Detecting Graph-Based Spatial Outliers: Algorithms and Applications(A Summary of Results) * †

Shashi Shekhar
Department of Computer Science and Engineering
University of Minnesota
200 Union ST SE, 4-192
Minneapolis, MN 55414
shekhar@cs.umn.edu

Chang-Tien Lu
Department of Computer Science and Engineering
University of Minnesota
200 Union ST SE, 4-192
Minneapolis, MN 55414
ctlu@cs.umn.edu

Pusheng Zhang
Department of Computer Science and Engineering
University of Minnesota
200 Union ST SE, 4-192
Minneapolis, MN 55414
pusheng@cs.umn.edu

ABSTRACT

Identification of outliers can lead to the discovery of unexpected, interesting, and useful 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 several fast algorithms to detect spatial outliers, and provide a cost model for outlier detection procedures. In addition, we provide experimental results from the application of our algorithms on a Minneapolis-St.Paul(Twin Cities) traffic dataset to show their effectiveness and usefulness.

Keywords

Outlier Detection, Spatial Data Mining, Spatial Graphs

1. INTRODUCTION

Outliers have been informally defined as observations which appear to be inconsistent with the remainder of that 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 [5]. 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, the performance analysis of athletes, voting irregularities, bankruptcy, and weather prediction.

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 attribute values of other objects in a given data set regardless of 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.

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 facilitate outlier query processing. We also use our basic algorithm to detect spatial 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

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 monitor the occupancy and volume of traffic on the road. At regular intervals, this information is sent to the Traffic Man-

*This work is supported 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, and by the National Science Foundation under grant 9631539

†A full version paper is available at `http://www.cs.umn.edu/~pusheng/pub/kdd2001/`

agement Center for operational purposes, e.g., ramp meter control, and research on traffic modeling and experiments.

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. 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. This outlier detection task is 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

We use three neighborhood definitions in this application as shown in Figure 1. First, we define a neighborhood based on spatial graph connectivity as a spatial graph neighborhood. In Figure 1, (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 1, (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 1, (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.

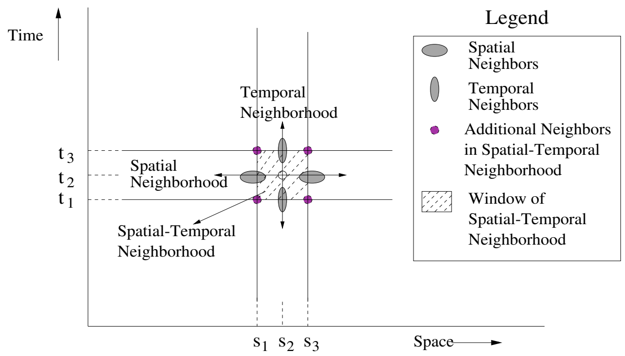


Figure 1: Spatial and Temporal outlier in traffic data

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 construct 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 is a collection of edges between locations in S
- A neighborhood relationship R 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 of spatial outliers

Objective:

- Correctness: outliers identified are significantly different from those of their neighborhood
- Efficiency: to minimize the computation time

Constraints:

- Attribute values have a normal distribution
- Size of the data set $\gg$ 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) the design of a statistical model M and a test for spatial outliers (b) the design of an efficient computation method to estimate parameters of the test, test whether a specific spatial location is an outlier, and test whether spatial locations on a given path are outliers.

2. RELATED WORK, CONTRIBUTION

Many outlier detection algorithms [1, 2, 3, 7, 8, 10, 12, 14] 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. The set-based outlier detection algorithms [2, 6] consider the statistical distribution of attribute values, ignoring the spatial relationships among items. Numerous outlier detection tests, known as discordancy tests [2, 6], 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.

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 multidimensional space, and provide tests based on concepts such as distance, density, and convex-hull depth. We discuss different example tests now. Knorr and Ng presented the notion of distance-based outliers [7, 8]. 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. [11] 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 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. In computational geometry, some depth-based approaches [12, 10] organize data objects in convex hull layers in data space according to their peeling depth [10], and outliers are expected to be found from data objects with a shallow depth value. Yu et al. [14] introduced an outlier detection approach, called *FindOut*, which identifies outliers by removing clusters from the original data.

