World Wide Web (2014) 17:539–568
DOI 10.1007/s11280-013-0218-4

Outlier detection from large distributed databases

Ji Zhang · Xiaohui Tao · Hua Wang

Received: 28 October 2012 / Revised: 23 March 2013 /
Accepted: 17 April 2013 / Published online: 1 May 2013

Abstract In this paper, we present an innovative system, coined as DISTROD (a.k.a **DISTR**ibuted **O**utlier **D**etector), for detecting outliers, namely abnormal instances or observations, from multiple large distributed databases. DISTROD is able to effectively detect the so-called global outliers from distributed databases that are consistent with those produced by the centralized detection paradigm. DISTROD is equipped with a number of optimization/boosting strategies which empower it to significantly enhance its speed performance and reduce its communication overhead. Experimental evaluation demonstrates the good performance of DISTROD in terms of speed and communication overhead.

Keywords Data mining · Distributed database · Outlier detection

1 Introduction

Outlier detection is an important research problem in data mining with an aim to find a specific number of objects that are considerably dissimilar, exceptional and inconsistent with respect to the majority records in the input databases. Outlier detection technologies have found a wide range of important applications in engineering, business, security and medicine, to name a few. For example, outlier detection are very useful for detecting the defects of engineering products, fraud credit card transactions and suspicious network traffic data, etc. This research problem has been intensively

J. Zhang (✉) · X. Tao · H. Wang
Department of Mathematics and Computing, University of Southern Queensland,
Toowoomba, QLD, Australia
e-mail: Ji.Zhang@usq.edu.au

X. Tao
e-mail: xtao@usq.edu.au

H. Wang
e-mail: wang@usq.edu.au

studied in the domain of data mining in the past 10 years. Nevertheless, most of the existing research work is focused on the centralized outlier detection problem where all the data are stored and processed in a central manner. The increasing number of applications that need to collect and store a large amount of data in multiple distributed (proprietary) databases push for the development of new outlier detection techniques that can work effectively and efficiently under distributed computing environments.

For distributed outlier detection, it is normally desirable to detect deviating observations in the whole database rather than in any individual distributed database. As an example, it may be necessary to find irregular or suspicious credit card spending patterns with relation to the whole city (where credit card transactions are scattered across a number of distributed community data centers), rather than to a single community. This kind of outliers are called *global outliers*, as opposed to the local outliers detected from any single participating database.

Given multiple distributed databases, the most straightforward solution to global outlier detection is to integrate them into a single centralized one and apply the existing centralized methods to detect outliers. However, this strategy will cause the following two major problems. First, the massive amount of data at different sites renders this solution prohibitively expensive and therefore totally infeasible for most of today's applications. In addition, data integration may result in violation of data privacy and leaking of sensitive information. In view of these, a general direction to approach distributed outlier detection, including other distributed data mining problems, is to devise communication efficient algorithms and mechanisms to accomplish the objectives without expensive data integration. Therefore, reducing communication overhead becomes a major focus of design consideration for developing these techniques. Great efforts have been taken in our work to reduce communication overhead by incorporating a number of optimization strategies into the technique we propose.

There are two major types of data partitioning in various distributed applications, *horizontal partitioning* and *vertical partitioning*, which fundamentally impacts the corresponding methods used to approach the problem of outlier detection. Horizontal partitioning is the most common type of data partitioning in real-life applications and is also the research focus of this paper. In horizontal partitioning, distributed sites host and process different subsets of data which have the same schema (i.e., same set of attributes). The union of the databases of all the distributed sites will form the whole database, which is called the global database. Mathematically speaking, if there are t participating distributed sites, then we have $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots \cup \mathcal{D}_t$, where $\mathcal{D}$ is the global database and $\mathcal{D}_i$ is the local database residing at site s_i. Normally, it is assumed that the data at different sites are independent and identically distributed (`iid`). In the case of vertical partitioning, all the sites have the same set of data observations but different subsets of attributes. We can obtain the complete set of attributes by concatenating the attribute sets from all the sites. Let A_i denote the attribute set for site s_i, then the complete attribute set $\mathcal{A}$ can be expressed as $\mathcal{A} = \mathcal{A}_1 : \mathcal{A}_2 : \ldots : \mathcal{A}_t$.

There are two major reasons why vertical partitioning are not considered in this paper. The primary reason is that vertical partitioning is not as common as horizontal partitioning in real-life distributed database applications. Second, vertical partitioning is generally more difficult to be dealt with than horizontal partitioning in terms

of outlier detection. Our technique are not applicable to the vertical partitioning because the data summary our technique uses needs to be additive per se, which is not possible in the case of vertical partitioning. Unless otherwise stated, we hereafter refer to horizontal data partitioning by default in this paper for distributed outlier detection.

The problem of detecting global outliers in a distributed environment can be formally formulated as follows. Let D denote the global database that is horizontally partitioned into t parts (i.e., $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \ldots \cup \mathcal{D}_t$) each residing at a distributed processing site, denoted respectively by $s_1, s_2, \ldots, s_t$. A data point p in $\mathcal{D}$ is detected and returned as one of the final top n global outliers (n) is a user specified parameter), if there does not exist more than $n-1$ other data points p_i in D that satisfy that $f(p) < f(p_i)$ $(0 < i < n-1)$. Here, $f(\cdot)$ denotes the outlier-ness score function that is used to quantitatively measure the strength/probability of p for being an outlier and may have different application-dependent definitions. When dealing with this problem, data integration is expensive and is thus disallowed.

To effectively and efficiently detect global outliers from large distributed databases, we present a new system, called DISTROD (short for **DISTR**ibuted **O**utlier **D**etector) in this paper. The innovative features and contributions of DISTROD are summarized as follows:

- DISTROD is able to produce global outlier detection results that are consistent with those produced in a centralized environment without expensive data integration;
- The communication between the mediator and the distributed sites only involves compact summary-level information, leading to a low data transfer overhead;
- DISTROD is efficient and scalable and therefore ideal to deal with applications with massive data volume;
- A number of optimization techniques have been incorporated into DISTROD to enable it to achieve even better communication and speed performance;
- Experimental results demonstrate that DISTROD is effective in detecting global outliers and outperforms existing methods in terms of communication overhead.

Roadmap The remainder of this paper is organized as follows. Section 2 presents discussions on the related research work in literature. In Section 3, we present some background information about different distributed computing architectures that are suitable for distributed outlier detection. An overview of DISTROD, the technique we propose for detecting outliers from distributed databases, is presented in Section 4, where we elaborate on its system architecture and algorithm. A number of optimization strategies are proposed in Section 5 to further reduce its communication overhead and improve DISTROD's speed performance. Section 6 presents the enhanced algorithm of DISTROD based on the optimization strategies we have developed. Detailed experimental evaluation results are reported in Section 7. The last section concludes the whole paper and points out future research directions.

2 Related work

Numerous research work on outlier detection, primarily dealing with a centralized database, has been proposed which can broadly categorized as the distribution-based

methods, the distance-based methods, the density-based methods and the clustering-based methods, etc.

Distribution-based methods [3, 8], also known as statistical methods, rely on the statistical approaches that assume a distribution or probability model to fit the given dataset. Outliers are those data that do not agree with or conform to the underlying model of the data. Distance-based methods [11, 12, 17] use distance-based metrics to quantify the proximity between the data point and its neighborhood. Most existing metrics are defined based upon the concepts of local neighborhood or k nearest neighbors (kNN). Those data points that are far from their respective neighbors are considered as outliers. Density-based methods [2, 10, 20, 22, 24] use more complex mechanisms to model the outlier-ness of data points than distance-based methods. They usually involve investigating not only the local density of the data being studied but also the local densities of its nearest neighbors. Thus, the outlier-ness metric of a data point is relative in the sense that it is normally a ratio of density of this data against the averaged densities around its nearest neighbors. The clustering-based methods regard outliers as a by-product of clustering algorithms [1, 7, 9, 15, 23, 25] themselves and define outliers as data that do not lie in or located far apart from any clusters. Thus, the clustering techniques implicitly define outliers as the background noise of clusters which are filtered out when clustering is performed.

The above-mentioned outlier detection methods are developed for centralized databases. Unfortunately, they are not able to be extended naturally to handle distributed databases efficiently and effectively. Recently, there have been some, but still very limited, research work aiming to develop new outlier detection methods exclusively for dealing with distributed databases, as discussed as follows.

Zhou et al. proposed an outlier detection technique for distributed databases using a distance-based approach [21]. The distance-based outlier definition is borrowed from Knorr and Ng [11, 12]. In each distributed site, a data point in a given distributed site is detected as a local outlier if its neighborhood in the radium of d contains less than $N_i(1-\rho)$ data points, where d is the scope of neighborhood, N_i is the number of data points at site s_i and ρ is a real-valued number that satisfies $0 < \rho < 1$. The final global outliers are defined as data points that have been detected as local outliers in *every* distributed site. The major disadvantage of this approach is that it is only able to detect a fraction of global outliers existing and, quite likely, a significant portion of global outliers will be missed out. This is because the definition of the global outlier is problematic. For example, if a data is not detected as an outlier in every distributed site, then, based on the definition of global outlier in [21], this data is not considered as a global outlier. However, the real situation is that a true global outlier is not necessarily an outlier in each of the distributed sites. In addition, the computation of distance-based outlier metric used in this work is inefficient when dealing with large databases as it will involve pairwise distance calculation for the data if no explicit indexing is built in advance.

A kernel density estimation technique is proposed to detect distributed outliers mainly from data streams [19]. The major problem of the kernel density estimation lies in that the kernel density function will become rather complex when the dimensionality of data increases. It is also computationally expensive to calculate kernel density functions as many of them are unbounded and costly integral is usually involved. In addition, selecting the kernel width, an important step when using the kernel function estimate, is not trivial. Finally, in order to detect outliers, a number

of so-called CF-tree based features need to be stored, maintained and communicated between the mediator and distributed sites. This leads to a higher communication expense than simply using density information to capture data distribution. A few other research work in the area of sensor networks uses kernel density estimation as well to detect outliers from inherently distributed nodes within the networks [4, 5, 18] and they suffer from the same limitations discussed above.

