Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 01-014

Detecting Graph-based Spatial Outliers: Algorithms and Applications

Shashi Shekhar, Chang-tien Lu, and Pusheng Zhang

March 08, 2001

Detecting Graph-based Spatial Outliers: Algorithms and Applications [*]

S. Shekhar, C. T. Lu, P. Zhang
Computer Science Department, University of Minnesota
200 Union Street SE, Minneapolis, MN-55455
[*shekhar, ctlu, pusheng*]@cs.umn.edu TEL:(612) 6248307 FAX:(612)6250572
http://www.cs.umn.edu/Research/shashi-group

March 8, 2001

Abstract

Identification of outliers can lead to the discovery of unexpected, interesting, and implicit 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 tests for spatial outliers in graph structured data sets, analyze the statistical foundation underlying our approach, design a fast algorithm to detect spatial outliers, provide the cost model for outlier detection procedures. In addition, we provide experimental results from the application of our algorithm on a Minneapolis-St. Paul(Twin-Cities) traffic dataset to show its effectiveness and usefulness.

Keywords: Outlier Detection, Spatial Data mining, Spatial Graphs

[*] 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.

1 Introduction

Data mining is a process to extract implicit, nontrivial, previously unknown and potentially useful information(such as knowledge rules, constraints, regularities) from data in databases [12, 4]. 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 huge amount of spatial data obtained from satellite images, medical images, GIS, etc., it is a non-trivial task for humans to explore spatial data in detail. Spatial data sets and patterns are abundant in many application domains related to NASA, National Imagery and Mapping Agency(NIMA), National Cancer Institute(NCI), and the Unite States Department of Transportation(USDOT).

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 noise, errors, exceptions, or deviation. Outliers have been informally defined as observations which appear to be inconsistent with the remainder of that set of data [2], or deviate so much from other observations as to arouse suspicions that it was 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, athlete performance analysis, voting irregularity, bankruptcy, and weather prediction.

Outliers in a spatial data set can be classified into three categories: set-based outliers, multi-dimension space-based outliers, and graph-based outliers. A set-based outlier is a data object whose attributes are inconsistent with attribute values of other objects in a given data set regardless of spatial relationships. Both multi-dimension space-based outliers and graph-based outliers are called spatial outliers, that is, data objects that are significantly different in the attribute space from the collection of data objects among spatial neighborhoods, however, multi-dimension space-based outliers and graph-based outliers are based on different spatial neighborhood definitions. In multi-dimension 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 some reasons. First, the choice of a neighborhood is non-trivial. Second,the design of statistical tests for the spatial outliers need to account for the distribution of the attribute values at various locations as well as the distribution of aggregate function 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

In this paper, we formulate a general framework for detecting outliers in spatial graph data sets, and propose an efficient graph-based outlier detection algorithm. We provide cost models for outlier detection queries, and compare underlying data storage and clustering methods that will facilitate outlier query processing. We also use our basic algorithm to detect spatial and temporal outliers in a Minneapolis-St. Paul(Twin-Cities) traffic data set, and show the correctness and effectiveness of our approach.

1.1 An Illustrative Application Domain: Traffic Data Set

In 1995, the University of Minnesota and the Traffic Management Center(TMC) 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 and interstate 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, as well as research on traffic modeling and experiments. Figure 1 shows a map of the stations on the highways within the Twin-Cities metropolitan area, where each polygon represents one station. The interstate freeways include I-35W, I35E, I-94, I-394, I-494, and I-694. The 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 South-North 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.

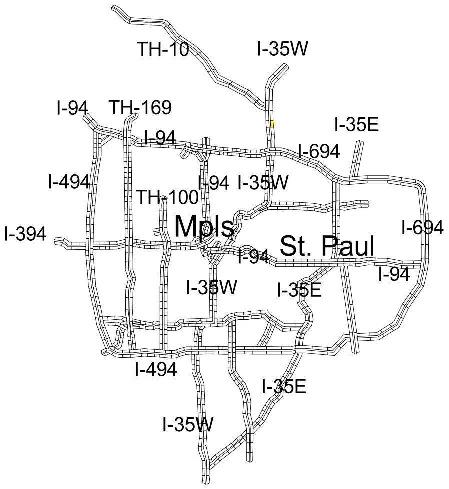
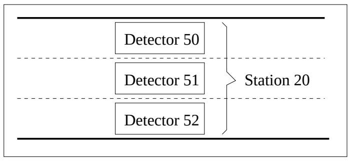


Figure 1: Detector map in station level

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.



