Intelligent Data Analysis 6 (2002) 451–468
IOS Press

Detecting graph-based spatial outliers[1]

Shashi Shekhar, Chang-Tien Lu* and Pusheng Zhang
Computer Science Department, University of Minnesota, 200 Union Street SE, Minneapolis, MN-55455, USA
Tel.: +1 612 6248307; Fax: +1 612 6250572; E-mail: [shekhar,ctlu,pusheng]@cs.umn.edu

Received 4 December 2001
Revised 28 December 2001
Accepted 28 January 2002

Abstract. Identification of outliers can lead to the discovery of unexpected and interesting knowledge. Existing methods are designed for detecting spatial outliers in multidimensional geometric data sets, where a distance metric is available. In this paper, we focus on detecting spatial outliers in graph structured data sets. We define statistical tests, analyze the statistical foundation underlying our approach, design a fast algorithm to detect spatial outliers, and provide cost models for outlier detection procedures. In addition, we provide experimental results from the application of our algorithm on a Minneapolis-St. Paul (Twin Cities) traffic data set to show its effectiveness and usefulness.

Keywords: Outlier detection, spatial data mining, spatial graphs

1. Introduction

Data mining is a process to extract nontrivial, previously unknown and potentially useful information(such as knowledge rules, constraints, regularities) from data in databases [4,11]. The explosive growth in data and databases used in business management, government administration, and scientific data analysis has created a need for tools that can automatically transform the processed data into useful information and knowledge. Spatial data mining is a process of discovering interesting and useful but implicit spatial patterns. With the enormous amounts of spatial data obtained from satellite images, medical images, and GIS, it is a nontrivial task for humans to explore spatial data in detail. Spatial data sets and patterns are abundant in many application domains related to NASA, the National Imagery and Mapping Agency, the National Cancer Institute, and the United States Department of Transportation.

Data Mining tasks can be classified into four general categories: (a) dependency detection (e.g., association rules) (b) class identification (e.g., classification, clustering) (c) class description (e.g., concept generalization), and (d) exception/outlier detection [9]. The objective of the first three categories is to identify patterns or rules from a significant portion of a data set. On the other hand, the outlier detection problem focuses on the identification of a very small subset of data objects often viewed as noises, errors,

[1] This work is supported in part by the Army High Performance Computing Research Center under the auspices of Department of the Army, Army Research Laboratory Cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, and by the National Science Foundation under grant 9631539.

*Corresponding author. Tel.: +1 612 626 7703; E-mail: ctlu@cs.umn.edu.

exceptions, or deviations. Outliers have been informally defined as observations which appear to be inconsistent with the remainder of a set of data [2], or which deviate so much from other observations so as to arouse suspicions that they were generated by a different mechanism [6]. The identification of outliers can lead to the discovery of unexpected knowledge and has a number of practical applications in areas such as credit card fraud, voting irregularities, severe weather prediction, and athlete performance analysis of athletes.

Outliers in a spatial data set can be classified into three categories: set-based outliers, multi-dimensional space-based outliers, and graph-based outliers. A set-based outlier is a data object whose attributes are inconsistent with the attribute values of other objects in a given data set regardless of the spatial relationships. Both multi-dimensional space-based outliers and graph-based outliers are spatial outliers, that is, data objects that are significantly different in attribute values from the collection of data objects among spatial neighborhoods. However, multi-dimensional space-based outliers and graph-based outliers are based on different spatial neighborhood definitions. In multi-dimensional space-based outlier detection, the definition of spatial neighborhood is based on Euclidean distance, while in graph-based spatial outlier detections, the definition is based on graph connectivity. Many spatial outlier detection algorithms have been recently proposed; however, spatial outlier detection remains challenging for various reasons. First, the choice of a neighborhood is nontrivial. Second, the design of statistical tests for spatial outliers needs to account for the distribution of the attribute values at various locations as well as the aggregate distribution of attribute values over the neighborhoods. In addition, the computation cost of determining parameters for a neighborhood-based test can be high due to the possibility of join computations.

1.1. An illustrative application domain: Traffic data set

In 1995, the University of Minnesota and the Traffic Management Center Freeway Operations group started the development of a database to archive sensor network measurements from the freeway system in the Twin Cities. The sensor network includes about nine hundred stations, each of which contains one to four loop detectors, depending on the number of lanes. Sensors embedded in the freeways monitor the occupancy and volume of traffic on the road. At regular intervals, this information is sent to the Traffic Management Center for operational purposes, e.g., ramp meter control and research on traffic modeling and experiments. Figure 1 shows a map of the stations on highways within the Twin-Cities metropolitan area, where each polygon represents one station. Interstate freeways include I-35W, I35E, I-94, I-394, I-494, and I-694. State trunk highways include TH-100, TH-169, TH-212, TH-252, TH-5, TH-55, TH-62, TH-65, and TH-77. I-494 and I-694 together form a ring around the Twin Cities. I-94 passes from East to North-West, while I-35W and I-35E run in a North-South direction. Downtown Minneapolis is located at the intersection of I-94, I-394, and I-35W, and downtown Saint Paul is located at the intersection of I-35E and I-94.

Figure 2(a) demonstrates the relationship between a station and its encompassing detectors. For each station, there is one detector installed in each lane. The traffic flow information measured by each detector can then be aggregated to the station level. Figure 2(b) shows the three basic data tables for the traffic data. The *station* table stores the geographical location and some related attributes for each station. The relationship between each detector and its corresponding station is captured in the *detector* table. The *value* table records all the volume and occupancy information within each 5-minute time slot at each particular station.

In this application, each station exhibits both graph and attribute properties. The topological space is the map, where each station represents a node, and the connection between each station and its

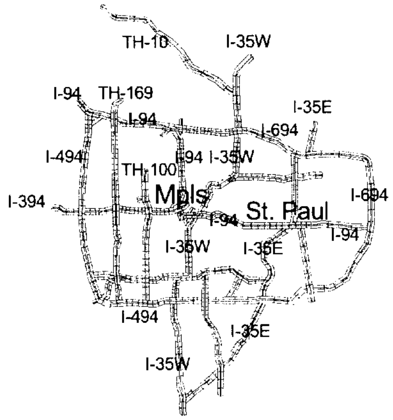
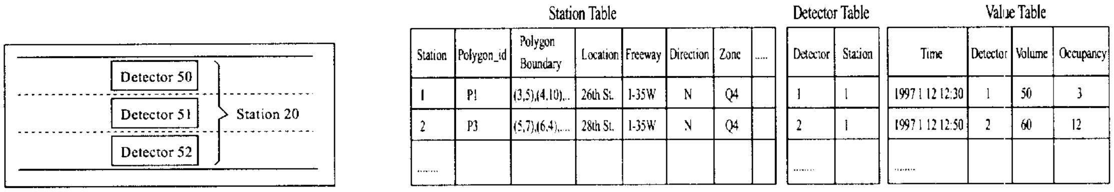


Fig. 1. Detector map at the station level.



