Information Systems 48 (2015) 89–112

Contents lists available at ScienceDirect

Information Systems

journal homepage: www.elsevier.com/locate/infosys

A fast MST-inspired *k*NN-based outlier detection method

Xiaochun Wang [a,*], Xia Li Wang [b], Yongqiang Ma [a], D. Mitchell Wilkes [c]

[a] *Xian Jiaotong University, ROC*
[b] *Changan University, ROC*
[c] *Vanderbilt University, United States*

ARTICLE INFO

Article history:
Received 11 June 2013
Received in revised form
25 August 2014
Accepted 9 September 2014
Recommended by F. Korn
Available online 26 September 2014

Keywords:
Distance-based outlier detection
Density-based outlier detection
Clustering-based outlier detection
Minimum spanning tree-based clustering
Approximate *k*-nearest neighbors' search

ABSTRACT

Today's real-world databases typically contain millions of items with many thousands of fields. As a result, traditional distribution-based outlier detection techniques have more and more restricted capabilities and novel *k*-nearest neighbors based approaches have become more and more popular. However, the problems with these *k*-nearest neighbors based methods are that they are very sensitive to the value of *k*, may have different rankings for top *n* outliers, are very computationally expensive for large datasets, and doubts exist in general whether they would work well for high dimensional datasets. To partially circumvent these problems, we propose in this paper a new global outlier factor and a new local outlier factor and an efficient outlier detection algorithm developed upon them that is easy to implement and can provide competing performances with existing solutions. Experiments performed on both synthetic and real data sets demonstrate the efficacy of our method.

1. Introduction

Aiming to discover observations that deviate from other observations so much as to arouse suspicions that they are generated by a different mechanism, outlier detection has become an important data mining task [1]. Being applied in many different fields such as intrusion detection for cyber-security [2,3], fraud detection for credit cards, insurance and tax [4], early detection of disease outbreaks in the medical field [5], fault detection in sensor networks for monitoring health, traffic, machine status, weather, pollution, surveillance [6], and so on [7,8], it has generated enormous interests and many techniques have been developed for this purpose in recent years, namely distribution-based approaches, depth-based approaches, distance-based approaches, density-based approaches and clustering-based approaches.

State-of-the-art *k*-nearest neighbors (*kNN*) based outlier detection algorithms, such as many distance-based and density-based methods, have demonstrated various ways to filter out the normal data and locate the small number of outliers. While they are simple to implement, other aspects concerning the algorithms are worth further exploration. Firstly, these methods usually return only top *n* outliers with two values. One is the outlier factor (also referred to as score in this paper) and the other is the ranking of the points according to the scores. Therefore, different methods may have different rankings for top *n* outliers. Secondly, it has been observed that *k*-nearest neighbors based outlier detection methods are sensitive to the parameter *k* and a small change in *k* can lead to changes in the scores and, correspondingly, the ranking. As a result, except for very strong outliers where the scores are distinct, the ranking is sensitive to *k* as well. Thirdly, for modern large datasets with N data items, the $O(N \log N)$

* Corresponding author.
E-mail addresses: xiaocchunwang@mail.xjtu.edu.cn (X. Wang), xlwang@chd.edu.cn (X.L. Wang), yongqiangma@stu.xjtu.edu.cn (Y. Ma), mitch.wilkes@vanderbilt.edu (D.M. Wilkes).

http://dx.doi.org/10.1016/j.is.2014.09.002
0306-4379/

running time involved in the exact search of the k-nearest neighbors has to be improved significantly. Finally, in high dimensional space, data points become equally close to each other, raising the so-called problem of "curse of dimensionality", and the issue exists whether the notion of these outlier definitions is still meaningful for the high-dimensional data.

To meet these challenges and to explore the impact of dimensionality on these *kNN*-based outlier detection algorithms, in this paper, we propose a new minimum spanning tree (MST)-inspired k-nearest neighbors (*kNN*)-based outlier detection method that continues the work presented in [9] and is both computationally efficient and competent with the state-of-the-art outlier detection techniques. Basically, our method starts by finding k-nearest neighbors for each data point. Then a mini MST is constructed upon each data point and its k-nearest neighbors. Finally a small number of outliers are identified relatively within each mini MST using our proposed outlier scores. It is based on the observation that one good point of MST clustering-based outlier detection methods reside in its taking distances between data points in an MST (rather than the distances to the kth or k nearest neighbor(s)) into account when clustering, thus making the outlier scores relatively less sensitive to k. However, the construction of an exact MST requires quadratic running time and is very computationally expensive for modern large datasets. Therefore, the aim of this hybridization (i.e., MST clustering-based outlier detection and *kNN* based outlier detection) is to increase the robustness and consistency of the detecting results and to significantly decrease the computation time of MST-clustering based outlier detection process. Our first contribution in this paper is two newly proposed MST-inspired *kNN*-based outlier scores, a global one and a local one, and an outlier detection method developed upon them. Our second contribution is the significant tempo efficiency of the proposed method through the application of an efficient approximate k-nearest neighbors' search framework to our *kNN*-based outlier detection techniques. This is an advantage over a preliminary work of this research presented in [9], which requires a construction of an approximate minimum spanning tree very close to a true one. Our third contribution is a study of a set of current outlier detection algorithms when applied to some high dimensional datasets. Finally, to be as general as possible, our algorithm has no specific requirements on the dimensionality of data sets and can be applied to outlier detection in large high-dimensional data sets. A number of experiments on both synthetic and real data sets demonstrate the robustness and efficiency of the proposed approach in comparison with several state-of-the-art outlier detection algorithms.

The rest of the paper is organized as follows. In Section 2, we review some existing work on state-of-the-art outlier detection approaches. We then present our proposed approaches in Section 3. In Section 4, an empirical study is conducted. Finally, conclusions are made in Section 5.

2. Related work

Related work in this paper falls into three main categories: distance -and density-based outlier detection methods, MST clustering-based outlier detection techniques, and outlier detection techniques for high dimensional data.

2.1. Distance-based and density-based outlier detection

Many efforts have been devoted to detecting outliers. Distance-based outlier detection method was originally proposed by Knorr and Ng in 1998 as an improvement over distribution based methods. Given a distance measure defined on a feature space, "an object O in a dataset T is a $DB(p,D)$-outlier if at least a fraction p of the objects in T lies greater than distance D from O", where the term $DB(p, D)$-outlier is a shorthand notation for a Distance-Based outlier (*DB*-outlier) detected using parameters p and D [10]. To suit for different theoretical and practical purposes, two *kNN*-based variants have been developed. "Given two integers, n and k, Distance-Based outliers are the data items whose distance to their kth nearest neighbor is among top n largest ones [11]" (referred to as "DB-MAX" in the following), and "Given two integers, n and k, Distance-Based outliers are the data items whose average distance to their k-nearest neighbors is among top n largest ones [12]" (referred to as "DB" in the following).

Though simple and elegant, distance-based outlier detection techniques work well for data sets that contain one or more clusters with similar densities and can detect more globally-orientated outliers. However, many real world data sets often have complex structures. A classic situation illustrating this deficiency is shown in Fig. 1, where o_1 and o_2 are global outliers and can easily be detected by distance-based methods while o_3 is a local one and cannot.

To deal with this situation, in 2000, Breunig et al. pioneered the density-based outlier detection research by introducing an indicator for each data item, called Local Outlier Factor (LOF), which is a ratio between the local density of an object and the average of those of its k-nearest neighbors [13]. The LOF method works by first calculating the LOF for each object, next ranking data points according to their LOF values, and, finally, returning objects with top-n largest LOF values as outliers.

Following the notion of local outlier factor, several extensions and refinements to the basic LOF model have been proposed. In 2002, Tang et al. proposed a connectivity-based

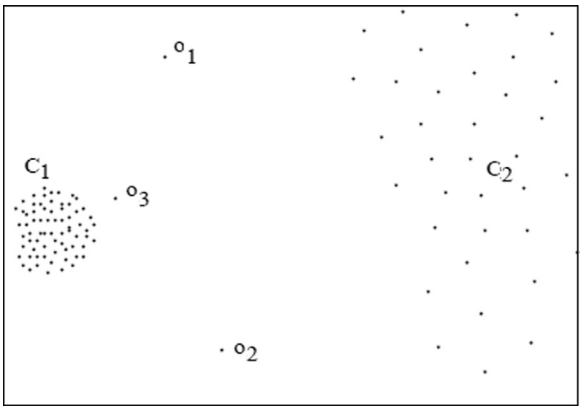

Fig. 1. A classic example of a local outlier.

outlier factor (COF) to deal with "isolativity" of outliers [14]. Isolativity implies low density, but the latter does not always imply the former. Given a data point o and its k-nearest neighbors, the first cost in the cost description is the distance from o to its closest neighbor. In general, the ith ($i \le k$) cost is the smallest distance from o and its $(i-1)$ closest objects to the rest of the k–i objects in the neighborhood. Finally, the COF is the ratio of a data point's cost description over the average of those of its k-nearest neighbors'. In 2003, Papadimitriou et al. proposed another local outlier detection scheme called Local Outlier Integral (LOCI) based on the concept of a multi-granularity deviation factor (MDEF). The main difference between LOF and LOCI is that the MDEF of LOCI uses ε-neighborhoods rather than k-nearest neighbors [15]. In 2004, Sun and Chawla proposed a spatial local outlier measure called SLOM [16]. In 2006, Jin et al. presented the INFLO method which considers the union of a point's k-nearest neighbors and its reverse nearest neighbors to obtain a measure of outlierness [17]. The reverse nearest neighborhood of a data point p is defined to consist of those of its k-nearest neighbors for which p is also among its k nearest neighbors. By this way, INFLO compares p's density with the average of densities of objects in the union as a measure of outlierness. Noticing that real-world data usually have a scattered distribution, in 2009, Zhang et al. proposed a new outlier definition, named Local Distance-Based Outlier Factor (LDOF), for detecting outliers in scattered datasets [18]. LDOF is the ratio of the average of distances from a data point to its k-nearest neighbors over the average of pairwise distances among these $k+1$ data points and, in this fashion, captures the degree to which an object deviates from its neighborhood system. In 2013, Huang et al. proposed a new approach for outlier detection, named RBDA, based on a ranking measure that focuses on the question of whether a point is 'central' from its nearest neighbors [19]. Eliminating the problem of density calculation in the neighborhood of a point, RBDA identifies outliers based on computing the ranks of a point among all its k-nearest neighbors but unfortunately with a high computation cost.

2.2. MST clustering-based algorithms

Distance-based as well as density-based outlier scores are sensitive to the setting of some parameters. This can be illustrated by a 2-dimensional dataset shown in Fig. 2. For distance-based outlier detection techniques, if $k=6$ nearest neighbors are considered, all the data points in cluster C_3 will not be detected as outliers, while if $k=7$, all the data points in cluster C_3 are regarded as outliers. Similar problems exist for density-based outlier detection techniques. The situation could be worse for the detection of outliers in high-dimensional feature space since data points there cannot be easily visualized.

This is where clustering-based algorithms can be more meaningful. Being a very important data mining tool, the main concern of clustering algorithms is to find clusters by optimizing some criterion, such as minimizing the intra-cluster distance and maximizing the inter-cluster distance. As a by-product, data items in small groups can often be regarded as outliers (noise) that should be removed to make clustering more reliable. Classic clustering algorithms,

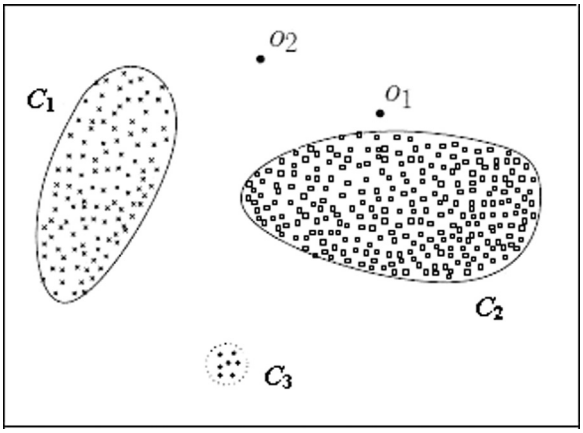

Fig. 2. Sample clusters in a 2-D data set.

such as K-means algorithm and PAM, rely on grouping the data points around some "centers" and do not work well when the boundaries of the clusters are irregular. As an alternate, graph theory based methods, typified by the MST-based clustering algorithms, can find clusters with irregular boundaries.

Being a connected weighted graph over a set of data points but with no closed paths, a minimum spanning tree has the minimal total weight. If a weight denoting a distance between two end points is assigned to each edge, any edge in an MST will be the shortest distance between two subtrees that are connected by that edge. This fact is referred to as the cut property of MST's. Therefore, removing the longest edges corresponds to choosing the breaks to form clusters. Minimum spanning tree (MST) based clustering was first proposed by Zahn in 1971 [20], and has so far been extensively studied [21,22,23,24,25]. Regarding that data points in smallest clusters formed by cutting the longest edges in an MST may likely be outliers, several MST-based outlier detection techniques have been proposed [26,27,28]. But for modern large and high-dimensional data sets where only a set of N data points is given, these MST-based outlier detection algorithms suffer from a quadratic running time required for the construction of an MST.

To be more computationally efficiency, Wang et al. proposed an efficient three-phase outlier detection technique (referred to as MST+LOF in the following) [9]. First, an efficient construction of a spanning tree very close to a minimum spanning tree of the data set is conducted. Second, the longest edges in the obtained spanning tree are removed to form clusters. Based on the intuition that the data points in small clusters may be most likely all outliers, they are selected and regarded as potential outlier candidates. Finally, density-based outlying factors, LOF, are calculated for potential outlier candidates and accessed to pinpoint the local outliers. The main advantage of the algorithm is its computation efficiency.

2.3. Projection based High dimensional outlier mining

The outlier detection algorithms presented so far make implicit assumptions of relatively low dimensionality of the data and use the distances in full-dimensional space to

find the outliers. However, in high-dimensional space, the data are sparse and the notion of finding meaningful outliers becomes substantially more complex and nonobvious. Aside from considering the behavior of the data in full dimensionality, the abnormal deviations may be embedded in some lower-dimensional subspace [29]. By examining the behavior of the data in subspace, it is possible to design more meaningful techniques for finding outliers that are specific to the particular subspace in question. In fact, it has been reported in [11] that more interesting outliers on the NBA98 basketball statistics database were obtained by using fewer features. Based on these observations, Aggarwal and Yu proposed new techniques for outlier detection (referred to as Projection Based Outlier Detection (PBOD) in the following) that define a data point to be an outlier if in some lower-dimensional projection it is present in a local region of abnormally low density [30]. To characterize and find such projections, a grid discretization of the data is first performed. Each of the d attributes of the data is divided into φ ranges and, thus, each range contains a fraction $f=1/\varphi$ of the records. For a k-dimensional cube that is created by picking grid ranges from $k \le d$ different dimensions, the sparsity coefficient $S(C)$ of the cube C is calculated as follows:

$$S(C) = \frac{n(C) - N \cdot f^k}{\sqrt{N \cdot f^k \cdot (1 - f^k)}} \tag{1}$$

where N is the number of data points and $n(C)$ denotes the number of points in the k-dimensional cube. Only sparsity coefficients that are negative indicate cubes in which the presence of the points is significantly lower than expected. Once such patterns have been identified, the outliers are defined as those records that have such patterns present in them. An interesting observation is that such lower-dimensional projections can be mined even in data sets that have missing attribute values [31]. The problem with PBOD is the exponentially increasing search space of possible projections with dimensionality. The algorithm is not feasible for a few hundred dimensions.

3. An MST-inspired outlier detection algorithm

From our study of distance-based, density-based and clustering-based outlier detection algorithms, we have obtained several observations.

Firstly, distance-based methods work theoretically by calculating *kNN* for each data point, computing the distance-based outlier scores for them, ranking all the objects according to their scores, and finally returning data points with top n largest scores as outliers. However, there is no reason to assume that this must be the case. For example, in Fig. 3, though distance-based outlier scores can be calculated for this data set, from an MST point of view, there are no outstanding outliers. Unfortunately, MST-based clustering criterion favors cutting small sets of isolated nodes in a graph and can give a bad partition when no outliers exist. This fact can be easily manifested from the dataset shown in Fig. 3. Therefore, there must be some way to judge whether outliers exist or not. To do so, as a first degree approximation, the edge weights (i.e., edge distances) within a cluster of an MST can be assumed to follow a uniform distribution and the corresponding mean and standard deviation can thus be calculated. The ratio of the standard deviation over the mean can be used to judge to some degree whether outliers exist or not.