(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

Figure 2: Detector-station Relationship and Basic Tables

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 surrounding stations can be represented as an edge. The attribute space for each station is the traffic flow information (e.g., volume, occupancy) stored in the *value* table.

In this application, we are interested in discovering the location of stations whose measurements are inconsistent with those of their graph-based spatial neighbors and time periods when those abnormalities arise. This outlier detection task is

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

We use three neighborhood definitions in this application as shown in Figure 3. First, we define a neighborhood based on the spatial graph connectivity as a spatial graph neighborhood. In Figure 3, (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 Figure 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 slots. In addition, we define a neighborhood based on both space and time series as a spatial-temporal neighborhood. In Figure 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 slots.

1.2 Problem Formulation

In this section, we formally define the spatial outlier detection problem. Given a spatial framework S for the underlying spatial graph G, an attribute f over S, and neighborhood relationship R, we can build a model and statistical tests for spatial outliers based on a spatial graph according to the given confidence level threshold. The 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 $s_1, s_2, \ldots, s_n$ and E ($E \subseteq S \times S$) is a collection of edges between locations in S.

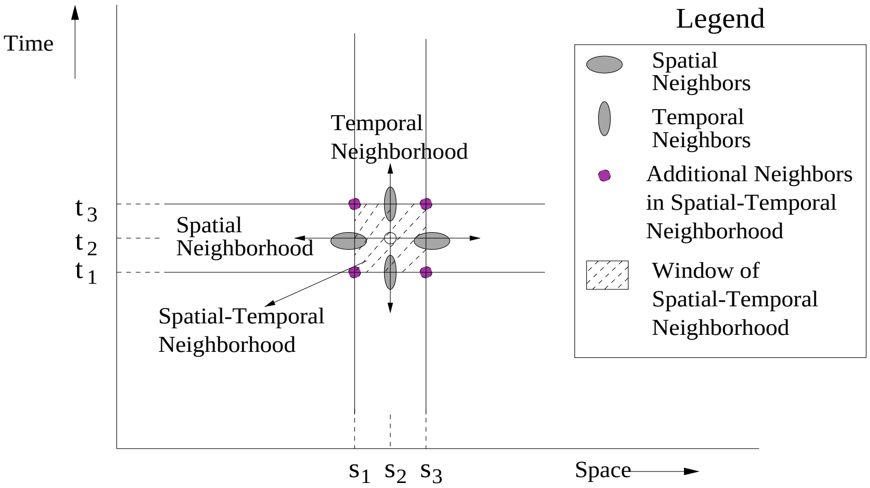


Figure 3: Spatial and Temporal outlier in traffic data

- 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 with those of their neighborhood
- Efficiency: to minimize the computation time

Constraints:

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

The formulation shows two subtasks in this spatial outlier detection problem: (a) design a statistical model M and test for spatial outliers (b) design an efficient computation method to estimate parameters of 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 the graph-based spatial outlier detection using a single attribute. Outlier detection in the multi-dimension space using multiple attributes is outside the scope of this paper.

The rest of the paper is organized as follows. Section 2 reviews related work and discusses our contributions. In Section 3, we propose our graph-based spatial outlier detection algorithm and discuss its computational complexity. The cost models for different outlier query processing are analyzed in Section 4. Section 5 presents our experiment 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, 2, 3, 8, 9, 13, 15, 17] have been recently proposed. As shown in Figure 4, these methods can be broadly classified into two categories, namely set-based outlier detection methods and spatial-set-based outlier detection methods. 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.