(a) Relationship between Detectors and Stations

Station Table

Station	Polygon_id	Polygon Boundary	Location	Freeway	Direction	Zone	.....
1	P1	(3,5),(4,10),..	26th St.	I-35W	N	Q4	
2	P3	(5,7),(6,4),....	28th St.	I-35W	N	Q4	
....							

Detector Table

Detector	Station
1	1
2	1
....	

Value Table

Time	Detector	Volume	Occupancy
1997 1 12 12:30	1	50	3
1997 1 12 12:50	2	60	12
....			

(b) Three Basic Tables

Fig. 2. Detector-station relationship and basic tables.

surrounding stations can be represented as edges. The attribute space for each station is the traffic flow information (e.g., volume, occupancy) stored in the *value* table. We are interested in discovering the location of stations whose measurements are inconsistent with those of their graph-based spatial neighbors and the time periods when those abnormalities arise. The outlier detection tasks are to:

- Build a statistical model for a spatial data set
- Check whether a specific station is an outlier
- Check whether stations on a route are outliers

As shown in Fig. 3, we use three neighborhood definitions in this application. First, we define a neighborhood based on graph connectivity as a spatial neighborhood. For example, (s_1, t_2) and (s_3, t_2) are the spatial neighbors of (s_2, t_2) if s_1 and s_3 are connected to s_2 in a spatial graph. Second, we define a neighborhood based on time series as a temporal neighborhood. In Fig. 3, (s_2, t_1) and (s_2, t_3) are the temporal neighbors of (s_2, t_2) if t_1, t_2, and t_3 are consecutive time periods. In addition, we define a neighborhood based on both spatial graph and time series as a spatial-temporal neighborhood. In Fig. 3, (s_1, t_1), (s_1, t_2), (s_1, t_3), (s_2, t_1), (s_2, t_3), (s_3, t_1), (s_3, t_2), and (s_3, t_3) are the spatial-temporal neighbors of (s_2, t_2) if s_1 and s_3 are connected to s_2 in a spatial graph, and t_1, t_2, and t_3 are consecutive time periods.

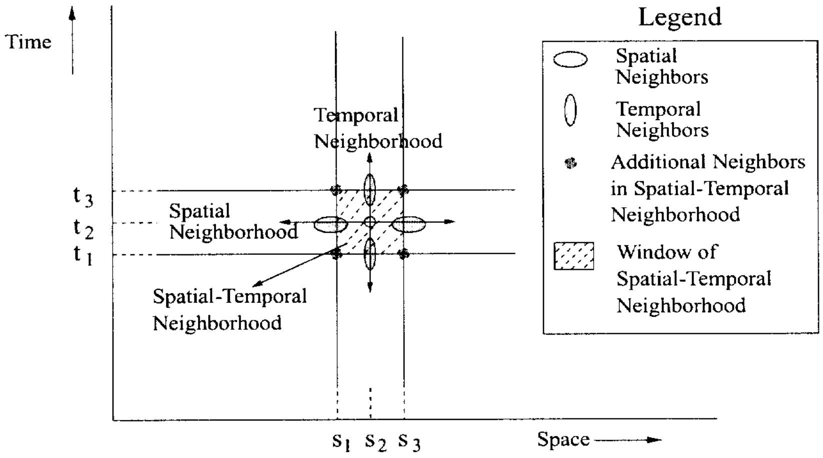

Time
Legend
Spatial Neighbors
Temporal Neighbors
Additional Neighbors in Spatial-Temporal Neighborhood
Window of Spatial-Temporal Neighborhood
Temporal Neighborhood
Spatial Neighborhood
Spatial-Temporal Neighborhood
t3
t2
t1
S1 S2 S3
Space

Fig. 3. Spatial and temporal outliers in traffic data.

1.2. Problem formulation

We formally define the spatial outlier detection problem. Given a spatial framework S for the underlying spatial graph G, an attribute function f over S, and neighborhood relationship R, we can build a model and construct statistical tests for spatial outliers based on a spatial graph according to a given confidence level threshold. This problem is formally defined as follows.

Spatial outlier detection problem

Given:

- A spatial graph $G = \{S, E\}$, where S is a spatial framework consisting of locations $s_1, s_2, \dots, s_n$ and E ($E \subseteq S \times S$) is a collection of edges between locations in S
- A neighborhood relationship $R \subseteq S \times S$ consistent with E
- An attribute function f: $S \rightarrow a\ set\ of\ real\ numbers$
- An aggregate function f_{aggr}: $R^N \rightarrow a\ set\ of\ real\ numbers$ to summarize values of attribute f over a neighborhood relationship $R^N \subseteq R$
- Confidence level threshold θ

Find: A set O of spatial outliers $O = \{s_i \mid s_i \in S, s_i\ is\ a\ spatial\ outlier\}$

Objective:

- Correctness: outliers identified by a method have significantly different
- attribute values from those of their neighborhood
- Efficiency: to minimize the computation time

Constraints:

- Attribute value for different locations in S is a normal distribution
- The size of the data set is much greater than the size of main memory
- The range of attribute function f is the set of real numbers

The formulation shows two subtasks in this spatial outlier detection problem: a) the design of a statistical model M and a test for spatial outliers b) the design of an efficient computation method to estimate the parameters of the test, test whether a specific spatial location is an outlier, and test whether spatial locations on a given path are outliers.

1.3. Paper scope and outline

This paper focuses on graph-based spatial outlier detection using a single attribute. Outlier detection in a multi-dimensional space using multiple attributes is beyond the scope of this paper.

The rest of the paper is organized as follows. Section 2 reviews related work. In Section 3, we propose our graph-based spatial outlier detection algorithm and discuss its computational complexity. The cost models for outlier query processing are analyzed in Section 4. Section 5 presents our experimental design. The experimental observation and results are shown in Section 6. We summarize our work in Section 7.