Otey et al. proposed a technique for distributed outlier detection from datasets with a mix of categorical and continuing attributes [16]. Frequent itemset mining approach is used to quantify the distance of the categorical attributes among data. The problem with this approach is that for distributed outlier detection, the support (i.e., frequency count) information needs to be transmitted to the mediator to generate the global support of all the itemsets. Due to the large number of itemsets especially for high-dimensional data, this technique usually incurs a very high communication overhead. Koufakou et. al proposed an improved distributed outlier detection method for datasets with mixed attributes [13]. It features a higher detection accuracy than the earlier method proposed in [16], however it still leverages the same idea of frequent itemsets mining to quantify the outlier-ness of data points, therefore it suffers from the same problem of high communication overhead as [16] when transmitting the support information of itemsets.

Besides the above-mentioned research work on distributed outlier detection methods that work for horizontally partitioned databases, there have been some research work on distributed outlier detection from vertically partitioned databases, represented by the work presented in [6]. Since outlier detection from vertically partitioned databases is out of the scope of this paper, we thus omit a detailed discussion on the papers and interested readers can refer to the relevant papers for more details.

3 Background

Before introducing our technique for distributed outlier detection, we first present the background information concerning various message passing mechanisms that are available for the applications of distributed outlier detection. All the architectures utilize Messaging Passing Interface (MPI), a widely used technology to accomplish efficient information exchange among a number of communicating sites. It is especially suitable for mathematical functions such as summation or accumulation of a particular number [14]. Three major kinds of MPI-based architectures for performing number summation in the distributed computing environment will be discussed in the sequel. Without losing generality, let's assume there are four participating sites denoted as $s_1, s_2, \ldots, s_4$, respectively, that have their own number values that are to be aggregated and shared among themselves.

The first kind of architecture, as illustrated in Figure 1a, is the naive one which involves broadcasting the value of each site to all the other peers, whereby each site obtains a copy of the value from each of other sites and the value aggregation can be performed in each site independently. The major downside of this architecture is obviously the massive communication overhead incurred which is in the quadratic order with regard to the number of sites.

More communication economic architectures are available which avoid the expensive broadcasting between each pair of sites. The first kind of such architectures

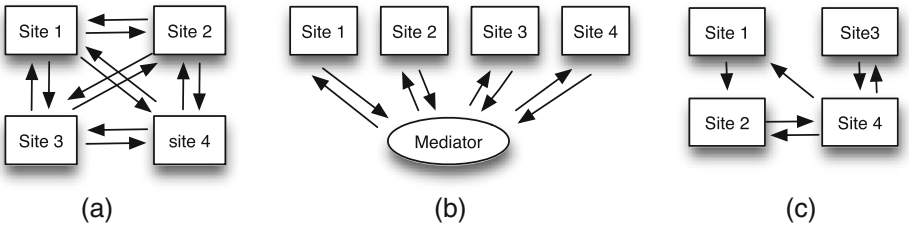

Figure 1 Architectures for value aggregation

uses a single site as the *mediator*, or the leading node, to perform all the operations including collection, aggregation and broadcasting, as shown in Figure 1b. In the given example, the mediator can be chosen from one of the four sites or preferably an additional dedicated one exclusively for performing all the tasks involved in aggregation. If an additional node is used as the mediator, then the value in each of the four sites will be passed to the mediator where the values are summed. The final summed value is sent from the mediator back to all the sites. This kind of architecture is simple and flat as it only contains two levels, the level for the mediator and the level for the participating sites. The second kind of architecture, as shown in Figure 1c, is more complex and multi-step value summation is performed. The value of a site will be passed to another site for summation which will be passed again to another site for further summation, and so on. The final site that calculates the total sum of the value is called the *converging node* in the architecture. In the example, the value of s_1 is passed to s_2 where aggregation can be done for s_1 and s_2. Similarly, the value of s_3 is shipped to s_4 for aggregation. The aggregated value of s_1 and s_2 will be passed from s_2 to s_4 where the aggregation for all the four sites can be done. The final aggregation value can be broadcast from s_4 to all the other sites. s_4 in this case is the *converging node*. When comparing these two communication efficient architectures, it is evident that both of them feature the same order of communication overhead which is in the linear, instead of quadratic, order of the number of sites. Nevertheless, the flat architecture is more structurally simple. Therefore, we adopt the mediator-based architecture in our work for the purpose of outlier detection.

4 DISTROD: our proposed technique

In this section, we will present our proposed technique, called DISTROD, for detecting global outliers from large distributed databases. The system architecture of DISTROD, the outlier measurement that is used to measure the strength of outlierness of data and finally the algorithm of DISTROD will be discussed in details (Figure 2). The notations that are used in this paper are presented in Table 1.

4.1 System architecture

DISTROD adopts the flat architecture for node communication, as discussed in Section 3. The system architecture diagram of DISTROD is given in Figure 2. There are three major parties in DISTROD system: *the end users*, *the mediator* and *the distributed sites*. The end users are the people who request outliers (typically the top n) for

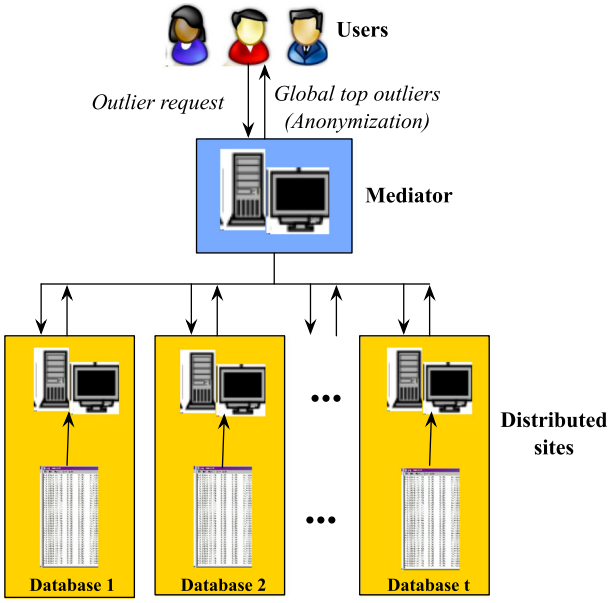

Figure 2 System architecture of DISTROD

certain purposes or applications. Requests from users are then passed to the mediator. The mediator is preferably a machine that can communicate with each distributor site but do not need to host or access the raw data. Upon receiving a request, the mediator starts to execute the outlier detection process. The major role of the mediator is to generate the global data summary and broadcast it to all the distributed sites for detecting outliers from each site. Each distributed site, having necessary computational capability and storage facility, collects and processes the data it receives. The final global top n outliers are generated by the mediator and returned to end users through it. In DISTROD, the mediator can communicate with each distributed site but it is stipulated that no communication is allowed amongst distributed sites themselves.

4.2 Outlierness measurement

In order to quantitatively measure the outlier-ness of data efficiently, a grid-based space partitioning performed at each distributed site. Outlier detection is performed

Table 1 Notations used in the paper

Notation symbols	Meaning
$\mathcal{D}$	The global database
$\mathcal{D}_i$	The local database collected and processed at site s_i, $1 < i < t$
n	The number of top global outliers to be returned
t	The number of participating distributed sites, excluding the mediator
δ	The dimension of the data
l	The number of intervals that each dimension is partitioned into
k	The number of neighbors considered when calculating k-ODF for data

based on a grid-like data structure with the aim to enhance the detection efficiency. The basic idea of grid-based space partitioning is to quantize each dimension into a finite number of intervals with equal length or, more complexly, with equal depth. To render the data partitioning consistent across all the distributed sites, an equal-width data partitioning is chosen for DISTROD. This effectively forms a grid structure on which all the operations for detection are performed. The main advantage of grid-based approaches is their fast processing time which is typically only dependent on the number of cells in the quantized space, rather than the number of data objects. In addition, the grid structure facilitates the efficient generation of both local and global data summaries that are critical for distributed outlier detection. After discussing space partitioning, we are now ready to introduce the outlier-ness measurement used in our work. It is defined as follows.

Definition 4.1 k-Outlying Degree Factor (k-ODF) k-ODF measures the strength of outlierness of data in the database. k-ODF of a data object p is defined as the averaged distance between p and the centroids of its k nearest global dense cells:

$$kODF(p) = \frac{\sum_{i=1}^{k} Dist(p, centroid(c_i))}{k}$$

where k is a user-specified parameter that typically takes a small value and c_i is one of the k nearest global dense cells of p. $Dist(\cdot)$ is the function calculating the distance between two data points.

The distance function $Dist(\cdot)$ can handle both numerical and categorical data points. The Euclidean distance is used between two numeric data points p_1 and p_2 as $Dist(p_1, p_2) = \sqrt{\sum_{i=1}^{\delta} \left(\frac{p_{1i}-p_{2i}}{\mathrm{Max}_i-\mathrm{Min}_i}\right)^2}$, where Max_i and Min_i denote the maximum and minimum data value of the ith data dimension and δ is the total number of attributes. The simple match method is used to measure the distance between two categorical data p_1 and p_2 as $Dist(p_1, p_2) = \frac{|p_{1i}-p_{2i}|}{\delta}$, where $|p_{1i} - p_{2i}|$ is 1 if p_{1i} equals to p_{2i} and is 0 otherwise. For data points with a mix of numeric and categorical attributes, the above two methods can be used together through weighted sum method.

k-ODF measures outlier-ness for data by calculating its distance to its nearby dense regions, which is intuitive and consistent with human perception. The other major existing outlier-ness measurements, such as [11, 12, 17], are not applicable in distributed computing environment due to the difficulty in obtaining the nearest neighboring data points for each data when data integration is prohibited. In comparison, our definition only requires identifying the dense regions for the global database, which can be obtained much easily in distributed environment without expensive data integration. The global dense cells are defined as follows.