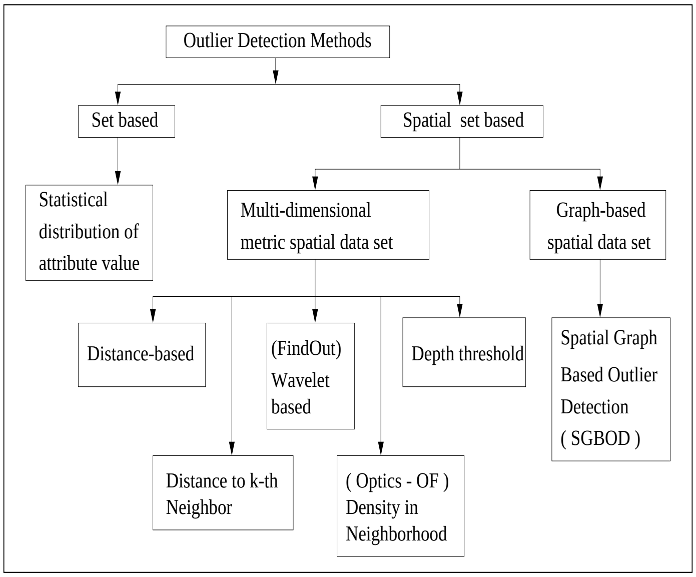

Outlier Detection Methods
Set based
Spatial set based
Statistical distribution of attribute value
Multi-dimensional metric spatial data set
Graph-based spatial data set
Distance-based
(FindOut) Wavelet based
Depth threshold
Spatial Graph Based Outlier Detection (SGBOD)
Distance to k-th Neighbor
(Optics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

Figure 4: Classification of outlier detection methods

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 collection of points in a multidimensional space, and provide tests based on concepts such as distance, density, convex-hull depth. We discuss different example tests now. 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. [14] proposed a formulation for distance-based outliers based on the distance of a point from 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 that 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. Their notions of outliers are based on the same theoretical foundation as density-based cluster analysis [1]. In computational geometry, some depth-based approaches [15, 13] organize data objects in convex hull layers in data space according to peeling depth [13], and outliers are expected to be found from data objects with shallow depth value. Conceptually, depth-based outlier detection methods are capable of processing multidimensional datasets. However, with the best case computational complexity $\Omega(N^{\lceil k/2 \rceil})$ 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. [17] 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.

Multi-dimensional Euclidean spatial based methods detect outliers in multidimensional data space. These approaches have some limitation. First, the multi-dimensional approaches assume that the data items are embedded in a 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 were on opposite sides of highway (e.g., I-35W north bound at exit 230, and I-35W south bound at exit 230), leading to potentially incorrect identification of bad detector. Secondly, they do not exploit apriori information about the statistical distribution of attribute data. Last, they seldom provide the 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 a neighborhood-based statistic and validate the statistical distribution. We 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 spatial outlier test on a given item. In addition, we evaluate our method in Twin-Cities traffic data set and show the effectiveness and usefulness of our approach.

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.

3.1 Choice of Spatial Statistic

For spatial statistics, several parameters should be pre-determined before running the spatial outlier test. First, the choice of neighborhood, 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, e.g., mean, variance, and auto-correlation. Third, the choice for comparing a location with its neighbors, either using just a number or a vector of attribute values. Finally, the choice of statistic.

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 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 values $f(x)$ are normally distributed and an average of normal variables is also normally distributed, the average attribute values $E_{y \in N(x)}(f(y))$ over neighborers is also a normal distribution for a fixed cardinality neighborhood.

Since the attribute value and the average attribute value over neighbors are two normal variable, the distribution of difference $S(x)$ of each data object x and the average attribute value of $x's$ neighbors is also normally distributed. ■

3.3 Test for Outlier Detection

The test for detecting an outlier can be described as follows. $|\frac{S(x)-\mu_s}{\sigma_s}| > \theta$. 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 is the mean value of all $S(x)$, and σ_s is the standard deviation of all $S(x)$. Choice of θ depends on 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 algorithm 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 validate the outlier of the incoming data set.

Given an attribute data set V and the connectivity graph G, the *TPC* algorithm first retrieves the neighbor nodes from G for each data object x, then it computes the difference of the attribute value of x to 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 computed to get the distribution value μ_s and σ_s. Note that the data object are processed on a page basis to reduce redundant I/O. In other words, all the nodes within the same disk page are processed before retrieving the nodes of the next disk page.

3.5 Computation of Test Results

The neighborhood aggregate statistics value, e.g., mean and standard deviation, computed in the TPC algorithm can be used to verify the outlier of an incoming data set. The two verification procedures 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

Test Parameters Computation(TPC) Algorithm

```
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 ($\mu_s$,$\sigma_s$).
```

Algorithm 1: Pseudo-code for test parameters computation

RNV procedure check the outlierness from a set of randomly generated nodes. The step 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/Os for Find_Neighbor_Nodes_Set() in different iteration 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 algorithm. The storage of data set should support I/O efficient computation of this operation. We discuss the choice for storage structure and provide experimental comparison in Section 5 and 6.

