A Unified Approach to Spatial Outliers Detection †

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) 624-8307 FAX:(612)625-0572
http://www.cs.umn.edu/Research/shashi-group

December 10, 2001

Abstract

Spatial outliers represent locations which are significantly different from their neighborhoods even though they may not be significantly different from the entire population. Identification of spatial outliers can lead to the discovery of unexpected, interesting, and implicit knowledge, such as local instability. In this paper, we first provide a general definition of S-outliers for spatial outliers. This definition subsumes the traditional definitions of spatial outliers. Second, we characterize the computation structure of spatial outlier detection methods and present scalable algorithms. Third, we provide a cost model of the proposed algorithms. Finally, we provide experimental evaluations of our algorithms using a Minneapolis-St. Paul(Twin Cities) traffic data set.

Keywords: Outlier Detection, Spatial Data Mining, Scalable Algorithm for Outlier Detection

---

†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, by the National Science Foundation under grant 9631539, and by the Minnesota Department of Transportation and the Center for Transportation Studies at the University of Minnesota under grant 1725-5216306

*The corresponding author. E-mail: ctlu@cs.umn.edu. Tel: (612) 626-7703

1 Introduction

Global outliers have been informally defined as observations in a data set 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 [9]. The identification of global 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, and severe weather prediction. This paper focuses on spatial outliers, i.e., observations which appear to be inconsistent with their neighborhoods. Detecting spatial outliers is useful in many applications of geographic information systems and spatial databases [23, 24]. These application domains include transportation, ecology, public safety, public health, climatology, and location based services.

We model a spatial data set to be a collection of spatially referenced objects, such as houses, roads, and traffic sensors. Spatial objects have two distinct categories of dimensions [27] along which attributes may be measured. Categories of dimensions* of interest are spatial and non-spatial. Spatial attributes of a spatially referenced object includes location, shape, and other geometric or topological properties. Non-spatial attributes of a spatially referenced object include traffic-sensor-identifiers, manufacturer, owner, age, and measurement readings. A spatial neighborhood [27] of a spatially referenced object is a subset of the spatial data based on a spatial dimension, e.g., location. Spatial neighborhoods may be defined based on spatial attributes, e.g., location, using spatial relationships such as distance or adjacency. Comparisons between spatially referenced objects are based on non-spatial attributes.

A spatial outlier is a spatially referenced object whose non-spatial attribute values are significantly different from those of other spatially referenced objects in its spatial neighborhood. Informally, a spatial outlier is a local instability (in values of non-spatial attributes) or a spatially referenced object whose non-spatial attributes are extreme relative to its neighbors, even though they may not be significantly different from the entire population. For example, a new house in an old neighborhood of a growing metropolitan area is a spatial outlier based on the non-spatial attribute house age.

In this paper, we provide a general definition of spatial outliers and propose efficient spatial outlier detection algorithms. We provide cost models for outlier detection algorithms, and compare alternative underlying data clustering methods. We also experimentally evaluate the proposed algorithm using a Minneapolis-St. Paul (Twin Cities) traffic data set.

*Examination of other categories of dimensions, e.g., temporal, is beyond the scope of this paper and may be explored in future work.

1.1 An Illustrative Application Domain

The Traffic Management Center [18] Freeway Operations group archives traffic measurements from the freeway system in Minneapolis-St. Paul (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 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.

In this application, each station is a spatially referenced object with spatial attributes(e.g., location) and non-spatial attributes(e.g., measurements). Spatial arrangement of stations can be modeled as a spatial graph [25]. A directed edge from station s_1 to station s_2 indicates the existence of a road segment allowing traffic to move from s_1 to s_2. This graph is called a spatial graph because nodes, i.e., stations, are located in a Euclidean space [27] where each node has a location specified by coordinates, e.g., <highway, mile point>. The non-spatial attributes include sensor-id and traffic measurements (e.g., volume, occupancy). We are interested in discovering the location of stations whose measurements are inconsistent with those of their neighbors. This spatial outlier detection task is formalized as follow.

Let the traffic sensors constitute a collection of spatially referenced objects. The location of a sensor represents a spatial attribute and is represented by the symbol x. A traffic measurement (e.g., volume) constitutes a non-spatial attribute space and is represented as $f(x)$. The neighborhood of x, $N(x)$, is the set of traffic sensors adjacent to the sensor located at x. We note that the neighborhood relationship is based on directed edges in the underlying spatial graph. Thus sensors on opposite sides (e.g., I-35W north bound and I-35W south bound) are not neighbors even if the pairwise Euclidean distance is small. A sensor is compared to its neighborhood using the function $S(x) = [f(x) - E_{y \in N(x)}(f(y))]$, where $f(x)$ is the attribute value for a location x, $N(x)$ is the set of neighbors of x, and $E_{y \in N(x)}(f(y))$ is the average attribute value for the neighbors of x. The statistic function $S(x)$ denotes the difference of the attribute value of a sensor located at x and the average attribute value of $x's$ neighbors.

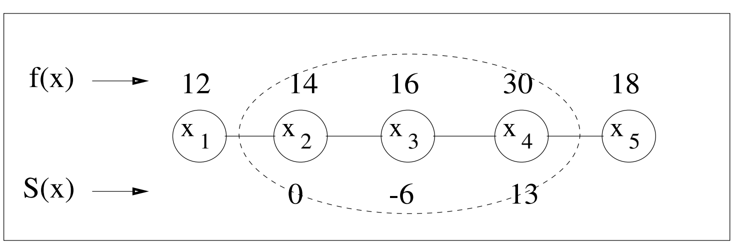


Figure 1: Example of a Spatial Statistic

Example: We illustrate the computation of the spatial statistic *S(x)* using an example, as shown in Figure 1. Consider location x_3 in which the attribute value

$f(x_3) = 16$, x_3's neighborhood set $N(x_3) = \{x_2, x_4\}$, the average neighborhood attribute value $E_{y \in N(x)}(f(y)) = \frac{14+30}{2} = 22$, and the spatial statistic function $S(x) = [f(x) - E_{y \in N(x)}(f(y))] = 16 - 22 = -6$.

Theorem 1 *Spatial Statistic $S(x)$ is normally distributed if the attribute value $f(x)$ is normally distributed.*

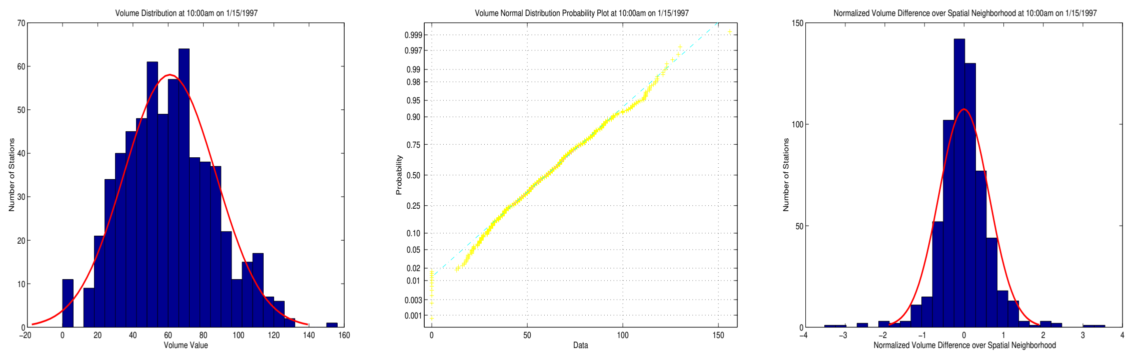
Proof: The formal proof is available in Appendix A.

A popular test for detecting spatial outliers for normally distributed $f(x)$ can be described as follows: Spatial statistic $Z_{s(x)} = |\frac{S(x)-\mu_s}{\sigma_s}| > \theta$. For each location x with an attribute value $f(x)$, the $S(x)$ is the difference between the attribute value at location x and the average attribute value of $x's$ neighbors, μ_s is the mean value of $S(x)$, and σ_s is the value of the standard deviation of $S(x)$ over all stations. The choice of θ depends on a specified confidence level. For example, a confidence level of 95 percent will lead to $\theta \approx 2$.