2. Related work and our contribution

Many outlier detection algorithms [1–3,8,9,12,14,16] have been recently proposed. These methods can be broadly classified into two categories, namely set-based outlier detection methods and spatial-set-based outlier detection methods, as shown in Fig. 4. The set-based outlier detection algorithms [2, 7] consider the statistical distribution of attribute values, ignoring the spatial relationships among items. Numerous outlier detection tests, known as discordancy tests [2,7], have been developed for different circumstances, depending on the data distribution, the number of expected outliers, and the types of expected outliers. The main idea is to fit the data set to a known standard distribution, and develop a test based on distribution properties. More than 100 discordancy tests [2] have been developed for normal, exponential, binomial, and Poisson distributions. In the cases that there is no test developed for a specified circumstance of given parameters or no standard distribution can model the observed distribution, other outlier detection methods should be used. For example, many multi-dimensional metric space-based methods detect outliers without considering the distribution of attribute value. We will introduce the multi-dimensional space methods in the following paragraph.

Spatial-set-based outlier detection methods consider both attribute values and spatial relationships. They can be further grouped into two categories, namely multi-dimensional metric space-based methods and graph-based methods. The multi-dimensional metric space-based methods model data sets as a collection of points in a multi-dimensional space, and provide tests based on concepts such as distance, density, and convex-hull depth. Knorr and Ng presented the notion of distance-based outliers [8,9]. For a k dimensional data set T with N objects, an object O in 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. Ramaswamy et al. [13] proposed a formulation for distance-based outliers based on the distance of a point to its k^{th} nearest neighbor. After ranking points by the distance to its k^{th} nearest neighbor, the top n points are declared as outliers. Breunig et al. [3] introduced the notion of a "local" outlier where the outlier-degree of an object is determined by taking into account the clustering structure in a bounded neighborhood of the object, e.g., k nearest neighbors. They formally defined the *outlier factor* to capture this relative degree of isolation or outlierness. In computational geometry, some depth-based approaches [12,14] organize data objects in convex hull layers in data space according to their peeling depth [12], and outliers are expected to be found from data objects with a shallow depth value. Conceptually, depth-based outlier detection methods are capable

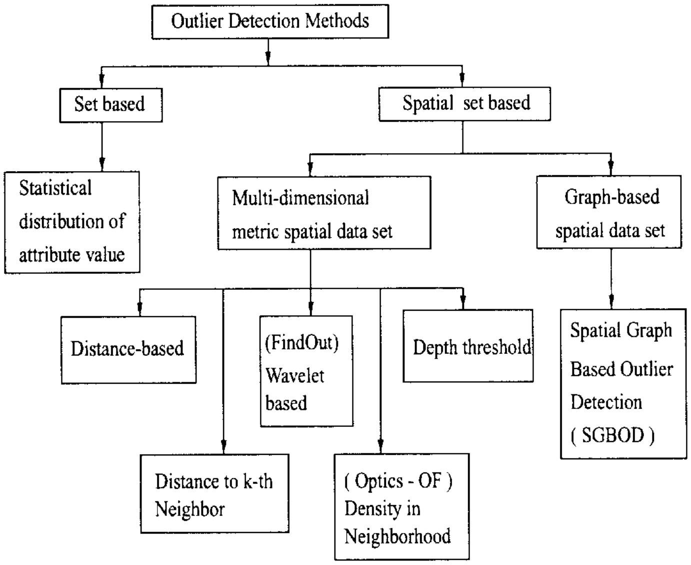

Outlier Detection Methods
Set based
Spatial set based
Statistical distribution of attribute value
Multi-dimensional metric spatial data set
Graph-based spatial data set
Distance-based
(FindOut) Wavelet based
Depth threshold
Spatial Graph Based Outlier Detection (SGBOD)
Distance to k-th Neighbor
(Optics - OF) Density in Neighborhood

Fig. 4. Classification of outlier detection methods.

of processing multidimensional datasets. However, with the best case computational complexity of $\Omega(N^{\lceil k/2 \rceil})$ for computing a convex hull, where N is the number of objects and k is the dimensionality of the dataset, depth-based outlier detection methods may not be applicable for high dimensional data sets. Yu et al. [16] introduced an outlier detection approach, called *FindOut*, which identifies outliers by removing clusters from the original data. Its key idea is to apply signal processing techniques to transform the space and find the dense regions in the transformed space. The remaining objects in the non-dense regions are labeled as outliers.

Methods for detecting outliers in multi-dimensional Euclidean space have some limitations. First, multi-dimensional approaches assume that the data items are embedded in an isometric metric space and do not capture the spatial graph structure. Consider the application domain of traffic data analysis. A multi-dimensional method may put a detector station in the neighborhood of another detector even if they are on opposite sides of the highway (e.g., I-35W north bound at exit 230, and I-35W south bound at exit 230), leading to the potentially incorrect identification of a bad detector. Secondly, multi-dimentional methods do not exploit apriori information about the statistical distribution of attribute data. Finally, they seldom provide a confidence measure of the discovered outliers.

In this paper, we formulate a general framework for detecting spatial outliers in a spatial data set with an underlying graph structure. We define neighborhood-based statistics and validate the statistical distribution. We then design a statistically correct test for discovering spatial outliers, and develop a fast algorithm to estimate model parameters, as well as to determine the results of a spatial outlier test on a given item. We also provide cost models for outlier detection procedures, and compare underlying data clustering methods that facilitate outlier query processing. In addition, we apply our proposed algorithm to detect spatial and temporal outliers in a Minneapolis-St.Paul traffic data set to show its the correctness and effectiveness.

3. Our approach: Spatial outlier detection algorithm

In this section, we list the key design decisions and propose an I/O efficient algorithm for spatial graph based outliers. The proposed algorithm contains two procedures. The first procedure computes the test parameters; the second procedure computers the test results, i.e., it verifies the outliers in an incoming data set.

3.1. Choice of spatial statistic

For spatial statistics, several parameters should be pre-determined before running the spatial outlier test. First, the neighborhood can be selected based on a fixed cardinality or a fixed graph distance or a fixed Euclidean distance. Second, the choice of neighborhood aggregate function can be mean, variance, or auto-correlation. Third, the choice for comparing a location with its neighbors can use either a number or a vector of attribute values. Finally, the statistic for base distribution can be selected from various choices.

The statistic we used is $S(x) = [f(x) - E_{y \in N(x)}(f(y))]$, where $f(x)$ is the attribute value for a data record x, $N(x)$ is the fixed cardinality set of neighbors of x, and $E_{y \in N(x)}(f(y))$ is the average attribute value for neighbors of x. Statistic $S(x)$ denotes the difference of the attribute value of each data object x and the average attribute value of $x's$ neighbors.