Secondly, density-based outlier detection algorithms may not do well in global outlier detection. For example, consider the data set in Fig. 4. Apparently data point A is farthest away from its six closest neighbors and therefore should be identified as a global outlier. However, for $k=6$, the LOF algorithm assigns a higher outlier score to data point B than to A. This is because that LOF is computed as a ratio and data point B gets a higher ratio for this k. As a result, LOF-based algorithms fail to identify A as the most significant outlier [20]. From an MST point of view as shown on the right of Fig. 4, if tree edge weight (i.e., distance) is used as the outlier factor, this will not happen and data A's score remains as a constant for different k's.

Thirdly, according to the cut property, MST-based clustering algorithms can be very useful in detecting small clusters that are connected to the normal data points by long edges and can be selected and regarded as outliers as illustrated in Fig. 4. However, standard MST clustering-based outlier detection algorithms usually take a quadratic running time to ensure the properties of MSTs to be satisfied. From our point of view, this is neither efficient nor necessary.

Based on these observations, the working ideas behind our efficient MST-inspired outlier detection algorithm are formalized in the following subsection.

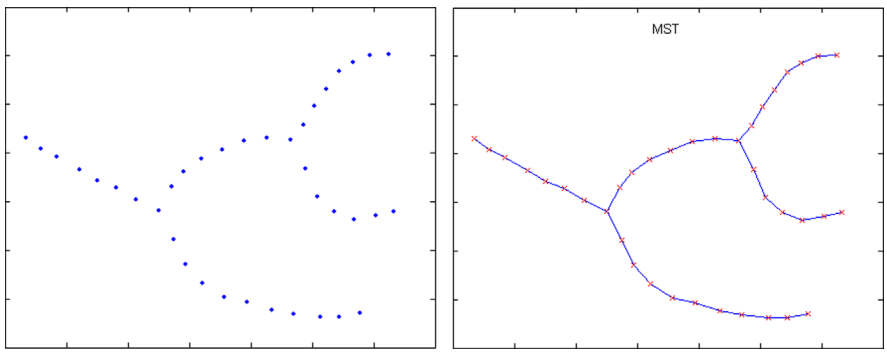

Fig. 3. A sample 2-D data set illustrating a deficiency of distance-based outlier detection (left) data set (right) its MST.

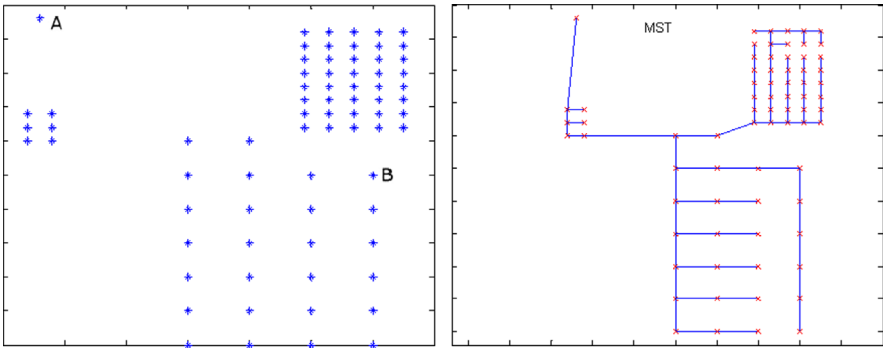

Fig. 4. A sample 2-D data set illustrating a deficiency of LOF-based outlier detection (left) data set (right) its MST.

3.1. Two new outlier factors

Definition 1. (Hawkins Outlier) An outlier is an observation that deviates from other observations so much as to arouse suspicions that it is generated by a different mechanism.

Definition 2. (Clustering) Given a dataset T, a general clustering algorithm attempts to partition T into K clusters, C_1, C_2,…, C_K, $C_i \neq \emptyset$, $C_i \cap C_j = \emptyset$, $T = C_1 \cup C_2 \cup \ldots \cup C_K$, $i,j = 1{:}K$, $i \neq j$, such that the distance between any closest pair of points in a cluster is less than the distance between any point in the cluster and any point not in the cluster (or, in its nearest cluster).

Definition 3. (Distance between two clusters) Let C_i, C_j be clusters of the dataset T, the distance between C_i and C_j is defined as $\rho(C_i,C_j) = \min\{\rho(x_i,x_j) | x_i, \in C_i, x_j \in C_j\}$.

Definition 4. (Minimum spanning tree) Given a weighted and connected graph, $G(T) = (V,E)$ with a vertex set $V = T$ (i.e., a set of T data points) and an edge set $E = \{e_{ij} = (x_i, x_j) | x_i, x_j \in T, i \neq j\}$. Each edge e_{ij} has a weight $w(x_i, x_j)$, a minimum spanning tree (MST) of graph $G(T)$ is an acyclic subset of E that connects all the vertices in V with minimum total weights $W(\text{MST}) = \min\{\Sigma_{xi,xj \in V, \text{eij} \in E} w(x_i, x_j)\}$ and satisfies the following two conditions:

(1) the cut property: the edge with the smallest weight crossing any 2 partitions of the vertex set must belong to an MST and
(2) the cycle property: the edge with the largest weight in any cycle in a graph cannot be in an MST.

If a weight (i.e., w) denoting a distance (i.e., ρ) between two end points is assigned to each edge, any edge in an MST will be the shortest distance between two subtrees that are connected by that edge. Therefore, removing the longest edges (i.e., the inconsistent edges) will theoretically result in clusters.

Definition 5. (Link) A partition of the nodes of a graph G is a division into two disjoint nonempty subsets (C_i, C_j). A link is any edge in G whose weight is equal to the distance $\rho(C_i, C_j)$.

Theorem 1. All MST edges are links of some partition in graph G. (The proof is given in [20].)

Definition 6. (Minimum spanning tree based clustering) Given a dataset T and an MST of it, MST_T, let C_m, C_n be two subsets of T such that C_m, $C_n \subseteq T$, $C_m \cap C_n = \emptyset$ and $C_m \neq \emptyset$, $C_n \neq \emptyset$. C_m and C_n are minimum spanning tree based clusters if they are formed by removing the link whose edge weight (i.e., distance) is significantly larger than the average of nearby edge weights on both sides of the link, that is, $\rho(C_m, C_n) \gg \max\{\rho(C_m), \rho(C_n)\}$, where $\rho(C_m) = \max\{\rho(x_i, x_j) | x_i, x_j \in C_m, e_{ij} \in MST_T, i \neq j\}$and $\rho(C_n) = \max\{\rho(x_i, x_j) | x_i, x_j \in C_n, e_{ij} \in MST_T, i \neq j\}$.

Definition 7. (Minimum spanning tree clustering based outliers) Let $C_1, C_2, \ldots, C_K$ be the clusters of the dataset T discovered by MST-based clustering in the sequence such that $|C_1| \geq |C_2| \geq \cdots \geq |C_K|$. Given parameters α and β, clustering based outliers are the clusters in C_i through C_K such that: $|C_1| + |C_2| + \cdots + |C_{i-1}| \geq |T| * \alpha$ and $|C_1| + |C_2| + \cdots + |C_{i-2}| \leq |T| * \alpha$. and $C_{i-1}/C_i > \beta$.

According to normal distribution, α can take a value of 0.3%. Without loss of generality, β can take a minimum value of 3.

Definition 8. (MST clustering based global outliers) Let C_1, C_2,…,C_K be the clusters of the dataset T discovered by MST-based clustering, and ρ_{MAX} be the largest intra-cluster edge weight, a cluster C_i is an MST-clustering based global outlier cluster if $\rho(C_i, C_j) \gg \rho_{MAX}$, where C_j is the nearest neighboring cluster of C_i.

Definition 9. (MST clustering based local outliers) Let C_1, C_2,…,C_K be the clusters of the dataset T discovered by MST-based clustering, and ρ_{MAX} be the largest intra-cluster edge weight, a cluster C_i is an MST-clustering based local outlier cluster if $\rho(C_i, C_j) < \rho_{MAX}$, but $\rho(C_i, C_j) \gg \rho(C_j)$, where C_j is the nearest neighboring cluster of C_i.

For the sample dataset shown in Fig. 5, it can be seen that o_1 (i.e., cluster C_3) is an MST-based global outlier since $E_1 \gg \rho_{MAX}$ (i.e., $\rho(C_1)$) and $E_1 < E_2$, while o_2 (i.e., cluster C_4) is an MST-based local outlier since $E_3 < \rho_{MAX}$ and $E_3 \gg \rho(C_2)$ where $E_3 = \rho(C_2, C_4)$.

Definition 10. (MST clustering based global outlier factor) Let C_1 be an MST-clustering based global outlier group and

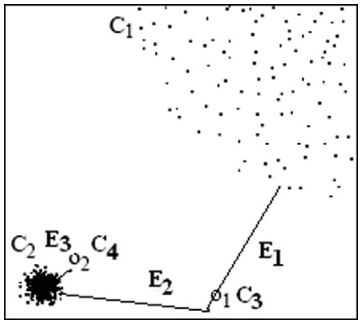

Fig. 5. Sample clusters in a 2-D data set.

C_2 be the nearest cluster of C_1. The MST clustering based global outlier factor is defined as $\rho(C_1,C_2)$.

Definition 11. (MST clustering based local outlier factor) Let C_1 be an MST-clustering based local outlier group and C_2 be the nearest cluster of C_1. The MST clustering based local outlier factor is defined as $\rho(C_1,C_2)/\rho(C_2)$.

Since there are only a small number of outliers in a dataset, MST clustering-based outlier detection algorithms can be more efficient if the longest edges connecting the outlier clusters to the normal data can be correctly identified and quickly located. In other words, some of the longest edges do not correspond to any cluster separations but are associated with the outliers. Basically, this observation for the design of a more efficient scheme is formalized in the following theorem.

Theorem 2. Given a data set and a minimum spanning tree constructed upon it, for an outlying group of k data points, the edge connecting these k data points to the rest of data set is the same as the one in a mini MST constructed upon these k data points and the data point at the other end of this edge in the minimum spanning tree constructed upon the data set.

Proof. To prove this, we use contradiction. Suppose there exists another smaller edge connecting these k outlying data points to the rest of the data set. This is impossible because otherwise the cut property will be violated. Therefore, they must have been the same one. This is true for both global and local outliers. □

Based on Theorem 2, locating the longest edges in an MST connecting the outliers to the rest of data is equivalent to locating the longest edges in mini MSTs constructed upon each data point and its k-nearest neighbors, which takes much less time since many efficient *kNN* computation methods have been proposed [32]. Therefore, we can carry out an outlier detection procedure (e.g., an MST-clustering based one) within each data point and its k-nearest neighbors to detect top outliers. On the other hand, since MST-based clustering has been an active research area and many sophisticated algorithms have been developed for it for small datasets, MST-based clustering algorithms can be dwelt upon and tapped for their potential for outlier detection within a mini MST. This is where the combination of *kNN* structure and the cutting property can be exploited.

In this sense, it will be more meaningful if k is defined to be the size of the largest outlying cluster plus 1 for outlier detection process. To identify the relatively small number of long edges connecting small outlier groups to the majority of normal data of an MST, a mini MST for every point and its *kNN* is first constructed.

To find global outliers, we first search for the edge in these mini MSTs that has the largest edge distance value. If it is significantly larger than the average values of neighboring edges, this longest edge among mini MSTs can be regarded as the global outlier score for those data items that are in the same partition as the data point itself. In other words, these data points are this longest edge away from the rest data points of the database. As illustrated in Fig. 4, it is evident that, compared to DB and DB-MAX, these global scores are less sensitive to k.

Further for global outlier detection, we are more interested in those mini MSTs in which the weights of the longest edges are significantly larger than the average values of those of the neighboring edges. To further quantify the significance of an edge's weight being larger than the average values of neighboring edges for global outlier detection, we use the uniform distribution as a first degree approximation for tree edge weights in mini MSTs.

Definition 12. (miniMST-based global outlier indicator) Let a mini MST be constructed upon a data point and its k-nearest neighbors, $dist[i]$ denote the ith edge weight of such mini MST starting at the point, miniMST-based global outlier indicator, SOM_{MST}, is define to be the ratios of the standard deviation, Std_{MST}, and the mean, $Mean_{MST}$, of these edge weights (i.e., edge distances) as

$$Mean_{MST} = \frac{1}{k}\sum_{i=1}^{k} dist[i] \tag{2}$$

$$Std_{MST} = \sqrt{\frac{1}{k}\sum_{i=1}^{k} (dist[i] - Mean_{MST})^2} \tag{3}$$

$$SOM_{MST} = \frac{Std_{MST}}{Mean_{MST}} \tag{4}$$

SOM_{MST} is a quantitative measure of deviation from normality and can be used as a threshold to rule out the large portion of normal data. Based on these ideas, our *kNN*-based global outlier score is given in the following definition.

Definition 13. (MST-inspired *kNN*-based global outlier factor) Let a mini MST be constructed upon a data point and its k-nearest neighbors, $dist[i]$ denote the ith edge weight of such mini MST and *cut-thred* be a user provided threshold for SOM_{MST} which measures the possibility of outlier existence, an MST-inspired *kNN*-based global outlier factor is defined as

$$MST - MAX = \max_{i=1:k}\{dist[i]\} \tag{5}$$

$$SOM_{MST} \geq cut - thred \tag{6}$$

For local outliers, since they are significantly far away only from their nearest neighboring clusters, the local outlier

score is defined as the ratio of the largest edge weight value over the smallest edge weight value in each mini MST.

Definition 14. (MST-inspired *kNN*-based local outlier factor) Let a mini MST be constructed upon a data point and its *k*-nearest neighbors and *dist*[*i*] denote the *i*th edge weight of such mini MST, the MST-inspired *kNN*-based local outlier factor is defined as

$$MST-MAX-MIN=\frac{\max_{i=1:k}\{dist[i]\}}{\min_{i=1:k}\{dist[i]\}} \quad (7)$$

The local outlier detection can be assumed to be over when the ratio drops below a threshold (say 3). Finally, the outlier scores are used to assign the returned data points a degree of being outlying.

3.2. An approximate nearest neighbor search structure

Since most data points are normal, the goal of finding top n outliers can be achieved by first quickly finding a good estimate of the outlying score for each data item and then focusing on top $m \geq n$ ones. By removing all the inliers among them, the required top n outliers show up. To meet this goal, we are particularly interested in an approximate nearest neighbor search facility, called the divisive hierarchical clustering algorithm (DHCA) [21]. Essentially, to start the DHCA, K centers at the top level are randomly selected from the whole data set. Next each data point is assigned to its closest center, creating K partitions. At each successive level in the iteration, for each of these K partitions, K random centers are recursively selected within each partition and the clustering process continues to form at most K^n partitions at the nth stage. The procedure continues until the number of elements in a partition is below $K+2$, at which time, a nearest neighbor search among all the data items in that partition is conducted. Such a strategy ensures that points that are close to each other in space are likely to be collocated in the same partition, and multiple runs of DHCA greatly enhance such possibilities. A more detailed demonstration and proof of the effectiveness of DHCA on approximate nearest neighbors' search have been given in [21] and will not be repeated here. After several iterations, exact *kNN*s and the correspondingly scores are computed for top outliers. The number of top outliers is small, thus the computation time is fast. With these observations in mind, a simple outlier detection method is developed in the following.