The assumption of a normal distribution for the $f(x)$ and $S(x)$ functions can be tested in our traffic data set. In this data set, the volume values of all stations at each time slot are approximately a normal distribution. A histogram of the numbers of stations for different intervals of a non-spatial attribute, volume, is shown in Figure 2(a), where a normal probability distribution curve is superimposed on the histogram. The normal distribution seems to approximate the volume distribution reasonably 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. These values are quite close to the corresponding values(68%, 95%, and 100%) for a normal distribution [4]. Moreover, we plot the normal probability plot in Figure 2(b), and it appears linear. Hence the values of the non-spatial attribute volume for all stations at the same time are approximately a normal distribution. The difference function, $S(x) = [f(x) - E_{y \in N(x)}(f(y))]$, which computes the difference of volume and the average volume of corresponding neighbors, also seems normally distributed, as shown in Figure 2(c). Given the confidence level 100(1-α)%, 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. Those stations with a spatial statistic $Z_{s(x)}$ value (standardized $S(x)$) greater than $z_{\alpha/2}$ or less than $-z_{\alpha/2}$ are classified as spatial outliers.

1.2 Definition of S-Outliers

Consider a spatial framework $SF =< S, NB >$, where S is a set of locations $\{s_1, s_2, \ldots, s_n\}$ and $NB : S \times S \rightarrow \{True, False\}$ is a neighbor relation over S. We define a neighborhood $N(x)$ of a location x in S using NB, specifically $N(x) = \{y \mid y \in S, NB(x, y) = True\}$.



(a) Histogram of traffic volume distribution

(b) Normal probability plot for traffic volume distribution

(c) Histogram of volume difference over neighborhood

Figure 2: Verification of normal distribution for traffic volumes and volume difference over neighbors

Definition: An object O is an S-outlier(f, f^N_{aggr}, F_{diff}, ST) if $ST\{F_{diff}[f(x), f^N_{aggr}(f(x),N(x))]\}$ is true, where $f : S \to R$ is an attribute function, $f^N_{aggr} : R^N \to R$ is an aggregation function for the values of f over neighborhood, R is a set of real numbers, $F_{diff} : R \times R \to R$ is a difference function, and $ST : R \to \{True, False\}$ is a statistic test procedure for determining statistical significance.

Example 1. The spatial outliers defined in Section 1.1 are examples of S-outliers. We can define respective components in the traffic application domain as follows. The f is the non-spatial attribute, namely, traffic volume. The neighborhood aggregate function $f^N_{aggr}(x) = E_{y \in N(x)}(f(y))$ is the average attribute value function over neighborhood $N(x)$. The difference function $F_{diff}(x)$ is $S(x) = [f(x) - E_{y \in N(x)}(f(y))]$, i.e., the arithmetic difference between attribute function $f(x)$ and neighborhood aggregate function $f^N_{aggr}(x)$. Let $\mu_{s(x)}$ and $\sigma_{s(x)}$ be the mean and standard deviation of the difference function F_{diff}; then the significance test function ST can be defined as $Z_{s(x)} = |\frac{S(x)-\mu_{s(x)}}{\sigma_{s(x)}}| > \theta$.

Example 2. A $DB(p, D)$-outlier [15] is also an example of an S-outlier. 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 [15]. Assuming f^N_{aggr} is the number of objects within distance D from object O, the statistical test function ST can be defined as $\frac{(Total\, number\, of\, objects) - f^N_{aggr}(x)}{(Total\, number\, of\, objects)} > p$. The DB-outlier subsumes many other definitions of global outliers [15].

1.3 Contribution, Outline, and Scope

This paper provides a general definition of spatial outliers and shows that various tests for detecting spatial outliers are special cases. We identify the basic spatial-self-join computational structure for the scalable implementation of spatial outlier tests and recognize clustering methods to be the primary design decision influencing the total computational cost. We also provide efficient strategies to implement a spatial outlier detection test and evaluate our method in a Twin-Cities traffic data set to show its effectiveness and usefulness.

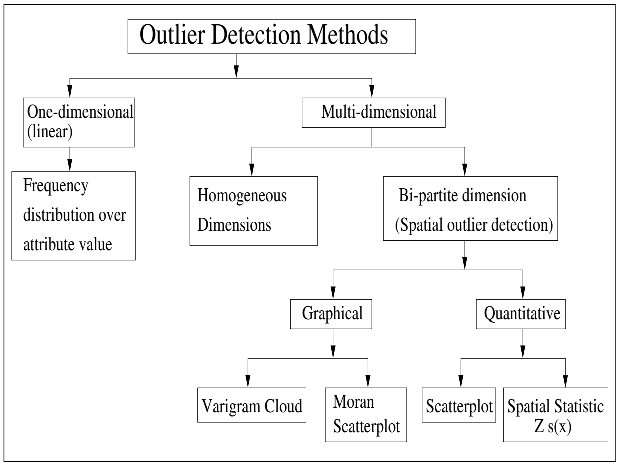
The rest of the paper is organized as follows. Section 2 reviews related work and discusses our contributions. In Section 3, we discuss the computation structure for detecting spatial outliers and propose our general outlier detection algorithms. The cost models for proposed algorithms are analyzed in Section 4. Section 5 presents our experiment design. The experimental observation and results are shown in Section 6. We summarize our work and describe the future direction of our research in Section 7.

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

2 Related Work

Many outlier detection algorithms [1, 2, 3, 13, 14, 20, 22, 28] have been recently proposed. As shown in Figure 3(a), these methods can be broadly classified into two categories, namely one-dimensional(linear) outlier detection methods and multi-dimensional outlier detection methods. The one-dimensional outlier detection algorithms [2, 10] consider the statistical distribution of non-spatial attribute values, ignoring the spatial relationships between items. Numerous outlier detection tests, known as discordancy tests [2, 10], 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. We use an example to illustrate the differences among one-dimensional and multidimensional outlier detection methods. In Figure 4(a), the X-axis is the location of data points in one dimensional space; the Y-axis is the attribute value for each data point. One-dimensional outlier detection methods ignore the spatial location of each data point, and fit the distribution model to the values of the non-spatial attribute. The outlier detected using a one-dimensional approaches is the data point G, which has an extremely high attribute value 7.9, exceeding the threshold of $\mu + 2\sigma = 4.49 + 2 * 1.61 = 7.71$, as shown in Figure 4(b). This test assumes a normal distribution for attribute values.

Multi-dimensional outlier detection methods can be further grouped into two categories, namely homogeneous multidimensional metric based methods and spatial methods. The homogeneous multidimensional methods model data sets as a collection of points in a multidimensional isometric space, and provide tests based on concepts such as distance, density, and convex-hull depth. These methods do not distinguish between attribute dimensions and geo-spatial dimensions, and use all dimensions for defining neighborhood as well as for comparison, as shown in Figure 3(b). We discuss representative methods now. Knorr and Ng presented the notion of distance-based outliers [13, 14].



