Fractal-Based Outlier Detection Algorithm over RFID Data Streams

http://dx.doi.org/10.3991/ijoe.v12i01.5171

Liansheng Li
Hunan University of Science and Engineering, Yongzhou City, Hunan Province, China

***Abstract*—Nowadays, Radio frequency identification (RFID) has been extensively deployed to retailing, supply chain management, object recognition, object monitoring and tracking and many other fields. Detecting outliers in RFID data streams can help us find abnormal activities and thus avoid disasters. In order to detect outliers in RFID data streams efficiently and effectively, we proposed a fractal based outlier detection algorithm. Firstly, we built a monotone searching space based on the self-similarity of fractal. Then, we proposed two piecewise fractal models for RFID data streams, and presented an outlier detection algorithm based on the piecewise fractal model. Finally, we validated the efficiency and effectiveness of the proposed algorithm by massive experiments.**

***Index Terms*—Radio Frequency Identification; outlier detection; fractal; data streams**

I. INTRODUCTION

Radio frequency identification (RFID) [1] is a kind of contactless automatic identification technique, which utilizes radio frequency signal to transmit data between electrical readers and tags, and thus to monitor and track tagged objects. RFID has the advantages of small tags, low cost and non-contact, and has been extensively deployed to retailing, supply chain management [2], target recognition [3], object monitoring and tracking [4] and so on. Nowadays, RFID is one of the most important techniques in Internet of Things.

In the real time monitoring applications [5, 6], electrical readers are deployed in different locations, and these readers monitor and track distributional tagged objects. When the status or data of the monitored object deviates from the normal value, an outlier occurs [7]. If the outlier is not reported and processed timely, then a devastating result will happen. Usually, RFID readers read data with some fixed frequency, so the data from the same reader can be assumed to be a data stream [8]. However, the huge amount of data, volatility, unreliability and distribution of RFID data streams pose a huge challenge for outlier detection in such systems.

In this paper, we apply fractal [9] to detect outliers in data streams. We use the self-similarity of fractal to sort the searching space into a monotone searching space, use piecewise fractal model to map a data stream into a number of buckets, and detect outliers in these buckets based on the piecewise fractal model. The rest of the paper is organized as follows. In section 2, we review related works about outlier detection in data streams. In section 3, we introduce some basis of fractal. In section 4, we build a monotone searching space based on fractal. In section 5, we propose two piecewise fractal models, and use them to detect outliers in data streams. Experiments and conclusion are given in section 6 and 7 respectively. Full-Sized Camera-Ready (CR) Copy

II. RELATED WORKS

There are lots of studies on outlier (or anomaly) detection in data mining, some classical outlier detection algorithms include FindAllOutsD [10], FindOut [11], LOF [12], LOCI [13], etc. However, these algorithms needs very large time and space complexity while processing large scale data, and cannot deal with streaming data with only one pass through the whole data. Moreover, the outlier detection algorithms proposed in [10, 11] depend on predefined parameters, and are not applicable to the dynamic streaming data.

The outlier detection on streaming data has attracted a lot of researchers. Jagadish et al. [14] proposed the deviant, a special kind of outlier, in a time series database. Based on [14], Muthukrishnan et al. [15] studied a special kind of deviant in time series data streams, and proposed a deviant detection algorithm for large scale data streams. Angiulli and Fassetti [16] proposed a distance based outlier detection algorithm, called STORM, on data streams. However, the STORM algorithm is only suitable to centralized data streams, and doesn't support cooperations between multiple nodes. In outlier detection on distributed data streams, Palpanas et al. [17] proposed an outlier detection framework for signal networks of a distributed environment. In the proposed framework, Palpanas et al. illustrated some problems that needed to be solved, but they didn't give the strategy of how to solve them. In addition, Babcock and Olston [18] proposed a Top-a detection algorithm for data streams; Cormode and Hadjieleftheriou [19] studied the problem of frequent item mining in data streams; Subramaniam et al. [20] analyzed the problem of outlier detection in sensor data streams; Cormode et al. [21] proposed a clustering algorithm for distributed data streams; and Su et al. [22] proposed a kernel density estimation based method for outlier detection. All of the above methods can deal with distributed data streams, but when the dimension of data is very high, performances of all above methods will decrease quickly. For outlier detection in RFID data, Masciari [23] proposed an outlier mining framework for RFID data, but the proposed framework cannot be applied to RFID data streams.