Definition 4.2 (Global dense cells) A global dense cell is a cell whose density is above or equal to the global average density level of all of the populate cells in the grid structure at the mediator, i.e.,

$$c \text{ is a global dense cell iff } density(c) > global\_ave\_density$$

where

$$global\_ave\_density = \frac{N}{N_{\text{populated}}}$$

Here, N is the number of data points in the global database and $N_{\text{populated}}$ is the number of the global populated cells, the cells that contain at least one data point.

The above definition of global dense cells is applicable to the possible extreme case that there is only one populated cell in the grid structure. In this case, this populated cell is considered dense because its density is equal to the average level which is the density of itself. Figure 3 presents an example of calculating k-ODF for data p when $k = 3$. The symbols of crosses in the figure represent the centroids of the global dense cells whose density is higher than the average density level.

The calculation of k-ODF needs the information about the centroids of the global dense cells in the grid structure, which are called the *representative data* of the global database. They are considered the "global data summary". Since each distributed site only maintains its own local data summary, therefore it is necessary to merge the local data summary in order to produce the global data summary.

When defining global dense cells, we only take into account the populated cells when calculating the average density level. Another possible way to calculate the average density level is to consider all the cells (both populated or empty) in the grid structure. Yet, considering all the cells may significantly lower the average density level in many cases due to the possibly high number of empty cells. As a result, many sparse cells will be mistakenly identified as dense ones. One may argue that it is still possible to use such definition to find the dense cells by slightly modifying the definition itself as follows: A cell is treated as dense if its density is q times higher than the average density, where $q > 1$. We admit that this method can in theory offset the adverse skewing effect posed by the empty cells, but the specification of a proper value for q is very difficult due to a lack of both priori knowledge about the datasets and the visualization facility for datasets whose dimensionality higher than 3. It is also possible to define a cell as dense if its density is higher than a fraction of α of the

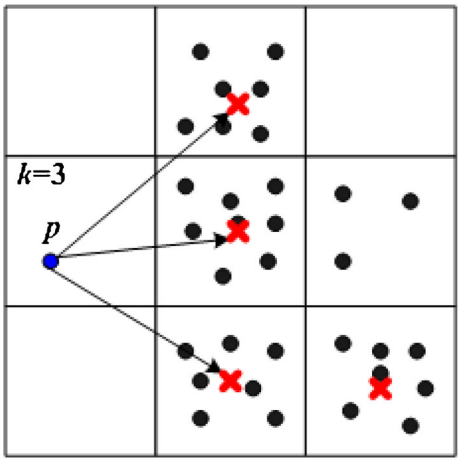

Figure 3 Calculate k-ODF for data p ($k = 3$)

total number of data in the database, where $0 \leq \alpha \leq 1$. Again, this definition of dense cells suffers from a similar drawback that, in practice, it is rather difficult to specify an appropriate value for α and the resulting dense cells are very sensitive to the value of α. Compared to the aforementioned two alternatives, our definition of dense cells is intuitive and easy to operate without the use of any sensitive parameters.

4.3 The baseline algorithm of DISTROD

We first present the baseline algorithm of DISTROD, the version of algorithm that is capable of detecting global outliers but with minimum performance tuning and optimization. Using the baseline algorithm, DISTROD detects global outliers from distributed databases in the following four steps. We highlight in each step of the algorithm the major task that needs to be conducted and the venues (either at different distributed sites or the mediator) where such task will be carried out. The baseline algorithm of DISTROD will undergo some significant optimizations/enhancements which will be presented in the subsequent sections.

Step 1: Assigning data into grid structure (distributed sites). The same grid is superimposed for all the distributed sites. Data in the local database $\mathcal{D}_i$ at each site s_i are read in sequentially and assigned into a cell in the grid. Instead of physically creating the grid structure whose number of cells will explode for high-dimensional data, we only maintain the list of populated cells. The incoming data is mapped into one appropriate cell in this list. If the data falls into a cell that is not yet in the list, then a new cell will be added into the list. The density of a cell will be incremented when a new data is assigned to it. For efficiently retrieving the density information of any particular cell, all the cells in the grid are uniquely numbered. A mapping function is devised to map each data to the identification number of the cell it belongs to. The major purpose for assigning data into the grid structure is to obtain the density information of cells in the grid which are considered as the local data summary, the statistical description of the data distribution at each of the distributed sites.

Step 2: Generating the global data summary (mediator). The density of local populated cells needs to aggregated in the mediator to generate the global dense cells. To do this, the identification number and density of local populated cells are first transferred to the mediator from each distributed site. Upon receiving the density information, the mediator carries out aggregation to generate the density of global populated cells and identify from them the global dense cells in the grid. The aggregation can be performed as follows to produce the global density of each populated cell at the mediator:

$$density(c) = \sum_{i=1}^{t} density(c)^{(i)}$$

The centroids of the global dense cells are extracted as the representative data, simply represented by the identification number of the cell they belong to, are then broadcast from the mediator to all the distributed sites. The densities of the representative data do not need to be transferred to the distributed sides.

Step 3: Generating Local top n outliers (distributed sites). When each distributed site receives the representative data from the mediator, a full scan of the local database

is performed and the k-ODF of each data is calculated. The top n local outliers will be picked up which have the highest k-ODF values at each distributed site. The local top n outliers are then sent to mediator for producing the global top outliers.

Step 4: Generating global top n outliers (mediator). When all the top n local outliers are collected, the mediator will merge them and generate the global top-n outliers. The final results are returned to end users.

Algorithm description The detailed description of the baseline algorithm of DISTROD is presented in Algorithm 1. The function of $Map(\cdot)$ in Line 4 performs assignment of an incoming data into a particular cell in the grid structure, returning an unique cell identification number. This identification number is used for fast retrieval of its density information in Line 5. $density[c]^{(i)}$ in Line 9 denotes the density of the cell c from site s_i.

Algorithm 1: Baseline algorithm of DISTROD

Input: Distributed databases $D_1, D_2, \ldots, D_t$, and number of global outliers requested n.
Output: Top n global outliers returned to human users.

```
1  for each distributed site s_i ∈ s_1, s_2, ..., s_t do
2      Superimpose a grid structure to data space;
3      for each data p ∈ D_i do
4          cell_Index ← Map(p);
5          density[cell_index]^i ++;
6      Transmit density information of populated cells from s_i to the mediator;
7  for the mediator do
8      global_dense_cells = ∅;
9      for each populated cell c received by the mediator do
10         density[c] = Σ_i density[c]^(i);
11     if density[c] > global_average_density then
12         global_dense_cells = global_dense_cells ∪ c;
13     Transmit the centroid of dense cells to each distributed site s_i;
14 for each distributed site s_i ∈ s_1, s_2, ..., s_t do
15     for each data p ∈ D_i do
16         calculate k_ODF(p);
17     Sorting data based on their k_ODF(p);
18     Transmit top n outlier candidates from the sorting list from s_i to the mediator;
19 for the mediator do
20     Generate the top n global outliers;
21     Return the top n global outliers;
```

5 Communication optimization techniques for DISTROD

Besides proposing the baseline algorithm, we have also devised a number of optimization techniques for effectively reducing the communication overhead of DISTROD. These optimization techniques enable DISTROD to generate the global data summary and the top n global outliers in a more communication efficient way. Optimization strategies 1, 2 and 3 are developed to reduce the communication overhead when generating the global data summary in the mediator and optimization strategy 4 aims to reduce the communication overhead in generating the top n local outliers.

5.1 Optimization 1: approximation and compression methods for obtaining global average density

Please recall that in order to calculate the global average density, We need to know N(the total number of data instances in the global database) and $N_{\text{populated}}$ (the number of populated cells). The value of N can be easily obtained as $N = \sum_{i=1}^{t} N_i$, where N_i is the number of data instances of local database $(D)_i$, $1 \leq i \leq t$. However, $N_{\text{populated}}$ cannot be obtained through the similar aggregation as N. In the baseline algorithm of DISTROD, the information of all the populated cells (including the cell identification numbers and their density) from each distributed site to the mediator for aggregation. This transmission process could still be expensive especially for large and high-dimensional databases that lead to a higher number of populated cells. We proposed two optimization methods to reduce the communication overhead when dealing with the above two issues respectively. The first one is an approximate method which produces an estimate for $N_{\text{populated}}$. The second one is an exact approach which is able to produce an accurate value for $N_{\text{populated}}$. The exact method achieves the reduction of communication overhead through a compression of the information of neighboring cells for efficient transmission. Next, we will discuss these two methods in greater details.

5.1.1 The approximation method

Let $N^{(i)}_{\text{populated}}$ be the number of local populated cells in distributed site s_i, $1 \leq i \leq t$, $N_{\text{populated}}$ is able to satisfy the following inequation:

$$\text{Max}\left(N^{(1)}_{\text{populated}}, N^{(2)}_{\text{populated}}, \cdots, N^{(t)}_{\text{populated}}\right) \leq N_{\text{populated}} \leq \sum_{i=1}^{t} N^{(i)}_{\text{populated}}$$

If $Lower\_bound(N_{\text{populated}})$ and $upper\_bound(N_{\text{populated}})$ are used to denote the lower and upper bounds of $N_{\text{populated}}$, i.e.,

$$Lower\_bound(N_{\text{populated}}) = \text{Max}\left(N^{(1)}_{\text{populated}}, N^{(2)}_{\text{populated}}, \cdots, N^{(t)}_{\text{populated}}\right)$$

$$Upper\_bound(N_{\text{populated}}) = \sum_{i=1}^{t} N^{(i)}_{\text{populated}}$$

Then $N_{\text{populated}}$ can then be estimated as

$$N^{*}_{\text{populated}} = \frac{Lower\_bound(N_{\text{populated}}) + Upper\_bound(N_{\text{populated}})}{2}$$

The above estimation comes with an error bound of

$$error\_bound = \frac{Upper\_bound(N_{\text{populated}}) - Lower\_bound(N_{\text{populated}})}{2}$$

Based on the error bound, we can define the error ratio as

$$error\_ratio = \frac{error\_bound}{N^{*}_{\text{populated}}} \times 100\%$$

The lower the error ratio is, the more accurate the estimation will be. Human users are given the discretion to decide the acceptability of the approximation method

based on the error ratio. For example, one may only use the estimation when its error ratio is lower than 5 %.