(a) Classification

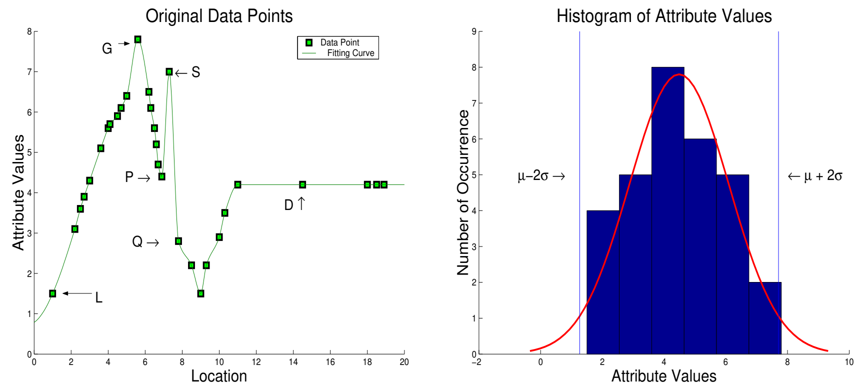
	One-dimensional (linear)	Multi-dimensional	
		Homogeneous	Bi-partite
Neighbor Definition	N/A	location and attribute	location
Comparison	with population distribution	location and attribute	attribute values of neighbors

(b) Comparison

Figure 3: Classification and comparison of outlier detection methods

As discussed in example 2 of Section 1, 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. [21] 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 in which 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, depth-based approaches [22, 20] organize data objects in convex hull layers in data space according to peeling depth [20], and outliers are expected to be found from data objects with shallow depth values. Yu et al. [28] introduced an outlier detection approach, called *FindOut*, which identifies outliers by removing clusters from the original data. The key idea of this approach 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. In Figure 4(a), for example, the outliers detected using homogeneous multidimensional approaches are the data point D and L, which lie in a low density area.

Homogeneous multidimensional methods have several limitations. First, they are designed to detect global outliers rather than spatial outliers. Second, they assume that the data items are embedded in a isometric metric space and do not distinguish between non-spatial attributes and spatial attributes. Third, they do not exploit apriori information about the statistical distribution of attribute data. Last, they seldom provide a confidence measure for the discovered outliers.



(a) An Example Data Set

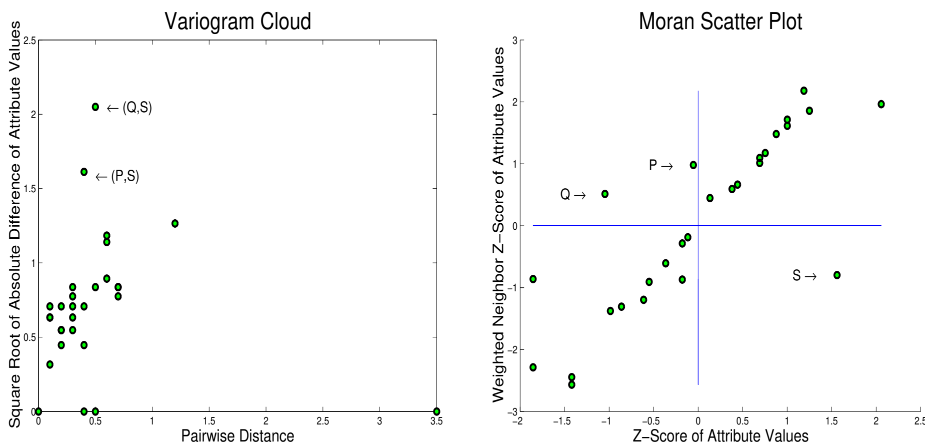
(b) Histogram

Figure 4: A Data Set for Outlier Detection

Bi-partite multidimensional tests are designed to detect spatial outliers. They separate spatial attributes from attribute attributes, as shown in Figure 3(b). Spatial attributes are used to characterize location, neighborhood, and distance. Non-spatial attribute dimensions are used to compare a spatially referenced object to its neighbors. Spatial statistics literature provides two kinds of bi-partite multidimensional tests, namely graphical tests and quantitative tests. Graphical tests are based on visualization of spatial data which highlight spatial outliers. Example methods include variogram clouds [5] and Moran scatterplots [17]. Quantitative methods provide a precise test to distinguish spatial outliers from the remainder of data. Scatterplots [16] are a representative technique from the quantitative family.

A variogram-cloud displays data points related by neighborhood relationships. For each pair of locations, the square-root of the absolute difference between attribute values at the locations versus the Euclidean distance between the locations are plotted. In data sets exhibiting strong spatial dependence, the variance in the attribute differences will increase with increasing distance between locations. Locations that are near to one another, but with large attribute differences, might indicate a spatial outlier, even though the values at both locations may appear to be reasonable when examining the data set non-spatially. Figure 5(a) shows a variogram cloud for the example data set shown in Figure 4(a). This plot shows that two pairs (P, S) and (Q, S) in the left hand side lie above the main group of pairs, and are possibly related to spatial outliers. The point S may be identified as a spatial outlier since it occurs in both pairs (Q, S) and (P, S). However, graphical tests of spatial outlier detection are limited by the lack of precise criteria to distinguish spatial outliers. In addition, a variogram cloud requires non-trivial post-processing of highlighted pairs to separate spatial outliers from their neighbors, particularly when multiple outliers are present or density varies greatly.

A Moran scatterplot [17] is a plot of normalized attribute value ($Z[f(i)] = \frac{f(i)-\mu_f}{\sigma_f}$)



(a) Variogram cloud (b) Moran scatterplot

Figure 5: Variogram Cloud and Moran Scatterplot to Detect Spatial Outliers

against the neighborhood average of normalized attribute values ($W \cdot Z$), where W is the row-normalized (i.e., $\sum_j W_{ij} = 1$) neighborhood matrix, (i.e., $W_{ij} > 0$ iff neighbor(i, j)). The upper left and lower right quadrants of Figure 5(b) indicate a spatial association of dissimilar values: low values surrounded by high value neighbors(e.g., points P and Q), and high values surrounded by low values (e.g,. point S). Thus we can identify points(nodes) that are surrounded by unusually high or low value neighbors. These points can be treated as spatial outliers.

Definition: $Moran_{outlier}$ is a point located in upper left and lower right quadrants of Moran scatterplot. This point can be identified by $(Z[f(i)]) \times (\sum_j (W_{ij} Z[f(j)])) < 0$.

Lemma 1 *$Moran_{outlier}$ is a special case of an S-outlier.*

Proof: For a $Moran_{outlier}$, the difference functions are Z_i and I_i, where $Z_i = \frac{f(i)-\mu_f}{\sigma_f}$, $I_i = \sum_j (W_{ij} Z[f(j)]))$, and μ_f and σ_f are the mean and standard deviation of the attribute function $f(i)$. The statistic test function ST is $(Z[f(i)]) \times (\sum_j (W_{ij} Z[f(j)])) < 0$.

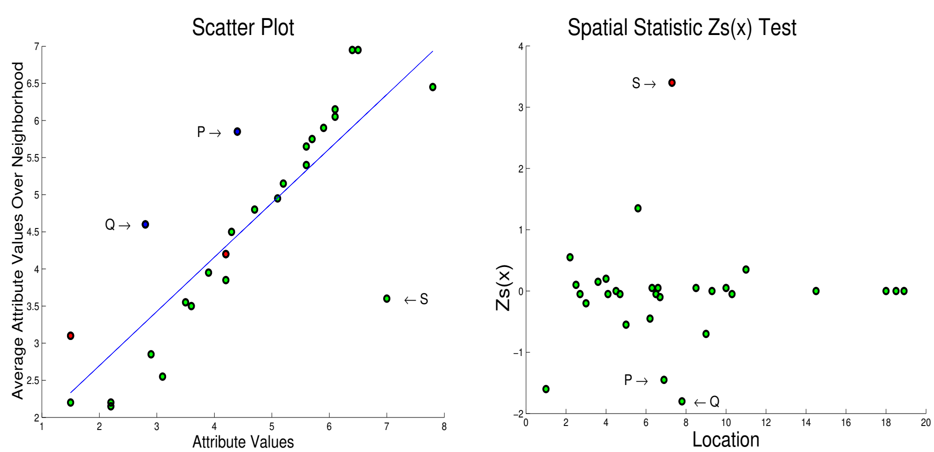
A scatterplot [8, 16] shows attribute values on the X-axis and the average of the attribute values in the neighborhood on the Y-axis. A least square regression line is used to identify spatial outliers. A scatter sloping upward to the right indicates a positive spatial autocorrelation (adjacent values tend to be similar); a scatter sloping upward to the left indicates a negative spatial autocorrelation. The residual is defined as the vertical distance (Y-axis) between a point P with location (X_p, Y_p) to the regression line $Y = mX + b$, that is, residual $\epsilon = Y_p - (mX_p + b)$. Cases with standardized residuals, $\epsilon_{standard} = \frac{\epsilon - \mu_\epsilon}{\sigma_\epsilon}$, greater than 3.0 or less than -3.0 are flagged as possible spatial outliers, where μ_ϵ and σ_ϵ are the mean and standard deviation of the distribution of the error term ϵ. In Figure 6(a), a scatter plot shows the attribute values plotted against the average of the attribute values in neighboring areas for the data set in Figure 4(a). The point S turns out to be the farthest from the regression line and may be identified as a spatial outlier.