3.2. Characterizing the distribution of the statistic

Lemma 1. *Spatial Statistic $S(x) = [f(x) - E_{y \in N(x)}(f(y))]$ is normally distributed if attribute value $f(x)$ is normally distributed.*

Proof:

Given the definition of neighborhood, for each data record x, the average attribute values $E_{y \in N(x)}(f(y))$ of $x's$ k neighbors can be calculated. Since attribute value $f(x)$ is normally distributed and an average of normal variables is also normally distributed, the average attribute value $E_{y \in N(x)}(f(y))$ over neighbors is also a normal distribution for a fixed cardinality neighborhood. Since the attribute value $f(x)$ and the average attribute value $E_{y \in N(x)}(f(y))$ over neighbors are two normal variables, the distribution of the difference function $S(x)$ is also normally distributed.

3.3. Test for outlier detection

The test for detecting a spatial outlier can be described as $|\frac{S(x)-\mu_s}{\sigma_s}| > \theta$. For each data object x with an attribute value $f(x)$, the $S(x)$ is the difference of the attribute value of data object x and the average attribute value of its neighbors, μ_s and σ_s are the mean and standard deviation of all $S(x)$. The choice of θ depends on the specified confidence interval. For example, a confidence interval of 95 percent will lead to $\theta \approx 2$.

3.4. Computation of test parameters

We now propose an I/O efficient procedure, Test Parameters Computation (TPC), to calculate the test parameters, e.g., mean and standard deviation for the statistics, as shown in Algorithm 1. The computed mean and standard deviation can then be used to detect the outlier in the incoming data set.

Given an attribute data set V and the connectivity graph G, the TPC procedure first retrieves the neighbor nodes from G for each data object x. It then computes the difference of the attribute value of x and the average of the attribute values of $x's$ neighbor nodes. These different values are then stored as a set in the AvgDist_Set. Finally, the AvgDist_Set is used to get the distribution value μ_s and σ_s. Note that the data objects are processed on a page basis to reduce redundant I/O.

Test Parameters Computation (TPC) procedure

Algorithm 1. Pseudo-code for test parameters computation

```
Input: $S$ is the multidimensional attribute space;
    $D$ is the attribute data set in $S$;
    $F$ is the distance function in $S$;
    $ND$ is the depth of neighbor;
    $G = (D, E)$ is the spatial graph;

Output: ($\mu_s$,$\sigma_s$).
for(i=1;i $\leq |D|$ ;i++){
    $O_i$=Get_One_Object(i,D); /* Select each object from $D$ */
    NNS=Find_Neighbor_Nodes_Set($O_i$,$ND$,$G$);
   /* Find neighbor nodes of $O_i$ from $G$ */
    Accum_Dist=0;
    for(j=1;j$\leq$ |NSS|;j++){
        $O_k$=Get_One_Object(j,NNS); /* Select each object from $NNS$ */
        Accum_Dist += $F(O_i, O_k, S)$
    }
    Avg_Dist = Accum_Dist / |NNS|;
    Add_Element(AvgDist_Set,i); /* Add the element to AvgDist_Set */
}
$\mu_s$ = Get_Mean(AvgDist_Set); /* Compute Mean */
$\sigma_s$ = Get_Standard_Dev(AvgDist_Set); /* Compute Standard Deviation */
return ($\mu_s$,$\sigma_s$).
```

3.5. Computation of test results

The neighborhood aggregate statistics value, e.g., mean and standard deviation, computed in the TPC procedure can be used to verify the outliers in an incoming data set. The two verification procedures to computer test results are Route Outlier Detection(ROD) and Random Node Verification(RNV). The ROD procedure detects the spatial outliers from a user specified route, as shown in Algorithm 2. The RNV procedure checks the outlierness from a set of randomly generated nodes. Given route RN in the data set D with graph structure G, the ROD procedure first retrieves the neighboring nodes from G for each data object x in the route RN, then it computes the difference $S(x)$ between the attribute value of x and the average of attribute values of $x's$ neighboring nodes. Each $S(x)$ can then be verified using the spatial outlier detection test $|\frac{S(x)-\mu_s}{\sigma_s}| > \theta$. The θ is predetermined by the given confidence interval. The steps to detect outliers in both ROD and RNV are similar, except that the RNV has no shared data access needs across tests for different nodes. The I/O operations for Find_Neighbor_Nodes_Set() in different iterations are independent of each other in RNV. We note that the operation Find_Neighbor_Nodes_Set()

is executed once in each iteration and dominates the I/O cost of the entire procedure. The I/O cost of Find_Neighbor_Nodes_Set() is determined by the Connectivity Residue Ratio (CRR), that is, how the nodes are grouped into disk pages. We will discuss the CRR measure in the following section. If the node and all of its neighbor nodes can be arranged in the same disk page, there will be no redundant I/O operation required. The storage of the data set should support the efficient I/O computation of this operation. We discuss the choice for storage structure and provide an experimental comparison in Sections 5 and 6.

Route Outlier Detection (ROD) procedure

Algorithm 2. Pseudo-code for route outlier detection

```
Input: $S$ is the multidimensional attribute space;
    $D$ is the attribute data set in $S$;
    $F$ is the distance function in $S$;
    $ND$ is the depth of neighbor;
    $G = (D, E)$ is the spatial graph;
    $CI$ is the confidence interval;
    ($\mu_s$,$\sigma_s$) are the mean and standard deviation calculated in TPC;
    $RN$ is the set of node in a route;

Output: Outlier_Set.
for(i=1;i $\leq$ $|RN|$ ;i++){
    $O_i$=Get_One_Object(i,D); /* Select each object from D */
    NNS=Find_Neighbor_Nodes_Set($O_i$,$ND$,$G$);
   /* Find neighbor nodes of $O_i$ from G */
    $Accum\_Dist$=0;
    for(j=1;j$\leq$ |NSS|;j++){
       $O_k$=Get_One_Object(j,NNS); /* Select each object from $NNS$ */
       Accum_Dist += $F(O_i, O_k, S)$
    }
    AvgDist = Accum_Dist/|NNS|;
    $T_{value}$=$\frac{AvgDist-\mu_s}{\sigma_s}$
    /*Check the normal distribution table */
    if( Check_Normal_Table($T_{value}$,$CI$)== True){
       Add_Element(Outlier_Set,i); /* Add the element to Outlier_Set */
    }
}
return Outlier_Set.
```

4. Analytical evaluation and cost models

In this section, we provide simple algebraic cost models for the I/O cost of outlier detection procedures, using the Connectivity Residue Ratio(CRR) measure of physical page clustering methods. The CRR value is defined as follows.

$$\text{CRR} = \frac{\text{Total number of unsplit edges}}{\text{Total number of edges}}$$

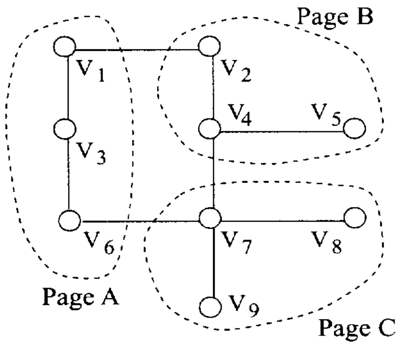


Fig. 5. Example of CRR.