III. BASIS OF FRACTAL

Fractal is the self-similar phenomenon in nature. Massive studies have validated that, self-similarity is ubiquitous in nature. For instance, communicating on the inter-

net, stock dealing, human physiological phenomenon and meteorological data are all self-similar. The power law scaling relationship [9] described in equation 1 is an important feature of fractal model, and it is a useful mathematical tool for describing self-similarity. In equation 1, s represents the feature, such as time or length; q is a measure with respect to s, such as aggregation function; p is proportionality constant; and d is a scaling Index. If we take logarithm of both sides of equation 1, we can get equation 2.

$$q = p \cdot s^d \tag{1}$$

$$\log q = \log p + d \cdot \log s \tag{2}$$

Given a point dataset, if every point (s, q) in the dataset is on the curve described by equation 1, and then the dataset is accurate fractal data. The pow law scaling relationship is a necessary and sufficient condition on an accurate fractal. For accurate fractal data, every point ($\log s, \log q$) is on the line described by equation 2, where d is the slope of the line. For approximate fractal data, $\log q$ and $\log p + d \cdot \log s$ have the same distribution, and d is the fit of all slopes of points $\log s, \log q$.

For discrete time series data or streaming data, equations 1 and 2 can be transformed to equations 3 and 4 [24].

$$q(s,t) = s^d \cdot q(t) \tag{3}$$

$$d = \frac{\log(q(s,t)) - \log(q(t))}{\log s} \tag{4}$$

Where t is the time unit, s is the number of time units for measuring q. In a data stream, s is the size of sliding window.

Piecewise fractal model consists of limited number of shrinking maps $\{M_i : i = 1, \cdots, k\}$, and each shrinking map M_i is related with a group of segments. The mathematical basis of piecewise fractal model is the recurrent iterated function system [25], which can be described by the following equation.

$$M_i \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}^i & a_{12}^i \\ a_{21}^i & a_{22}^i \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1^i \\ a_2^i \end{bmatrix} \tag{5}$$

Where, $i = 1, \cdots, m$, and the shrinking factor a_{22}^i satisfies that $| a_{22}^i | \leq 1$. In order to make the mapping injective, let $| a_{12}^i | = 0$.

In this paper, we aim to build piecewise fractal model for data streams, and detect outliers on the shrinking maps of the piecewise fractal model.

IV. BUILDING SEARCHING SPACE

Current outlier detection algorithms optimize the time overhead via different index techniques while detecting multiple sliding windows. In the worst case, it still needs to check the sliding windows with all lengths, and thus the time complexity is $O(m)$, where m the number of sliding windows with different lengths is.

In this paper, we sort all sliding windows required to be checked, and turn the unordered searching space into an ordered searching space. In an ordered searching space, if an outlier (measured by aggregating function) is detected in some sliding window, then the aggregating value in previous sliding windows can also be assumed to be outliers. So, we can do binary search in the ordered searching space, and the time complexity is only $O(\log m)$.

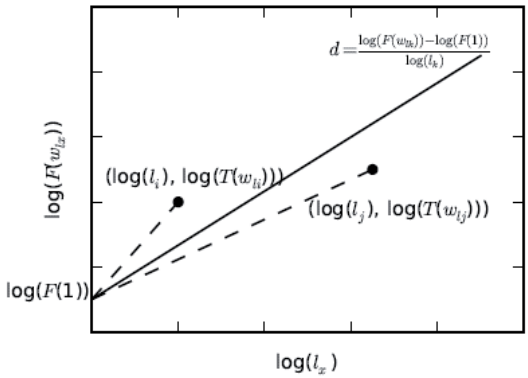

Figure 1. Monotone searching space on an accurate fractal

A. Fractal based searching space

We assume that the data stream $\{X_i : i = 1, \cdots, n\}$ is an accurate fractal. Let the value of aggregating function for one time unit be $F(1)$, then we have $F(1) > 0$. The accurate fractal of $\{X_i : i = 1, \cdots, n\}$ can determine the slope d in equation 2, where d is the exponential of aggregating function F, and can be computed by equation 4. Different aggregating functions on different datasets will result in different exponentials. Given an aggregating function F and its corresponding exponential d, $T(w_{li})$ and $T(w_{lj})$ are the thresholds for checking outliers on the sliding windows w_{li} and w_{lj}, and figure 1 illustrates their relationships. From the figure we can see that, the point $(\log(l_i), \log(T(w_{li})))$ is above the line, and the point $(\log(l_j), \log(T(w_{lj})))$ is below the line. So, we have the following two in equations:

$$\frac{\log(T(w_{li})) - \log(F(1))}{\log(l_i)} > d \tag{6}$$

$$\frac{\log(T(w_{lj})) - \log(F(1))}{\log(l_j)} < d \tag{7}$$

Where, $d = \frac{\log(T(w_{lk})) - \log(F(1))}{\log(l_k)}$.