Given a route RN in the data set D with graph structure G, the ROD algorithm 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 tested using the spatial outlier detection test $|\frac{S(x)-\mu_s}{\sigma_s}| > \theta$. The θ is predetermined by the given confidence interval.

The I/O cost of ROD and RNV are also dominated by the I/O cost of Find_Neighbor_Nodes_Set() operation.

4 Analytical Evaluation and Cost Models

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

Route Outlier Detection(ROD) Algorithm

```
**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 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.
```

Algorithm 2: Pseudo-code for route outlier detection

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

The CRR value is determined by the 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 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 memory page. Λ is the average number of nodes in the neighbor list of a node. N is the total number of node in the data set, L is the number of node along a route, and R is the number of nodes randomly generated by users for spatial outlier verification.

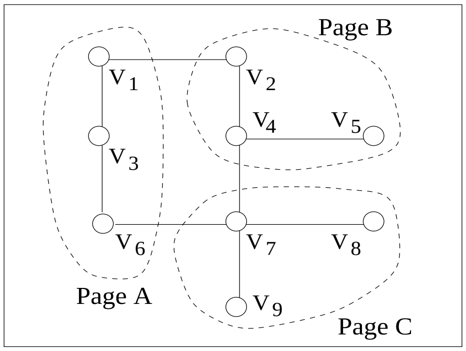


Figure 5: Example of CRR

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

Table 1: Symbols used in Cost Analysis

4.1 Cost Modeling for Test Parameters Computation(TPC) Algorithm

The TPC algorithm is a nest loop index join. Suppose that we use two memory buffers, 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 number of page accesses.

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

The outer loop retrieves all the data records on the page basis, and has an aggregated cost of $\frac{N}{\beta}$. For each node x, on the 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 expected total cost for the inner loop is $N * \Lambda * (1 - \alpha)$.

4.2 Cost Modeling for Route Outlier Detection(ROD) algorithm

We get the following cost function to estimate the number of page accesses with two memory buffers for ROD algorithm. One memory buffer is reserved for processing the node x to be verified, 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 the average, its successor node y are in 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 in 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 Random Node Verification(RNV) algorithm

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

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

Since the memory buffer is assumed to be 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 in 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 expected total cost to process the neighbor of R nodes is $R*\Lambda*(1-\alpha)$.

5 Experiment Design

In this section, we describe the layout of our experiments and then illustrate the candidate clustering methods.

5.1 Experimental Layout

The design of our experiments is shown is Figure 6. Using the Twin-Cities Highway Connectivity Graph(TCHCG), we took data from the TCHCG and physically stored the data set into data pages using different clustering strategies and page sizes. These data pages were then precessed to generate the global distribution or sampling distribution, depending on the size of the data sets.

We compared different data page clustering schemes CCAM [16], Z-ordering [11], and Cell-tree [5]. Other parameters of interest were the size of the memory buffer, the buffering strategies, the memory block size(page size), and the number of neighbors. The measures of our experiments are the CRR values and I/O cost for each outlier detection procedures.

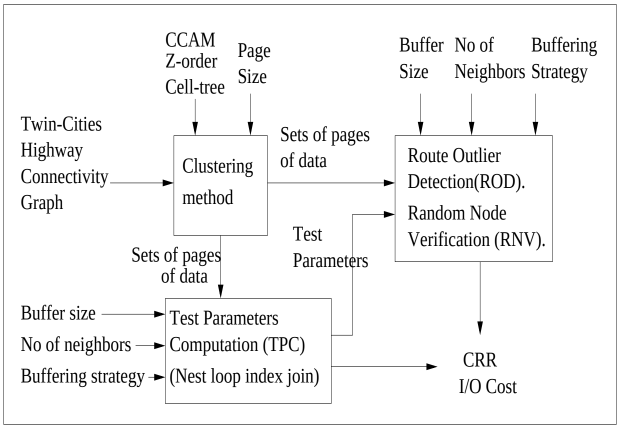


Figure 6: Experimental Layout

The experiments are 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, as shown in Figure 1. This data set is provided by the Minnesota Dept. of Transportation(MnDot).

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 I/O cost for outlier-detection query processing.

5.2 Candidate Clustering Methods

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

Connectivity-Clustered Access Method(CCAM): CCAM [16] 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. We used the B^+ tree with Z-order in our experiments, since the benchmark graph was embedded in graphical space. 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 [11] 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 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), 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.