Table 1
Symbols used in cost analysis

Symbol	Meaning
α	The CRR value
β	Average blocking factor
N	Total number of nodes
L	Number of nodes in a route
R	Number of nodes in a random set
Λ	Average number of neighbors for each node

The CRR value is determined by the disk page clustering method, the data record size, and the page size. Figure 5 gives an example of CRR value calculation. The blocking factor, i.e., the number of data records within a disk page, is three, and there are nine data records. The data records are clustered into three pages. There are a total of nine edges and six unsplit edges. The CRR value of this graph can be calculated as $6/9 = 0.66$.

Table 1 lists the symbols used to develop our cost formulas. α is the CRR value. β denotes the blocking factor, which is the number of data records that can be stored in one disk page. Λ is the average number of nodes in the neighbor list of a node. N is the total number of nodes in the data set, L is the number of nodes along a route, and R is the number of nodes randomly generated by users for spatial outlier verification.

4.1. Cost modeling for the test parameters computation (TPC) *procedure*

The TPC algorithm is a nest loop index join. Suppose that we use two memory buffers. If one memory buffer stores the data object x used in the outer loop and the other memory buffer is reserved for processing the neighbors of x, we get the following cost function to estimate the number of page accesses.

$$C_{\text{TPC}} = \frac{N}{\beta} + N * \Lambda * (1 - \alpha)$$

The outer loop retrieves all the data records on a page basis and has an aggregated cost of $\frac{N}{\beta}$. For each node x, on average, $\alpha * \Lambda$ neighbors are in the same page as x, and can be processed without redundant I/O. Additional data page accesses are needed to retrieve the other $(1 - \alpha) * \Lambda$ neighbors, and it takes at most $(1 - \alpha) * \Lambda$ data page accesses. Thus the aggregated cost for the inner loop is $N * \Lambda * (1 - \alpha)$.

4.2. *Cost modeling for the route outlier detection* (ROD) *procedure*

We get the following cost function to estimate the number of page accesses with two memory buffers for the ROD procedure. One memory buffer is reserved for processing the node x to be verified, and the other is used to process the neighbors of x.

$$C_{ROD} = L * (1 - \alpha) + L * \Lambda * (1 - \alpha) = L * (1 - \alpha) * (1 + \Lambda)$$

For each node x, on average, its successor node y is on the same page as x with probability α, and can be processed with no redundant page accesses. The cost to access all the nodes along a route is $L * (1 - \alpha)$. To process the neighbors of each node, $\alpha * \Lambda$ neighbors are on the same page as x. Additional data page accesses are needed to retrieve the other $(1 - \alpha) * \Lambda$ neighbors, and it takes at most $(1 - \alpha) * \Lambda$ data page accesses.

4.3. *Cost modeling for the random node verification* (RNV) *procedure*

Assume two memory buffers are used, one memory buffer is reserved for processing the node x to be verified, the other is used to process the neighbors of x. The following cost function can be applied to estimate the number of page accesses for the RNV procedure.

$$C_{RNV} = R + R * \Lambda * (1 - \alpha)$$

Since the memory buffer is cleared for each consecutive random node, we need R page accesses to process all these random nodes. For each node x, $\alpha * \Lambda$ neighbors are on the same page as x, and can be processed without extra I/O. Additional data page accesses are needed to retrieve the other $(1 - \alpha) * \Lambda$ neighbors, and it takes at most $(1 - \alpha) * \Lambda$ data page accesses. Thus, the aggregated cost to process the neighbor of R nodes is $R * \Lambda * (1 - \alpha)$.

5. Experiment design

As discussed in the cost model analysis, the I/O cost for each outlier detection procedure depends on the the CRR value α, which is determined by the data clustering method. In this section, we describe the layout of our experiments and then illustrate the candidate data clustering methods.

5.1. *Experimental layout*

The design of our experiments is shown in Fig. 6. The input data set, Twin Cities Highway Connectivity Graph, is provided by the Minnesota Department of Transportation and physically stored into disk pages using various clustering strategies and different page sizes. These data pages were then processed to generate the global distribution or sampling distribution, depending on the size of the data sets.

We compared different data page clustering schemes: Connectivity-Clustered Access Method(CCAM) [15], Z-ordering [10], and Cell-tree [5]. Other parameters of interest were the size of the memory buffer, the memory block size (page size), the number of neighbors, and the neighborhood depth. The measures of our experiments were the CRR value and I/O cost for each outlier detection procedure.

The experiments were conducted on many graphs. We present the results on a representative graph, which is a spatial network with 990 nodes that represents the traffic detector stations for a 20-square-mile section of the Twin Cities area. We used a common record type for all the clustering methods. Each record contains a node and its neighbor-list, i.e., a successor-list and a predecessor-list. We also conducted performance comparisons of the I/O cost for outlier-detection query processing.

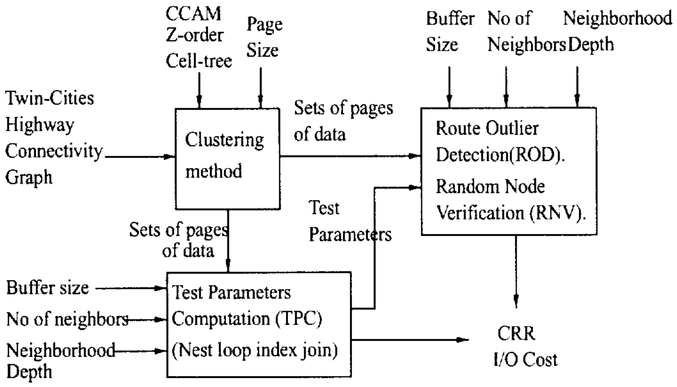


Fig. 6. Experimental layout.

5.2. Candidate clustering methods

In this section we describe the candidate clustering methods used in the experiments.