Definition: $Scatterplot_{outlier}$ is a point with significant standardized residual error from the least square regression line in a scatter plot. Assuming errors are normally distributed, then $\epsilon_{standard} = |\frac{\epsilon - \mu_\epsilon}{\sigma_\epsilon}| > \theta$ is a common test. Nodes with standardized residuals $\epsilon_{standard} = \frac{\epsilon - \mu_\epsilon}{\sigma_\epsilon}$ from regression line $Y = mX + b$ and greater than θ or less than $-\theta$ are flagged as possible spatial outliers. The μ_ϵ and σ_ϵ are the mean and standard deviation of the distribution of the error term ϵ.

Lemma 2 *$Scatterplot_{outlier}$ is a special case of an S-outlier.*

Proof: For a $Scatterplot_{outlier}$, the neighborhood aggregate function $f^N_{aggr} = E(x) = \frac{1}{k}\sum_{y \in N(x)} f(y)$ is the average attribute value of neighbors. The difference function is $F_{diff} = \epsilon = E(x) - ((m * f(x)) + b)$, where m and b characterize the slope and intercept of the least square line fitting $(f(x), E(x))$. The spatial outliers are tested using statistic test function $ST = (|\frac{\epsilon - \mu_\epsilon}{\sigma_\epsilon}| > \theta)$.

Figure 6(b) shows the visualization of spatial statistic $Z_{s(x)}$ method described earlier in Section 1.1 and Example 1. The X-axis is the location of data points in one dimensional space; the Y-axis is the value of spatial statistic $Z_{s(x)}$ for each data point. We can easily observe that the point S has the $Z_{s(x)}$ value exceeding 3, and will be detected as spatial outlier. Note the two neighboring points P and Q of S have $Z_{s(x)}$ values close to -2 due to the presence of spatial outlier in their neighborhoods. Example 1 has already shown that $Z_{s(x)}$ is a special case of S-outlier.



(a) Scatterplot

(b) Spatial statistic $Z_{s(x)}$

Figure 6: Scatterplot and Spatial Statistic $Z_{s(x)}$ to Detect Spatial Outliers

3 Spatial Outlier Detection: Problem Definition and Proposed Algorithms

In this section, we provide a formal definition of the problem of designing computationally efficient techniques for detecting spatial outliers. Earlier sections presented a definition of spatial outliers and showed that the definition subsumes other quantitative spatial outlier definitions. Table 1 shows examples of difference function F_{diff} and statistic test function ST for different quantitative spatial outlier detection methods. Difference function F_{diff} computes parameters that are used by statistical test function ST to verify the outlierness of a node. We show F_{diff} and ST functions for Spatial statistic $Z_{s(x)}$, Scatterplot, and Moran scatterplot approaches to summarize the lemmas presented in the earlier section. For example, in the scatterplot approach, the difference function computes the error term ϵ, which is the value of the vertical distance between a node and the regression line in the $X-Y$ plane and is defined as $F_{diff} : \epsilon = E(x) - (m * f(x) + b)$, where $E(x)$, the average attribute value of neighbor nodes of x, is the Y-axis value; $f(x)$, the attribute value of node x, is the X-axis value; the m and b are the slope and intercept of the scatterplot line in the $X-Y$ plane.

The computation needs of spatial outlier detection are divided into two parts, model building and test result computation. Model building computes aggregate functions used by the difference function F_{diff} and statistic test function ST, as shown in the last row of Table 1. We discuss the computation of the aggregate functions and propose algorithms for model building and test result computation.

Test Computation			
Spatial Outlier Definition	Spatial statistic $Z_{s(x)}$	Scatterplot	Moran scatterplot
Difference function F_{diff}	$S(x) = [f(x) - E(x)]$	$\epsilon = E(x) - (m * f(x) + b)$	$Z_i = \frac{f(x)-\mu_f}{\sigma_f}$, $I_i = \sum_j W_{ij} Z_j$
Statistic test function ST	$\lvert \frac{S(x)-\mu_s}{\sigma_s} \rvert > \theta$	$\lvert \frac{\epsilon-\mu_\epsilon}{\sigma_\epsilon} \rvert > \theta$	$(Z[f(i)]) \times (\sum_j (W_{ij} Z[f(j)])) < 0$
Aggregate function used in F_{diff} and ST	μ_s, σ_s	m, b, μ_ϵ, σ_ϵ	μ_f, σ_f

Table 1: Examples of F_{diff} and ST functions for different approaches

3.1 Problem Definition

Given the components of the S-outlier definition, the objective is to design a computationally efficient algorithm to detect the S-outliers. The components of the S-outlier definition are restricted via constraints to allow computational efficiency while preserving the correctness of Lemmas showing that various existing spatial outlier detection tests (e.g., Scatterplot, Moran scatterplot, Spatial statistic $Z_{s(x)}$) are special cases of S-outliers. Thus the algorithms proposed in this section are useful in building models to detect spatial outliers via a variety of existing techniques. The following optimization problem characterizes the problem of designing efficient algorithms for detecting spatial outliers:

Spatial Outlier Detection Problem

Given:

- A spatial framework S consisting of locations $s_1, s_2, \ldots, s_n$
- A neighborhood relationship $N \subseteq S \times S$
- An attribute function $f : s_i \rightarrow R$
- A neighborhood aggregate function $f^N_{aggr} : R^N \rightarrow R$, where N is the maximum neighbor number for a location
- A comparison function $F_{diff}(f, f^N_{aggr})$
- Statistic test function $ST : R \rightarrow \{True, False\}$

Design: An efficient algorithm to detect S-outliers,
i.e., $\{s_i \mid s_i \in S, s_i\, is\, an\, S - outlier\}$

Objective:

- Efficiency: to minimize the computation time

Constraints:

- F_{diff} and ST are algebraic aggregate functions of values of $f(x)$ and f^N_{aggr}
- The size of the data set is much greater than the main memory size
- Computation time is determined by I/O time

Aggregate functions can be grouped into three categories, namely, distributive, algebraic, and holistic [6, 26]. An aggregate function F is called distributive if there exists a function G such that the value of F for a data set can be computed by applying a G function to the value of F in each partition of the data set. In most cases, $F = G$. Examples of distributive aggregate functions include *count*, *max*, and *sum*, as shown in Appendix B. An aggregate function F is algebraic if F of a data set can be computed using a fixed number of distributive aggregates from each partition of the data set. $Average$, $variance$, $standard\ deviation$, $maxN$, $minN$ are all algebraic aggregate functions. Illustrations are available in Appendix B. An aggregate function F is called holistic if the value of F for a data set cannot be computed using a constant number of distributive aggregates from each partition of the data set. Example of a holistic aggregate function includes *median*. We note that algebraic and distributive aggregate functions can be computed by a single scan of a data set even when the data set is too large to fit in the main memory. In processing a data set with a size greater than the size of memory, extra disk scans are required to calculate the holistic aggregate function.