5.3 Candidate Buffering Strategies

We evaluated three buffering strategies to replace the page in the memory buffer. The simplest page replacement algorithm is First In First Out(FIFO) algorithm. A FIFO replacement algorithm marks the time when each page was bought into memory buffer. When a page must be replaced, the oldest page is chosen. The Least Recently Used(LRU) algorithm selects the page that has not been referenced for the longest period of time for replacement. In contrast, The Most Recently Used(MRU) algorithm replaces the page which has been just recently referenced.

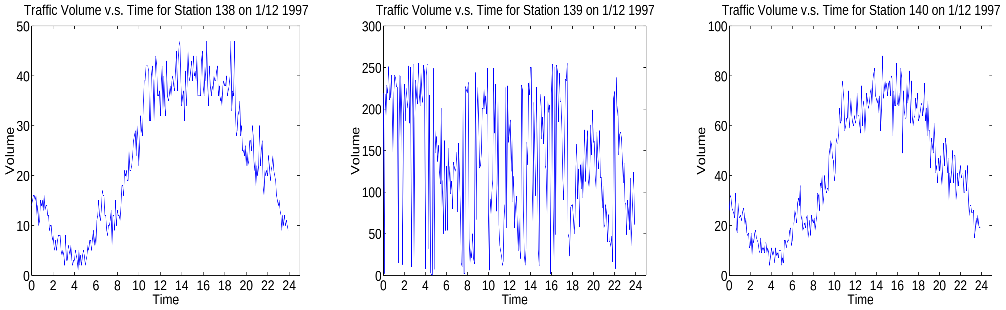
6 Experimental Observations and Results

In this section, we illustrate outlier examples detected in the traffic data set, test the statistical assumption, present the results of our experiments, and test the effectiveness of different page 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. We examined the CRR measures in the set of experiments that deals with range outlier detection queries.

6.1 Outliers Detected

We tested the effectiveness of our algorithm on the Twin-Cities traffic data set and detect numerous outliers, as described in the following examples.

In Figure 7, the abnormal station(Station 139) was detected whose volume values are 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 to daily basis.



(a) Station 138

(b) Outlier Station 139

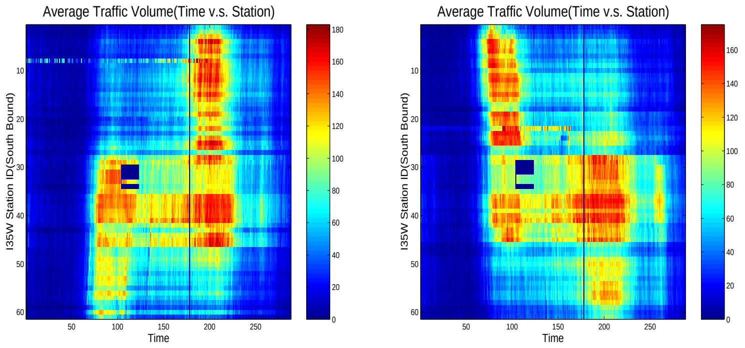
(c) Station 140

Figure 7: Outlier station 139 and its neighbor stations on 1/12 1997

Figure 8 shows another example of traffic flow outliers. Figure 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 the 5-minute time slot for the whole day and the Y-axis is the label of the stations installed on the highway, starting from 1 in the north end to 61 in the south end. The abnormal dark line at time slot 177 and the dark rectangle during time slot 100 to 120 on X-axis and between station 29 to 34 on Y-axis can be easily observed from both (a) and (b). This dark line at time slot 177 is an instance of temporal outliers, where the dark rectangle is a spatial-temporal outlier. Moreover, station 9 in Figure 8(a) exhibits inconsistent traffic flow compared with its neighboring stations, and was detected as a spatial outlier.

6.2 Testing 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 on different volumes are shown in Figure 9(a) with a normal probability distribution superimposed. As can be seen in Figure 9(a), the normal distribution approximates the volume distribution very well. We calculated the interval of



(a) I-35W North Bound

(b) I-35W South Bound

Figure 8: An example of outlier

$[\bar{v}-\sigma, \bar{v}+\sigma]$, $[\bar{v}-2\sigma, \bar{v}+2\sigma]$, and $[\bar{v}-3\sigma, \bar{v}+3\sigma]$ where $\bar{v}$ 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, which 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 Figure9(b), and it appears linear. Hence the volume values of all stations at the same time are approximately a normal distribution.