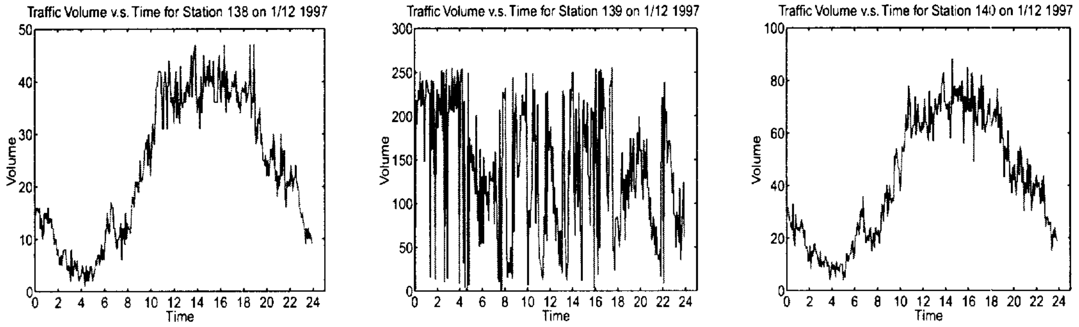
Connectivity-Clustered Access Method(CCAM): CCAM [15] clusters the nodes of the graph via graph partitioning, e.g., Metis. Other graph-partitioning methods can also be used as the basis of our scheme. In addition, an auxiliary secondary index is used to support query operations. The choice of a secondary index can be tailored to the application. Since the benchmark graph was embedded in graphical space, We used the B^+ tree with Z-order in our experiments. Other access methods such as the R-tree and Grid File can alternatively be created on top of the data file as secondary indices in CCAM to suit the application.

Linear Clustering by Z-order: Z-order [10] utilizes spatial information while imposing a total order on the points. The Z-order of a coordinate (x,y) is computed by interweaving the bits in the binary representation of the two values. Alternatively, Hilbert ordering may be used. A conventional one-dimensional primary index (e.g. B^+-tree) can be used to facilitate the search.

Cell Tree: A cell tree [5] is a height-balanced tree. Each cell tree node corresponds not necessarily to a rectangular box but to a convex polyhedron. A cell tree restricts polyhedra to partitions of a BSP(Binary Space Partitioning) in order to avoid overlaps among sibling polyhedra. Each cell tree node corresponds to one disk space, and the leaf nodes contain all the information required to answer a given search query. The cell tree can be viewed as a combination of a BSP- and R^+-tree, or as a BSP-tree mapped on paged secondary memory.

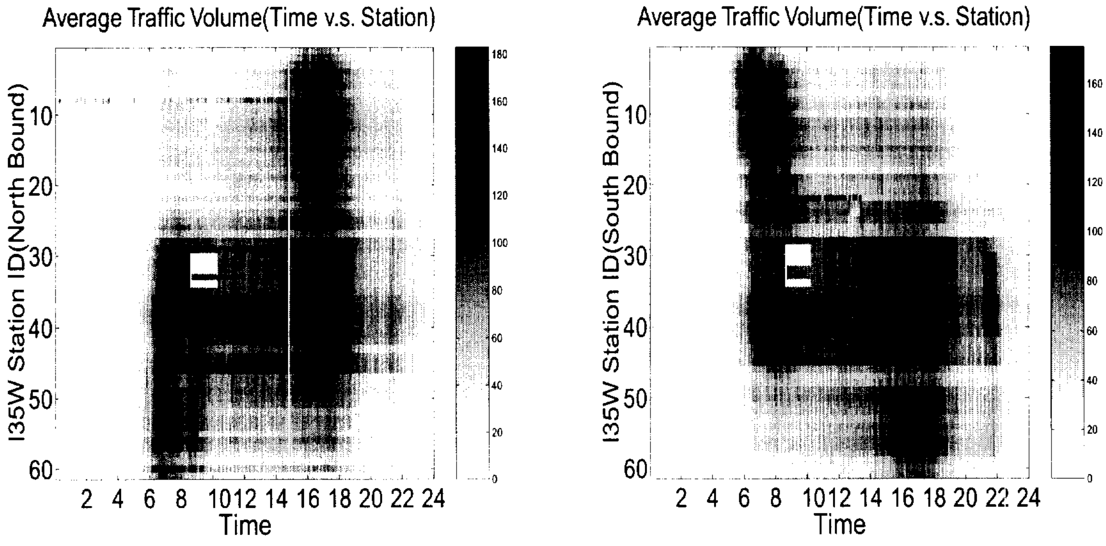
6. Experimental results

In this section, we illustrate the outlier examples detected in the traffic data set, present the results of our experiments, and test the effectiveness of the different data clustering methods. To simplify the comparison, the I/O cost represents the number of data pages accessed. This represents the relative performance of the various methods for very large databases. For smaller databases, the I/O cost associated with the indices should be measured. Here we present the evaluation of I/O cost for the TPC algorithm.



(a) Station 138 (b) Outlier Station 139 (c) Station 140

Fig. 7. Outlier station 139 and its neighbor stations on 1/12/1997.



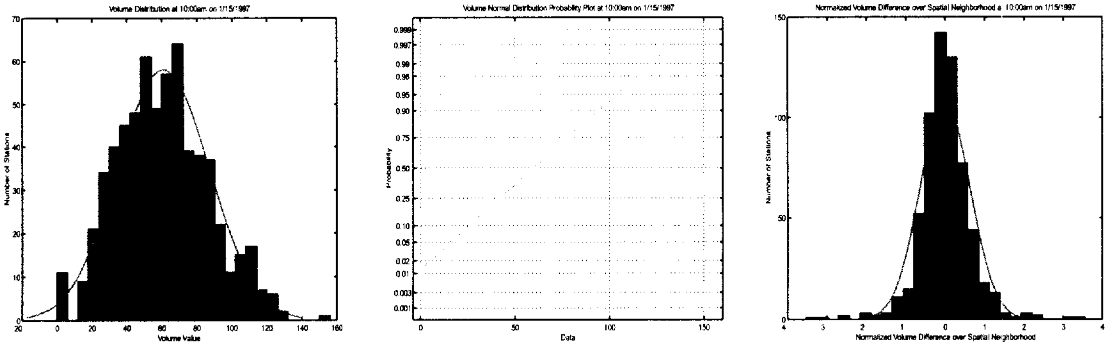
(a) I-35W North Bound (b) I-35W South Bound

Fig. 8. An example of outliers.