When processing data stream X, outlier is always detected on w_{lj}, and is never detected on w_{li}. Because of inequations 6 and 7, we have $T(w_{lj}) < F(w_{lj})$ and

$T(w_{li}) < F(w_{li})$. So, we can detect outlier by comparing $d = \frac{\log(T(w_{li})) - \log(F(1))}{\log(l_i)}$ with d.

B. Searching space building algorithm

In this section, we propose an algorithm for building the monotone searching space, and the algorithm is in algorithm 1. Algorithm 1 is an incremental algorithm, and it describes how to update the searching space when new sliding window w_{lk} needs to be checked. When the number of sliding windows required to be checked is 2, and the thresholds are $T(w_{li})$ and $T(w_{lj})$ ($w_{li} < w_{lj}$), then the line constructed by $(\log(l_i), \log(T(w_{li})))$ and $(\log(l_j), \log(T(w_{lj})))$ intersects with the y axis at the point A. While building searching space for two sliding windows, if the point $(0, \log(F(1)_{d^p}))$ is above the point A, then the searching space corresponding to d_p is $< w_{li}, w_{lj} >$; and if the point d_p is below the point A, then the searching space corresponding to d_p is $< w_{lj}, w_{li} >$ When there are more sliding windows, we can use algorithm 1 to update the searching space dynamically.

Algorithm 1: Algorithm of Building monotonic search space ($w_i, T(w_i)$)

1 Create a new point $T_i(\log(l_i), \log(T(w_{li})))$;
2 if $m \geq 1$ then
3 for $j = 1$ to m do
4 line $T_i T_j$ intersect vertical axis at point X_i;
5 if $w_{lj} < w_{li}$ then
6 add $< w_{lj}, w_{li} >$ to lower intervals of X_i;
7 add $< w_{li}, w_{lj} >$ to upper intervals of X_i;
8 else
9 add $< w_{li}, w_{lj} >$ to lower intervals of X_i;
10 add $< w_{lj}, w_{li} >$ to upper intervals of X_i;
11 increase m by 1;

The threshold for detecting outlier is usually predefined, and this algorithm build the monotone searching space offline, so it wouldn't affect the performance of outlier detection for a data stream. If m is the number of sliding windows needed to be checked, then the time complexity of algorithm is $O(m^2)$.

V. OUTLIER DETECTION ALGORITHM FOR PIECEWISE FRACTAL MODEL ON A DATA STREAM

After building the monotone searching space, in order to finish the task of outlier detection efficiently, it only needs to compute the current d_p and compare it with $\frac{\log(T(w_{li})) - \log(F(1))}{\log(l_i)}$. Therefor, in order to maintain the aggregating value of sliding windows with any lengths on a data stream and compute d_p, we propose a piecewise fractal model on a data stream. The proposed model utilizes the self-similarity of aggregating values of sliding windows with any lengths, and thus can compress both the monotone and un-monotone aggregating values. Based on the monotone searching space and the piecewise fractal model on a data stream, we propose an outlier detection algorithm on a data stream.

The power law exponential d_p of the current data stream is the slope of linear regression on point set $\{(\log(l_i), \log(F(w_{li}))) : l_i = 1, \cdots, n\}$, where $F(w_{li})$ is the aggregating function value of the sliding window being checked. If F is the sum function, then $F(w_{li})$ is the sum of suffix of the current data stream $x_i : l_i = 1, \cdots, n$, i.e. $F(w_{li}) = \sum_{j=n-l_i+1}^{n} x_j$. Therefor, the overhead for maintaining d_p on a data stream is very high, and it cannot be implemented incrementally. However, if the points is piecewise in set $\{(\log(l_i), \log(F(w_{li}))) : l_i = 1, \cdots, n\}$, then it only needs to keep the endpoints for each segment, for the regression line with these endpoints is equal to the regression line with all points.

In the rest of the paper, we assume F is the sum aggregating function.