3.3. Our MST- inspired outlier detection algorithm

We combine the above three factors to create our MST-inspired *kNN*-based outlier detection algorithm:

1. set k to be the largest outlying cluster size plus 1;
2. sequentially read the data set in and initialize k neighbors of each data item from its immediate predecessors or successors on the fly (referred to as sequential initialization (SI));
3. run DHCA multiple times, and, for each iteration, calculate a 1-dimensional array of the average distance of each data item to its *kNN* and then the array mean; stop this step when the percentage differential of the mean between two consecutively iterations is below 10^{-6};
4. construct the mini MST over each data item and its k nearest neighbors;
5. compute three 1-dimensional arrays of the estimated outlying scores for *iNN* (where $i=2{:}k$), and sort them in a non-increasing order;
6. for data items with top global outlying scores, find their true *iNN* and calculate their true outlying scores, check its SOM_{MST}, return it if the SOM_{MST} is larger than a threshold (say 1.5), then if $i=k$, the global outlier detection is completed, otherwise, $i=i+1$ and go to 5;
7. for data items with top local outlying scores, find their true *iNN* and calculate their true outlying scores, return it if the score is larger than a threshold (say 3), then if $i=k$, the local outlier detection is completed, otherwise, $i=i+1$ and go to 5;
8. repeat steps 5, 6 and 7 until all the outliers (i.e., whose score is above the global or local thresholds) are mined.

Since the number of outliers is expected to be relatively small, the number of distance computations consumed is expected to be relatively small as well.

Physically, the resource consumed by our algorithm includes the space to hold the whole data set in memory, space to store their k neighbors, and some temporary space for mini MSTs. The numerical parameters our algorithm needs from the user include the number of nearest neighbors (i.e., k), the global and local outlier thresholds, while the outputs include a set of ranked outliers from the dataset. To improve the readability, our outlier detection algorithm is presented in a pseudo code format in Table 1.

3.4. Time complexity analysis

From the description in the previous subsections, it can be seen that our algorithm mainly consists of three steps, a sequential initialization, a DHCA updating, and finally top outliers' mining. The time complexity for sequential initialization is dNk. Since the number of outliers is small, the third step takes a nearly linear time on average. For the runs of DHCA, we use a tree structure to analyze its time complexity.

In d-dimensional space, the time complexity of the *kNN* computation is upper bounded by $O(dN^2)$. However, if fortunately, a dataset can be evenly partitioned into several well separated clusters, such as the one shown on the left of Fig. 6, the computation cost can be reduced to $dN^2/4$. Then how to quickly find *kNN* for a data point becomes how to quickly partition data into size reduced well separated equally distributed clusters. It is in this sense that DHCA provides a very good way to partition data quickly.

At the top level of DHCA, the number of distance computations is dNK, where K is the number of subsets for DHCA. For each subsequent level l, if it is a balanced tree, there are K^l subsets of size N/K^l, and the partition cost is still dNK. At the lowest lever, since $K+2$ number of data points is quite small, the pairwise distance computation

Table 1
Our MST-inspired outlier detection algorithm.

Input:
data a set of *N* data points
k the number of *NN*s of a data item
K the number of clusters at each step
SOM-TH global outlier detection termination threshold
MAX-MIN-TH local outlier detection termination threshold
Output:
a set of ranked global outliers *GO*, a set of ranked local outliers *LO*
Begin
set *k* to be the largest outlying cluster size plus 1
perform a sequential initialization (SI)
run DHCA multiple times until the percentage difference between two consecutively updated *kNN* is below 10^{-6};
find miniMST for each data point
for each *i*=2:*k*
{ compute three 1-dimensional arrays of the estimated outlying scores for *iNN*, namely, *MST-MAX*, *MST-MAX-MIN* and SOM_{MST}, and sort the first two in a non-increasing order and the orders are remembered in *MST-MAX–INDEX* and *MST-MAX-MIN-INDEX*, respectively;
GO_T=[];
for *j*=1:*N*
{
find true *iNN* for *MST-MAX-INDEX*[*j*];
recomputed the global score and SOM_{MST};
if (SOM_{MST}[*MST-MAX-INDEX*[*j*]] > *SOM-TH*)
GO_T=[*GO_T MST-MAX-INDEX*[*j*]];
end
}
GO=[*GO GO_T*];
LO_T=[];
for *j*=1:*N*
{
find *iNN* for *MST-MAX-MIN-INDEX*[*j*];
recomputed the local score;
if(*MST-MAX-MIN*[*j*]] > *MAX-MIN-TH*)
LO_T=[*LO_T MST-MAX-MIN-INDEX*[*j*]];
end
}
LO=[*LO LO_T*];
}
End

GO_T and *LO_T* are temporary arrays to hold ranked outliers for each *i*.

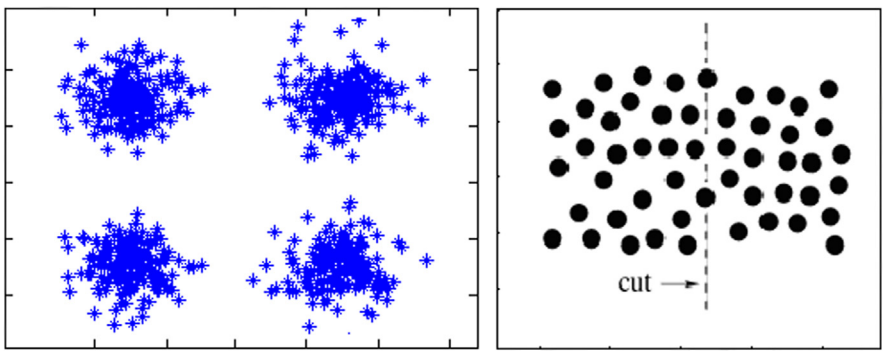

Fig. 6. An illustration of the effect of DHCA.

takes a nearly constant time. For a balanced tree, the tree height *l* is $\log_K N$. The best case time complexity of a DHCA is $O(dNK \log_K N+N)$.

However, in practice, we may not have a balanced tree, and it can happen that a partition may result in two completely unbalanced subsets where one set contains most of the points. Therefore, the worst time complexity to DHCA could be $O(dN^2)$. Fortunately, by randomly selecting the partition centers, the probability of isolating a point from its neighbors is small, and the worst case when all the data items are clustered into the same partition will be diminished. The average time complexity of DHCA can be roughly approximated as the case of the balanced tree. On the other hand, as shown on the right of Fig. 6, any data point in a partition is closer to its cluster center (not its nearest neighbor) than to the center of any other partition, the data points in the clusters' boundaries can be misclassified into a wrong partition. Fortunately, MSTs are

Table 2
The node class.

Name	Explanation
public data members:	
sampleNumbers;	An array holding the indices of all samples in the cluster
centroid;	An array holding the indices of the randomly chosen cluster centers
childNodes;	An array holding the indices of its child Nodes in the Node array
parent;	An integer holding the index of its parent Node in the Node array
vecindex;	An integer holding the index of the current Node in the Node array
public Methods:	
void DHCA	Our DHCA Procedure

Table 3
The DHCA member function.

Procedure name	DHCA
Input:	
dist_knn, edge_knn	The auxiliary arrays to remember *k*-nearest neighbors(*kNN*) for each data item
k	The number of *NN*s of a data item
nodeArray	An array of the *Node* structures
currentNode	The current *Node* in the *Node* array
K	The number of clusters at each step
data	The input data set
clustersize	The maximum size of each clusters
Output:	
updated *dist_knn, edge_knn*, and newly generated $\leq K$ Nodes	
Begin	
randomly select *K* centers from *sampleNumbers* of *currentNode;*	
generate *K* newNodes;	
for each sample *i* in *sampleNumbers* of *currentNode* that is not a center	
{ find its nearest center *j* out of *K*;	
if (*dist_knn*[*i*].*max* > distance(*i*,*j*)	
update *dist_knn*, *edge_knn;*	
end	
assign *sampleNumbers*[*i*] to the group of center *j*;	
}	
for each newNode *j*=1: *K*	
{ if (newNode[*j*].*sampleNumbers*.size > *clustersize*)	
push newNode[*j*] to the end of *nodeArray*;	
assign values to data members parent and vecindex;	
end	
}	
End	

dist_knn[*i*].*max* is the *kNN*th nearest neighbor of data item *i*.

relatively insensitive to this effect, which can be easily figured out from Fig. 6 Therefore, a few DHCA will meet our needs.

As a result, we expect the time complexity of our algorithm to be $O(dfNK \log_K N)$, where *f* denotes the number of DHCA run.

3.5. Pseudo code for DHCA

The implementation of the DHCA in our approach is through the design of a C++ data structure called Node, which has several member variables for bookkeeping and a main member function that clusters its own set into *K* subclusters. The outputs of the Node data structure are at most *K* new Nodes as the descendents of the current one.

The divisive hierarchical clustering process starts with a Node instance, called the topNode, which has every data item in the data set as its samples, and generates *K* data subsets in the form of *K* Nodes. Only when the number of samples in a Node is larger than a predefined cluster size will that Node be pushed to the back of the topNode, forming an array of Nodes. This process continues recursively until no new Nodes are generated and the end of the existing Node array is reached. The Node class is summarized in Table 2 and the DHCA procedure is given in Table 3.

4. A performance study

In this section, we present the results of three sets of experiments performed to evaluate our MST-inspired outlier detection algorithm. In Experiment 1, three 2-dimensional synthetic data sets are used to show that our MST-inspired outlier detection method can outperform classic outlier detection algorithms in classification accuracy. In Experiment 2, five real data sets obtained from the UCI Machine Learning Repository [33] are used to check the technical robustness of this study and to illustrate the effectiveness of our method in real-world situation.

Table 4
the sets of data.

Data set	Size (N)	Dimensionality
Dataset1	515	2
Dataset2	78	2
Dataset3	473	2
Shuttle	14 500	9
Lymphography	148	18
Ionosphere	351	34
Optical digits	5620	64
Multiple features	2000	649
IPUMS	88 443	68
Covertype	581 012	55
UScensus	2 458 285	61

In Experiment 3, we evaluate the run time performance of the proposed algorithm on three large real data sets and compare it with that of MST+LOF outlier detection algorithm to show the impact of different data sizes and input of k's on our algorithm. All the data sets are briefly summarized in Table 4. All the algorithms are implemented in C/C++ and run on a computer with Intel Core 2 Duo Processor E6550 2.33 GHz CPU and 2 GB RAM. The operating system running on this computer is Windows XP. We use the timer utilities defined in the C standard library to report the CPU time. The results show that, overall, our MST-inspired outlier detection algorithm is superior to other state-of-the-art outlier detection algorithms in both classification accuracy and the execution time. For consistency, we only use the parameter k to represent the neighborhood size in the investigation of these methods.

4.1. Performance of our algorithm on synthetic datasets

In this subsection, we use three synthetic data sets to show that the proposed method can efficiently identify local and global outliers in various scenarios. In each dataset, there are multiple clusters and six outliers (A, B, C, D, E, and F) are planted in the vicinities of the clusters. A particular challenging feature of these data sets is that clusters are of different sizes and have different densities. We exclusively use 2-dimensional data sets due to their convenience for visual inspection.

To study the relative effects of our proposed outlier detection algorithm to parameter k, for comparison with DB, DB-MAX, LOF, COF, INFLO, LDOF, RBDA, and MST+LOF, we first decide the largest number of data points for an outlying cluster, next calculate the outlier scores given by each outlier definition, sort them in a nondecreasing order and return top ones.

Synthetic dataset 1 consists of 515 instances, including six planted outliers, one large normally-distributed cluster and two small uniform clusters. This is a global outlier detection task. For our approach, if the size of the largest outlying group is set to 2, k is set to 3 for all the algorithms in this case. The experimental results are depicted in Fig. 7 where 6 top outliers are marked with red plus, red star, red cross, red triangle, red circle and red square in order, respectively.

From the figures, it can be seen that, DB, DB-MAX, LOF, INFLO MST+LOF and our method (i.e., *MST-MAX*) correctly detect all the outliers, though the rankings are slightly different for different methods. Unfortunately, COF does not do very well, and misses A, D, F. Since the average of k nearest distances and that of the $k(k-1)$ pairwise inner distances are both large for B,D,F, their ratios are near 1 and they are missed by LDOF. RBDA misses one global outlier, (i.e., A) in Fig. 7. This phenomenon is due to the fact that the rank of A in its 3 nearest neighbors is relatively lower than the ranks of H in its 3 nearest neighbors. PBOD detects 3 right for top 6 outliers.

So far, it is implicitly assumed that the number of outliers is given beforehand and it is not clear when the detection process should come to a close. As an improvement, SOM_{MST} can be used as a first degree approximation for such purpose.

To test our termination mechanism, we show top 7 outliers detected using our algorithm for $k=3$ in the middle plot on the last line of Fig. 7. From the figure, we can observe the limitation of our global outlier factor, that is, data point K is mistakenly detected by our global outlier definition. Fortunately, our termination mechanism for global outlier detection identifies data object K to have a very low SOM_{MST} value (which is actually 0) and declares the end of global outlier detection.

In addition to 6 planted outliers, there are three clusters existing. Correspondingly, if the size of the largest outlying group is set to 9, the dataset has only a major data pattern. Depicted in Fig. 8 are the experimental results for each method when k is set to 10, at which case, there are totally 24 outliers. To simplify the illustration, totally 24 top outliers are marked with red plus.

From the figure, it can be seen that DB misses one for $k=10$. LOF identifies only 10 out of 24 correctly. COF does not do very well, either and misclassifies 19 out of 24. INFLO gets only 9 out of 24. LDOF does not detect two clusters of size 9 and therefore misses 18 out of 24. RBDA misses 9 points in one of the two clusters of size 9. PBOD works a little better for this case and only misses 2 for top 24 outliers. LOCI misses 2 but with quite some false positives. DB-MAX, MST+LOF and our method (i.e., *MST-MAX*) detect all the outliers correctly.

To test if our approach can effectively find meaningful outliers in a little more complex dataset, we use Synthetic Dataset 2 which contains 78 instances, including five planted global outliers, one planted local outlier B, and four clusters of different densities consisting of 36, 8, 12 and 16 uniformly distributed instances.

To demonstrate the effectiveness of our approach in finding both global and local outliers, we compare the effectiveness of the distance-based and density-based methods on this dataset. For these *kNN*-based methods, we first set $k=3$ and the screen shots in Fig. 9 show the results for mining top 6 outliers. For this case, DB and DB-MAX both miss B and C, and have the same ranks for A, D, E and F. LDOF, our *MST-MAX* and MST+LOF all miss one of them while PBOD misses 5. RBDA, INFLO, COF, LOF and our local outlier factor *MST-MAX-MIN* detect all six outliers correctly. The plot at lower bottom-right corner shows those data points whose SOM_{MST} values above 1.5, which correctly identifies the six outliers.

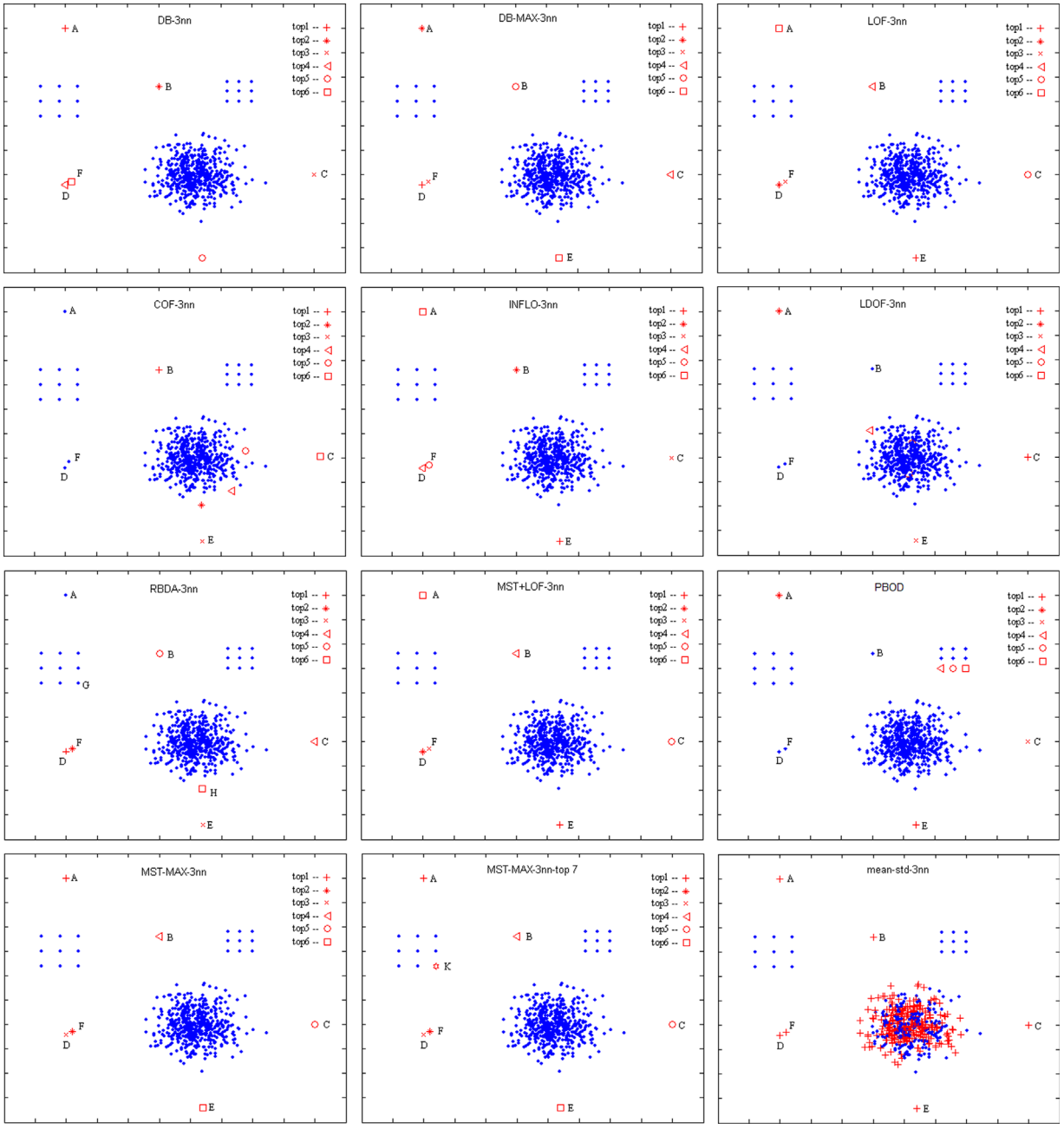

Fig. 7. The outlier detecting results on synthetic dataset 1 for $k=3$.