For each node, say x, the attribute function $f(x)$ contains the attribute value of x. The neighborhood aggregate function f^N_{aggr} computes a value using the attribute value of x and the attribute value of $x's$ neighboring nodes. The distributive aggregate function computes the aggregate value (e.g., *sum*, *count*) of the attribute value and neighborhood aggregate value for all nodes. The algebraic aggregate function computes the statistic values for all nodes, e.g., $mean$ and $standard\, deviation$, and can be derived using the values computed in the distributive aggregate functions. The comparison function F_{diff} and statistic test function ST for the quantitative spatial outlier definition can be computed using algebraic aggregate functions of values from $f(x)$ and f^N_{aggr}. Table 2 shows the algebraic aggregate functions for different quantitative definitions of spatial outliers. Each column shows the computation structure of the attribute function, neighborhood aggregate function, distributive aggregate functions, and algebraic aggregate functions for each spatial outlier detection approach. For example, in the scatterplot approach, the attribute function is $f(x)$; the neighborhood aggregate function f^N_{aggr} is $E(x) = \frac{1}{k}\sum_{y \in N(x)} f(y)$; the distributive aggregate functions $D^{G_i}_{aggr}$ are $\sum f(x)$, $\sum E(x)$, $\sum f(x)E(x)$, $\sum f^2(x)$, $\sum E^2(x)$; and the algebraic aggregate functions $A^{G_i}_{aggr}$ are the slope m and the intercept b of the regression line, and the standard deviation σ_ϵ of the error term ϵ, all of which can be derived using the distributive aggregate functions.

By utilizing Table 2, we can compute the algebraic aggregate functions in one single scan of the spatial self-join of the station data set using the neighbor relationship. For example, the standard deviation of the error term ϵ in the scatterplot approach can be computed using the values computed in the distributive aggregate functions. In a naive approach, however, two data scans of the spatial self-join may be used, where the first scan computes the slope and intercept of the regression line, and the second scan calculates the statistic values (e.g., *mean* and *standard deviation*) of the error term.

Model Building			
Outlier Definition	Spatial statistic $Z_{s(x)}$	Scatterplot	Moran scatterplot
Attribute function f	f(x)	f(x)	f(x)
Neighborhood aggregate function f^N_{aggr}	$S(x) = f(x) - E(x)$	$E(x) = \frac{1}{k}\sum_{y \in N(x)} f(y)$	
Distributive aggregate functions: $D^{G1}_{aggr}, D^{G2}_{aggr}, .. , D^{Gk}_{aggr}$	$\sum S(x)$, $\sum S^2(x)$, n(count)	$\sum f(x)$, $\sum E(x)$, $\sum f(x)E(x)$, $\sum f^2(x)$, $\sum E^2(x)$, n(count)	$\sum f(x)$, $\sum f^2(x)$, n(count)
Algebraic aggregate functions: $A^{G1}_{aggr}, A^{G2}_{aggr}, .. , A^{Gk}_{aggr}$	$\mu_s = \frac{\sum S(x)}{n}$, $\sigma_s = \sqrt{\frac{1}{n}[\sum S^2(x) - \frac{(\sum S(x))^2}{n}]}$	$m = \frac{N\sum f(x)E(x) - \sum f(x)\sum E(x)}{N\sum f^2(x) - (\sum f(x))^2}$, $b = \frac{\sum f(x)\sum E^2(x) - \sum f(x)\sum f(x)E(x)}{N\sum f^2(x) - (\sum f(x))^2}$, $\mu_\epsilon = 0$, $\sigma_\epsilon = \sqrt{\frac{S_{yy} - (m^2 S_{xx})}{(n-2)}}$, where $S_{xx} = \sum f^2(x) - [\frac{(\sum f(x))^2}{n}]$, $S_{yy} = \sum E^2(x) - [\frac{(\sum E(x))^2}{n}]$,	$\mu_f = \frac{\sum f(x)}{n}$, $\sigma_f = \sqrt{\frac{1}{n}[\sum f^2(x) - \frac{(\sum f(x))^2}{n}]}$

Table 2: Model building to compute the aggregate functions

3.2 Our Approach

The computational task in the spatial outlier detection problem can be divided into two subtasks: a) design an efficient computation method to compute the global statistical parameters using a spatial join and b) test whether spatial locations on a given path are outliers. The first task is called model building; the second task is called test result computation.

3.2.1 Model Building

An I/O efficient model building algorithm computes the algebraic aggregate functions, e.g., the *mean* and *standard deviation*, in a single scan of a spatial self-join from a spatial data set using a neighbor relationship. The computed values from the algebraic aggregate functions can be used by the difference function F_{diff} and statistic test function ST to validate the outlierness of an incoming data set. Algorithm 1 shows the steps of the Model Building algorithm. In the first step, the algorithm retrieves the neighbor nodes for each data object, say x; then it computes the neighborhood aggregate function f^N_{aggr}. The distributive aggregate functions are then aggregated using the attribute function $f(x)$ and the neighborhood aggregate function f^N_{aggr}. Finally, the algebraic aggregate functions are computed using the values from the distributive aggregate functions. Note that the data objects 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.

Model Building Algorithm

Input: S is a spatial framework;
f is an attribute function;
N is the neighborhood relationship;
f^N_{aggr} is the neighborhood aggregate function;
$D^{G1}_{aggr}, D^{G2}_{aggr}, \ldots, f^{Gk}_{aggr}$ are the distributive aggregate functions;

Output: Algebraic aggregate functions $A^{G1}_{aggr}, A^{G2}_{aggr}, \ldots, A^{Gk}_{aggr}$

```
for(i=1;i ≤ |S| ;i++){
    O_i=Get_One_Object(i,S); /* Select each object from S */
    NNS=Find_Neighbor_Nodes_Set(O_i,N,S); /* Find neighbor nodes of O_i from S */
    for(j=1;j≤ |NSS|;j++){
        O_j=Get_One_Object(j,NNS); /* Select each neighbor of O_i */
        f^N_aggr = Compute_and_Aggregate(f(O_i), f(O_j));
    }
    /* Add the element to global aggregate functions */
    Aggregate_Element(D^G1_aggr, D^G2_aggr, ..., D^Gk_aggr, f^N_aggr, i);
}
/* Compute the algebraic aggregate functions*/
< A^G1_aggr, A^G2_aggr, ..., A^Gk_aggr > = Compute_Algebraic_Aggregate(D^G1_aggr, D^G2_aggr, ..., D^Gk_aggr);
return (A^G1_aggr, A^G2_aggr, ..., A^Gk_aggr).
```

Algorithm 1: Pseudo-code for model construction

Assuming each node has k neighbors, the first operation Get_One_Object() retrieves data points on the basis of the disk page, thus reducing redundant disk I/O operation. The Find_Neighbor_Nodes_Set() operation retrieves the data records of the k neighbors of the current processing node . If the neighbor nodes are not in the memory buffer, extra I/O operations are required to retrieve the disk pages which contain the data records of the neighbor nodes. The time needed to locate and transfer a disk block to memory buffer is in the order of milliseconds, usually ranging from 15 to 60 msec. Therefore, the operation Find_Neighbor_Nodes_Set() dominates the computation time. The I/O cost of Find_Neighbor_Nodes_Set() is determined by the clustering efficiency(CE), that is, how the nodes are grouped into disk pages. If a node and all of its neighbor nodes can be arranged in the same disk page, no redundant I/O operation will be required. The execution time for each data object can be estimated as follows.

For instance, we assume a 500MHz machine, and that the Cycle Per Instruction(CPI) is 5, the instruction count is 150, the number of neighbors for each data node is 10, that each disk I/O operation takes $15 * 10^{-3}$ sec (typical: 15 - 60 msec), and that CE denotes the clustering efficiency. For each data record, the execution time will be CPU time + (I/O time)*(1-CE)*k = $150 * 5 * \frac{1}{500*10^6} = 1.5 * 10^{-6} + 1.5 * 10^{-1} * (1-\text{CE})$ sec. The CE value determines the execution time. The higher the CE value, the shorter the execution time. We will formally define clustering efficiency in the following section.