Communication overhead analysis The salient advantage of this method is that it features a very low communication overheads; each distributed site only needs to transmit its respective number, rather than the whole list, of local populated cells fix fix to the mediator for producing the estimation.

5.1.2 The exact method

The performance of the approximation method proposed above deteriorates when there is a large error ratio, which typically happens when the number of distributed sites is large and the number of local populated cells at different sites are relatively close to each other. In response to this, we also propose an exact method that can produce the accurate value of $N_{\text{populated}}$ at the mediator. The basic idea of this exact method is to encode the information of the neighboring populated cells using a single integer in a bid to reduce communication overhead. We first present the definition of *neighboring populated cells* as follows before discussing how the encoding is performed.

Definition 5.1 (Neighboring populated cells) For a cell c, a neighboring populated cell of c is defined as one of the populated cells that is adjacent to c in at least one of dimensions. c is called the central cell with regard to its neighbors.

Please note that the diagonal neighboring cells are excluded as per the above definition. As a result, the number of neighboring populated cells for any given cell falls into the range of $[0, 2\delta]$ for δ-dimensional database and grows in a linear, instead of exponential, order with respect to the dimensionality of data. This ensures the efficiency of evaluating the neighboring cells in this method. Figure 4 illustrates a 2-dimensional grid structure in which cell #1, #2, #3 and #4 are the neighboring cells of cell c but only cell #1 and #3 are the populated neighboring cell of the central cell c. The remaining four cells in the corner of the grid, which are what we call the diagonal neighboring cells, are not the neighboring cells as defined in this paper.

For easy referral of different neighboring cells and, more importantly, for the purpose of neighboring cell encoding, we assign different label numbers to neighboring

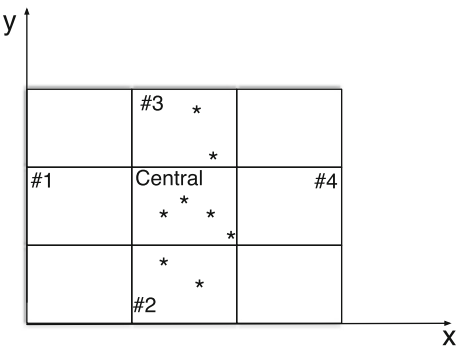

Figure 4 Neighboring cell labeling

cells by abiding by the following convention. A priority level is first given to different dimensions. For example, for 2-dimensional data, we can simply specify that x dimension has a higher priority than y dimension. The general rule for assigning neighboring cell label numbers is specified as: for a given central cell c, starting from the dimension with the highest priority, a smaller label number is always assigned to its neighboring cell that contains data with lower values in that dimension. If there are more than one cell having the same value for a particular dimension, then the next dimension with lower priority will be considered. Let's go back to Figure 4 to demonstrate this idea. In this example, let's assume that x dimension has a higher priority than y dimension. For the central cell c, its left cell is the #1 neighbor because it contains data with a smaller value than that of c in x dimension. The #2 and #3 neighbors have the same value for x dimension but the #2 neighbor has a smaller value than the #3 neighbor in y dimension. Finally, the right cell is the #4 neighbor as it has a larger value in x dimension than the other 3 neighboring cells. This rule can be applied to grid structures with any number of dimensions. Once the neighboring populated cells have been numbered, hash function is utilized to encode them. The purpose of using hash function is to generate a unique integer value that corresponds to a specific possible combination of populated neighboring cells of a given central cell. Instead of transmitting the information of each neighboring cell, we only need to transmit the single integer obtained from the hash function. In addition, the transmitted encoded information can be easily decoded without information loss at the mediator.

To encode the neighboring populated cells for a central cell, we first arrange the label numbers of the neighboring populated cells in an ascending order. For example, if the neighoring populated cells are the #1, #2 and #4 cells, then the sorted set is $\{1, 2, 4\}$. $S_{\text{nei}}(c)$ is used to denoted the set of neighboring populated cells of c and $S_{\text{nei}}(c)[i]$ is used to refer to the $(i+1)$th element in $S_{\text{nei}}(c)$ (note that the index starts from 0). For instance, if $S_{\text{nei}}(c) = \{1, 2, 4\}$, then $S_{\text{nei}}(c)[1] = 2$. The hash value of a set of neighboring populated cell $S_{\text{nei}}(c)$ is calculated as

$$hash\_value(S_{\text{nei}}(c)) = \sum_{i=0}^{|S_{\text{nei}}(c)|-1} \left[(2\delta)^{|S_{\text{nei}}(c)|-i-1} \cdot S_{\text{nei}}(c)[i]\right]$$

As an example, the hash value of the neighboring cell set $\{1, 2, 4\}$ in a 2-dimensional space is calculated as

$$hash\_value(\{1, 2, 4\}) = 1 \cdot 4^2 + 2 \cdot 4 + 4 = 28$$

Lemmas 8.1 and 8.2 in the Appendix prove the uniqueness of the hash value of the neighboring populated cells. They show that the hash function is "perfect" in the sense that they do not produce any collisions for any two different sets of neighboring populated cells, no matter whether these two sets have the same cardinality or not, their hash values are guaranteed to be different.

Table 2 is a complete hash table containing the encoding values for all the different possible combinations of neighboring cells using the aforementioned hash function for 2-dimensional data. The table can be easily generated at both the mediator and each of the distributed sites independently without involving any data communication. It is actually unnecessary to physically create the hash table because on-the-fly encoding and decoding of neighboring cells is possible.

Besides developing the hash function, another issue that needs to be investigated is how to choose the central cells in the encoding process. Apparently, it is desirable

Table 2 Hash table of neighboring cells for 2-dimensional data

1-neighbor sets		2-neighborsets		3-neighbor sets		4-neighbor sets	
Sets	Hash value	Sets	Hash value	Sets	Hash value	Sets	Hash value
{1}	1	{1, 2}	6	{1, 2, 3}	27	{1, 2, 3, 4}	112
{2}	2	{1, 2}	7	{1, 2, 4}	28		
{3}	3	{1, 3}	8	{1, 3, 4}	32		
{4}	4	{2, 3}	11	{2, 3, 4}	48		
		{2, 4}	12				
		{3, 4}	16				

to choose as few central cells as possible because the communication overhead is dependent upon the number of resulting central cells. The problem of choosing central cells can be formulated as follows. Given a set of populated cells V, choose a smallest subset of V, denoted as V', such that V' covers V. Here, V' covers V means that, for each populated cell $c_i \in V$, we have either $c_i \in V'$ or $\exists c_j \in V'$ such that c_i is a neighboring populated cell of c_j.

Algorithm 2: Algorithm for selecting the central cells

Input: The array storing the identification number and density of the local populated cells in a particular distributed site s_i, which is denoted as Pop_cells.

Output: The central cells, stored in the array called $Central\_cells$, that are selected for encoding for the site.

```
1  Mark every cell in Pop_cells active;
2  Central_cells = ∅ ;
3  for each cell c ∈ Pop_cell do
4      Calculate the number of neighboring populated cells of c;
5  while not all the cells in Pop_cell become inactive OR 2 · |Centra_cells| < |Pop_cells| do
6      Choose the cell from Pop_cells that has the largest number of neighboring populated cells as a
       central cell, denoted by c*;
7      Cental_cells = Cental_cells ∪ c*;
8      Pop_cells = Pop_cells − c*;
9      Mark c* inactive;
10     for each neighboring populated cell of c*, denoted by c_i do
11         Decrement the density of c_i by 1;
12         Mark c_i inactive;
13 Return(Central_cells);
```

To efficiently select the central cells at each distributed site, we develop an algorithm, as presented in Algorithm 2. As the initialization steps, all the populated cells are marked as active at the beginning of the algorithm (Line 1) and the number of neighbors for each populated cell are calculated (Lines 3–4). The algorithm then enters into an iterative loop (Lines 5–12) where it first searches for the cell with the highest number of neighbors in the current list of active populated cells as the first central cell, denoted as c^*. After being selected as a central cell, c^* is removed from the list of populated cells to ensure that it won't be selected as the central cell again later in the algorithm (Line 8). Then for each neighbor of c^*, decrease their respective number of neighbors by 1 (Line 11), and mark itself and all its neighboring populated cells inactive, indicating that they have been covered (Lines 9 and 12). Please note that the inactive cells can still be selected as the central cell. This process continues until one of the following two conditions is met:

1. All the populated cells become inactive, indicating that all the populated cells have been covered by the central cells selected;

2. If we have $2N^{(i)}_{\text{central}} \geq N^{(i)}_{\text{populated}}$, suggesting that there is no any benefit in communication overhead reduction for performing this compression algorithm for site s_i as its associated communication overhead has already exceeded that of directly transmitting uncompressed populated cells to the mediator.