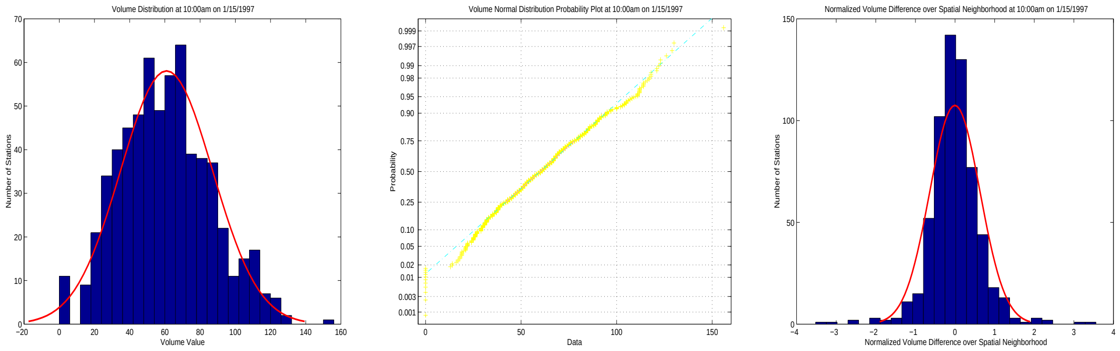
Given the definition of neighborhood, we then calculate the average volume value $(\bar{v_N})$ around its k neighbors according to topological relationship for each station. Since the volume values are normally distributed, the average of the normal variables is also a normal distribution.

Since the volume values and the average volume values over neighborhoods are normally distributed, the difference$(v - \bar{v})$ between these volumes and their corresponding average volume values over neighborhoods is also a normal distribution since the difference of two random normal random variables is always normal, as shown in Figure 9(c). Given the confidence level 100(1-α)%, we can calculate the confidence interval for the difference distribution, i.e., 100(1-α) percentage of difference value distribution lies between $-z_{\alpha/2}$ and $z_{\alpha/2}$ standard deviation of the mean in the sample space. So we can classify the spatial outliers at the given confidence level threshold.

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 1K bytes, 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 4.2.

As shown in Table 1, CCAM produced the lowest number of data page accesses for the



(a) Histogram of traffic volume distribution

(b) Normal probability plot for traffic volume distribution

(c) Histogram of volume difference over neighborhood

Figure 9: Verification of normal distribution for traffic volumes and volume difference over neighbors

Clustering Method	Parameters Computation		Random Node Verification		Route Outlier Detect		$\alpha =$ CRR
	Actual	Predicted	Actual	Predicted	Actual	Predicted	
CCAM	628	687	241	246	30	36	0.68
CELL	834	919	279	291	45	53	0.53
Zord	1263	1269	349	357	78	79	0.31
$N = 773$, $L = 38$, $R = 150$, $\beta = 4$, $\Lambda = 2$							

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

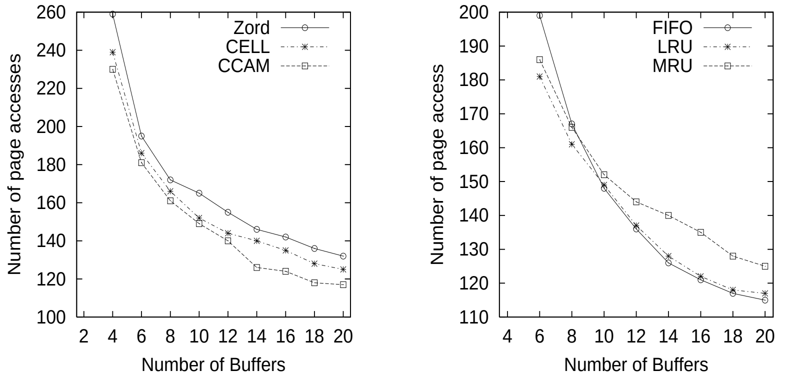
outlier detection procedures. This is to be expected, since CCAM generated the highest CRR value.

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 buffer size, page size, number of neighbor, and neighborhood depth.

6.4.1 The Effect of Buffering

We evaluated the effect of buffering on the performance of the page clustering methods and buffer replacement strategies. The variable parameters were the number of buffers available. Figure 10(a) shows the effect of buffering on the performance of model construction for various clustering methods with fixed page size 2K. As can be seen, the performance improves as the number of buffers increases. The performance ranking for each clustering methods remains the same for different buffer sizes. Figure 10(b) demonstrates the effect of different buffering strategies on the number of page accesses. When the buffer size is small (e.g., 4-8), the LRU algorithm has the best performance. As the number of buffers increases to greater than 10, both FIFO and LRU have better performance than MRU.