Lemma 3 *Algebraic aggregate functions needed by the difference function F_{diff} and statistic test function ST can be computed by the Model Building Algorithm in one scan of the spatial self-join of the data set.*

Proof: By definition, an algebraic function F of a data set can be computed using a fixed number of distributive aggregates from each partition of the data set. In the Model Building algorithm, a fixed k number of distributive aggregate functions $D_{aggr}^{G1}, D_{aggr}^{G2}, \ldots, D_{aggr}^{Gk}$ are used to store the aggregate values in memory, and the algebraic aggregate functions are then computed using these aggregate values. If the distributive aggregate functions can fit inside the memory buffer, the algebraic aggregate functions can be computed using a single disk scan of the self-join of the data set. Distributive aggregate functions needed for various quantitative spatial outlier definitions are shown in Table 2. In all cases, one needs a very small number (less than a dozen) of distributive aggregate functions to compute the algebraic aggregate functions needed by each spatial outlier definition.

3.2.2 Test Result Computation

The algebraic aggregate functions, e.g., *mean* and *standard deviation*, computed in the Model Building algorithm can be used to verify the spatial outlier of incoming data sets. The two verification algorithms are Route Outlier Detection (ROD) and Random Node Verification (RNV). The ROD algorithm 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. 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 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 efficient I/O computation of this operation. We discuss the choice for storage structure and provide experimental comparison in Sections 5 and 6.

Given a route RN within the data set S, the ROD algorithm first retrieves the neighboring nodes from S for each data object, say x, in the route RN; then it computes the neighborhood aggregate function f_{aggr}^N using the attribute value of x and the attribute values of $x's$ neighbors. The difference function F_{diff} is computed using the attribute function $f(x)$, neighborhood aggregate function f_{aggr}^N, and the algebraic aggregate functions computed in the Model Building algorithm. Node x can then be tested for outlierness using the statistical test function ST.

Route Outlier Detection(ROD) Algorithm

```
Input: S is a spatial framework;
    f is an attribute function;
    N is the neighborhood relationship;
    f^N_aggr is a neighborhood aggregate function;
    F_diff is a difference function;
    A^G1_aggr, A^G2_aggr, ..., A^Gk_aggr are algebraic aggregate functions;
    ST is the spatial outlier test function;
    RN is the set of node in a route;

Output: Outlier_Set.
for(i=1;i ≤ |RN| ;i++){
    O_i=Get_One_Object(i,RN); /* Select each object from RN */
    NNS=Find_Neighbor_Nodes_Set(O_i,N,S);
   /* Find neighbor nodes of O_i from S */
    for(j=1;j≤ |NSS|;j++){
      O_j=Get_One_Object(j,NNS); /* Select each neighbor of O_i */
      f^N_aggr = Compute_and_Aggregate(f(O_i), f(O_j));
    };
    F_diff = Compute_Difference(f,f^N_aggr, A^G1_aggr, A^G2_aggr, ..., A^Gk_aggr);
    if(ST(F_diff, A^G1_aggr, A^G2_aggr, ..., A^Gk_aggr)== True){
      Add_Element(Outlier_Set,i); /* Add the element to Outlier_Set */
    }
}
return Outlier_Set.
```

Algorithm 2: Pseudo-code for route outlier detection

4 Analytical Evaluation and Cost Models

The computation of outlier detection algorithms is dominated by the operation Find_Neighbor_Nodes_Set(), which is determined by the clustering efficiency (CE) parameter of disk page clustering. In this section, we provide simple algebraic cost models for the I/O cost of outlier detection operations, using the CE measure of physical page clustering methods. The CE value is defined as follows:

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

The CE value is determined by the disk page clustering method, the data record size, and the disk page size. Figure 7 gives an example of CE 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 CE value of this graph can be calculated as $6/9 = 0.66$.

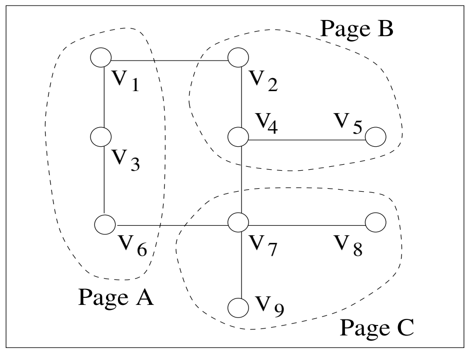


Figure 7: Example of Clustering Efficiency (CE)

Symbol	Meaning
α	The CE 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 3: Symbols used in the Cost Analysis

Table 3 lists the symbols used to develop our cost formulas. α is the CE 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.

Lemma 4 *The cost function for the Model Building algorithm is* $C_{MB} = \frac{N}{\beta} + N * \Lambda * (1 - \alpha)$

Proof: The Model Building 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. 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 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 $N * \Lambda * (1-\alpha)$. ■

Lemma 5 *The cost function for the* ROD *algorithm is* $C_{ROD} = L*(1-\alpha)+L*\Lambda*(1-\alpha) = L*(1-\alpha)*(1+\Lambda)$

Proof: Assume two memory buffers are used for the ROD algorithm; one memory buffer is reserved for processing the node x to be verified, and the other is used to process the neighbors of x. 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. ■

Lemma 6 *The cost function for the* RNV *algorithm is* $C_{RNV} = R + R * \Lambda * (1-\alpha)$

Proof: Suppose two memory buffers are used for the RNV algorithm; one memory buffer is reserved for processing the node x to be verified, and the other is used to process the neighbors of x. 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 the spatial outlier detection algorithm, clustering efficiency(CE) is a dominant factor for computation cost. The CE value is determined by the disk page clustering method. In this section, we describe the layout of our experiments and illustrate the candidate data clustering methods.

5.1 Experimental Layout

The design of our experiments is shown is Figure 8. The input data set, a Twin Cities Highway Connectivity Graph, was provided by the Minnesota Department of Transportation and physically stored into data pages using different page clustering strategies and page sizes. These data pages were then precessed to generate sets of pages of data to be used by the Model Building algorithm and Test Result Computation algorithm. The Model Building algorithm computes the algebraic aggregate functions to be used by the Test Result Computation algorithm to detect spatial outliers.

We compared three different data page clustering schemes: the Connectivity-Clustered Access Method (CCAM) [25], Z-ordering [19], and Cell-tree [7]. 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 experimental measures for the Model Building procedure and the Test Result Computation procedures are the CE value and I/O cost.

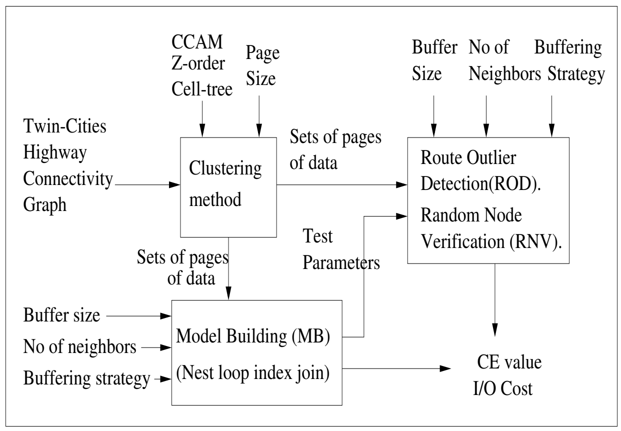


Figure 8: Experimental Layout

The experiments were conducted on many spatial frameworks. We present the results on a representative framework, 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. The size of each record is 256 bytes.

5.2 Candidate Clustering Methods

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

Connectivity-Clustered Access Method(CCAM): CCAM [25] clusters the nodes of the graph via graph partitioning, e.g., Metis [11, 12]. 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. In Figure 9, a simple graph and its CCAM are shown. The left half of Figure 9 shows a spatial graph. Nodes are annotated with the node-id and geographical coordinates. The node-id is an integer representing the Z-order of the (x,y) coordinates. For example, the node with the coordinates (1,1) gets a node-id of 3. The solid lines that connect nodes represent edges. The dashed lines show the cuts and partitioning of the spatial graph into data pages. The partitions are (0,1,4,5), (3,6,8,9), (7,12,13,18), and (11,14,15,26). The right half of Figure 9 shows the data pages and the secondary index. Nodes in the same partition set are stored on the same data page.