In addition to six planted outliers, there is also a small cluster consisting of 8 points. If it is also regarded as an outlying group, we set k to 9. For this case, there are 14 outliers. The detection results are shown in Fig. 10. From the figure, it can be observed that DB-MAX does not perform well when local outlier exists. DB misses 10, DB-MAX misses 11, LOF misses 3, COF misses 6, INFLO misses 4, LDOF misses 8, RBDA misses 6, PBOD misses 12. LOCI misses 11. MST+LOF misses 1. Our local outlier detector, *MST-MAX-MIN*, detects all right.

From Figs. 9 and 10, the advantage of our local outlier detection factor, *MST-MAX-MIN*, is very evident on this comparably low dimensional data set having not only global outliers, but also local outliers. The experiment shows that our method is suitable to mixed type data as well in this sense.

To test our method as a whole, Synthetic dataset 3, which has 473 data points and consists of 5 clusters with different densities and 6 outliers, is used. A particular challenging feature of this data set is that three denser clusters are buried into one sparse cluster on the upper right corner. Since the largest outlying group has 2 data points (i.e., E and F), k is first set to 3. The performance of outlier detection methods is shown in Fig. 11. The same

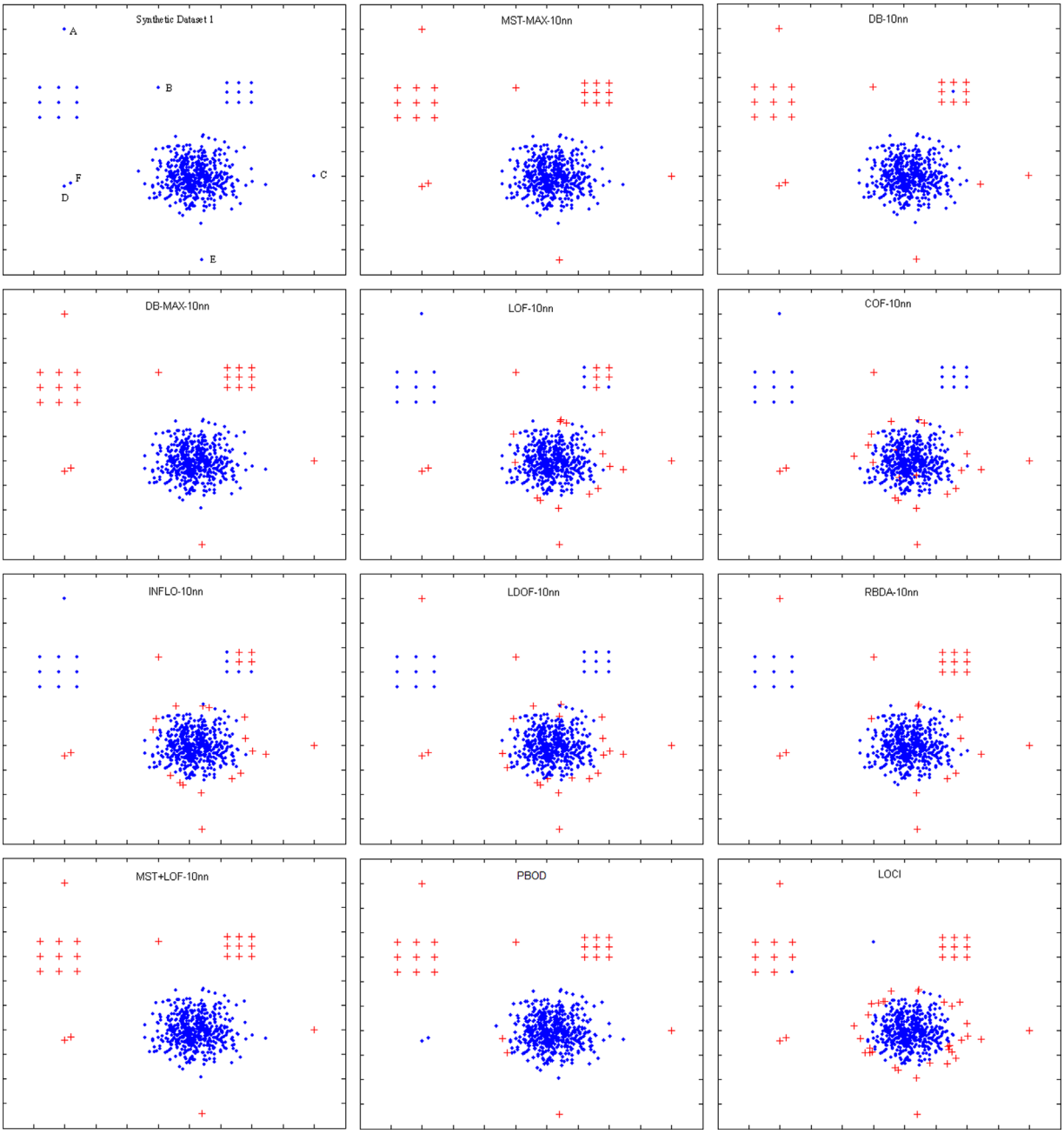

Fig. 8. The outlier detecting results on synthetic dataset 1 for $k=10$.

conventions are employed for the plots. From the figure, it is easily seen that this is also a global outlier detection situation with the difference from Synthetic dataset 1 being that the detection process is disturbed by the immediate connection of clusters with different densities.

For detecting top 6 outliers, COF misses B and C, LDOF misses E and F, RBDA misses C and PBOD misses all. The good news is that DB, DB-MAX, LOF, INFLO, MST+LOF and our method detect all six outliers correctly but with different rankings. If the sparse cluster (i.e., the one consisting of 7 data points) is also regarded as an outlying group, k is set to 8. For this situation, both DB-MAX and LOF misses 1, COF misses 7, INFLO misses 2, LDOF misses 8, RBDA misses 6, PBOD misses 6, LOCI misses 8, while DB, MST+LOF, and our outlier detector get all right, as shown in Fig. 12.

To summarize, it can be observed from Figs. 7–12 that our method has no problems detecting all outliers and clearly offer the best ranking in three synthetic datasets while all other methods do not perform competently with detecting all the outliers for two different k's one way or the other. In other words, the advantage of our outlier detection factors is very evident on these 2-dimensional data sets.

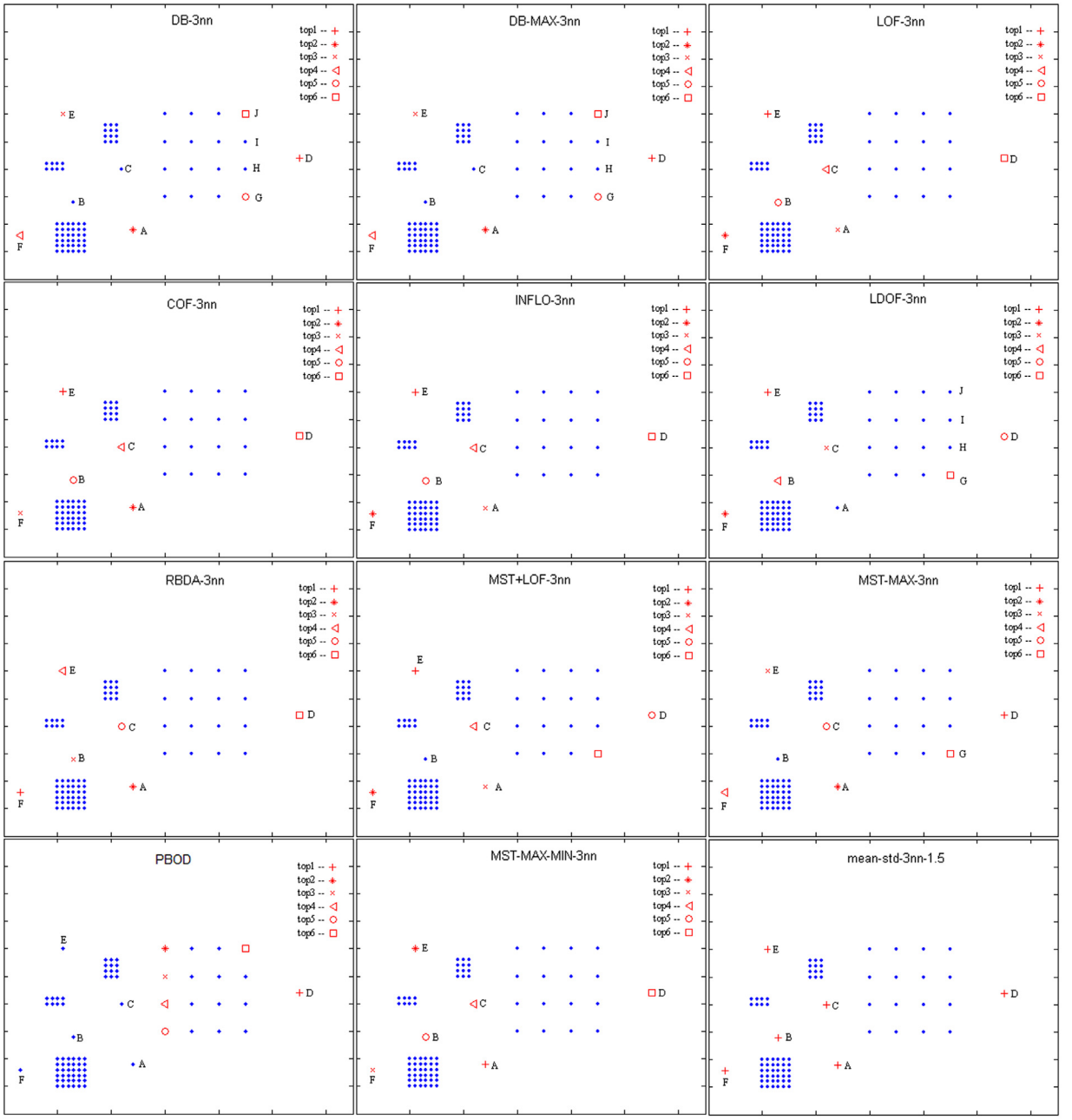

Fig. 9. The outlier detecting results for synthetic dataset 2 for $k=3$.

4.2. Performance on real datasets

As pointed out by Aggarwal and Yu, one way to test how well the outlier detection algorithm works is to run the method on the dataset and test the percentage of points which belongs to the rare classes [34]. To evaluate the effectiveness and accuracy of our proposed method on real data, we compare the algorithms by their performance on detecting rare classes in five real datasets, namely, the Lymphography, Shuttle, Ionosphere, Optical Digits and Multiple Features datasets, which are downloaded from UCI [33].

To quantitatively measure the performance of an outlier detection scheme, three popular metrics, namely, *precision*, *recall*, and *rank power*, are employed here. Assuming that a dataset $D=Do\cup Dn$ where Do denotes the set of all outliers and Dn denotes the set of all normal data. Given any integer $m\geq 1$, if O_m denotes the set of outliers among the objects in the top m positions returned by an outlier detection scheme, *precision* and *recall* are defined as

$$precision=\frac{|O_m|}{m} \tag{8}$$

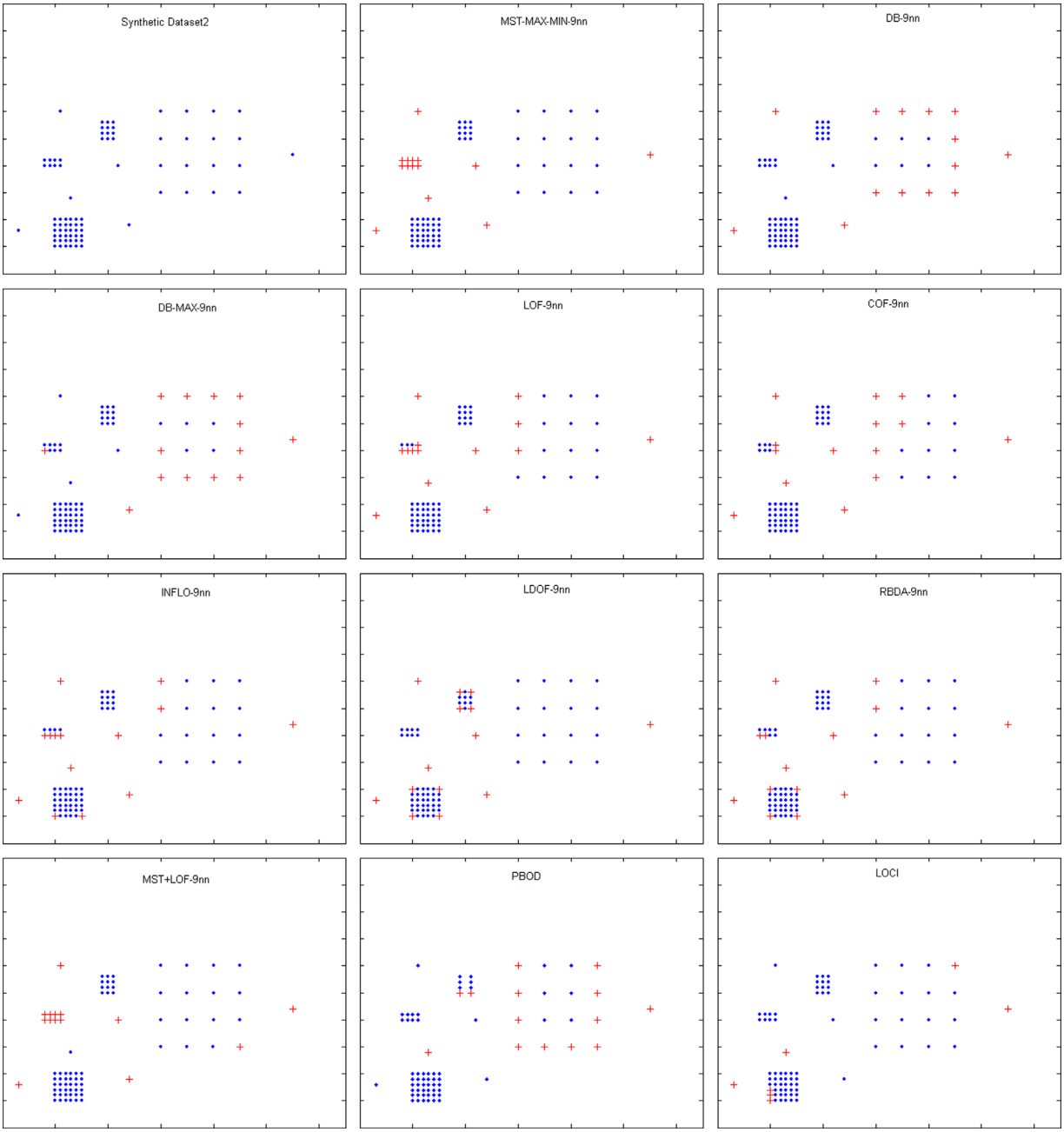

Fig. 10. The outlier detecting results for synthetic dataset 2 for $k=9$.