A. Piecewise fractal model on a data stream

In a data stream, let the time stamp be i and the value be y_i, then the data stream is a series of points (i, y_i). Given a data stream P, let the start point and end point of P be (s, y_s) and (e, y_e) respectively, where $s < e$. As the points in a data stream are discrete, we define M as a shrinking map of the form $M(i, y_i) = (\text{int}(i \cdot a_{11} + b_1), a_{21} \cdot i + a_{22} \cdot y_i + b_2)$. The shrinking map M maps the points in $P\{(s, y_s), (e, y_e)\}$ to a smaller segment $P'\{(s', y_{s'}), (e', y_{e'})\}$, i.e. $\boldsymbol{P' \subset P}$ and $e - s > e' - s'$. At the same time, different segments overlap only at the endpoint, and the union of all segments equals to the original data stream, i.e. $\gamma_{i=1}^{k} P_i = \gamma_{i=1}^{k} P_i' = \{x_j : j = 1, \cdots, n\}$. In this section, we aim to solve the following problem: Given P and P', how to set shrinking factor a_{22} to minimize the L_2 error $E = (M(P) - P')^2$.

The error of M is $E = \sum_{i=s}^{e} (A_{ia_{22}} - B_i)^2$, where $A_i = y_i - (\frac{e-i}{e-s} y_s + \frac{i-s}{e-s} y_e)$, $B_i = y_i - (\frac{e-i}{e-s} y_s + \frac{i-s}{e-s} y_e)$. According to [26], the shrinking factor a_{22} of the shrinking map M_{opt}, which has the minimal error, can be computed by the partial derivative of E with respect to a_{22}, i.e. $\frac{\partial E}{\partial a_{22}} = 2\sum_{i=s}^{e} (A_{ia_{22}} - B_i) A_i = 0$ and $a_{22} = \frac{\sum_{i=s}^{e} A_i B_i}{\sum_{i=s}^{e} A_i^2}$. Therefore, in order to find the optimal shrinking map M_{opt}, it only needs to maintain $\sum_{i=s}^{e} A_i B_i$ and $\sum_{i=s}^{e} A_i^2$ on the data stream. Because the maintain of $\sum_{i=s}^{e} A_i B_i$ on a data stream cannot be done in sub-linear passes of the data stream, we need to compute $\sum_{i=s}^{e} A_i B_i$ approximately. In order to approximate $\sum_{i=s}^{e} A_i B_i$, we substitute a_{22} with $a'_{22} = \sqrt{\frac{\sum_{i=s}^{e} B_i^2}{\sum_{i=s}^{e} A_i^2}}$, and propose two piecewise fractal models, which are illustrated in figures 2 and 3 respectively.

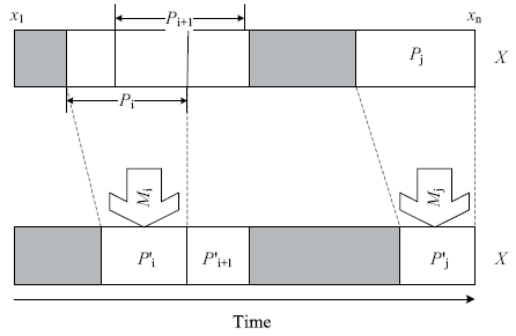

Figure 2. Piecewise fractal model

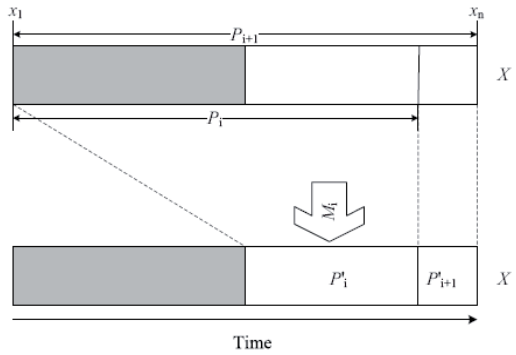

Figure 3. Extended piecewise fractal model

In figure 2, piecewise fractal model is made up of B closest buckets, and each bucket b_i corresponds to a shrinking map M_i and the segments of P_i and P_i'. In each b_i, it only needs to keep the suffix of endpoint $(s_{i'}, y_{si'})$ of p_i', $F(w_{n-si'})$ and the number of points in it. Under the constraint of $(a'_{22})^2 < 1$, add the arrival data x_n to P_i and P_i' of the current bucket b_i, and use x_n as the end point of both P_i and P_i'. When $(a'_{22})^2 \geq 1$, make a new bucket b_{i+1}, let the new segment P_{i+1} equal to P_i', and make a new segment P_{i+1}', Herein, let x_{n-1} be the start point of P_{i+1}', and x_n be the end point of both P_{i+1} and P_{i+1}', then we have $P_{i+1} = P_i' \cup P_{i+1}'$.