Linear Clustering by Z-order: Z-order [19] 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 a search. Figure 10 shows an example of using Z-order as the page clustering method and B-tree as the primary index for accessing the data file. The nodes of different partitions are (0,1,3,4), (5,6,7,8), (9,11,12,13), and (14,15,18,26).

Cell Tree: A Cell tree [7] 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. Figure 11 is an example of using Cell Tree to cluster the node. The first level binary partition is line H1, and the second level partitions are lines H2 and H3. The partitions are (0,1,3,6), (4,5,7,18), (8,9,11,14), and (12,13,15,26).

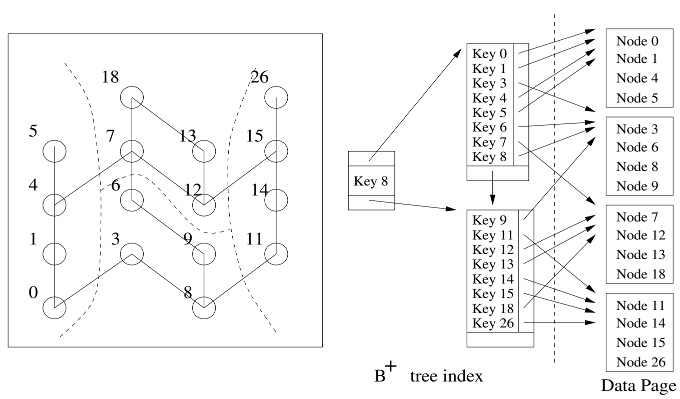


Figure 9: CCAM Clustering Method

5.3 Candidate Buffering Strategies

We evaluated three buffering strategies to replace the page in the memory buffer. The simplest page replacement algorithm is the First In First Out (FIFO) algorithm. A FIFO replacement algorithm marks the time when each page was bought into the 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, 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 CE 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.

Figure 12 shows one example of traffic flow outliers. Figures 12(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

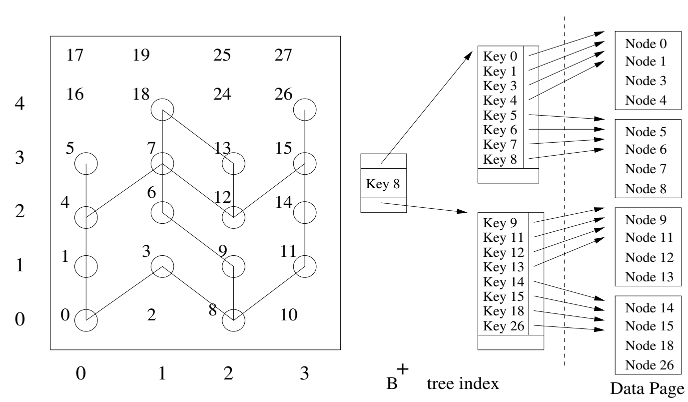


Figure 10: Z-order Clustering Method

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 12(a) exhibits inconsistent traffic flow compared with its neighboring stations, and was detected as a spatial outlier.

6.2 Evaluation of the Proposed Cost Model

We evaluated the I/O cost for different clustering methods for outlier detection procedures, namely, Model Building (MB), 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 4 shows the number of data page accesses for each procedure under various clustering methods. The CE value for each method is also listed in the table. The cost function for MB is $C_{MB} = \frac{N}{\beta} + N * \Lambda * (1 - \alpha)$. 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.

Clustering Method	Parameters Computation		Random Node Verification		Route Outlier Detect		$\alpha =$ CE
	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
$N = 773,\ L = 38,\ R = 150,\ \beta = 4,\ \Lambda = 2$							

Table 4: The Actual I/O Cost and Predicted Cost Model for Different Clustering Methods

As shown in Table 3, CCAM produced the lowest number of data page accesses for the outlier detection procedures. This is to be expected, since CCAM generated the

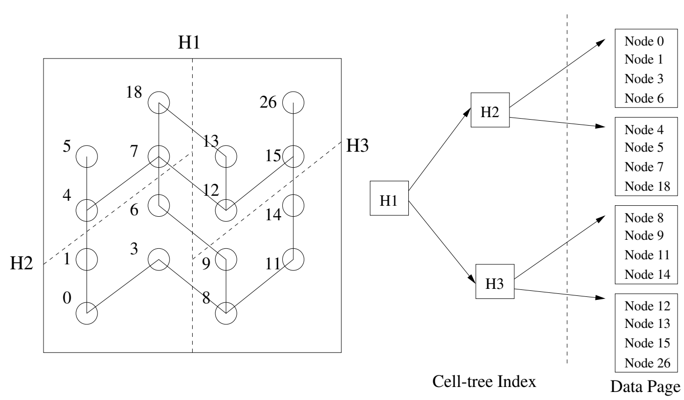
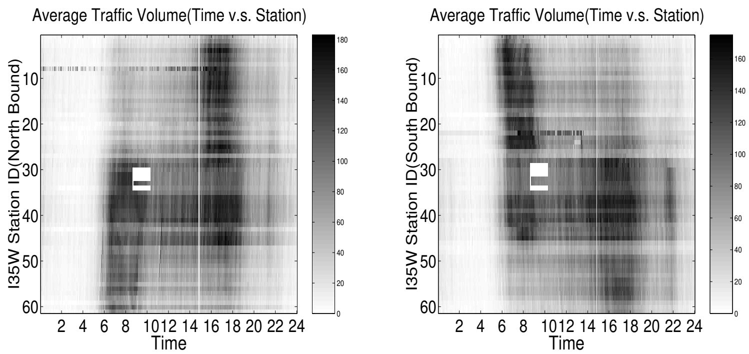


Figure 11: Cell-tree Clustering Method



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

Figure 12: An Example of an Outlier

highest CE value.

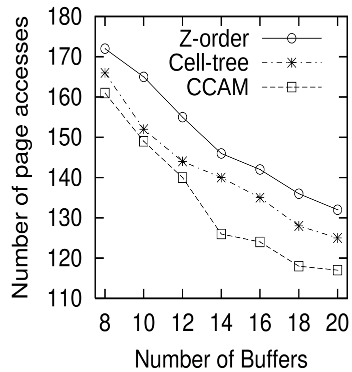
6.3 Evaluation of I/O Cost for the Model Building Algorithm

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

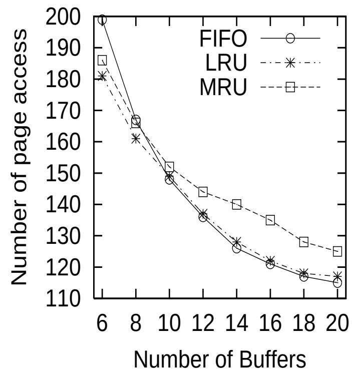
The Effect of Buffering: We evaluated the effect of buffering on the performance of the page clustering methods and buffer replacement strategies. The variable parameters were the number of buffers available. Figure 13(a) shows the effect of buffering on the performance of model construction for various clustering methods with fixed page size 2 Kbytes. 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 13(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.

The Effect of Page Size and CE Value: Figures 14 (a) and (b) show the number of data pages accessed and the CE 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 CE 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.