$$recall = \frac{|O_m|}{|D_o|} \tag{9}$$

Therefore, *precision* measures the percentage of true outliers among top m ranked objects returned by a method, while *recall* measures the percentage of total outlier set included in top m ranked objects. Usually, users are not only interested in how many true outliers being returned by a method, but also in where they are placed. *rank power* is a metric that considers both the placements and the number of results returned by a method. Here, the *rank power* given in [35] is used. Suppose that a method returns m objects, n of which are true outliers. For $1 \le i \le n$, if L_i denotes the position of the ith outlier, the *rank power* of the method with respect to m can be defined as

$$Rank\ Power = \frac{n(n+1)}{2\sum_{i=1}^{n} L_i} \tag{10}$$

As can be seen from Eq. (10), *rank power* weighs the placements of the returned outliers heavily. An outlier placed earlier in the returned list adds less to the denominator of the *rank power* (and thus contributes more to the

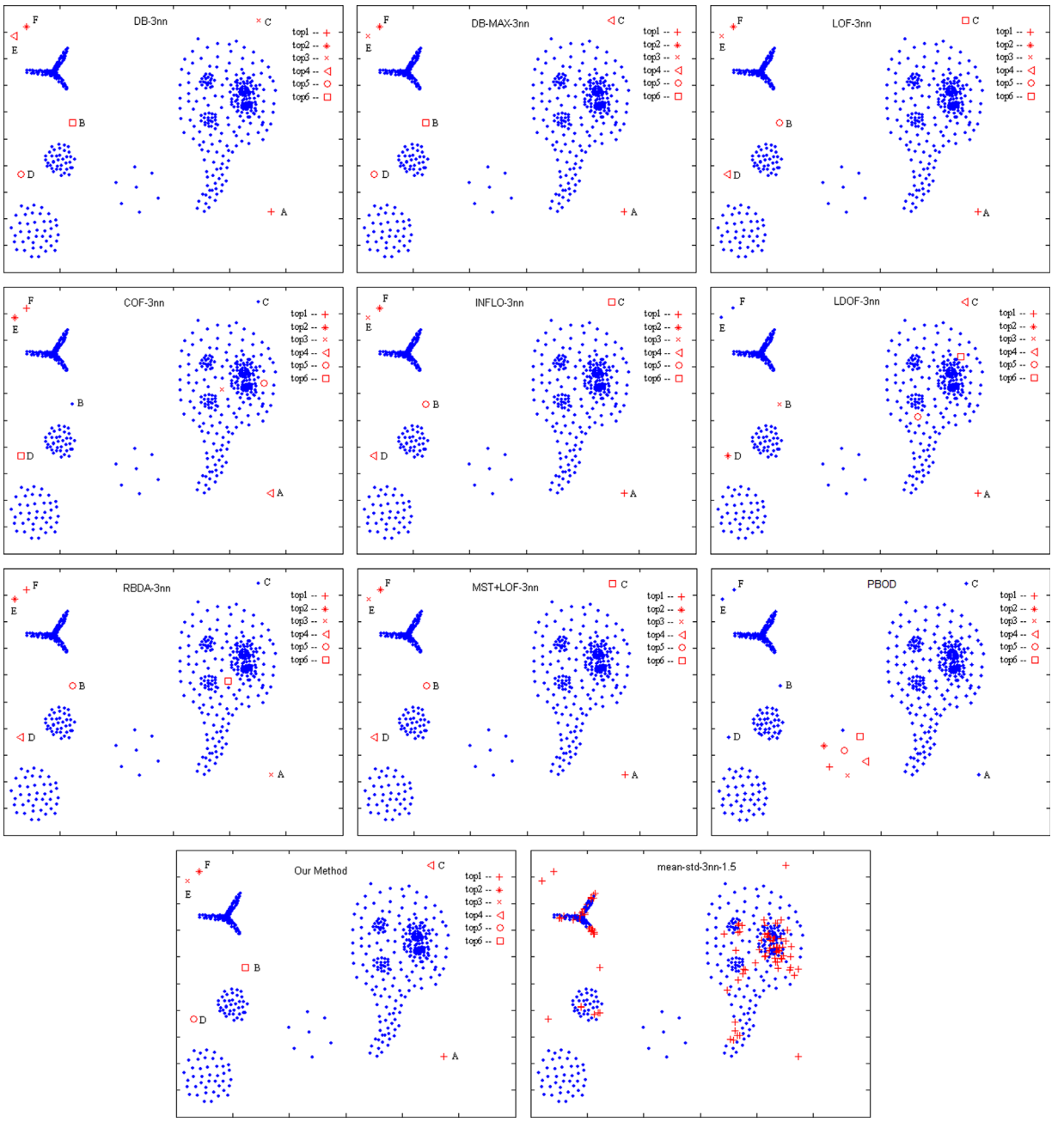

Fig. 11. The outlier detecting results for synthetic dataset 3 using $k=3$.

rank power metric) than those placed later in the list. A value of 1 indicates the best performance and 0 indicates the worst.

To measure the capability of each algorithm to retrieve the most likely outliers and to compare the quality of the ranking provided by each algorithm, in the following, the performance is measured with the three metrics of *recall*, *precision*, and *rank power*, denoted by p, r and rp, respectively. The value of m indicates top m ranked records returned by our scheme. *SOM-TH* is set to 1.5 and *MAX-MIN-TH* is set to 3 for all the experiments in this subsection using our method.

4.2.1. Lymphography data

The Lymphography dataset has 148 instances with 18 attributes and contains a total of 4 classes. Classes 2 and 3 have the largest number of instances (81 and 61, respectively). The remaining two classes have 6 instances in total (2 and 4, respectively) and are regarded as outliers (i.e., rare classes) for they are small in size. We report the corresponding detecting results of our method, PBOD method and three best cases of the rest methods in Table 5 for four values of k and five values of m.

For k equal to 7, PBOD is the first one to mine all six outliers within top 10 ranked instances. LOF and our

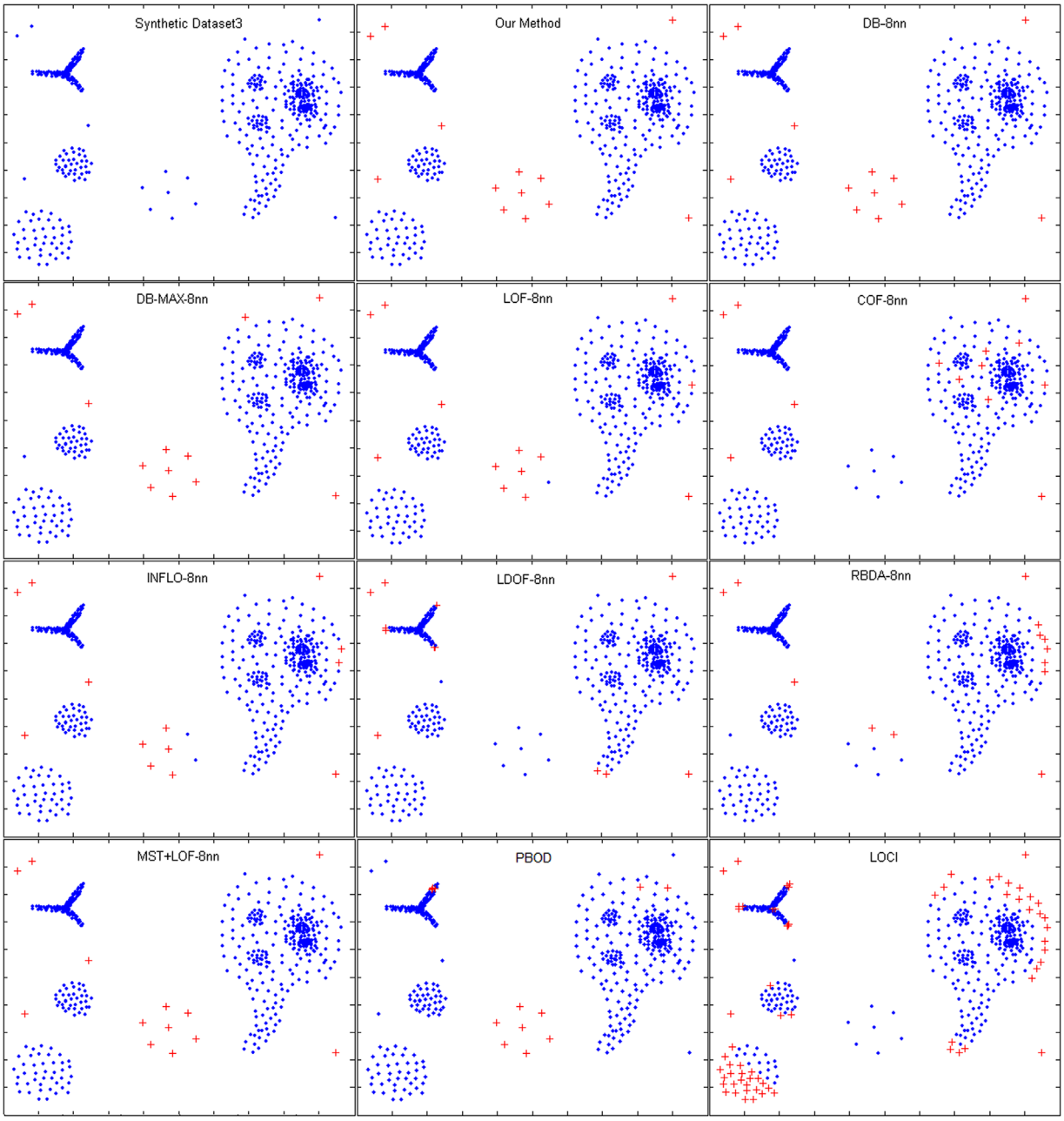

Fig. 12. The outlier detecting results for synthetic dataset 3 for $k=8$.

method are the second one. For k equal to 30, LOF, INFLO and RBDA perform better than PBOD. The detection performances of our method are similar with the other four methods and stay the same for all k's, that is, our method is less sensitive to parameter k, which agrees with our expectation.

4.2.2. Space shuttle data

The Shuttle dataset contains 14500 objects, in which each object has 9 real-valued features and an integer label (1–7), and has 7 clusters. We regard those objects with labels 2, 6 and 7 (with 13, 4 and 2 objects respectively) as outliers, and the rest 4 classes as normal data (i.e., classes 1, 3, 4, 5, with 11478, 39, 2155 and 809 instances, respectively). We show in Table 6 the experimental results of our method, PBOD method and three best cases of the rest methods for four different values of k's and seven values of m's. As reflected in the table, this is not a very easy task for all other techniques. Our method performs significantly better in comparison with others for all k's and all m's. More specifically, our method has the highest *precision* and *recall* and is less sensitive to k as in the case

Table 5
Lymphography, κ=7,10,20,30.

m	LOF				INFLO				RBDA				Our method				PBOD			
	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp
k=7																				
5	4	0.80	0.67	0.83	3	0.60	0.50	0.60	3	0.60	0.50	0.86	4	0.80	0.67	0.71	2	0.40	0.33	**1.00**
10	5	0.50	0.83	0.75	5	0.50	0.83	0.58	4	0.40	0.67	0.71	5	0.50	0.83	0.68	**6**	**0.60**	**1.00**	0.88
15	**6**	**0.40**	**1.00**	0.64	5	0.33	0.83	0.58	5	0.33	0.83	0.60	**6**	**0.40**	**1.00**	0.57	**6**	**0.40**	**1.00**	0.88
20	**6**	**0.30**	**1.00**	0.64	5	0.25	0.83	0.58	5	0.25	0.83	0.60	**6**	**0.30**	**1.00**	0.45	**6**	**0.30**	**1.00**	0.88
30	**6**	**0.20**	**1.00**	0.64	**6**	**0.20**	**1.00**	0.40	**6**	**0.20**	**1.00**	0.42	**6**	**0.20**	**1.00**	0.37	**6**	**0.20**	**1.00**	0.88
k=10																				
5	4	0.80	0.67	0.83	3	0.60	0.50	0.60	3	0.60	0.50	1.00	4	0.80	0.67	0.71	–	–	–	–
10	5	0.50	0.83	0.79	5	0.50	0.83	0.65	5	0.50	0.83	0.65	5	0.50	0.83	0.68	–	–	–	–
15	5	0.33	0.83	0.79	5	0.33	0.83	0.65	5	0.33	0.83	0.65	**6**	**0.40**	**1.00**	0.49	–	–	–	–
20	**6**	**0.30**	**1.00**	0.58	5	0.25	0.83	0.65	5	0.25	0.83	0.65	**6**	**0.30**	**1.00**	0.49	–	–	–	–
30	**6**	**0.20**	**1.00**	0.58	5	0.17	0.83	0.65	5	0.17	0.83	0.65	**6**	**0.20**	**1.00**	0.40	–	–	–	–
k=20																	–	–	–	–
5	4	0.80	0.67	**1.00**	4	0.80	0.67	**1.00**	4	0.80	0.67	**1.00**	4	0.80	0.67	0.77	–	–	–	–
10	5	0.50	0.83	0.94	5	0.50	0.83	0.94	5	0.50	0.83	0.94	5	0.50	0.83	0.60	–	–	–	–
15	5	0.33	0.83	0.94	**6**	**0.40**	**1.00**	0.68	5	0.33	0.83	0.94	**6**	**0.40**	**1.00**	0.40	–	–	–	–
20	**6**	**0.30**	**1.00**	0.66	**6**	**0.30**	**1.00**	0.68	**6**	**0.30**	**1.00**	0.58	**6**	**0.30**	**1.00**	0.40	–	–	–	–
30	**6**	**0.20**	**1.00**	0.66	**6**	**0.20**	**1.00**	0.68	**6**	**0.20**	**1.00**	0.58	**6**	**0.20**	**1.00**	0.32	–	–	–	–
k=30																	–	–	–	–
5	4	0.80	0.67	**1.00**	4	0.80	0.67	0.83	4	0.80	0.67	**1.00**	4	0.80	0.67	0.77	–	–	–	–
10	**6**	**0.60**	**1.00**	0.84	**6**	**0.60**	**1.00**	0.78	**6**	**0.60**	**1.00**	0.84	5	0.50	0.83	0.71	–	–	–	–
15	**6**	**0.40**	**1.00**	0.84	**6**	**0.40**	**1.00**	0.78	**6**	**0.40**	**1.00**	0.84	**6**	**0.40**	**1.00**	0.44	–	–	–	–
20	**6**	**0.30**	**1.00**	0.84	**6**	**0.30**	**1.00**	0.78	**6**	**0.30**	**1.00**	0.84	**6**	**0.30**	**1.00**	0.38	–	–	–	–
30	**6**	**0.20**	**1.00**	0.84	**6**	**0.20**	**1.00**	0.78	**6**	**0.20**	**1.00**	0.84	**6**	**0.20**	**1.00**	0.31	–	–	–	–

Note: maximum values are marked bold.