Piecewise fractal model describes the similarity between the whole data with part of it. In order to better approximate the whole data, we propose an extended piecewise fractal model described in figure 3. In the extended piecewise fractal model, P_i is extended to include all data before x_n. The extended piecewise fractal model consists B buckets, and each bucket b_i corresponds to a shrinking map M_i and the segments of P_i and P_i'. b_i only keeps the suffix of P_i' starting from $(s_{i'}, y_{si'})$, $F(w_{n-si'})$ and its number of data point. When a new data comes, we add it to the current buckets. Under the constraint of $\frac{\sum_{i=1}^{n} B_i^2}{\sum_{i=1}^{n} A_i^2} < 1$, let $\sum_{i=1}^{n} B_i^2 = y_i$, add the arrival data x_n to P_i and P_{i+1} of the current bucket b_i, and let x_n be the end point of P_i and P_i'. When $\frac{\sum_{i=1}^{n} B_i^2}{\sum_{i=1}^{n} A_i^2} \leq 1$, make a new bucket b_{i+1}, let the new segment P_{i+1} equal to P_i and make a new segment P_{i+1}'. Moreover, let x_{n-1} be the start point of P_{i+1}', and x_n be the end point of both P_{i+1} and P_{i+1}', then we have $P_{i+1} = P_i' \cup P_{i+1}'$. When the number of buckets is bigger than B, delete the oldest bucket, and keep the most recent B buckets. The details of maintaining a piecewise fractal model on a data stream are in algorithm 2.

Algorithm 2: Algorthm of Building pievewise fractal model (x_n)

```
compute ∑_{i=1}^{n} B_i^2 approximately;
compute ∑_{i=1}^{n} A_i^2
if (∑_{i=1}^{n} B_i^2) / (∑_{i=1}^{n} A_i^2) < 1 then
    add x_n to current bucket;
else
    create a new bucket for x_{n-1} and x_n;
    bucket Num ++;
    if bucket Num>B then
        delete the oldest bucket;
```

B. Outlier detection algorithm

When a new data point x_n comes into the data stream, we add it into the piecewise fractal model, and compte the slope d_p with the set $\{(\log(n-si'),\log(F(w_n-si'))): i=1,...,B\}$. After computing the slope d_p, we use algorithm 3 to detect outliers on a data stream.

Algorithm 3: Algorithm of outlier detection

```
locate log(F(1)_{d_p}) at interval I_k;
Array ← retrieve monotonic search space of I_k;
low ← 0;
high ← Array.size() − 1
while low ≤ high do
    mid ← (low + high) / 2;
    if log(T(Array[mid] / F(1))) / (low ← mid + 1) ≤ d_p then
        low ← mid + 1;
    else
        high ← mid − 1;
return high;
```

The algorithm 3 doesn't contain any parameter, and its output is the alert of outliers. In addition, we return the size of the sliding window where the outliers occur. The time complexity of algorithm 3 is $O(\log m)$, where m is the number of sliding windows needed to be checked. If users want to detect outliers only when x_n arrives, and don't care about the number of outliers and the size of the sliding window, then the time complexity is only $O(1)$. So, if we use algorithm 3 to detect outliers on m sliding windows with different sizes, then the space complexity is $O(B)$, and the time complexity is $O(n\log m)$.

VI. EXPERIMENTS

A. Experimental setup

The experiments are implemented on a cluster of 4 nodes, and each node is a PC with a 3.2GHz Pentium4 CPU and 2GB memory. In the cluster, we use one node as the centralized server node, and the other three nodes as the distributed nodes. We apply the SR2240 reader to observe the real distribution of the 2.4Ghz active tags, and generate the experimental data streams with the observed distribution. The observed data is the form of $V =< t_id, r_id, t, ss, st >$, where t_id is the id of tag, r_id is the id of reader, t is the observed time stamp, ss is the observed signal intensity, and st is the status of object.

In order to validate the efficiency and effectiveness of the proposed algorithm, we denote the proposed fractal-based outlier detection algorithm as FBOD, and compare it with STORM [16] and KNOD [22]. The metrics that we use are as follows:

1) running time is the required time while processing data streams in stable status;

2) memory usage is the maximal memory usage of algorithms while processing data streams;

3) precision is the proportion of real detected outliers to total detected outliers;

4) recall is the proportion of real detected outliers to total real outliers.

The experimental parameter setting is in table 1.