6.1. Outliers detected

We tested the effectiveness of our algorithm on the Twin Cities traffic data set and detected numerous outliers. In Fig. 7, the abnormal station (Station 139) was detected with volume values significantly inconsistent with the volume values of its neighboring stations 138 and 140. Note that our basic algorithm detects outlier stations in each time slot; the detected outlier stations in each time slot are then aggregated on a daily basis.

Figure 8 shows another example of traffic flow outliers. Figures 8(a) and (b) are the traffic volume maps for I-35W North Bound and South Bound, respectively, on 1/21/1997. The X-axis is a 5-minute



(a) Histogram of Traffic Volume Distribution

(b) Normal Probability Plot for Traffic Volume Distribution

(c) Histogram of Volume Difference over Neighborhood

Fig. 9. Verification of normal distribution for traffic volumes and volume difference over neighbors.

time slot for the whole day and the Y-axis is the label of the stations installed on the highway, starting from 1 on the north end to 61 on the south end. The abnormal white line at 2:45 pm and the white rectangle from 8:20 am to 10:00 am on the X-axis and between stations 29 to 34 on the Y-axis can be easily observed from both (a) and (b). The white line at 2:45 pm is an instance of temporal outliers, where the white rectangle is a spatial-temporal outlier. Moreover, station 9 in Fig. 8(a) exhibits inconsistent traffic flow compared with its neighboring stations, and was detected as a spatial outlier.

6.2. *Testing the statistical assumption*

In this traffic data set, the volume values of all stations at one moment are approximately a normal distribution. The histogram of stations with different traffic volumes are shown in Fig. 9(a) with a normal probability distribution superimposed. As can be seen, the normal distribution approximates the volume distribution very well. We calculated the interval of $[\mu - \sigma, \mu + \sigma]$, $[\mu - 2\sigma, \mu + 2\sigma]$, and $[\mu - 3\sigma, \mu + 3\sigma]$ where μ and σ are the mean and standard deviation of the volume distribution, and the percentages of measurements falling in the three intervals are equal to 68.27%, 95.45%, and 99.73%, respectively. This pattern fits well with a normal distribution since the expected values in a normal distribution are 68%, 95%, and 100%. Moreover, we plot the normal probability plot in Fig. 9(b), and it appears linear. Hence the volume values of all stations at the same time are approximately a normal distribution.

Given the definition of neighborhood, we then calculate the average volume value (μ_k) around its k neighbors according to the topological relationship for each station. Since the volume values are normally distributed, the average of the normal variables is also a normal distribution.

Since the volume value and the average volume value over neighborhoods are normally distributed, the difference between them is also normal distributed, as shown in Fig. 9(c). Given the confidence level $100(1 - \alpha)\%$, we can calculate the confidence interval for the difference distributions, i.e., the difference value distribution lies between the $-z_{\alpha/2}$ and $z_{\alpha/2}$ standard deviation of the mean. The spatial outiers are detected based on this confidence level threshold.

Table 2
The actual I/O cost and predicted cost model for different clustering methods

Clustering	Parameters computation		Random node verification		Route outlier detect		α = CRR
Method	Actual	Predicted	Actual	Predicted	Actual	Predicted	
CCAM	628	687	241	246	30	36	0.68
Cell-tree	834	919	279	291	45	53	0.53
Z-order	1263	1269	349	357	78	79	0.31
$N = 773$, $L = 38$, $R = 150$, $\beta = 4$, $\Lambda = 2$							

6.3. *Evaluation of proposed cost model*

We evaluated the I/O cost for different clustering methods for outlier detection procedures, namely, Test Parameters Computation (TPC), Route Outlier Detection (ROD) and Random Node Verification (RNV). The experiments used Twin-Cities traffic data with page size 1 Kbytes, and two memory buffers. Table 2 shows the number of data page accesses for each procedure under various clustering methods. The CRR value for each method is also listed in the table. The cost function for TPC is $C_{TPC} = \frac{N}{\beta} + N * \Lambda * (1 - \alpha)$. The cost function for RNV is $C_{RNV} = R + R * \Lambda * (1 - \alpha)$. The cost function for ROD is $C_{ROD} = L * (1 - \alpha) * (1 + \Lambda)$, as described in Section 4.

As shown in Table 2, CCAM produced the lowest number of data page accesses for the outlier detection procedures. This is to be expected, since CCAM generated the highest CRR value.

6.4. *Evaluation of I/O cost for* TPC *algorithm*

In this section, we present the results of our evaluation of the I/O cost and CRR value for alternative clustering methods while computing the test parameters. The parameters of interest are page size, number of neighbors, and neighborhood depth.

6.4.1. *The effect of page size and CRR value*

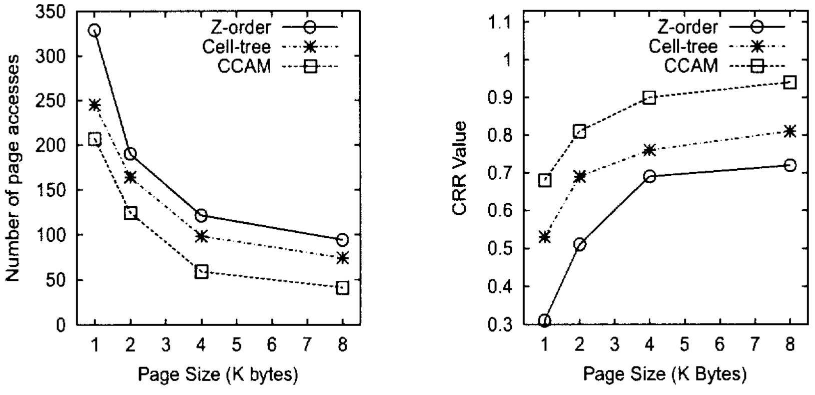
Figures 10 (a) and (b) show the number of data pages accessed and the CRR values respectively, for different page clustering methods as the page sizes change. The buffer size is fixed at 32 Kbytes. As can be seen, a higher CRR value implies a lower number of data page accesses, as predicted in the cost model. CCAM outperforms the other competitors for all four page sizes, and Cell-tree has better performance than Z-order clustering.

6.4.2. *The effect of neighborhood cardinality*

We evaluated the effect of varying the number of neighbors and the depth of neighbors for different page clustering methods. We fixed the page size at one Kbyte, the buffer size at four Kbytes. Figure 11 shows the number of page accesses as the number of neighbors for each node increases from 2 to 10. CCAM has better performance than Z-order and Cell-tree. The performance ranking for each page clustering method remains the same for different numbers of neighbors. Figure 11 shows the number of page accesses as the neighborhood depth increases from 1 to 5. CCAM has better performance than Z-order and Cell-tree for all the neighborhood depths.