If the algorithm terminates due to the first stopping condition, then the central cells is output for the subsequent decoding operation. After encoding, each distributed site will transmit the following set of pairs to the mediator:

$$\{(centra_1), (hash\_value(centra_1)), (centra_2), (hash\_value(centra_2)), \ldots\}$$

Where $centra_i$ denotes the ith central cell selected at the site. The encoding is exact and is not associated with with any information loss. The total number of pairs may vary for different sites but will all be upper bounded by $N^{(i)}_{\text{populated}}$, the number of local populated cell at the site. If the second condition is the reason for the early stopping of the algorithm, then we simply give up cell compression and choose to transmit all the populated cells to the mediator instead.

The decoding carried out at the mediator is simpler than the encoding process. Upon receiving the central cells and their respective encoded neighbors, the mediator will start to decode the hash value to generate the original set of local populated cells for each distributed site. All these sets will then be merged to generate the populated cells for the global database. Formally, the set of local populated cells for distributed the site $s_i, i \in [1, t]$ can be generated as

$$S^{(i)}_{\text{populated}} = \cup_{j=1}^{|N^{(i)}_{\text{central}}|}(central^{(i)}[j] \cup decode(hash\_value(central^{(i)}[j])))$$

Where $central^{(i)}[j]$ refers to the jth central cell selected from the ith distributed site and the function $decode(\cdot)$ decodes the hash value of the central cell back to its original set of neighboring populated cells. The populated cells of the global database can then be generated as

$$S_{\text{populated}} = \cup_{i=1}^{t} S^{(i)}_{\text{populated}}$$

Once $S_{\text{populated}}$ is generated, its cardinality as well as the global average density can be easily calculated.

Computational complexity analysis At a particular site i, calculating the number of neighbors for each populated cell incurs a complexity of $O\left(2\delta \cdot N^{(i)}_{\text{populated}}\right)$ as there are at most 2δ populated cells for any given cell in δ-dimensional data. The iterative process for selecting the central cell has an ideal complexity of $O\left(2\delta \cdot \frac{N^{(i)}_{\text{populated}}}{2d+1}\right)$ when each central cell is able to cover a maximum of 2δ neighboring cells and a worst complexity of $O\left(2\delta \cdot N^{(i)}_{\text{populated}}\right)$ when each central cell doesn't cover any other populated cells except itself. In a summary, this algorithm has a computational complexity that is in the linear order of δ and $N^{(i)}_{\text{populated}}$, showing that this algorithm is well scalable with regard to the number of populated cells and data dimensionality.

Communication overhead analysis The benefit of the compression algorithm in reducing the communication overhead is dependent upon the overall adjacency of populated cells in the grid structure at each distributed site. Without the use of the compression algorithm, each distributed site needs to transmit all the populated cells from its local grid structure, resulting in a total communication overhead of

$O\left(N^{(i)}_{\text{populated}}\right)$ for each site i. Nevertheless, if the compression algorithm is used and it terminates due to the first stopping condition, then the communication overhead will be $2N^{(i)}_{\text{central}}$, which is guaranteed to be smaller than the total number of populated cells in respective site.

5.2 Optimization 2: transmitting only density information of local dense cell candidates

In the baseline algorithm of DISTROD, both the identification number and density of populated cells need to be sent from distributed sites to the mediator to generate the representative data. However, optimization can be performed on this step to save the transferring of density information for many cells. Instead of transmitting density information of all the populated cells from each distributed site to the mediator, we developed a pruning method to efficiently remove the local populated cells that cannot possibly become a global dense cell. The remaining cells are the local dense cell candidates. The global dense cells will only be selected from this candidate list collected from each distributed site. That is, if a cell is not one of the local dense cell candidates, then it is impossible for it to become a global dense cell. As a result, each distributed site only needs to transmit the density information of local dense cell candidates, which is typically a much smaller set than that of the populated cells. The definition of the local dense cell candidates are presented as follows.

Definition 5.2 (Local Dense Cell Candidate) A local dense cell candidate is defined as a cell whose local density is above or equal to $\frac{global\_ave\_density}{t}$, where t represents the number of distributed sites.

The pruning mechanism is devised based upon the following lemma that unveils the relationship between the global dense cells and the local dense cell candidates.

Lemma 5.3 *A global dense cell must be a local dense cell candidate at one or more distributed sites.*

Proof This lemma can be easily proved by mathematical contradiction. Let's assume that a cell c is a global dense cell but it is not a local dense cell candidate at any of the distributed sites, then for each distributed site s_i, $1 \leq i \leq t$, we have $density(c)^{(i)} < \frac{global\_ave\_density}{t}$, then we have $density(c) = \sum_{i=1}^{t} density(c)^{(i)} < global\_ave\_density$, meaning that the global density of c is lower than the global average density level. Therefore, c cannot be a global dense cell. This leads to a contradiction with the assumption that c is a global dense cell. The lemma is thus proved as required. □

Communication overhead analysis Let us assume that there are on average a ($0 < a < 1$) of the populated grid cells in distributed sites. The communication overhead in transmitting their identification information to the mediator is $O(al^{\delta}t)$. The number of dense cell candidates is at most $O(al^{\delta})$, making the overhead of both broadcasting dense cell candidates and final density information transmission be $O(al^{\delta}t)$. Thus, the total overhead is $O(3al^{\delta}t)$.

5.3 Optimization 3: early-stopping transmission of density information

A more refined progressive density aggregation approach can be developed to further reduce the overhead in the process for communicating density information. To this end, a particular transmission order needs to be established among all the sites. This can be easily done by assigning a sequential number to each site (e.g., $s_1, s_2, \ldots, s_t$). Under such order, the density transmission is first conducted from site s_1 to the mediator, followed by the transmission from site s_2 to the mediator and so on. Instead of waiting until receiving the density information of the dense cell candidates from all sites before kicking off aggregation operation, the mediator performs aggregation every time when it receives the density information from a distributed site. By doing so, the aggregation process can be early stopped; at any moment in this process if we find the aggregated cell density exceeds the pre-determined threshold, then this cell is considered dense and the aggregation process for this cell can be terminated.

Communication overhead analysis For a dense cell candidate, the most ideal case involves only transmitting its density from a single site to the mediator, while in the worse case, all the t sites needs to transmit the density information. The average complexity will be $O(\frac{t+1}{2})$. Therefore, the communication overhead of collecting the density information of all the dense cell candidates in the mediator is $O\big(al^{\delta}\frac{t(t+1)}{2}\big)$.

5.4 Optimization 4: implementing global top n outliers merging algorithm

In the baseline algorithm, each distributed site will ship the top n local outliers to the mediator to produce the global top n outliers. This scheme can be optimized to achieve a better communication performance by transmitting only some of the local top outliers to the mediator from each site. The basic idea of this optimization is similar to the the early-stopping transmission of density information. Again, a particular transmission order needs to be established for all the distributed sites. We can do this by simply following the same sequential numbers that have been assigned to different distributed sites when employing the early-stopping method for density transmission. Under such order, the top local outliers are first transmitted from site s_1 to the mediator. The minimum k-ODF value of the outliers from s_1, denoted as Min_{kODF}, is extracted and broadcast to all the distributed sites. We can then prune away local top outliers in each of the remaining distributed sites whose k-ODF value is lower than Min_{kODF} as it is guaranteed that they cannot possibly be included in the global top n list later. After pruning, the surviving outliers at s_2 are transmitted to the mediator and Min_{kODF} will be updated. This process will continue until no local top outlier are left after the pruning for all the remaining sites that have yet transmitted their local top outliers.

Communication overhead analysis In the most ideal situation, the first batch of transmitted outliers which are from s_1 to the mediator, are the global top n outliers with an overhead of only ($O(n)$). However, we end up with an overhead of $O(nt)$ in the worst case when we have to exhaust the transmission of the top n outliers for all the t distributed sites. Thus, the average communication overhead is $O\big(\frac{n(1+t)}{2}\big)$.

6 Enhanced algorithm of DISTROD: a big picture

After having presented various optimization and enhancement strategies for improving communication and speed performance of DISTROD, it is now ready for us to present the complete enhanced algorithm of DISTROD, which detect distributed outliers in the following six steps. The detailed algorithmic description is given in Algorithm 3.

Step 1: Assigning data into grid structure (distributed sites). The first step of the enhanced algorithm is identical to that of the baseline version which mainly involves superimposing the grid structure, assigning data points into the grid structure and obtaining the local density of each populated cell.

Step 2: Calculating the global average density (mediator). The second step of the enhanced algorithm calculates the global average density at the mediator. Human users can choose to use either the approximate or exact methods to do this. If the approximate method is chosen, then only the number of data points for each local database and the number of local populated cells need to be transmitted from distributed sites to the mediator. If users utilize the exact method, then neighboring cell encoding will be first performed at each distributed site and the encoding results will be transmitted thereafter. The mediator, depending on the methods that user choose at the distributed sites, will produce either the estimated or accurate global average density and broadcast it to all distributed sites.

Step 3: Generating local dense cell candidates (distributed sites). Each distributed site will generate the local dense cell candidates when receiving the global average density from the mediator. This involves the pruning away those cells that cannot possibly become the global dense cells. The remaining populated cells are the local dense cell candidates that will be transmitted to the mediator.

Step 4: Generating the global data summary (mediator). The local dense cell candidates will be merged at the mediator to generate the global dense cell candidates. The mediator will then request the density of the global dense cell candidates from distributed sites using early-stopping density transmission method. The global densities of the global dense cell candidates are then calculated, whereby the global dense cells can be generated at the mediator.

Step 5: Generating Local top n outliers (distributed sites). When each distributed site receives the global data summary from the mediator, distribute sites will start to detect local top n outliers. By performing a full scan of each local database, the local top n outliers are detected and then sent to mediator for producing the global top outliers.

Step 6: Generating global top n outliers (mediator). When all the top n local outliers are collected using early-stopping outlier transmission method, the mediator will merge them and generate the global top-n outliers. The final results are returned to end users.