TABLE I.
TABLE OF PARAMETER SETTING

Parameters	Descriptions	Values
W	Sliding window size (KB)	$1\sim100$
R	Neighbor radius	10
k	Neighbor number	20
ρ	Proportion of safe inliers saved in buckets	$0\sim1$
P_i	Local outlier probability	$0\sim0.1$

B. Experimental results

In order to test the execution efficiency of the proposed algorithm, we compare the running time of the three algorithms, and the result is in figure 4. The running time of all three algorithms is nearly linear with the volume of data streams. At the same time, the growth rate of KNOD is obviously bigger than that of STORM and FBOD, and the reason is that the computation of KNOD increases more quickly than STORM and FBOD. In addition, the proposed FBOD algorithm optimizes the structure of data streams, and it reduces the deleting and querying time for nodes, so it has the lowest running time.

In order to test the memory usage, we generate 2-dimension, 3-dimension and 4-dimension three datasets, where each sliding window contains 100 data. The maximal memory usages of different algorithms are in figure 5. From the figure we can see that, all algorithms use almost the same amount of memory on the same dataset, and the increase of data dimension only increases a small quantity of memory.

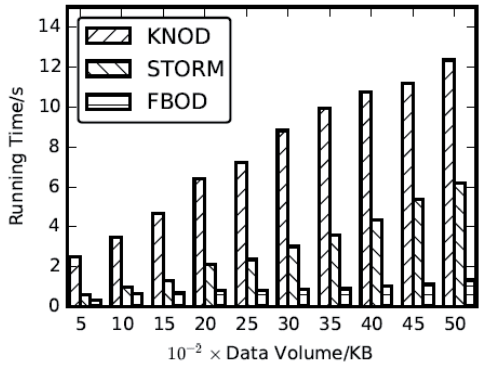

Figure 4. Comparison of running time

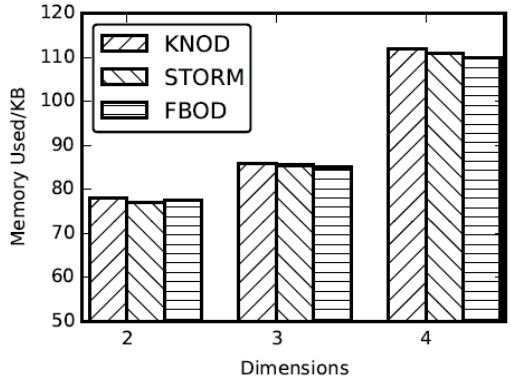

Figure 5. Comparison of memory usage

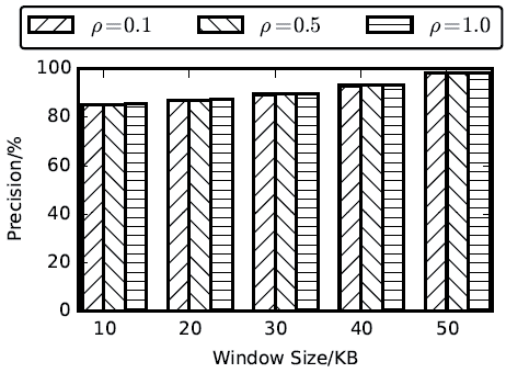

Figure 6. Precision comparison upon different window size

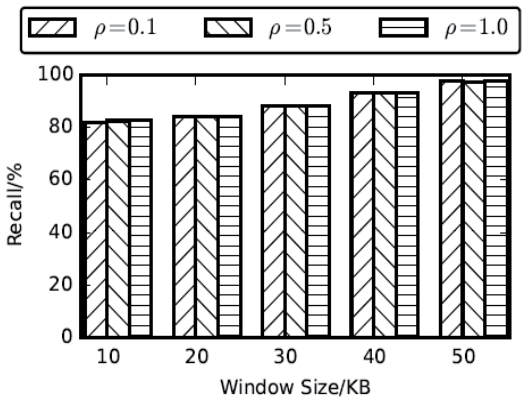

Figure 7. Recall comparison upon different window size

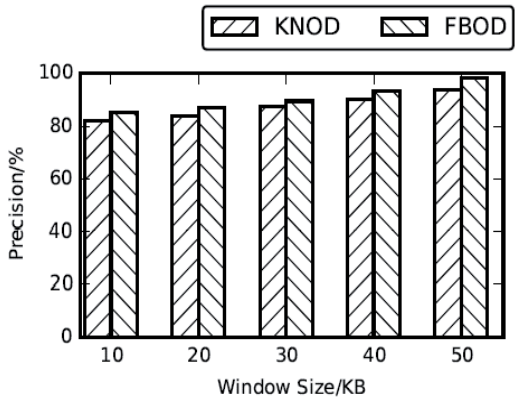

Figure 8. Precision comparison between FBOD with KNOD

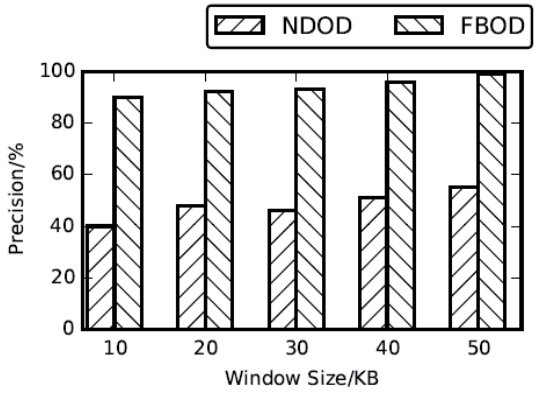

Figure 9. Precision comparison between FBOD with NDOD

We let $\rho = 0.1$, $\rho = 0.5$ and $\rho = 1.0$, and observe the precision and recall of the proposed algorithm while increasing the size of sliding window. The results of precision and recall are in figure 6 and 7 respectively. With the increasing of sliding window size, precision and recall of FBOD increase. However, ρ has little effect on precision and recall of FBOD. The reason is that ρ is used to approximate the segments of each data stream, and the precision and recall can be assured while ρ exceeds some fixed value.

In addition, we compared the precision between FBOD and KNOD, and the result is in figure 8. Both of the two algorithms have very high precision, but the precision of FBOD is a little higher than that of KNOD. The reason is that, we need to smooth parameters in KNOD, and this would decrease the precision, but we don't need to do this in FBOD.