Table 6
Shuttle, κ=7,20,40,60.

m	DB				LOF				RBDA				Our method				PBOD			
	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp
k=7																				
20	6	0.30	0.32	0.23	1	0.05	0.21	0.06	1	0.05	0.05	0.20	9	0.45	0.47	0.35	6	0.30	0.32	0.34
30	6	0.20	0.32	0.23	2	0.07	0.10	0.07	2	0.07	0.11	0.09	12	0.40	0.63	0.36	6	0.20	0.32	0.34
60	8	0.13	0.42	0.18	10	0.17	0.53	0.14	8	0.13	0.42	0.13	16	0.27	0.84	0.33	7	0.12	0.37	0.27
90	9	0.10	0.47	0.16	11	0.12	0.58	0.14	10	0.11	0.53	0.13	17	0.19	0.89	0.31	7	0.08	0.37	0.27
120	10	0.08	0.53	0.14	11	0.09	0.58	0.14	12	0.10	0.63	0.13	18	0.15	0.95	0.25	7	0.06	0.37	0.27
140	11	0.08	0.58	0.13	11	0.08	0.58	0.14	15	0.11	0.79	0.12	18	0.1	0.95	0.25	7	0.05	0.37	0.27
200	16	0.08	0.84	0.10	13	0.06	0.68	0.11	16	0.08	0.84	0.12	**19**	**0.09**	**1.00**	0.12	7	0.04	0.37	0.27
k=20																				
20	6	0.30	0.32	0.23	4	0.20	0.21	0.32	1	0.05	0.05	0.08	11	0.55	0.58	0.40	–	–	–	–
30	6	0.20	0.32	0.23	5	0.17	0.26	0.28	4	0.13	0.21	0.10	14	0.47	0.74	0.42	–	–	–	–
60	7	0.12	0.37	0.19	9	0.15	0.47	0.20	7	0.12	0.37	0.13	18	0.30	0.95	0.37	–	–	–	–
90	9	0.10	0.47	0.15	13	0.14	0.68	0.17	9	0.10	0.47	0.12	**19**	**0.21**	**1.00**	0.33	–	–	–	–
120	10	0.08	0.53	0.13	15	0.12	0.79	0.16	13	0.11	0.68	0.11	**19**	**0.16**	**1.00**	0.27	–	–	–	–
140	11	0.08	0.58	0.12	17	0.12	0.89	0.15	15	0.11	0.79	0.11	**19**	**0.14**	**1.00**	0.27	–	–	–	–
200	**19**	**0.10**	**1.00**	0.10	**19**	**0.09**	**1.00**	0.15	**19**	**0.10**	**1.00**	0.11	**19**	**0.09**	**1.00**	0.12	–	–	–	–
k=40																				
20	6	0.30	0.32	0.23	4	0.20	0.21	0.29	0				11	0.55	0.58	0.40	–	–	–	–
30	6	0.20	0.32	0.24	5	0.17	0.26	0.25	3	0.10	0.16	0.07	14	0.47	0.74	0.42	–	–	–	–
60	6	0.10	0.32	0.24	8	0.13	0.42	0.18	7	0.12	0.37	0.12	18	0.30	0.95	0.37	–	–	–	–
90	9	0.10	0.47	0.15	9	0.10	0.47	0.16	8	0.09	0.42	0.11	**19**	**0.21**	**1.00**	0.33	–	–	–	–
120	9	0.08	0.47	0.15	11	0.09	0.58	0.14	13	0.11	0.68	0.11	**19**	**0.16**	**1.00**	0.27	–	–	–	–
140	11	0.08	0.58	0.12	16	0.11	0.84	0.12	15	0.11	0.79	0.11	**19**	**0.14**	**1.00**	0.27	–	–	–	–
200	18	0.09	0.95	0.10	**19**	**0.09**	**1.00**	0.12	**19**	**0.10**	**1.00**	0.10	**19**	**0.09**	**1.00**	0.12	–	–	–	–
k=60																				
20	6	0.30	0.32	0.23	4	0.20	0.21	0.23	0				11	0.55	0.58	0.40	–	–	–	–
30	6	0.20	0.32	0.24	5	0.17	0.26	0.22	3	0.10	0.16	0.07	14	0.47	0.74	0.42	–	–	–	–
60	6	0.10	0.32	0.24	8	0.13	0.42	0.17	7	0.12	0.37	0.12	18	0.30	0.95	0.37	–	–	–	–
90	8	0.09	0.42	0.17	8	0.09	0.42	0.17	7	0.08	0.37	0.12	**19**	**0.21**	**1.00**	0.33	–	–	–	–
120	9	0.08	0.47	0.15	11	0.09	0.58	0.12	13	0.11	0.68	0.11	**19**	**0.16**	**1.00**	0.27	–	–	–	–
140	10	0.07	0.53	0.13	16	0.11	0.84	0.11	14	0.10	0.74	0.11	**19**	**0.14**	**1.00**	0.27	–	–	–	–
200	16	0.08	0.84	0.09	**19**	**0.09**	**1.00**	0.11	15	0.08	0.79	0.11	**19**	**0.09**	**1.00**	0.12	–	–	–	–

Table 7
Ionosphere, κ=5,10,20,30.

m	DB				RBDA				MST+LOF				Our method				PBOD			
	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp
k=5																				
5	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	4	0.80	0.03	**1.00**	**5**	**1.00**	**0.04**	**1.00**	4	0.80	0.03	**1.00**
10	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	9	0.90	0.07	**1.00**	**10**	**1.00**	**0.08**	**1.00**	7	0.70	0.06	0.76
30	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	29	0.97	0.23	**1.00**	**30**	**1.00**	**0.24**	**1.00**	20	0.67	0.16	0.73
60	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	59	0.98	0.47	**1.00**	59	0.98	0.47	**1.00**	32	0.53	0.25	0.72
90	88	0.98	0.70	**1.00**	88	0.98	0.70	0.99	89	0.99	0.71	0.99	83	0.92	0.66	**1.00**	34	0.38	0.27	0.69
120	107	0.90	0.85	0.98	98	0.82	0.78	0.98	106	0.88	0.84	0.94	94	0.78	0.75	0.97	58	0.48	0.46	0.51
130	109	0.84	0.87	0.98	100	0.77	0.79	0.97	109	0.84	0.86	0.92	96	0.74	0.76	0.96	68	0.52	0.54	0.51
140	110	0.79	0.87	0.97	103	0.74	0.82	0.96	113	0.81	0.90	0.91	98	0.70	0.78	0.95	70	0.50	0.56	0.51
k=10																				
5	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	4	0.80	0.03	**1.00**	**5**	**1.00**	**0.04**	**1.00**	–	–	–	–
10	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	9	0.90	0.07	**1.00**	**10**	**1.00**	**0.08**	**1.00**	–	–	–	–
30	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	29	0.97	0.23	**1.00**	**30**	**1.00**	**0.24**	**1.00**	–	–	–	–
60	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	59	0.98	0.47	**1.00**	**60**	**1.00**	**0.48**	**1.00**	–	–	–	–
90	88	0.98	0.70	**1.00**	88	0.98	0.70	0.99	89	0.99	0.71	0.99	89	0.99	0.71	**1.00**	–	–	–	–
120	107	0.90	0.85	0.98	97	0.81	0.77	0.98	106	0.88	0.84	0.95	96	0.80	0.76	0.99	–	–	–	–
130	109	0.84	0.87	0.98	100	0.77	0.79	0.97	111	0.85	0.88	0.93	99	0.76	0.79	0.97	–	–	–	–
140	112	0.80	0.89	0.96	101	0.72	0.80	0.96	114	0.81	0.90	0.93	100	0.71	0.79	0.97	–	–	–	–
k=20																				
5	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	4	0.80	0.03	**1.00**	**5**	**1.00**	**0.04**	**1.00**	–	–	–	–
10	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	9	0.90	0.07	**1.00**	**10**	**1.00**	**0.08**	**1.00**	–	–	–	–
30	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	29	0.97	0.23	**1.00**	**30**	**1.00**	**0.24**	**1.00**	–	–	–	–
60	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	59	0.98	0.47	**1.00**	**60**	**1.00**	**0.48**	**1.00**	–	–	–	–
90	85	0.94	0.67	**1.00**	84	0.93	0.67	0.99	89	0.99	0.71	**1.00**	**90**	**1.00**	**0.71**	**1.00**	–	–	–	–
120	103	0.86	0.82	0.97	98	0.82	0.78	0.96	106	0.88	0.84	0.94	103	0.86	0.82	**1.00**	–	–	–	–
130	107	0.82	0.85	0.96	102	0.78	0.81	0.95	110	0.85	0.87	0.90	105	0.81	0.83	0.99	–	–	–	–
140	111	0.79	0.88	0.95	105	0.75	0.83	0.93	111	0.79	0.88	0.89	106	0.76	0.84	0.98	–	–	–	–
k=30																				
5	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	4	0.80	0.03	**1.00**	**5**	**1.00**	**0.04**	**1.00**	–	–	–	–
10	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	9	0.90	0.07	**1.00**	**10**	**1.00**	**0.08**	**1.00**	–	–	–	–
30	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	29	0.97	0.23	**1.00**	**30**	**1.00**	**0.24**	**1.00**	–	–	–	–
60	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	59	0.98	0.47	**1.00**	**60**	**1.00**	**0.48**	**1.00**	–	–	–	–
90	83	0.92	0.66	0.99	86	0.96	0.68	0.99	89	0.99	0.71	**1.00**	**90**	**1.00**	**0.71**	**1.00**	–	–	–	–
120	98	0.82	0.78	0.96	98	0.82	0.78	0.97	106	0.88	0.84	0.93	107	0.89	0.85	**1.00**	–	–	–	–
130	103	0.79	0.82	0.94	102	0.78	0.81	0.95	111	0.85	0.88	0.90	108	0.83	0.86	0.99	–	–	–	–
140	106	0.76	0.84	0.93	105	0.75	0.83	0.94	112	0.80	0.89	0.89	109	0.78	0.86	0.98	–	–	–	–

Table 8
Optical digits, κ=7.

m	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp
	DB				DB-MAX				RBDA				Our method				MST+LOF			
Class '2'																				
5	0	0.00	0.00	0.00	2	0.40	0.40	0.50	0	0.00	0.00	0.00	1	0.20	0.20	0.33	0	0.00	0.00	0.00
10	1	0.10	0.20	0.13	3	0.30	0.60	0.40	1	0.10	0.20	0.20	3	0.30	0.60	0.43	0	0.00	0.00	0.00
15	2	0.13	0.40	0.16	4	0.27	0.80	0.40	1	0.07	0.20	0.20	4	0.27	0.80	0.42	0	0.00	0.00	0.00
20	3	0.15	0.60	0.18	4	0.20	0.80	0.40	1	0.05	0.20	0.20	**5**	**0.25**	**1.00**	0.38	0	0.00	0.00	0.00
30	4	0.13	0.80	0.16	**5**	**0.17**	**1.00**	0.33	4	0.13	0.80	0.13	**5**	**0.17**	**1.00**	0.38	0	0.00	0.00	0.00
40	4	0.10	0.80	0.16	**5**	**0.13**	**1.00**	0.33	4	0.10	0.80	0.13	**5**	**0.13**	**1.00**	0.38	0	0.00	0.00	0.00
50	4	0.08	0.80	0.16	**5**	**0.10**	**1.00**	0.33	4	0.08	0.80	0.13	**5**	**0.10**	**1.00**	0.38	1	0.02	0.20	0.20
	DB				DB-MAX				RBDA				Our method				LOF			
Class '0'																				
5	1	0.20	0.20	0.33	2	0.40	0.40	0.75	2	0.40	0.40	0.43	3	0.60	0.60	**1.00**	0	0.00	0.00	0.00
10	1	0.10	0.20	0.33	3	0.30	0.60	0.67	3	0.30	0.60	0.50	3	0.30	0.60	**1.00**	0	0.00	0.00	0.00
15	1	0.07	0.20	0.33	4	0.27	0.80	0.33	4	0.27	0.80	0.40	**5**	**0.33**	**1.00**	0.60	0	0.00	0.00	0.00
20	1	0.05	0.20	0.33	4	0.20	0.80	0.33	4	0.20	0.80	0.40	**5**	**0.25**	**1.00**	0.35	0	0.00	0.00	0.00
30	2	0.07	0.40	0.10	**5**	**0.17**	**1.00**	0.33	4	0.13	0.80	0.40	**5**	**0.17**	**1.00**	0.35	1	0.03	0.20	0.04
40	4	0.10	0.80	0.10	**5**	**0.13**	**1.00**	0.33	4	0.10	0.80	0.40	**5**	**0.13**	**1.00**	0.35	1	0.03	0.20	0.04
50	4	0.08	0.80	0.10	**5**	**0.10**	**1.00**	0.33	4	0.08	0.80	0.40	**5**	**0.10**	**1.00**	0.35	1	0.02	0.20	0.04

Table 9
Multiple features, $\kappa=7$.

m	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp
	DB				LOF				RBDA				Our method				COF			
Class '9'																				
5	0	0.00	0.00	0.00	0	0.00	0.00	0.00	0	0.00	0.00	0.00	4	0.80	0.80	**1.00**	0	0.00	0.00	0.00
10	1	0.01	0.20	0.11	0	0.00	0.00	0.00	0	0.00	0.00	0.00	**5**	**0.50**	**1.00**	0.75	1	0.10	0.20	0.11
15	1	0.07	0.20	0.11	0	0.00	0.00	0.00	0	0.00	0.00	0.00	**5**	**0.33**	**1.00**	0.75	1	0.07	0.20	0.11
20	1	0.05	0.20	0.11	1	0.05	0.20	0.05	2	0.10	0.40	0.08	**5**	**0.25**	**1.00**	0.75	1	0.05	0.20	0.11
30	2	0.07	0.40	0.08	2	0.07	0.40	0.07	2	0.07	0.40	0.08	**5**	**0.17**	**1.00**	0.75	1	0.03	0.20	0.11
40	2	0.05	0.40	0.08	2	0.05	0.40	0.07	3	0.08	0.60	0.08	**5**	**0.13**	**1.00**	0.28	1	0.03	0.20	0.11
50	2	0.04	0.40	0.08	2	0.04	0.40	0.07	3	0.06	0.60	0.08	**5**	**0.10**	**1.00**	0.28	1	0.02	0.20	0.11
	DB				DB-MAX				RBDA				Our method				COF			
Class '6'																				
5	0	0.00	0.00	0.00	1	0.20	0.20	0.33	0	0.00	0.00	0.00	1	0.20	0.20	0.33	0	0.00	0.00	0.00
10	2	0.20	0.40	0.21	3	0.30	0.60	0.30	0	0.00	0.00	0.00	3	0.30	0.60	0.43	0	0.00	0.00	0.00
15	3	0.20	0.60	0.25	4	0.27	0.80	0.29	0	0.00	0.00	0.00	4	0.27	0.80	0.40	0	0.00	0.00	0.00
20	4	0.20	0.80	0.24	**5**	**0.25**	**1.00**	0.29	0	0.00	0.00	0.00	4	0.20	0.80	0.40	0	0.00	0.00	0.00
30	**5**	**0.17**	**1.00**	0.22	**5**	**0.17**	**1.00**	0.29	0	0.00	0.00	0.00	**5**	**0.17**	**1.00**	0.33	1	0.03	0.20	0.05
40	**5**	**0.13**	**1.00**	0.22	**5**	**0.13**	**1.00**	0.29	1	0.03	0.20	0.03	**5**	**0.13**	**1.00**	0.33	1	0.03	0.20	0.05
50	**5**	**0.10**	**1.00**	0.22	**5**	**0.10**	**1.00**	0.29	1	0.02	0.20	0.03	**5**	**0.10**	**1.00**	0.33	1	0.02	0.20	0.05
	DB				DB-MAX				RBDA				Our method				LOF			
Class '4'																				
5	2	0.40	0.40	0.60	1	0.20	0.20	0.50	1	0.20	0.20	0.33	2	0.40	0.40	0.60	1	0.20	0.20	
10	3	0.30	0.60	0.55	3	0.30	0.60	0.46	1	0.10	0.20	0.33	2	0.20	0.40	0.60	1	0.10	0.20	
15	3	0.20	0.60	0.55	4	0.27	0.60	0.38	2	0.13	0.40	0.20	3	0.20	0.60	0.33	1	0.07	0.20	
20	**5**	**0.25**	**1.00**	0.33	4	0.20	1.00	0.38	3	0.15	0.60	0.19	**5**	**0.25**	**1.00**	0.31	1	0.05	0.20	
30	**5**	**0.17**	**1.00**	0.33	**5**	**0.17**	**1.00**	0.28	4	0.10	0.80	0.19	**5**	**0.17**	**1.00**	0.31	2	0.07	0.40	
40	**5**	**0.13**	**1.00**	0.33	**5**	**0.10**	**1.00**	0.28	**5**	**0.13**	**1.00**	0.15	**5**	**0.13**	**1.00**	0.31	3	0.08	0.60	
50	**5**	**0.10**	**1.00**	0.33	**5**	**0.13**	**1.00**	0.28	**5**	**0.10**	**1.00**	0.15	**5**	**0.10**	**1.00**	0.31	4	0.07	0.80	

Table 10
Shuttle, $k=7,20,40,60$.