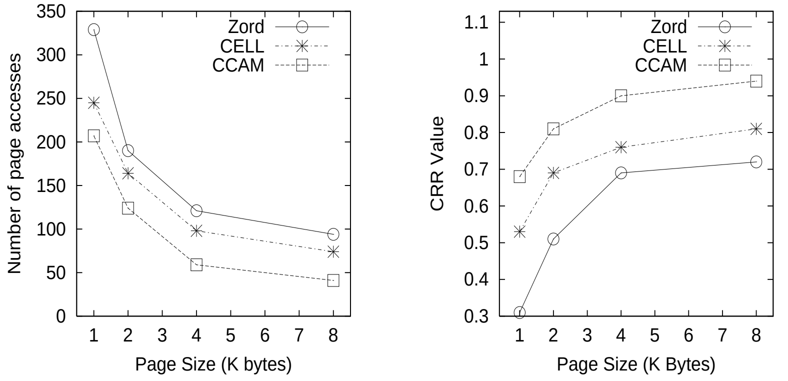


(a) Page clustering methods (b) Buffer replacement strategy

Figure 10: Effect of Buffering

6.4.2 The effect of page size and CRR value

Figures 15 (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 has better performance than Z-order clustering.

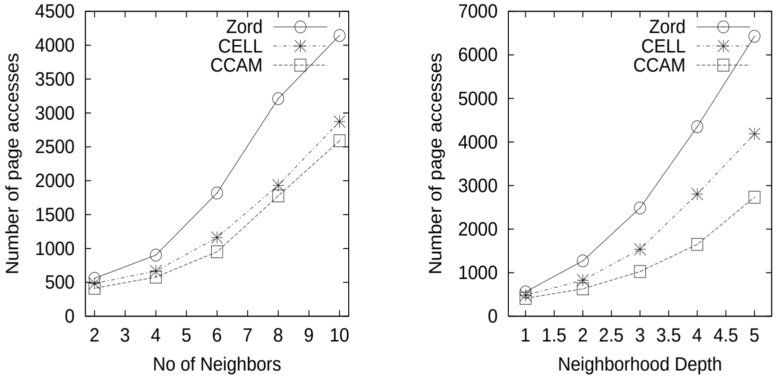


(a) Page Accesses (b) CRR

Figure 11: Effect of page size on data page accesses and CRR (buffer size = 32K)

6.4.3 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 1K, buffer size at 4K, and used the LRU buffering strategy. Figure 12 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. The performance ranking for each page clustering method remains the same for different numbers of neighbors. Figure 12 shows the number of page accesses as the neighborhood depth increases from 1 to 5. CCAM has better performance than Z-order and CELL for all the neighborhood depths.



(a) Number of Neighbors

(b) Neighborhood Depth

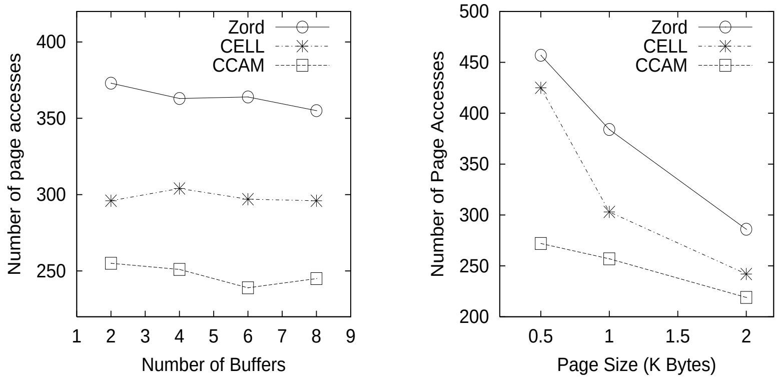
Figure 12: Effect of neighborhood cardinality on data page accesses (Page size = 1K, Buffer size = 4K)

6.5 Evaluation of I/O cost for RNV algorithm

We also evaluated the performance for different page clustering methods, page sizes, and buffer sizes when the user issues a query to verify if a node is an outlier. In this experiment, we randomly generated 150 nodes for verification, and accumulated the total number of page accesses. Figure 13(a) shows the effect of increasing buffer size from two to eight, when the page size is fixed at 1 Kbyte. As can be easily observed, the increase of buffer number does not reduce the number of page accesses, and CCAM has the lowest number of page accesses. Figure 13(b) shows the effect of varying page size from 0.5 Kbytes to 2 Kbytes. The number of page accesses decreases as we increase the page size, and CCAM outperforms Zord and CELL for all page sizes.