Remark After a close comparison between the baseline and enhanced algorithms of DISTROD, we can see that the performance improvements of the enhanced algorithm mainly lie in, but not limited to, Step 2 and 3 described above that involves generating the global average density and the local dense cell candidates, respectively. Despite that it features more rounds of communication than the baseline version, the enhanced algorithm is able to achieve a much lower communication overhead in

transmitting cell density information in most cases. This overweights the slightly higher overhead in initiating communication sessions in the enhanced algorithm.

Algorithm 3: Enhanced Algorithm of DISTROD

Input: Distributed databases $D_1, D_2, \ldots, D_t$, number of global outliers requested n and the number of circular zones for each cell in Group 3 m.
Output: Top n global outliers returned to human users.

```
for each distributed site s_i ∈ s_1, s_2, ..., s_t do
    Superimpose a grid structure to data space;
    for each data p ∈ D_i do
        cell_Index ← Map(p);
        density[cell_index]^(i) + +;
    Transmit the number of populated cells from s_i to the mediator;
for the mediator do
    global_average_density = estimate(|D_1|, |D_2|, ..., |D_t|);
    Broadcast global_average_density to all distributed sites;
for each distributed site s_i ∈ s_1, s_2, ..., s_t do
    local_dense_cells^(i) = ∅;
    for each populated cell c in s_t do
        if density[c]^(i) ≥ global_average_density / t then
            local_dense_cells^(i) = local_dense_cells^(i) ∪ c;
    Transmit local_dense_cells^(i) to the mediator;
for the mediator do
    global_dense_cells = ∅;
    Calculate the global density of the global dense cell candidates c using the early-stopping density
    transmission method;
    if density[c] > global_average_density then
        global_dense_cells = global_dense_cells ∪ c;
    Transmit the centroid of global dense cells to each distributed site s_i;
for each distributed site s_i ∈ s_1, s_2, ..., s_t do
    Detect the top n local outliers;
    Transmit the top n local outliers from s_i to the mediator using early-stopping approach;
for the mediator do
    Generate the top n global outliers;
    Return the top n global outliers;
```

7 Experimental evaluations

In this section, we will report results of the extensive experimental evaluation conducted to evaluate DISTROD's effectiveness, speed and communication overhead. We used both synthetic and real-life datasets in the experimental evaluation. A synthetic dataset generator is utilized to generate datasets that feature various desired number of data instances and dimensions. This gives us a great deal of control over the scale of the datasets to experiment with. The KDD Cup 99 network intrusion detection dataset is also used as a real-life dataset to detect intrusions which manifest as outlying observations. To construct the distributed computing environment for the experimental purpose, we use five PCs to simulate distributed sites and one PC as the mediator. For the experiments that require more than five distributed sites, simulation is done through multiple sequential sessions on each of the five machines. The execution time and the communication overhead are recorded for each session

to produce the final experimental results. Each machine is configured with a 2.4 GHz Intel Processor, 4G RAM and 250 G hard disc storage.

7.1 Scalability study

In the scalability experiment, we conduct a study to evaluate the efficiency of DISTROD when it deals with large datasets. The number of data instances in each distributed site in this experiment ranges from 100,000 to 1,000,000 to give a good simulation for large datasets. Another dimension of the scalability study is to investigate how well DISTROD can scale up with regard to the number of distributed sites. We simulate the situations where there are varying number of participating distributed sites ranging from 100 to 500. Figures 5 and 6 show the scalability performance of DISTROD with respect to the number of data instances and distributed sites, respectively. From Figure 5, we can see that DISTROD exhibits a linear increase of execution time when the number of data instances increases from 100,000 to 1,000,000 for all sites. Such good linear behaviour is contributed by the grid-based structure and efficient algorithm in each site to construct the local data summary and detect outliers. This experiment establishes that DISTROD is well able to handle large data sources in a promising efficiency. Figure 6 shows that DISTROD is rather insensitive to the number of sites, an advantageous feature that enables DISTROD to easily handle large-scale network. The slight increase of its execution time when the number of sites increases is due to the fact that some sites that finish their processing earlier have to wait for other sites to finish the processing. A large number of sites will statistically increase the length of such waiting period.

In this scalability study, we also evaluate the speed improvement when utilizing the three-group method for evaluating different cell groups. The performance of the three-group method is compared with the baseline method where no cell pruning is performed (i.e., each data point in the database will be evaluated). The execution time of the baseline method is considered as unit 1. The comparison result, as presented in Figure 7, shows that the three-group method is able to save up to 75–93 % of the computation by utilizing the pruning strategies for datasets with varying

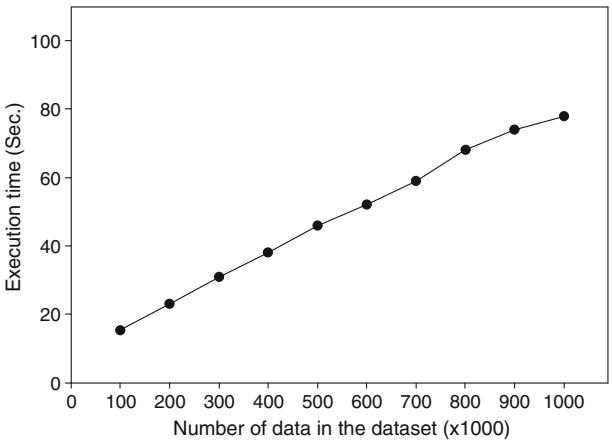

Figure 5 Scalability w.r.t database size

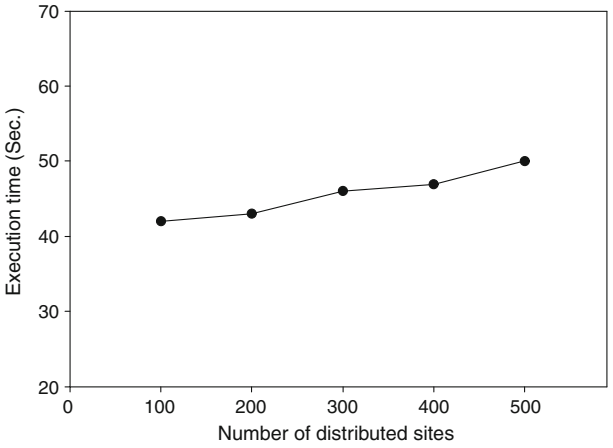

Figure 6 Scalability w.r.t number of distributed sites

sizes. The speed contribution of the three-group method is more prominent when the size of the database is large, making it ideal for handling large databases.

7.2 Communication overhead evaluation

A major part of our experimental evaluation is to analyze how each of the four optimization strategies contributes to the reduction of communication overhead for DISTROD. In order to get a better idea as to the extent of contribution from each strategy, we compare the communication overhead of the baseline algorithm and the enhanced algorithm that only implements a single communication optimization strategy each time. The communication overhead of the baseline algorithm is considered as unit 1 in the experiments. The performance contribution analysis for all the optimization strategies is presented in the following subsections.

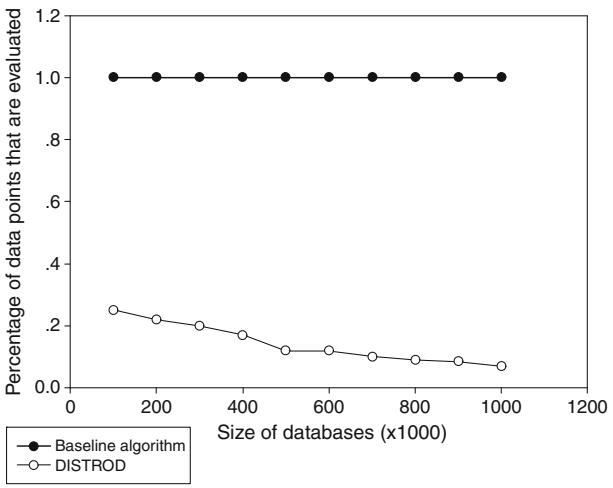

Figure 7 Speed improvement by using the three-group pruning method

7.2.1 Contribution of compressing the information of neighboring populated cells

The contribution of transmitting the compressed information of the neighboring populated cells for generating the global average density is jointly affected by the overall data distribution of the dataset and the granularity of the gird structure. As the data distribution of the dataset is an uncertain factor, thus we only investigate the effect of the granularity of the grid structure. Grid granularity can be represented by the number of cells in the grid, which are further determined by the dimension of data δ and the number of intervals l that each dimension is partitioned into in the grid. In this experiment, we study the contribution of this strategy when δ is increased from 5 to 30 and l is increased from 5 to 20. The result is presented by a 3D mesh plot in Figure 8. When comparing with the baseline algorithm, this optimization strategy is able to achieve 30–60 % communication reduction for the task of transmitting local populated cells to the mediator for generating global average density. This experimental result also indicates that the encoding method delivers a better job to reduce communication overhead when the data space is more coarsely partitioned due to an overall higher number of neighboring populated cells for the selected central cells in the grid.