m	1.0				1.5				2.0				2.5				3.0			
	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp
k=7																				
20	11	0.55	0.58	0.42	9	0.45	0.47	0.35	5	0.25	0.26	0.23	5	0.25	0.26	0.23	5	0.25	0.26	0.23
30	14	0.47	0.74	0.45	12	0.40	0.63	0.36	9	0.30	0.47	0.28	9	0.30	0.47	0.28	9	0.30	0.47	0.28
60	17	0.28	0.89	0.32	16	0.27	0.84	0.33	13	0.22	0.68	0.24	13	0.22	0.68	0.24	13	0.22	0.68	0.24
90	18	0.20	0.95	0.28	17	0.19	0.89	0.31	15	0.17	0.79	0.22	15	0.17	0.79	0.22	15	0.17	0.79	0.22
120	**19**	**0.16**	**1.00**	0.23	18	0.15	0.95	0.25	16	0.13	0.84	0.21	16	0.13	0.84	0.21	16	0.13	0.84	0.21
140	**19**	**0.14**	**1.00**	0.23	18	0.1	0.95	0.25	17	0.12	0.89	0.20	17	0.12	0.89	0.20	17	0.12	0.89	0.20
200	**19**	**0.14**	**1.00**	0.14	**19**	**0.09**	**1.00**	0.12	18	0.10	0.95	0.10	18	0.10	0.95	0.10	18	0.10	0.95	0.10
k=20																				
20	11	0.55	0.58	0.42	11	0.55	0.58	0.40	8	0.40	0.42	0.31	8	0.40	0.42	0.31	8	0.40	0.42	0.31
30	14	0.47	0.74	0.45	14	0.47	0.74	0.42	13	0.43	0.68	0.39	13	0.43	0.68	0.39	13	0.43	0.68	0.39
60	18	0.30	0.95	0.32	18	0.30	0.95	0.37	18	0.30	0.95	0.33	18	0.30	0.95	0.33	18	0.30	0.95	0.33
90	**19**	**0.21**	**1.00**	0.28	**19**	**0.21**	**1.00**	0.33	**19**	**0.21**	**1.00**	0.27	**19**	**0.21**	**1.00**	0.27	**19**	**0.21**	**1.00**	0.27
120	**19**	**0.16**	**1.00**	0.23	**19**	**0.16**	**1.00**	0.27	**19**	**0.16**	**1.00**	0.26	**19**	**0.16**	**1.00**	0.26	**19**	**0.16**	**1.00**	0.26
140	**19**	**0.14**	**1.00**	0.23	**19**	**0.14**	**1.00**	0.27	**19**	**0.14**	**1.00**	0.23	**19**	**0.14**	**1.00**	0.23	**19**	**0.14**	**1.00**	0.23
200	**19**	**0.14**	**1.00**	0.14	**19**	**0.09**	**1.00**	0.12	**19**	**0.10**	**1.00**	0.10	**19**	**0.10**	**1.00**	0.10	**19**	**0.10**	**1.00**	0.10
k=40																				
20	11	0.55	0.58	0.42	11	0.55	0.58	0.40	11	0.55	0.58	0.51	11	0.55	0.58	0.51	11	0.55	0.58	0.51
30	14	0.47	0.74	0.45	14	0.47	0.74	0.42	14	0.47	0.74	0.44	14	0.47	0.74	0.44	14	0.47	0.74	0.44
60	18	0.30	0.95	0.32	18	0.30	0.95	0.37	18	0.30	0.95	0.33	18	0.30	0.95	0.33	18	0.30	0.95	0.33
90	**19**	**0.21**	**1.00**	0.28	**19**	**0.21**	**1.00**	0.33	**19**	**0.21**	**1.00**	0.27	**19**	**0.21**	**1.00**	0.27	**19**	**0.21**	**1.00**	0.27
120	**19**	**0.16**	**1.00**	0.23	**19**	**0.16**	**1.00**	0.27	**19**	**0.16**	**1.00**	0.26	**19**	**0.16**	**1.00**	0.26	**19**	**0.16**	**1.00**	0.26
140	**19**	**0.14**	**1.00**	0.14	**19**	**0.14**	**1.00**	0.27	**19**	**0.14**	**1.00**	0.23	**19**	**0.14**	**1.00**	0.23	**19**	**0.14**	**1.00**	0.23
200	**19**	**0.14**	**1.00**	0.14	**19**	**0.09**	**1.00**	0.12	**19**	**0.10**	**1.00**	0.10	**19**	**0.10**	**1.00**	0.10	**19**	**0.10**	**1.00**	0.10
k=60																				
20	11	0.55	0.58	0.42	11	0.55	0.58	0.40	11	0.55	0.58	0.51	11	0.55	0.58	0.51	11	0.55	0.58	0.51
30	14	0.47	0.74	0.45	14	0.47	0.74	0.42	14	0.47	0.74	0.44	14	0.47	0.74	0.44	14	0.47	0.74	0.44
60	18	0.30	0.95	0.32	18	0.30	0.95	0.37	18	0.30	0.95	0.33	18	0.30	0.95	0.33	18	0.30	0.95	0.33
90	**19**	**0.21**	**1.00**	0.28	**19**	**0.21**	**1.00**	0.33	**19**	**0.21**	**1.00**	0.27	**19**	**0.21**	**1.00**	0.27	**19**	**0.21**	**1.00**	0.27
120	**19**	**0.16**	**1.00**	0.23	**19**	**0.16**	**1.00**	0.27	**19**	**0.16**	**1.00**	0.26	**19**	**0.16**	**1.00**	0.26	**19**	**0.16**	**1.00**	0.26
140	**19**	**0.14**	**1.00**	0.14	**19**	**0.14**	**1.00**	0.27	**19**	**0.14**	**1.00**	0.23	**19**	**0.14**	**1.00**	0.23	**19**	**0.14**	**1.00**	0.23
200	**19**	**0.14**	**1.00**	0.14	**19**	**0.09**	**1.00**	0.12	**19**	**0.10**	**1.00**	0.10	**19**	**0.10**	**1.00**	0.10	**19**	**0.10**	**1.00**	0.10

Table 11
Ionosphere, $k=5,10,20,30$.

m	1.0				1.5				2.0				2.5				3.0			
	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp	n	p	r	rp
k=5																				
5	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**
10	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**
30	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	27	0.90	0.21	0.93	25	0.83	0.20	0.90	25	0.83	0.20	0.90
60	**60**	**1.00**	**0.48**	**1.00**	59	0.98	0.47	**1.00**	47	0.78	0.37	0.85	44	0.73	0.35	0.80	44	0.73	0.35	0.80
90	**90**	**1.00**	**0.71**	**1.00**	83	0.92	0.66	**1.00**	63	0.70	0.50	0.78	59	0.66	0.47	0.73	59	0.66	0.47	0.73
120	109	0.91	0.87	0.95	94	0.78	0.75	0.97	72	0.60	0.57	0.74	67	0.56	0.53	0.70	67	0.56	0.53	0.70
130	113	0.87	0.90	0.93	96	0.74	0.76	0.96	74	0.57	0.59	0.73	69	0.53	0.55	0.69	69	0.53	0.55	0.69
140	116	0.83	0.92	0.92	98	0.70	0.78	0.95	79	0.56	0.63	0.71	72	0.51	0.57	0.67	72	0.51	0.57	0.67
k=10																				
5	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**
10	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**
30	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**
60	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**
90	**90**	**1.00**	**0.71**	**1.00**	89	0.99	0.71	**1.00**	76	0.84	0.60	0.90	73	0.81	0.58	0.87	73	0.81	0.58	0.87
120	109	0.91	0.87	0.95	96	0.80	0.76	0.99	83	0.70	0.66	0.82	79	0.66	0.63	0.79	79	0.66	0.63	0.79
130	114	0.88	0.90	0.93	99	0.76	0.79	0.97	86	0.66	0.68	0.80	82	0.63	0.65	0.77	82	0.63	0.65	0.77
140	116	0.83	0.92	0.92	100	0.71	0.79	0.97	89	0.64	0.71	0.76	85	0.61	0.67	0.74	85	0.61	0.67	0.74
k=20																				
5	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**
10	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**
30	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**
60	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**
90	**90**	**1.00**	**0.71**	**1.00**	**90**	**1.00**	**0.71**	**1.00**	88	0.98	0.70	0.96	86	0.96	0.68	0.97	86	0.96	0.68	0.97
120	117	0.97	0.93	0.97	103	0.86	0.82	1.00	96	0.80	0.76	0.89	93	0.78	0.74	0.86	93	0.78	0.74	0.86
130	121	0.93	0.96	0.97	105	0.81	0.83	0.99	97	0.75	0.77	0.86	94	0.72	0.75	0.83	94	0.72	0.75	0.83
140	123	0.88	0.98	0.96	106	0.76	0.84	0.98	99	0.71	0.79	0.81	96	0.69	0.76	0.79	96	0.69	0.76	0.79
k=30																				
5	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**	**5**	**1.00**	**0.04**	**1.00**
10	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**	**10**	**1.00**	**0.08**	**1.00**
30	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**	**30**	**1.00**	**0.24**	**1.00**
60	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**	**60**	**1.00**	**0.48**	**1.00**
90	**90**	**1.00**	**0.71**	**1.00**	**90**	**1.00**	**0.71**	**1.00**	**90**	**1.00**	**0.73**	**1.00**	**90**	**1.00**	**0.73**	**1.00**	**90**	**1.00**	**0.73**	**1.00**
120	**120**	**1.00**	**0.71**	**1.00**	107	0.89	0.85	**1.00**	98	0.82	0.78	0.91	96	0.80	0.76	0.87	96	0.80	0.76	0.87
130	123	0.95	0.98	0.99	108	0.83	0.86	0.99	98	0.75	0.78	0.87	96	0.74	0.76	0.84	96	0.74	0.76	0.84
140	124	0.89	0.98	0.97	109	0.78	0.86	0.98	100	0.71	0.79	0.82	97	0.70	0.77	0.80	97	0.70	0.77	0.80

of Lymphography Data, which again manifests the intuition that using tree edge weights in mini MST or MST as indication of outlying degree is less sensitive to the different values of k and, therefore, more reliable.

4.2.3. Ionosphere data

The Johns Hopkins University Ionosphere dataset contains 351 instances with 34 attributes, of which 225 are labeled as good instances while the rest 126 are labeled as bad instances. All attributes are normalized in the range from 0 to 1 and there are no duplicate instances in the dataset. Instead of randomly selecting sample instances from the bad class to form the rare class, we use all 351 instances and compare the performances of all the methods for four different k's and eight different m's. The experimental results obtained by PBOD, our method and three best cases from the rest methods are presented in Table 7.

From the table, it can be clearly seen that, overall, our method performs the best, and DB method performs the next. RBDA performs a little better than MST+LOF. PBOD performs the worst. For this test dataset, all the methods are less sensitive to k than they are in the previous experiments.

For real world high-dimensional data sets, we pick two datasets (Multiple Features and Optical Digits) from UCI [33].

4.2.4. Optical digits

The Optical Digits dataset consists of the data extracted from handwritten numbers ('0'–'9') and contains 5620 instances with 64 attributes. To obtain the dataset, normalized bitmaps of handwritten digits are extracted from a preprinted form. 32×32 bitmaps are then divided into nonoverlapping blocks of 4×4 and the number of pixels is counted in each block, generating an input matrix of 8×8 where each element is an integer in the range 0…16. Since there are no small rare classes to be regarded as outliers, we randomly pick 5 data points from one of the classes to make it rare (i.e., as outliers). We show in Table 8 the best experimental results of our method for two classes (i.e., '2', '0') and four best cases of the rest methods for $k=7$ and seven values of m's.

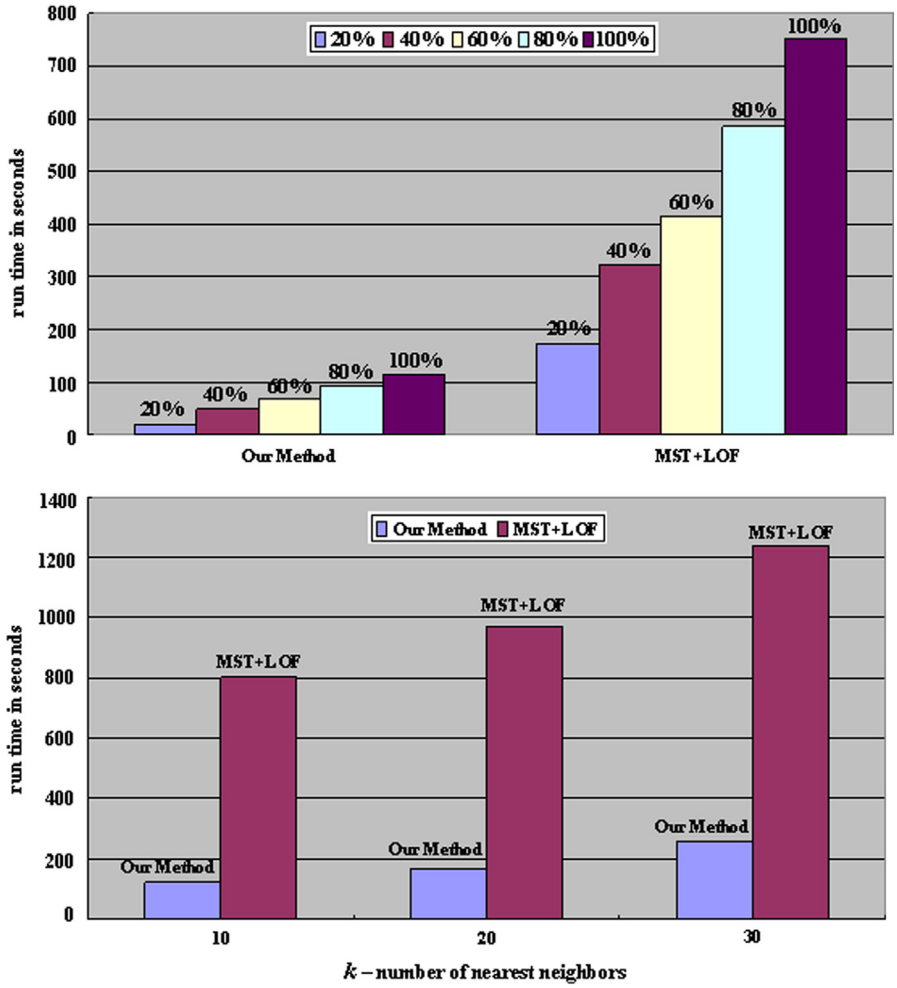

Fig. 13. Run time performance of our algorithm for IPUMS data.

From the table, it can be clearly seen that, overall, our method performs the best, and DB-MAX method performs the next. RBDA performs a little better than DB method.

4.2.5. Multiple features

The Multiple Features dataset consists of the data of handwritten numbers ('0'–'9') as well but extracted from a collection of Dutch utility maps and contains 2000 instances with 649 attributes including six feature sets (i.e., 76 Fourier coefficients, 216 profile correlations, 64 Karhunen–Love coefficients, 240 pixel averages in 23 windows, 47 Zernike moments and 6 morphological features). Since all the classes contain 200 data points each and there are no rare classes to be regarded as outliers, we randomly pick 5 data points from one of the classes to make it rare (i.e., as outliers). Table 9 summarizes the best findings from the experimental results of our method for three classes (i.e., '9', '6', '4') and four best cases of the rest methods for $k=7$ and seven values of m's.

From the table, it can be clearly seen that, overall, our method is the only one that performs best for class '9' for which others perform much worse. For the other two cases, our method and DB-MAX method perform similarly. DB performs well for classes '6' and '4'. RBDA performs well for class '4'. Therefore, overall, our method performs the best.

4.3. Performance of our algorithm with SOM-TH

In the above experiments, we keep the threshold, *SOM-TH*, being fixed to be 1.5. In this subsection, we conduct experiments to study the detecting effectiveness of the proposed method by varying the *SOM-TH* in the range [1.0, 1.5, 2.0, 2.5, 3.0] and the results for two datasets, Shuttle and Ionosphere, are shown in Tables 10 and 11. From the tables, we can see that our methods can obtain better results when *SOM-TH* is set to take the value of 1.0.