In order to test the effectiveness of algorithms while detecting global outliers, we independently generate three data streams with different distributions on three distributed nodes. We compare the precision of our FBOD algorithm with the Naive distributed outlier detection (NDOD) algorithm, and figure 9 illustrates the result when $\rho = 0.1$. The NDOD algorithm detects outliers locally, and our FBOD algorithm detects outliers globally, so FBOD is obviously better than NDOD.

VII. CONCLUSION

In this paper, we study outlier detection in RDID data streams, which is one of the most important topics in RFID. We apply fractal to detect outliers in data streams, use the self-similarity of fractal to sort the searching space into a monotone searching space, use piecewise fractal model to map a data stream into a number of buckets, and propose an outlier detecting algorithm in these buckets based on the piecewise fractal model. Finally, we validate the efficiency and effectiveness of the proposed algorithm by massive experiments.

REFERENCES

[1] Roberts, C.M.: Radio frequency identification (rfid). Computers & Security 25(1), 18–26 (2006) http://dx.doi.org/10.1016/j.cose.2005.12.003

[2] Sarac, A., Absi, N., Dauz`ere-P´er`es, S.: A literature review on the impact of rfid technologies on supply chain management. International Journal of Production Economics 128(1), 77–95 (2010) http://dx.doi.org/10.1016/j.ijpe.2010.07.039

[3] Roh, S.-g., Choi, H.R.: 3-d tag-based rfid system for recognition of object. Automation Science and Engineering, IEEE Transactions on 6(1), 55–65 (2009) http://dx.doi.org/10.1109/TASE.2008.2008119

[4] Roh, S.-g., Choi, H.R.: 3-d tag-based rfid system for recognition of object. Automation Science and Engineering, IEEE Transactions on 6(1), 55–65 (2009) http://dx.doi.org/10.1109/TASE.2008.2008119

[5] Chung, K.K.-T., Shi, X., Li, J.J.: RFID device for object monitoring, locating, and tracking. Google Patents. US Patent 7,319,397 (2008)

[6] Hu, J., Lewis, F.L., Gan, O.P., Phua, G.H., Aw, L.L.: Discrete-event shop-floor monitoring system in rfid-enabled manufacturing. Industrial Electronics, IEEE Transactions on 61(12), 7083–7091 (2014) http://dx.doi.org/10.1109/TIE.2014.2314068

[7] Badia-Melis, R., Ruiz-Garcia, L., Garcia-Hierro, J., Villalba, J.I.R.: Refrigerated fruit storage monitoring combining two different wireless sensing technologies: Rfid and wsn. Sensors 15(3), 4781–4795 (2015) http://dx.doi.org/10.3390/s150304781

[8] Gupta, M., Gao, J., Aggarwal, C., Han, J.: Outlier detection for temporal data. Synthesis Lectures on Data Mining and Knowledge Discovery 5(1), 1–129 (2014) http://dx.doi.org/10.2200/S00573ED1V01Y201403DMK008