7.2.2 Contribution of transmitting only the density information of global dense cell candidates

Similar to the overhead reduction analysis of the first optimization strategy presented above, the contribution provided by transmitting only the density of the global dense cell candidates is also affected by data distribution and the grid structure. A similar 3D mesh plot is presented in Figure 9 to demonstrate how the data dimension and the number of partition intervals affect the communication overhead reduction offered by this strategy. As unveiled by the figure, given a fixed dataset, the saving of communication overhead from this strategy can be 10–20 %, and such contribution becomes more significant when the granularity of the grid structure is finer. This is because that, when the grid structure is finer, there tends to be more sparse cells at each

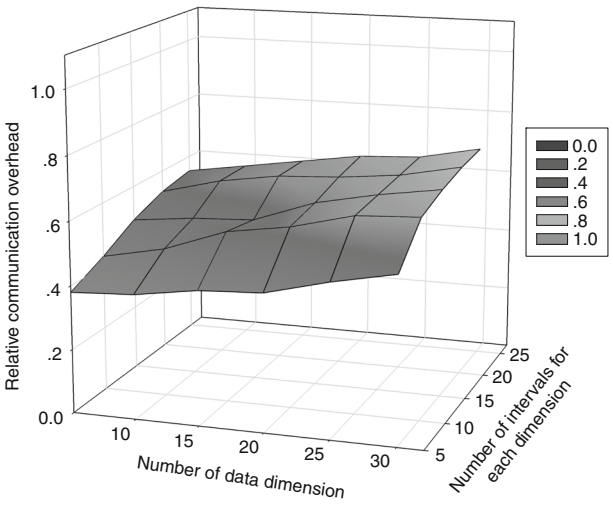

Figure 8 Contribution of transmitting only the density information of dense cell candidate

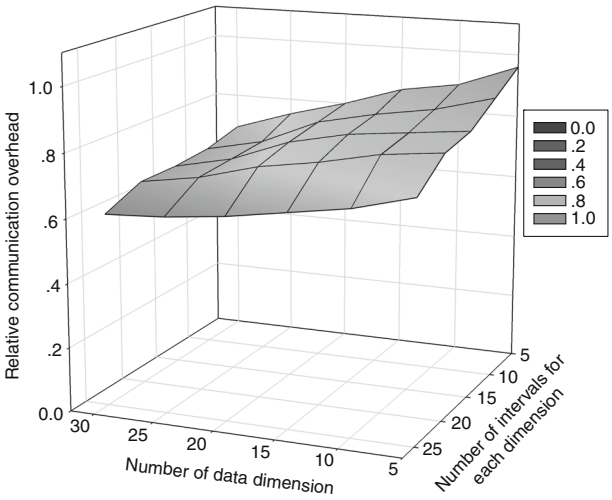

Figure 9 Contribution of transmitting the density of global dense cell candidates

distributed sites which can be pruned away. This results in a smaller set of global dense cell candidates that needs to be transmitted to the mediator.

7.2.3 Contribution of early stopping transmission of density information

As we have discussed earlier, the number of communication overhead for the density information of a global dense cell candidate ranges from $[1, t]$. Even though the worst theoretical complexity is still t, the complexity in most of practical cases is less than t. In this experiment, we test the contribution of the method of early stopping transmission of density information from distributed sites to the mediator. Figure 10 shows the experimental result of the communication overhead saving using this early stopping method under varying number of participating sites t (from 5 to 30). As indicated by the figure, this strategy can help reduce the communication overhead by

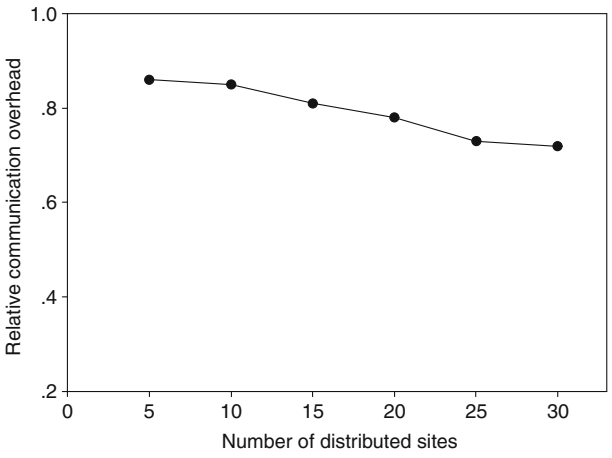

Figure 10 Contribution of early stopping transmission of density information

20–30 % for this step, and it can achieve better communication overhead reduction as the number of sites increases.

7.2.4 Contribution of the global top global outlier merging algorithm

Utilizing the top global outlier merging algorithm will further help reduce to number of local outliers to be transmitted from the distributed sites to the mediator. Figure 11 shows the contribution of this strategy in communication overhead reduction under varying values of site number t ranging from 5 to 30. We can see from the result that the communication overhead saving ranges from 4 to 17 % and becomes more significant when the value of t goes up.

7.2.5 Relative performance contribution analysis

After experimentally evaluating the contribution for the four optimization techniques individually under their respective affecting factors, we now evaluate these four strategies together in order to get a bigger picture about their relative performance contribution under the same experimental setup. Figure 12 presents a pie chart showing the portion of contribution for all the four strategies. This result is obtained under the same reasonable experimental setup, namely $N = 100{,}000$, $\delta = 10$, $n = 20$, $t = 10$ and $k = 2$.

Among the four communication optimization strategies, Strategy 1 (transmitting the compressed information of the neighboring populated cells) is the biggest contributor accounting for 55 % of total communication overhead reduction, followed by Strategy 2 (transmitting density information of the global dense cell candidates) that registers a 25 % contribution. Strategy 3 (the early stopping method for transmitting the density for dense cell candidates) and Strategy 4 (the global outlier merging algorithm) respectively achieve the remaining 12 and 8 % shares.

Another interesting finding from this experiment is that the optimization strategies for reducing the communication overhead incurring in generating the global data summary (i.e., Strategy 1, 2 and 3) are much more important than that utilized for generating the final global outliers (i.e., Strategy 4). This is because that the number

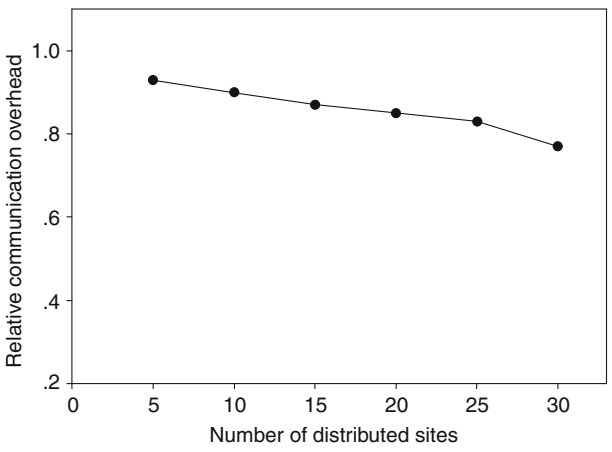

Figure 11 Contribution of the global outlier merging algorithm

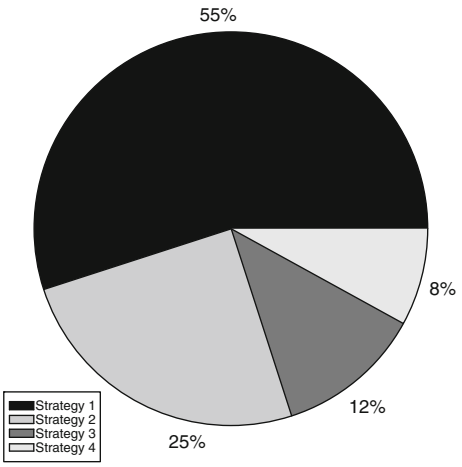

Figure 12 Performance contribution analysis for all communication optimization strategies

of populated cells in the grid structure is typically much larger than the number of the top global outliers users may request.

It is also important to note that the result presented in the pie chart in Figure 12 is largely dependent on a snapshot of experimental setup. We may end up with different pie chart results under different parameter values. For example, if we are dealing with a relatively small database with lower dimensionality, then the contribution of the Strategy 1 and 2 will become less significant. Also, when the number of the distributed sites is large, Strategy 1 and 2 will play a more important role. Finally, if human users request a larger number of global outliers, then the contribution of Strategy 3 and 4 will have higher share of contributions.

7.3 Comparison with mixed attribute method

In this experiment, comparison is conducted between DISTROD and the existing methods dealing with distributed outlier detection from data with mixed attributes [13, 16]. Communication overhead under different dimensions of data is studied. Dimensionality of data will affect the total number of itemsets that need to be evaluated in terms of their support values in [13, 16], similarly it will also exert an influence on the number of cells in the grid structure in DISTROD. The data dimension is set from 5 to 30 in this experiment. We also specified different combination ratios for the continuous and categorical attributes of the data. The percentage of continuous attributes is defined as $\rho = \frac{|S_{\text{continuous}}|}{\delta}$, where $S_{\text{continuous}}$ represents the set of continuous attributes of the data. For ease of presentation, the communication overhead of the mixed attribute method is normalized to unit 1. The comparative result is presented in Figure 13. It is observed that the communication overhead of the mixed attribute approach is significantly higher (may be up to 3 times) than that of DISTROD. Moreover, the benefit of communication overhead saving is more prominent for DISTROD when the percentage of continuing attributes is increased (i.e., ρ goes up). The underlying reason is that those mixed attribute methods, besides

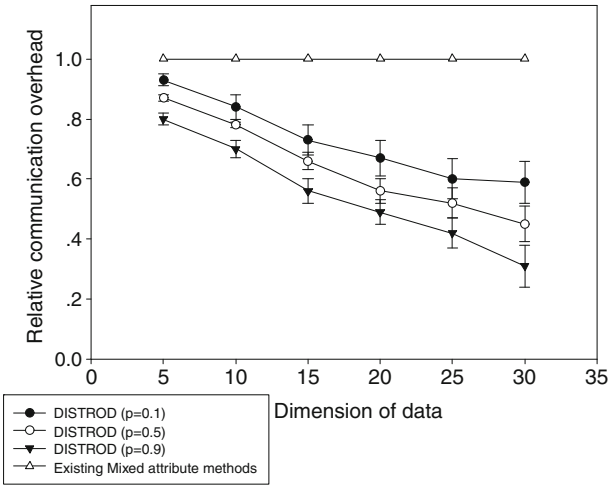

Figure 13 Comparison with existing mixed attribute method

transmitting supports of frequent itemsets, need to transmit additional high volume information for the continuing attributes to maintain the global covariance matrix.