Methods for detecting outliers in multi-dimensional Euclidean space have several limitations. First, multi-dimensional approaches assume that the data items are embedded in a isometric metric space and do not capture the spatial graph structure. Secondly, they do not exploit apriori information about the statistical distribution of attribute data. Last, 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. In addition, we evaluate our method in a Twin Cities traffic data set and show the effectiveness and usefulness of our approach.

3. OUR APPROACH

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 predetermined 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 be either just a number or a vector of attribute values. Finally, the statistic for the 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.

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 neighbors 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 the difference of $S(x)$ of each data object x and the average attribute value of $x's$ neighbors is also normally distributed.

3.2 Test for Outlier Detection

The test for detecting an 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 is the mean value of all $S(x)$, and σ_s is the 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 $\theta \approx 2$.

3.3 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 detect the outlier in the incoming data set.

Given an attribute data set V and the connectivity graph G, the Test Parameters Computation(TPC) algorithm 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) Algorithm

```
Input: S is the 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: (μs,σs).
for(i=1;i ≤ |D| ;i++){
    Oi=Get_One_Object(i,D); /* Select each object from D */
    NNS=Find_Neighbor_Nodes_Set(Oi,ND,G);
    /* Find neighbor nodes of Oi from G */
    Accum_Dist=0;
    for(j=1;j≤ |NSS|;j++){
        Ok=Get_One_Object(j,NNS); /* Select each object */
        Accum_Dist += F(Oi, Ok, S)
    }
    Avg_Dist = Accum_Dist / |NNS|;
    Add_Element(AvgDist_Set,i); /* Add the element */
}
μs = Get_Mean(AvgDist_Set); /* Compute Mean */
σs = Get_Standard_Dev(AvgDist_Set);
return (μs,σs).
```

Algorithm 1: Pseudo-code for test parameters computation

3.4 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 outliers in 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 RNV procedure checks the outliers from a set of randomly generated nodes. Given 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 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 algorithm. The storage of the data set should support the I/O efficient 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) Algorithm

```
Input: S is the 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;
    (μs,σs) are mean and standard deviation calculated in TPC;
    RN is the set of node in a route;

Output: Outlier_Set.
for(i=1;i ≤ |RN| ;i++){
    Oi=Get_One_Object(i,D); /* Select each object from D */
    NNS=Find_Neighbor_Nodes_Set(Oi,ND,G);
    /* Find neighbor nodes of Oi from G */
    Accum_Dist=0;
    for(j=1;j≤ |NSS|;j++){
        Ok=Get_One_Object(j,NNS); /* Select each object */
        Accum_Dist += F(Oi, Ok, S)
    }
    AvgDist = Accum_Dist/|NNS|;
    Tvalue=(AvgDist−μs)/σs
    /*Check the normal distribution table */
    if( Check_Normal_Table(Tvalue,CI)== True){
        Add_Element(Outlier_Set,i); /* Add the element */
    }
}
return Outlier_Set.
```

Algorithm 2: Pseudo-code for route outlier detection

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

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 determined by the page clustering method, the data record size, and the page size. The CRR value is defined as follows.

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

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

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 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

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 number of page accesses.

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

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 expected total cost for the inner loop is $N * \Lambda * (1-\alpha)$.

In a similar way, a cost model for ROD algorithms can be derived as follows: $C_{ROD} = L*(1-\alpha) + L*\Lambda*(1-\alpha) = L*(1-\alpha)*(1+\Lambda)$; and a cost model for RNV algorithms can be derived as follows: $C_{RNV} = R + R*\Lambda*(1-\alpha)$

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 in Figure 2. 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 processed to generate the global distribution or sampling distribution, depending on the size of the data sets.

We compared different data page clustering schemes: CCAM [13], Z-ordering [9], and Cell-tree [4]. 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 were the CRR values and I/O cost for each outlier detection procedure.

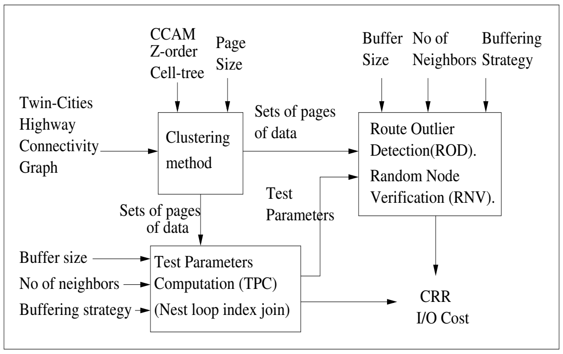


Figure 2: Experimental Layout

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. 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 the 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 [13] 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 [9] 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 [4] 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.