[9] Wang, X., Ji, Y., Zhao, B.: An approximate duplicate-elimination in rfid data streams based on d-left time bloom filter. In: Web Technologies and Applications, pp. 413–424. Springer, (2014) http://dx.doi.org/10.1007/978-3-319-11116-2_36

[10] Schroeder, M.R.: Fractals, Chaos, Power Laws: Minutes from an Infinite Paradise. Courier Corporation, (2012)

[11] Knox, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large datasets. In: Proceedings of the International Conference on Very Large Data Bases, pp. 392–403 (1998). Citeseer

[12] Yu, D., Sheikholeslami, G., Zhang, A.: Findout: finding outliers in very large datasets. Knowledge and Information Systems 4(4), 387–412 (2002) http://dx.doi.org/10.1007/s101150200013

[13] Breunig, M.M., Kriegel, H.-P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: ACM Sigmod Record, vol. 29, pp. 93–104 (2000). ACM http://dx.doi.org/10.1145/335191.335388

[14] Papadimitriou, S., Kitagawa, H., Gibbons, P.B., Faloutsos, C.: Loci: Fast outlier detection using the local correlation integral. In: Data Engineering, 2003. Proceedings. 19th International Conference On, pp. 315–326 (2003). IEEE

[15] Jagadish, H., Koudas, N., Muthukrishnan, S.: Mining deviants in a time series database. In: VLDB, vol. 99, pp. 7–10 (1999)

[16] Muthukrishnan, S., Shah, R., Vitter, J.S.: Mining deviants in time series data streams. In: Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference On, pp. 41–50 (2004). IEEE http://dx.doi.org/10.1109/ssdm.2004.1311192

[17] Angiulli, F., Fassetti, F.: Detecting distance-based outliers in streams of data. In: Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management, pp. 811–820 (2007). ACM http://dx.doi.org/10.1145/1321440.1321552

[18] Palpanas, T., Papadopoulos, D., Kalogeraki, V., Gunopulos, D.: Distributed deviation detection in sensor networks. ACM SIGMOD Record 32(4), 77–82 (2003) http://dx.doi.org/10.1145/959060.959074

[19] Babcock, B., Olston, C.: Distributed topk monitoring. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, pp. 28–39 (2003). ACM http://dx.doi.org/10.1145/872757.872764

[20] Cormode, G., Hadjieleftheriou, M.: Finding frequent items in data streams. Proceedings of the VLDB Endowment 1(2), 1530–1541 (2008) http://dx.doi.org/10.14778/1454159.1454225

[21] Cormode, G., Muthukrishnan, S., Zhuang, W.: Conquering the divide: Continuous clustering of distributed data streams. In: Data gineering, 2007. ICDE 2007. IEEE 23rd International Conference On, pp. 1036–1045 (2007). IEEE

[22] Su, L., Han, W., Zou, P., Jia, Y.: Continuous kernel-based outlier detection over distributed data streams. In: Frontiers of High Performance Computing and Networking ISPA 2007 Workshops, pp. 305–314 (2007). Springer http://dx.doi.org/10.1007/978-3-540-74767-3_32

[23] Masciari, E.: A framework for outlier mining in rfid data. In: Database Engineering and Applications Symposium, 2007. IDEAS 2007. 11th International, pp. 263–267 (2007). IEEE http://dx.doi.org/10.1109/ideas.2007.4318112

[24] Borgnat, P., Flandrin, P., Amblard, P.-O.: Stochastic discrete scale invariance. Signal Processing Letters, IEEE 9(6), 181–184 (2002) http://dx.doi.org/10.1109/LSP.2002.800504

[25] Barnsley, M.F., Elton, J.H., Hardin, D.P.: Recurrent iterated function systems. Constructive approximation 5(1), 3–31 (1989) http://dx.doi.org/10.1007/BF01889596

[26] Mazel, D.S., Hayes, M.H.: Using iterated function systems to model discrete sequences. Signal Processing, IEEE Transactions on 40(7), 1724–1734 (1992) http://dx.doi.org/10.1109/78.143444

AUTHOR

Liansheng Li is with the School of Electronics and Information Engineering, Hunan University of Science and Engineering, 425100 Yongzhou City, Hunan Province, China (Email: lls5111@sina.com).

This work was supported by Science Research Foundation of Hunan Province Education Department (14C0483); Science Research Foundation for Distinguished Young Scholars of Hunan Province Education Department(14B070). Science and Technology Project of Hunan Province of China(2014FJ6095). Submitted 26 October 2015. Published as resubmitted by the author 30 December 2015.