4.4. Run time performance of our algorithm on large High-dimensional datasets

In this subsection, we focus our study on the running time behavior of the proposed method with varying parameter, k, and under different workloads. Particularly, we want to show the impact of the size of the dataset on the performance of our algorithm (i.e., the scalability of our algorithm) and compare the running time of our algorithm with that of

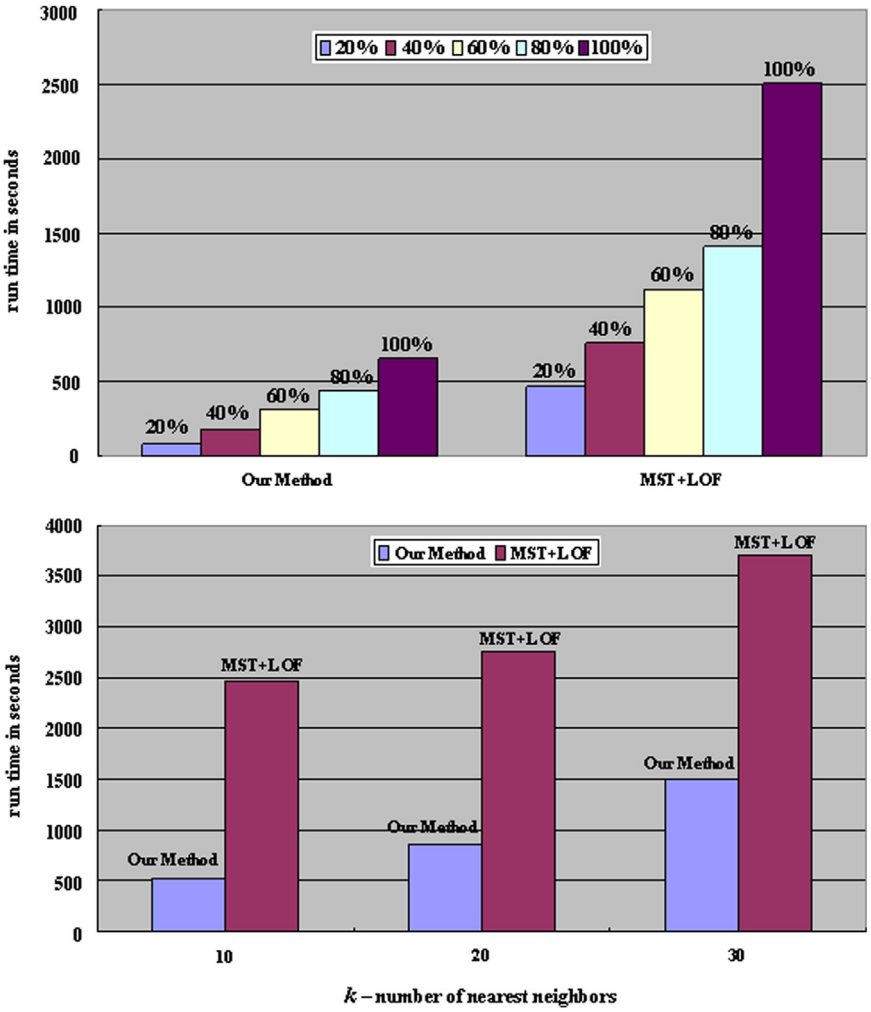

Fig. 14. Run time performance of our algorithm for covertype data.

MST+LOF algorithm on three different large high-dimensional datasets which can be downloaded from UCI [33]. These datasets are briefly summarized in Table 1. All the categorical features in them, if existing, have been cast to integer values.

A large problem when evaluating outlier detection methods for large high-dimensional data is that there are very few real world data sets where it is exactly known which objects are really behaving differently due to belonging to a different and rare mechanism. More often than not, it is not given beforehand what and how much objects are outliers. Although there exist multiple case studies on outlier detection, the question whether an object is an outlier or not is often depending on the point of view. Another problem is that the list of possible outliers is often incomplete, making it hard to evaluate whether the algorithm ranks all outliers in the database properly. Since it is shown from the previous subsections that our outlier detection method is comparable with (if not more effective than) other state-of-the-art outlier detection schemes, we focus on comparing our approach with MST+LOF algorithm on the running time issue. The sensitivity of DHCA to the input parameter K (the number of partition centers at each stage of the DHCA) has been studied in [21]. For large K, more distances to the partition centers need to be computed and the running time increases with K. In this set of experiments, the input K to DHCA is set to be 5.

Since outliers account for only a very small portion in a dataset, in this set of experiments, the running time results of our method and MST+LOF method to mine top 30 outliers for three real data sets under different workloads for $k = 10$ and with varying k's (i.e., the number of nearest neighbors from 10 to 30) are presented in Fig. 13 for IPUMS data, Fig. 14 for Covertype data, and Fig. 15 for UScensus data, respectively. For this comparison, the threshold of our global outlier detection is $SOM_{MST}=3$, and that of our local outlier detection is set to be 2.

There are two bar charts for each dataset in each figure. The upper part of each figure represents the running time performance of our algorithm and MST+LOF method for five different data sizes (in five different colors) obtained by varying the data sizes between 20% and 100% of the whole of each dataset. The results of our method are

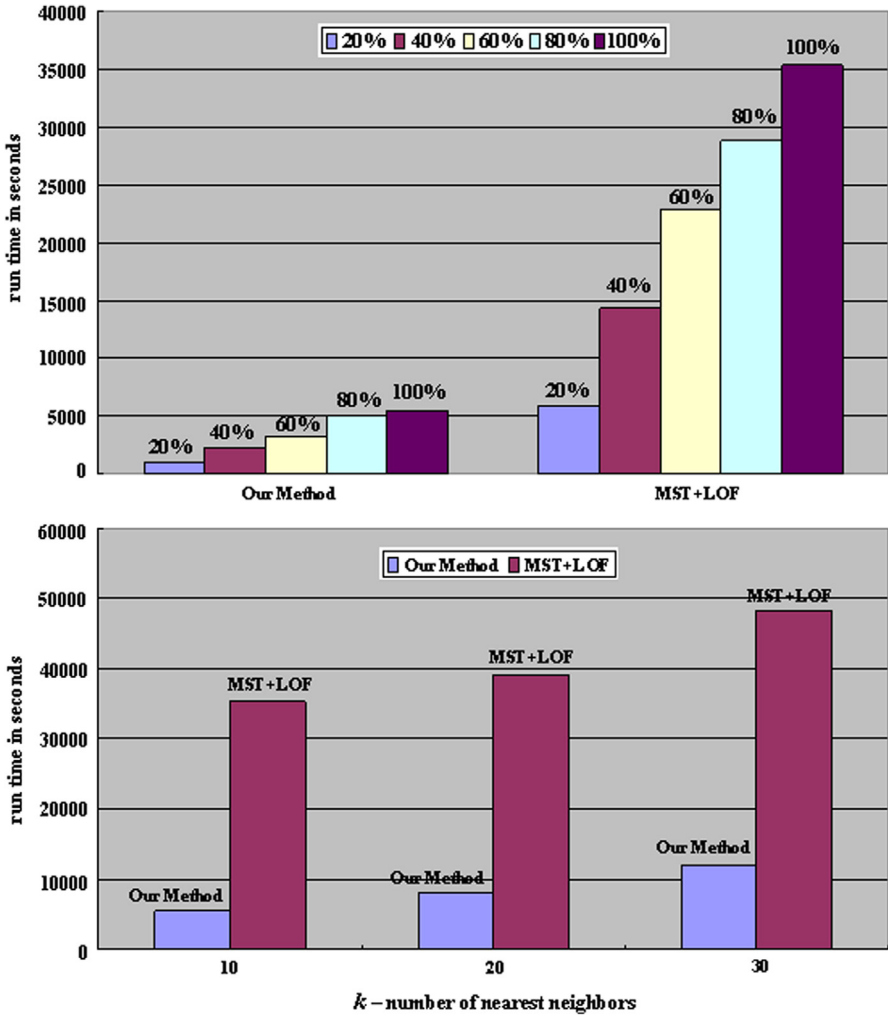

Fig. 15. Run time performance of our algorithm for UScensus data.

shown on the left while those of MST+LOF algorithm are shown on the right. The lower part of each figure represents the running time performance of our algorithm (in light blue) and MST+LOF method (in purple) for three different k's (i.e., $k=10, 20, 30$). Clearly, the execution time of our algorithm increases with data sizes and k's in a sublinear form and is about five times faster than those of MST+LOF. Since these datasets have very different characteristics and are of different dimensionalities, overall, it can be seen that our algorithm outperforms MST+LOF algorithm.

5. Conclusion

In this paper, we have proposed an efficient MST-inspired *kNN*-based outlier detection method that can detect both global and local outliers. In addition to being compatible with the traditional distance-based outlier detection methods, our approach also performs better in identifying local outliers that deviate from the main patterns in a given dataset. Candidate outliers are ranked based on the MST-inspired outlier scores that are assigned to each data point. To demonstrate the utility of our proposed outlier factors, we have performed a detailed comparison of its performance with a number of state-of-the-art outlier detection methods. Through a thorough evaluation, our method manifests its ability to rank the best candidates for being an outlier with high *precision* and *recall*. In addition to the basic approach, we propose to use DHCA as an acceleration suitable for high-dimensional large data sets. It has been shown that a small number of consecutive running of DHCA can facilitate the detection of top outliers, which are only a small fraction of the whole dataset. Both theoretical justification and empirical validation show the effectiveness of the proposed method.

Our study also shows that we should not view one method as being superior to others in all aspects but use it as a compliment to, instead of as a replacement of, others in applications with different requirements. This is intuitive because, in reality, it is usually difficult to detect all the outliers that fit user's intuitions. Thus, it is probably meaningful to incorporate our proposed outlier detection method as a component into current outlier detection framework.

Acknowledgment

The authors would like to thank the Chinese National Science Foundation for its valuable support of this work under award 61473220 and all the anonymous reviewers and editors for their valuable comments.

References

[1] D.M. Hawkins, Identification of outliers, Monographs on Applied Probability and Statistics, Chapman and Hall, London, 1980.
[2] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. in: Daniel Barbará and Sushil Jajodia (Eds.), Data Mining for Security Applications, 2002, pp.77-101.
[3] T. Lane, C.E. Brodley, Temporal sequence learning and data reduction for anomaly detection, ACM Trans. Inf. Syst. Secur. 2 (3) (1999) 295–331.
[4] R.J. Bolton, J.H. David, Unsupervised profiling methods for fraud detection, Stat. Sci. 17 (3) (2002) 235–255.
[5] W. Wong, A. Moore, G. Cooper, and M. Wagner, Rule-based anomaly pattern detection for detecting disease outbreaks, in: Proceedings of the 18th National Conference on Artificial Intelligence, 2002, pp. 217–223.
[6] B. Sheng, Q. Li, W. Mao, W. Jin, Outlier detection in sensor networks, in: Proceedings of ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2007, pp. 219–228.
[7] V.J. Hodge, J. Austin, A survey of outlier detection methodologies, Artif. Intell. Rev. 22 (2004) 85–126.
[8] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: a survey, ACM Comput. Surv. 41 (July) (2009). (Article 15).
[9] X. Wang, X.L. Wang, D.M. Wilkes, A spanning tree-inspired clustering based outlier detection technique, in: Proceedings of the 12th Industry Conference on Data Mining, Berlin, Germany, 2012, pp. 209–223.
[10] E.M. Knorr, R.T. Ng, Algorithms for mining distance-based outliers in large datasets, in: Proceedings of the 24th VLDB Conference, New York, USA,1998, pp. 392–403.
[11] S. Ramaswamy, R. Rastogi, K. Shim, Efficient algorithms for mining outliers from large data sets, in: Proceedings of the ACM SIGMOD Conference, 2000, pp. 427–438.
[12] F. Angiulli, C. Pizzuti, Fast outlier detection in high dimensional spaces, Proceedings of the Sixth European Conference on the Principles of Data Mining and Knowledge Discovery (2002) 15–26.
[13] M.M. Breuning, H.P. Kriegel, R.T. Ng, J. Sander, LOF: identifying density-based local outliers, in: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, 2000, pp. 93–104.
[14] J. Tang, Z. Chen, A.W.C. Fu, D.W. Cheung, Enhancing effectiveness of outlier detections for low density patterns, in: Proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), vol. 2336, Taipei, Taiwan, 2002, pp. 535–548.
[15] P.B. Gibbons, S. Papadimitriou, H. Kitagawa, C. Faloutsos, LOCI: fast outlier detection using the local correlation integral, in: Proceedings of the IEEE 19th International Conference on Data Engineering, Bangalore, India, 2003, pp. 315–328.
[16] P. Sun, S. Chawla, On local spatial outliers, in: Proceedings of the 4th International Conference on Data Mining (ICDM), Brighton, UK, 2004, pp. 209–216.
[17] W. Jin, A.K.H. Tung, J. Han, W. Wang, Ranking outliers using symmetric neighborhood relationship, in: Proceedings of the 10th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), vol. 3918, Singapore, 2006, pp. 577–579.
[18] K. Zhang, M. Hutter, H. Jin., A new local distance-based outlier detection approach for scattered real-world data, Adv. Knowl. Discov. Data Min. 5476 (2009) 813–822.
[19] H. Huang, K. Mehrotra, C.K. Mohan, Rank-based outlier detection, J. Stat. Comput. Simul. 83 (3) (2013) 1–14.
[20] C.T. Zahn, Graph-theoretical methods for detecting and describing gestalt clusters, IEEE Trans. Comput. C-20 (1971) 68–86.
[21] X. Wang, X.L. Wang, D.M. Wilkes., A divide-and-conquer approach for minimum spanning tree-based clustering, IEEE TKDE 21 (7) (2009) 945–958.
[22] C. Zhong, D. Miao, R. Wang., A graph-theoretical clustering method based on two rounds of minimum spanning trees, Pattern Recognit. 43 (3) (2010) 752–766.
[23] T. Luo, C. Zhong., A neighborhood density estimation clustering algorithm based on minimum spanning tree, LNAI 6401 (2010) 557–565.
[24] T. Luo, C. Zhong, H. Li, X. Sun, A multi-prototype clustering algorithm based on minimum spanning tree, in: Proceedings of 2010 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010), 2010, pp. 1602–1607.
[25] C. Zhong, D. Miao, P. Franti, Minimum spanning tree based split-and-merge: a hierarchical clustering method, Inf. Sci. 181 (2011) 3397–3410.
[26] F.J. Rohlf, Generalization of the gap test for the detection of multivariate outliers, Biometrics 31 (1975) 93–101.
[27] M.F. Jiang, S.S. Tseng, C.M. Su., Two-phase clustering process for outliers detection, Pattern Recognit. Lett. 22 (2001) 691–700.
[28] J. Lin, D. Ye, C. Chen, M. Gao, Minimum spanning tree based spatial outlier mining and its applications, Lecture Notes Computer Science, vol. 5009, Springer-Verlag, Berlin/Heidelberg, 2008, 508–515.
[29] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is nearest neighbors meaningful?, in: Proceedings of ICDT, 1999, pp. 217–235.
[30] C. Aggarwal, P. Yu, An effective and efficient algorithm for high-dimensional outlier detection, Int. J Very Large Data Bases 14 (2) (2005) 211–221.
[31] S. Parthasarathy, C.C. Aggarwal, On the use of conceptual reconstruction for mining massively incomplete data sets, IEEE TKDE 15 (6) (2003) 1512–1531.
[32] X. Wang, X.L. Wang, D.M. Wilkes, Modifying iDistance for a fast CHAMELEON with application to patch based image segmentation, in: Proceedings of the 9th IASTED International Conference on Signal Processing, Pattern Recognition and Applications (SPPRA 2012), Crete, Greece, 2012, pp. 107–114.
[33] UCI: The UCI KDD Archive, University of California, Irvine, CA. ⟨http://kdd.ics.uci.edu/⟩.
[34] C. Aggarwal, P. Yu, Outlier detection for high-dimensional data, in: Proceedings of SIGMOD'01, Santa Barbara, CA, USA, 2001, pp. 37–46.
[35] X. Meng, Z. Chen, On user-oriented measurements of effectiveness of web information retrieval systems, in: H.R. Arabnia, O. Droegehorn (Eds.), Proceedings of the International Conference on Internet Computing, vol. 1, Las Vegas, Nevada, USA, 2004, pp. 527–533.