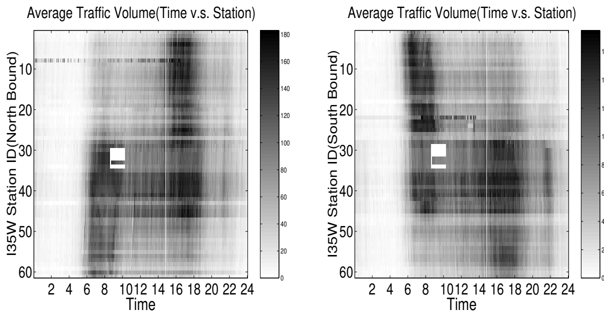
6. EXPERIMENTAL RESULTS

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 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. Here we present the evaluation of I/O cost for the TPC algorithm. The evaluations of I/O cost for the RNV and ROD algorithms are available in the full version paper.

6.1 Outliers Detected

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

Figure 3 shows one example of traffic flow outliers. Figures 3(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 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 2:45pm and the white rectangle from 8:20am to 10:00am 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 2:45pm is an instance of temporal outliers, where the white rectangle is a spatial-temporal outlier. Moreover, station 9 in Figure 3(a) exhibits inconsistent traffic flow compared with its neighboring stations, and was detected as a spatial outlier.



(a) I-35W North Bound

(b) I-35W South Bound

Figure 3: An example of outliers

6.2 Evaluation of I/O cost for TPC algorithm

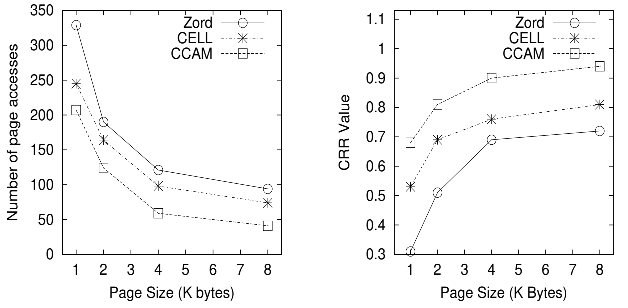
In this section, we present the results of our evaluation of the I/O cost and CRR value for alternative clustering methods while computing the test parameters. The parameters of interest are buffer size, page size, number of neighbors, and neighborhood depth.

6.2.1 *The effect of page size and CRR value*

Figures 4 (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.

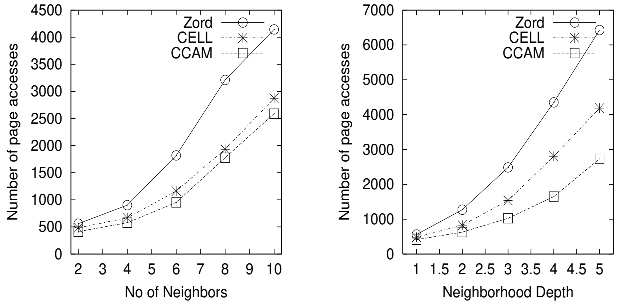
6.2.2 *The effect of neighborhood cardinality*

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 1K, and the buffer size at 4K, and used the LRU buffering strategy. Figure 5 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 5 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) Page Accesses (b) CRR

Figure 4: Effect of page size on data page accesses and CRR (buffer size = 32K)



(a) Number of Neighbors (b) Neighborhood Depth

Figure 5: Effect of neighborhood cardinality on data page accesses (Page size = 1K, Buffer size = 4K)

7. CONCLUSIONS

In this paper, we focused on detecting outliers in spatial graph data sets. We proposed the notion of a neighbor outlier in graph structured data sets, designed a fast algorithm to detect outliers, analyzed the statistical foundation underlying our approach, provided the cost models for different outlier detection procedures, and compared the performance of our approach using different data clustering approaches. In addition, we provided 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.

9. REFERENCES

[1] M. Ankerst, M. Breunig, H. 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. Breunig, H. 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] 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.

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

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

[7] 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.

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

[9] 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.

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

[11] S. Ramaswamy, R. Rastongi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of ACM SIGMOD International Conference on Management of Data, Dallas, Texas*, 2000.

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

[13] 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.

[14] 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.