8 Conclusions and future work

In this paper, we study the problem of global outlier detection from large distributed databases. A new technique called DISTROD is proposed in this paper to address this problem. DISTROD is effective and it is able to detect global outliers that are consistent with the centralized detection approaches. It is characterized with fast speed and low communication overhead due to a number of optimization/enhancement strategies implemented. It also provides strong protection for data privacy throughout the detection process. Its good performance has been demonstrated by experimental evaluations.

In the future, we are interested in using DISTROD to deal with distributed high-dimensional datasets. The concept of subspace outliers will be considered in the problem and we will be working on developing algorithms that enable DISTROD to handle subspace outliers in distributed high-dimensional datasets efficiently.

It is also very interesting to investigate the dynamic databases or data streams that continuously arrive at each distributed site which are under restrictions of limited caching and secondary storage capabilities. When handling data streams, the global data summary needs to be refreshed regularly to keep it current and special considerations needs to take on the best mechanism to update the data summary in order to achieve the best possible detection accuracy.

Appendix

The following are the two lemmas proving the uniqueness of the hash value devised for the neighboring populated cells to support our claim in Section 5.1.

Lemma 8.1 *Let S_1 and S_2 be two different sets of neighboring populated sets such that $|S_1| = |S_2|$, then we have hash_value$(S_1) \neq$ hash_value(S_2).*

Proof For the two sets, we can always find the smallest index j such that $S_1[j] \neq S_2[j]$. Without losing generality, it is assumed that $S_1[j] > S_2[j]$. For example, if $S_1 = \{1, 3, 4\}$ and $S_2 = \{1, 2, 4\}$, then the smallest index $i = 1$ as $S_1[1] > S_2[1]$. Given $S_1[j] > S_2[j]$, we need to prove that $hash\_value(S_1) > hash\_value(S_2)$, which are presented as follows.

Let $|S_1| = |S_2| = q$.

$$hash\_value(S_1) = p + \sum_{i=j}^{q-1} S_1[i](2\delta)^{q-i-1}$$

$$hash\_value(S_2) = p + \sum_{i=j}^{q-1} S_2[i](2\delta)^{q-i-1}$$

Where p refers to the partial hash value for the elements indexed from 0 to $j-1$, which are identical for S_1 and S_2.

Thus,

$$\begin{aligned} hash\_value(S_1) - hash\_value(S_2) &= \sum_{i=j}^{q-1} S_1[i](2\delta)^{q-i-1} - \sum_{i=j}^{q-1} S_2[i](2\delta)^{q-i-1} \\ &= S_1[j](2\delta)^{q-j-1} - S_2[j](2\delta)^{q-j-1} \\ &\quad + \sum_{i=j+1}^{q-1} S_1[i](2\delta)^{q-j-1} - \sum_{i=j+1}^{q-1} S_2[i](2\delta)^{q-j-1} \end{aligned}$$

Dividing both sides of the above inequity by $(2\delta)^{q-j-1}$, we get

$$\begin{aligned} \frac{hash\_value(S_1) - hash\_value(S_2)}{(2\delta)^{q-j-1}} &= S_1[j] - S_2[j] + \frac{\sum_{i=j+1}^{q-1} S_1[i](2\delta)^{q-i-1}}{(2\delta)^{q-j-1}} \\ &\quad - \frac{\sum_{i=j+1}^{q-1} S_2[i](2\delta)^{q-i-1}}{(2\delta)^{q-j-1}} \end{aligned}$$

where

$$S_1[j] - S_2[j] \geq 1$$

$$\frac{\sum_{i=j+1}^{q-1} S_1[i](2\delta)^{q-i-1}}{(2\delta)^{q-j-1}} - \frac{\sum_{i=j+1}^{q-1} S_2[i](2\delta)^{q-i-1}}{(2\delta)^{q-j-1}} < 1$$

Therefore, $\frac{hash\_value(S_1) - hash\_value(S_2)}{(2\delta)^{q-j-1}} > 0$, leading to $hash\_value(S_1) > hash\_value(S_2)$, as required. □

Lemma 8.2 *Let S_1 and S_2 be two different sets of neighboring populated sets such that $|S_1| \neq |S_2|$, then we have hash_value$(S_1) \neq$ hash_value(S_2).*

Proof Without losing generality, we assume that $|S_1| > |S_2|$, then we need to prove that $hash\_value(S_1) > hash\_value(S_2)$.

Let us set $|S_1| = p$ and $S_2 = q$ where $p > q$. Based on the definition of the hash function, we have

$$hash\_value(S_1) = \sum_{i=0}^{p-1} S_1[i](2\delta)^{p-i-1}$$

$$hash\_value(S_2) = \sum_{i=0}^{q-1} S_2[i](2\delta)^{q-i-1}$$

We can find the lower bound of $hash\_value(S_1)$ and the upper bound of $hash\_value(S_2)$ as follows:

$$hash\_value(S_1) > \sum_{i=0}^{p-1} (2\delta)^{p-i-1}$$

$$hash\_value(S_2) < \sum_{i=1}^{q} (2\delta)^{q-i-1}$$

Thus,

$$\begin{aligned} hash\_value(S_1) - hash\_value(S_2) &> \sum_{i=0}^{p-1} (2\delta)^{p-i-1} - \sum_{i=1}^{q} (2\delta)^{q-i-1} \\ &= \sum_{i=0}^{p-1-q} (2\delta)^{p-i-1} + 1 > 0 \end{aligned}$$

Which means that $hash\_value(S_1) > hash\_value(S_2)$, as required. □

References

1. Agrawal, R., Gehrke, J., Gunopulos, D., Raghavan, P.: Automatic subspace clustering of high dimensional data for data mining applications. In: SIGMOD'98, pp. 94–105. Seattle, Washington (1998)
2. Breuning, M., Kriegel, H-P., Ng, R., Sander, J.: LOF: identifying density-based local outliers. In: SIGMOD'00, pp. 93–104. Dallas, Texas (2000)
3. Barnett, V., Lewis, T.: Outliers in Statistical Data, 3rd edn. John Wiley (1994)
4. Branch, J.W., Szymanski, B.K., Giannella, C., Wolff, R., Kargupta, H.: In-network outlier detection in wireless sensor networks. In: ICDCS'06, pp. 51. Lisboa, Portugal (2006)
5. Chhabra, P., Scott, C., Kolaczyk, E.D., Crovella, M.: Distributed spatial anomaly detection. In: INFOCOM'08, pp. 1705–1713. Phoenix, AZ (2008)
6. Dutta, H., Giannella, C., Borne, K.D., Kargupta, H.: Distributed top-K outlier detection from astronomy catalogs using the DEMAC system. In: SDM'07. Minneapolis, Minnesota (2007)
7. Ester, M., Kriegel, H-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: SIGKDD'96, pp. 226–231. Portland, Oregon, USA (1996)
8. Hawkins, D.: Identification of Outliers. Chapman and Hall, London (1980)
9. Hinneburg, A., Keim, D.A.: An efficient approach to cluster in large multimedia databases with noise. In: SIGKDD'98, pp. 58–65. New York, NY (1998)
10. Jin, W., Tung, A.K.H., Han, J.: Finding top n local outliers in large database. In: SIGKDD'01, pp. 293–298. San Francisco, CA (2001)

11. Knorr, E.M., Ng, R.T.: Algorithms for mining distance-based outliers in large dataset. In: VLDB'98, pp. 392–403. New York, NY (1998)
12. Knorr, E.M., Ng, R.T.: Finding intentional knowledge of distance-based outliers. In: VLDB'99, pp. 211–222. Edinburgh, Scotland (1999)
13. Koufakou, A., Georgiopoulos, M.: A fast outlier detection strategy for distributed high-dimensional data sets with mixed attributes. Data Min. Knowl. Disc. **20**(2), 259–289 (2010)
14. Kaosar, M.G., Xu, Z., Yi, X.: Distributed Association rule mining with minimum communication overhead. In: AusDM'09. Melbourne, Australia (2009)
15. Ng, R., Han, J.: Efficient and effective clustering methods for spatial data mining. In: VLDB'94, pp. 144–155. Santiago, Chile (1994)
16. Otey, M., Ghoting, A., Parthasarathy, S.: Fast distributed outlier detection in mixed attribute data sets. Data Min. Knowl. Disc. **12**(2), 203–228 (2006)
17. Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. In: SIGMOD'00, pp. 427–438. Dallas, Texas (2000)
18. Sheng, B., Li, Q., Mao, W., Jin, W.: Outlier detection in sensor networks. In: MobiHoc'07, pp. 219–228. Montral, Qubec, Canada (2007)
19. Su, L., Han, W., Yang, S., Zou, P., Jia, Y.: Continuous adaptive outlier detection on distributed data streams. In: HPCC'07, pp. 74–85. Houston, TX, USA (2007)
20. Tang, J., Chen, Z., Fu, A., Cheung, D.W.: Enhancing effectiveness of outlier detections for low density patterns. In: PAKDD'02, pp. 535–548. Taipei, Taiwan (2002)
21. Zhou, J. et al: A novel outlier detection algorithm for distributed databases. In: FSKD'05, pp. 293–297. Shangdong, China (2008)
22. Zhang, J., Wang, H.: Detecting outlying subspaces for high-dimensional data: the new task, algorithms and performance. Knowl. Inf. Syst. (KAIS) **10**(3), 333–355 (2006)
23. Zhang, J., Hsu, W., Lee, M.L.: Clustering in dynamic spatial databases. J. Intell. Inf. Syst. (JIIS) **24**(1), 5–27 (2005)
24. Zhang, J., Lou, M., Ling, T.W., Wang, H.: HOS-Miner: a system for detecting outlying subspaces of high-dimensional data. In: VLDB'04, pp. 1265–1268. Toronto, Canada (2004)
25. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. In: SIGMOD'96, pp. 103–114. Montreal, Canada (1996)