6.6 Evaluation of I/O cost for ROD algorithm

We evaluated the performance for different page clustering methods, page size, and buffer size when users request an outlier detection along a given route (e.g., I-35W North Bound) on a highway.



(a) Effect of buffer size (page size = 1K)

(b) Effect of page size (Buffer size = 4K)

Figure 13: Effect of buffer size and page size

6.6.1 The effect of buffering

Finally, we evaluated the effect of buffering for the outlier detection along a route. Figure 14 shows the number of page accesses as we increase the buffer number from 2 to 8. As can be seen, the increase of buffer size does not improve the performance after a certain buffer size, and CCAM has the best performance.

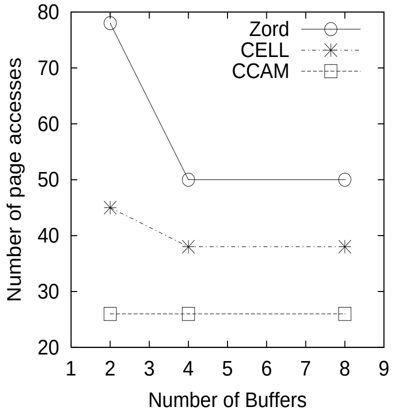
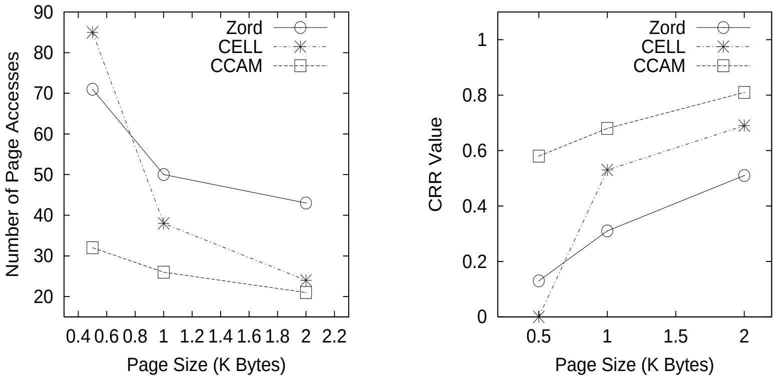


Figure 14: Effect of Buffering

6.6.2 The effect of page size and CRR value

Figures 15 (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 4

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 three page sizes. Note that the Cell tree has a CRR value of 0 and generates the highest number of page accesses when page size is 0.5 Kbytes.



(a) Page Access

(b) CRR

Figure 15: Effect of page size on data page accesses and CRR (buffer size = 32K)

7 Conclusions

In this paper, we focus on detecting outliers in spatial graph data sets. We propose the notion of a neighbor outlier in graph structured data sets, design a fast algorithm to detect outliers, analyze the statistical foundation underlying our approach, provide the cost models for different outlier detection procedures, and compare the performance of our approach using different data clustering approaches. In addition, we provide experimental results from the application of our algorithm on Twin-Cities traffic archival to show its effectiveness and usefulness.

We have evaluated alternative clustering methods for neighbor outlier query processing, including model construction, random node verification, and route outlier detection. Our experimental results show that the CCAM, which achieves the highest CRR, provides the best overall performance.

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*, pages 49–60, 1999.

[2] V. Barnett and T. Lewis. *Outliers in Statistical Data.* John Wiley, New York, 3rd edition, 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), pp. 262-270, Springer Verlag*, 1999.

[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*, pages 219–222, 1997.

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

[10] J. T. McClave and T. Sincich. *Statistics (Eighth Edition).* Prentice Hall, 2000.

[11] A. Orenstein and T. Merrett. A Class of Data Structures for Associative Searching. In *Proc. Symp. on Principles of Database Systens*, pages 181–190, 1984.

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

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

[14] S. Ramaswamy, R. Rastongi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Bell Laboratories, Murray Hill, NJ.*

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

[16] 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.

[17] 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.