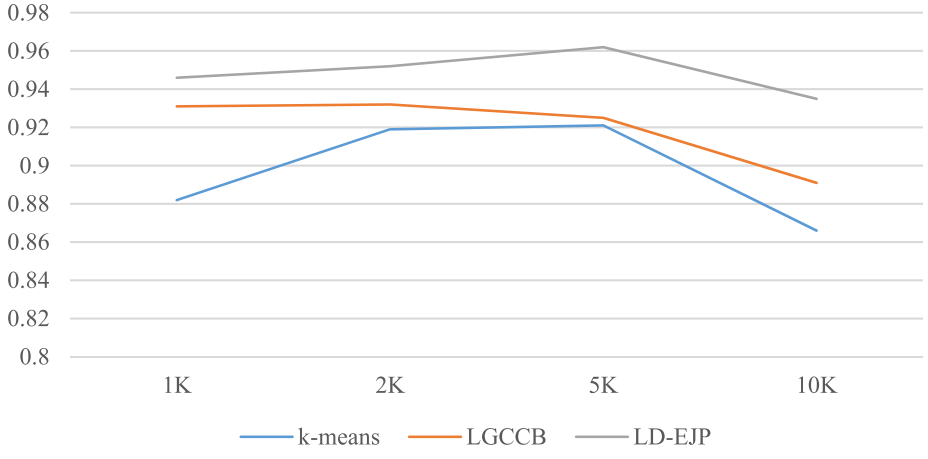

Fig. 10 Detection rate comparison to other cluster-based anomaly detection methods.

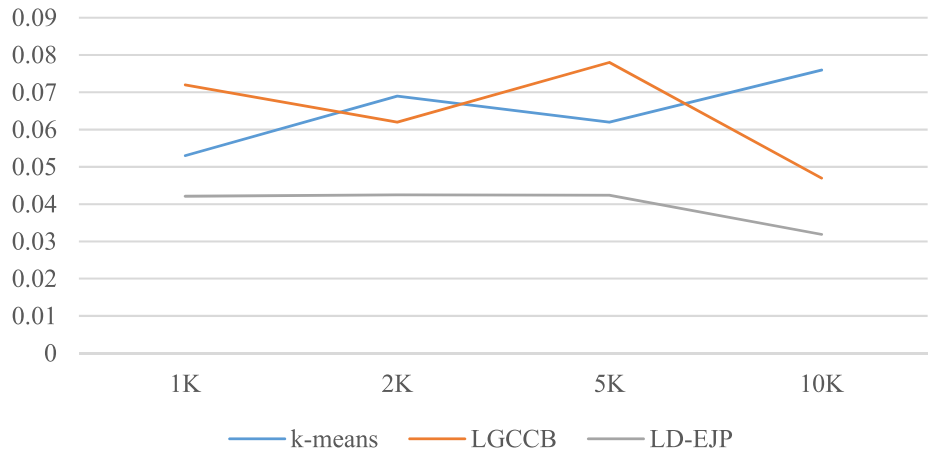

Fig. 11 False alarm rate comparison of other cluster-based anomaly detection methods.

Fig. 10 and Fig. 11 show a comparison among some of these cluster-based anomaly detection methods (i.e., *k*-means and LGCCB) in terms of the detection rate and false alarm rate. These methods use different parameters and cannot be further compared, but through the detection rate and false alarm rate of these two aspects, these figures reveal that the detection rate and false alarm rate of our LD-EJP are better than those of the traditional cluster-based and GB clustering-based anomaly detection.

5. Conclusion and future work

In this paper, we first introduced the Jarvis-Patrick clustering algorithm in the field of anomaly detection and proposed an extended JP clustering algorithm to overcome the shortcomings in the experimental process of JP clustering. After experimentation, we proved that the detection rate of the extended JP clustering algorithm is greatly improved. Then, based on the local outlier coefficient outlier detection algorithm, a new anomaly detection method called *LD-EJP* is proposed to further improve the false alarm rate of the extended JP clustering algorithm. The extended algorithm and outlier detection do not change the time complexity of the JP clustering algorithm, which is $o(m^2)$. For low-dimensional Euclidean data, we can use the method such as k-d tree to compute the *k*-nearest neighbor list, so the time complexity can be reduced to $o(nlogn)$.

In addition, the detection rate and false positives of the algorithm also have some room for improvement. In the labeling process, the proportion of anomaly clusters to normal clusters needs to be manually adjusted to find a better detection rate. In addition, the detection rate we chose can consume some of the detection rate gained in extended JP clustering to have the LD-EJP obtain a better FAR. Therefore, our future work contains finding or proposing another outlier detection algorithm with better performance than our LD-EJP method.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This paper acknowledges funding supported by National Key R&D Program(2019YFB1405100).