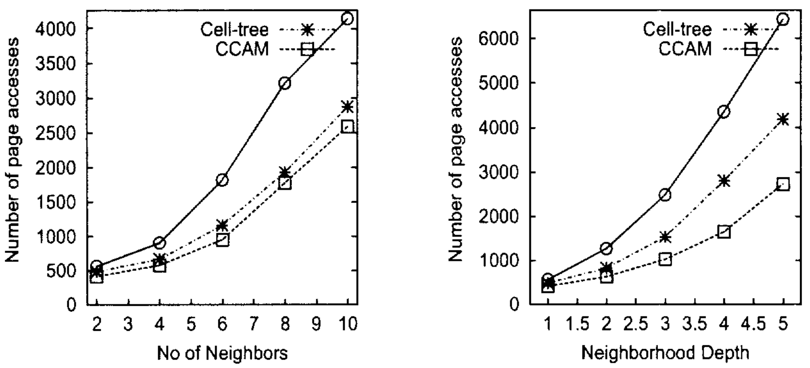
7. Conclusions

In this paper, we formulated a general framework for detecting spatial outliers in a spatial data set with an underlying graph structure. We proposed the notion of a spatial outlier in graph structured data sets,



(a) Page Accesses (b) CRR

Fig. 10. Effect of page size on data page accesses and CRR (buffer Size = 32 Kbytes).



(a) Number of Neighbors (b) Neighborhood Depth

Fig. 11. Effect of neighborhood cardinality on data page accesses (page size = 1 Kbyte, buffer size = 4 Kbytes).

designed a statistically correct test for discovering spatial outliers, and developed an efficient algorithm to estimate model parameters, as well as to determine the results of a spatial outlier test on a given item set. Since the Find_Neighbor_Nodes_Set() operation dominates the I/O cost of the entire algorithm and the I/O cost of Find_Neighbor_Nodes_Set() is determined by the Connectivity Residue Ration (CRR), i.e., how the nodes are grouped into disk pages, we analyzed the cost models for the three outlier detection procedures using CRR value and other given parameters. We also provided experimental results from the application of our algorithm on a Twin Cities traffic archival to show its effectiveness and usefulness.

We have evaluated alternative data clustering methods for neighbor outlier query processing, including test parameter computation procedure, random node verification procedure, and route outlier detection procedure. Our experimental results show that the CRR is a good predicator of relative I/O performance, i.e., higher CRR leads to lower I/O cost, and that the connectivity-clustered access method (CCAM), which achieves the highest CRR value for most cases, provides the best overall performance than other access methods, such as Z-order and Cell-tree.

Our current algorithm is designed to detect spatial outliers using a single non-spatial attribute from a data set. In the future, we are planning to investigate spatial outliers with multiple non-spatial attributes, such as the combination of volume, occupancy, and speed in the traffic data set. For multiple attributes, the definition of spatial neighborhood will be the same, but the neighborhood aggregate function, comparison function, and statistic test function need to be redefined. The key challenge is to define a general distance function in a multi-attribute data space.

8. Acknowledgment

We are particularly grateful to Professor Vipin Kumar, and our Spatial Database Group members, Weili Wu, Yan Huang, Xiaobin Ma,and Hui Xiong for their helpful comments and valuable discussions. We would also like to express our thanks to Kim Koffolt for improving the readability and technical accuracy of this paper.

This work is supported in part by USDOT grant on high performance spatial visualization of traffic data, and is sponsored in part by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAH04-95-2-0003/contract number DAAH04-95-C-0008, the content of which does not necessarily reflect the position or the policy of the government, and no official endorsement should be inferred. This work was also supported in part by NSF grant #9631539.

References

[1] M. Ankerst, M.M. Breunig, H.P. Kriegel and J. Sander, Optics: Ordering points to identify the clustering structure, in: *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data,* Philadephia, Pennsylvania, USA, 1999, pp. 49–60.

[2] V. Barnett and T. Lewis, *Outliers in Statistical Data,* (3rd ed.), John Wiley, NewYork, 1994.

[3] M.M. Breunig, H.P. Kriegel, R.T. Ng and J. Sander, Optics-of: Identifying local outliers, in: *Proc. of PKDD '99, Prague, Czech Republic, Lecture Notes in Computer Science (LNAI 1704),* Springer Verlag, 1999, pp. 262–270.

[4] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth and R. Uthurusamy, *Advances in Knowledge Discovery and Data Mining,* MIT Press, Cambridge, MA, 1996.

[5] O. Gunther, The Design of the Cell Tree: An Object-Oriented Index Structure for Geometric Databases, in: *Proc. 5th Intl. Conference on Data Engineering,* Feb. 1989.

[6] D. Hawkins, *Identification of Outliers,* Chapman and Hall, 1980.

[7] R. Johnson, *Applied Multivariate Statistical Analysis,* Prentice Hall, 1992.

[8] E. Knorr and R. Ng, A unified notion of outliers: Properties and computation, in: *Proc. of the International Conference on Knowledge Discovery and Data Mining,* 1997, pp. 219–222.

[9] E. Knorr and R. Ng, Algorithms for mining distance-based outliers in large datasets, in: *Proc. 24th VLDB Conference,* 1998.

[10] A. Orenstein and T. Merrett, A Class of Data Structures for Associative Searching, in: *Proc. Symp. on Principles of Database Systens,* 1984, pp. 181–190.

[11] G. Piatetsky-Shapiro and W.J. Frawley, *Knowledge Discovery in Databases,* AAAI/MIT Press, 1991.

[12] F. Preparata and M. Shamos, *Computatinal Geometry: An Introduction,* Springer Verlag, 1988.

[13] S. Ramaswamy, R. Rastongi and K. Shim, Ecient algorithms for mining outliers from large data sets, in: *Bell Laboratories,* Murray Hill, NJ.

[14] I. Ruts and P. Rousseeuw, Computing depth contours of bivariate point clouds, *Computational Statistics and Data Analysis* **23** (1996), 153–168.

[15] S. Shekhar and D.-R. Liu, Ccam: A connectivity-clustered access method for aggregate queries on transportation networks – a summary of results, *IEEE Trans. on Knowledge and Data Eng.* **9**(1) (January 1997).

[16] D. Yu, G. Sheikholeslami and A. Zhang, Findout: Finding outliers in very large datasets, in: *Department of Computer Science and Engineering State University of New York at Buffalo Buffalo, Technical report 99-03*, http://www.cse.buffalo.edu/tech-reports/, 1999.