References

[1] L. Akoglu, H. Tong, D. Koutra, Graph based anomaly detection and description: a survey, Data Min. Knowl. Disc. (29)3 (2015) 626–688.
[2] Amuthan Prabakar Muniyandia, R. Rajeswarib, R. Rajaramc. Network Anomaly Detection by Cascading K-Means Clustering and C4.5 Decision Tree algorithm. International Conference on Communication Technology and System Design. Procedia Engineering, 30 (2) (2012) 174182
[3] Bigdeli, Elnaz, et al. "A fast noise resilient anomaly detection using GMM-based collective labelling." Science and Information Conference (SAI), 2015. IEEE, 2015,2:102-110.
[4] D. Chakrabarti, Y. Zhan, C. Faloutsos, R-MAT: A Recursive Model for Graph Mining, SDM 4 (2004) 12–17.
[5] Chen, Shi, et al. "A graphical feature generation approach for intrusion detection." MATEC Web of Conferences. Vol. 44. EDP Sciences, 2016,11(2):41-50.
[6] D. Denning, An intrusion-detection model, IEEE computer society Symposium on research security and privacy (1986) 118–131.
[7] S.R. Gaddam, V.V. Phoha, K.S. Balagani, *K*-Means+ ID3: A novel method for supervised anomaly detection by cascading *K*-Means clustering and ID3 decision tree learning methods, IEEE Trans. Knowl. Data Eng. 19 (3) (2007) 88–96.
[8] P. Garcia-Teodoro et al, Anomaly-based network intrusion detection: Techniques, systems and challenges, computers & security 28 (1) (2009) 18–28.
[9] Guan, Yu, Ali-Akbar Ghorbani, and Nabil Belacel. "Y-means: A clustering method for intrusion detection." 2003,15(3):78-85.
[10] Han, Li. "Research of K-MEANS algorithm based on information entropy in anomaly detection." 2012 Fourth International Conference on Multimedia Information Networking and Security. IEEE, 2012,45-64.
[11] Z. He, X.u. Xiaofei, S. Deng, Discovering cluster-based local outliers, Pattern Recogn. Lett. 24 (3) (2003) 1641–1650.
[12] R.A. Jarvis, E.A. Patrick, Clustering using a similarity measure based on shared near neighbors, IEEE Trans. Comput. 100 (11) (1973) 1025–1034.
[13] W. Jiang, M. Yao, J. Yan, Intrusion detection based on improved fuzzy c-means algorithm, Information Science and

Engineering, 2008. ISISE'08. International Symposium on. IEEE, 2008, 2: 326-329.
[14] M. Jianliang, S. Haikun, B. Ling, The application on intrusion detection based on k-means cluster algorithm. In Information Technology and Applications, 2009. IFITA'09. International Forum , 1, pp. 150-152
[15] KDD.KDD Cup1999Data.http://kdd.ics.uci.edu/databases/kddcup99/kdd- cup99.html, 1999.
[16] Kumar, Manoj, and Robin Mathur. "Unsupervised outlier detection technique for intrusion detection in cloud computing." Convergence of Technology (I2CT), 2014 International Conference for. IEEE, 2014.
[17] Leung, Kingsly, and Christopher Leckie. "Unsupervised anomaly detection in network intrusion detection using clusters." Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38. Australian Computer Society, Inc., 2005,12-18.
[18] Lin Ni, Hong-Ying Zheng, An unsupervised intrusion detection method combind clustering with chaos simulated annealing, Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, August 2007.
[19] Ibéria Medeiros, Nuno Neves, Miguel Correia, Detecting and removing web application vulnerabilities with static analysis and data mining, IEEE Trans. Reliab. 65 (1) (2016) 54–69.
[20] Zhou Mingqiang, Huang Hui, Wang Qian, A graph-based clustering algorithm for anomaly intrusion detection, Computer Science & Education (ICCSE), 2012 7th International Conference on, 2012.
[21] Caleb C. Noble, Diane J. Cook, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003, pp. 12–15.
[22] Salima Omar, Asri Ngadi, Hamid H. Jebur, Machine learning techniques for anomaly detection: an overview, International Journal of Computer Applications 79 (2) (2013) 43–52.
[23] N.R. Pal, J.C. Bezdek, On cluster validity for the fuzzy c-means model, IEEE Trans. Fuzzy Syst. 3 (3) (1995) 370–379.
[24] Portnoy, Leonid, Eleazar Eskin, and Sal Stolfo. "Intrusion detection with unlabeled data using clustering." In Proceedings of ACM CSS Workshop on Data Mining Applied to Security (DMSA-2001. 2001.
[25] B. Qiu, C. Jia, J. Shen, Local outlier coefficient-based clustering algorithm[C]//Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on. IEEE, 2: 5859-5862.
[26] Taeshik Shon, Jongsub Moon, A hybrid machine learning approach to network anomaly detection, Inf. Sci. 177 (18) (2007) 3799–3821.
[27] C.F. Tsai, C.Y. Lin, A triangle area based nearest neighbors approach to intrusion detection, Pattern Recogn. 43 (1) (2010) 222–229.
[28] Gang Wang et al, A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering, Expert Syst. Appl. 37 (9) (2010) 6225–6232.