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. The neighborhood depth defines the levels of the neighborhood relationship. When the neighborhood depth D is set to one, only directly connected nodes are considered as



(a) Page Clustering Methods



(b) Buffer replacement strategy

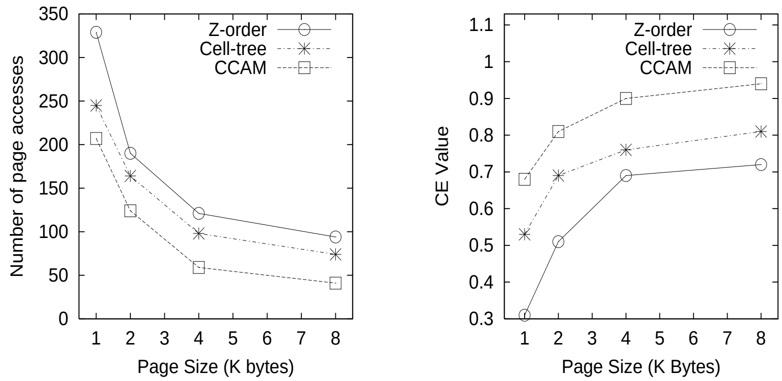
Figure 13: Effect of Buffering

neighbors; when D is set to be greater than one, then node n_2 is considered a neighbor of node n_1 provided there is a path connecting n_1 to n_2 with number of edges less than or equal to D. For example, there are three nodes, n_a, n_b, and n_c, with directed graph relationship: edge(n_a,n_b) and edge(n_b,n_c). If the neighborhood depth is set to two, node n_c will be considered as neighbor of node n_a due to a path of length two via node n_b. We fixed the page size at 1 Kbytes, buffer size at 4 Kbytes, and used the LRU buffering strategy. Figure 15 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 15 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.

6.4 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. We also evaluated the performance of RNV algorithm. The experiment results showed the similar trend as the ROD algorithm.

The effect of buffering: We evaluated the effect of buffering for the outlier detection along a route. Figure 16 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.



(a) Page Accesses

(b) Clustering Efficiency (CE)

Figure 14: Effect of Page Size on Data Page Accesses and Clustering Efficiency (Buffer Size = 32 Kbytes)

The effect of page size and CE value: Figures 17 (a) and (b) show the number of data pages accessed and the CE 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 CE 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 CE value of 0 and generates the highest number of page accesses when page size is 0.5 Kbytes and record size is 256 bytes. Cell-tree clusters stations by Euclidean distance even when there is no edge connecting the stations. This can lead to low CE values and CE value of 0 when each data page (disk block) can hold only two records.

7 Conclusions and Future Work

In this paper, we focus on detecting spatial outliers in spatial data sets. We propose a definition of S-outliers which generalizes traditional spatial outliers; we also analyze computation structures for detecting spatial outliers, design efficient algorithms to detect outliers, provide cost models for 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 a Twin Cities traffic archival to show its effectiveness and usefulness.

We have evaluated alternative clustering methods for neighbor outlier query processing, including model building, random node verification, and route outlier detection. Our experimental results show that the connectivity-clustered access method (CCAM), which achieves the highest clustering efficiency (CE) value, provides the best overall performance.

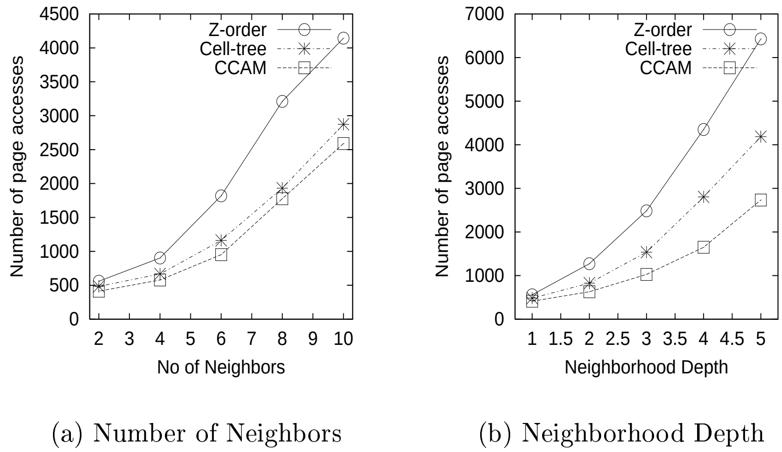


Figure 15: Effect of Neighborhood Cardinality on Data Page Accesses (Page Size = 1 Kbytes, Buffer Size = 4 Kbytes)

Our algorithm is designed to detect spatial outiers outliers using a single non-spatial attribute from a data set. 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.

We will also explore graphical methods for spatial outlier detection. The key issue is to facilitate the visualization of spatial relationships while highlighting spatial outliers. For instance, in variogram cloud and scatterplot visualizations, the spatial relationship between a single spatial outlier and its neighbors is not obvious. It is necessary to transfer the information back to the original map to check neighbor relationships. As a single spatial outlier tends to flag not only the spatial location of local instability but also its neighboring locations, it is important to group flagged locations and identify real spatial outliers from the group in the post-processing step.

Although spatial outlier detection is the focus of this paper, Figure 12 shows other types of outliers, such as temporal outliers and spatial-temporal outliers. While our proposed algorithm can efficiently detect spatial outliers, temporal and spatial-temporal outliers are detected by post-processing and data visualization. We are planning to investigate the definitions of temporal and spatial-temporal outliers, as well as to expand our algorithm to directly detect these outliers.

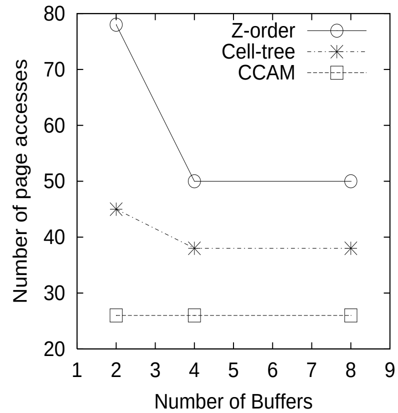


Figure 16: Effect of Buffering

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 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, by the National Science Foundation under grant 9631539, and by the Minnesota Department of Transportation and the Center for Transportation Studies at the University of Minnesota under grant 1725-5216306.

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.

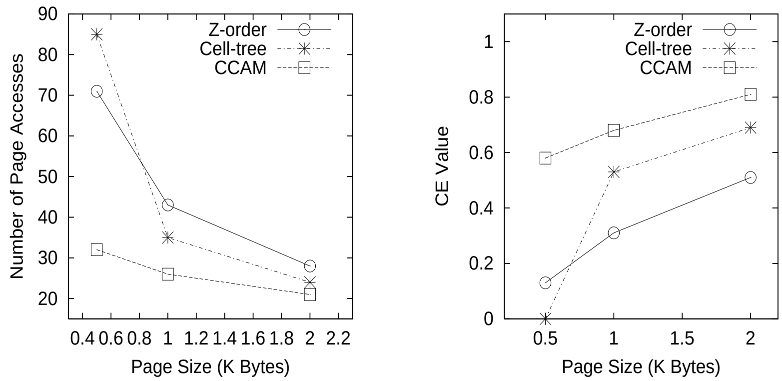
[3] M.M. Breunig, H.P. Kriegel, R. T. Ng, and J. Sander. Optics-of: Identifying local outliers. In *Proc. of PKDD '99, Prague, Czech Republic, Lecture Notes in Computer Science (LNAI 1704), pp. 262-270, Springer Verlag*, 1999.

[4] B.W. Lindgren. *Statistical Theory.* Chapman-Hall, 1998.

[5] D. Cook, J. Symanzik, and J.J. Majure. The Variogram Cloud Link. In *http://www.public.iastate.edu/~dicook/compgeo/VariogramCloudExample.html*, 1996.

[6] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-by, Cross-tab, and Sub-total. In *Proceedings of the Twelfth IEEE International Conference on Data Engineering*, pages 152–159, 1995.

[7] 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.



(a) Page Access

(b) Clustering Efficiency

Figure 17: Effect of Page Size on Data Page Accesses and Clustering Efficiency (Buffer Size = 32 Kbytes)

[8] R.P. Haining. *Spatial Data Analysis in the Social and Environmental Sciences*. Cambridge University Press, 1993.

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

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

[11] G. Karypis and V. Kumar. Metis Home Page. *http://www-users.cs.umn.edu/ ~karypis/metis/metis/main.html*.

[12] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

[13] 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.

[14] E. Knorr and R. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *Proc. 24th VLDB Conference*, 1998.

[15] Edwin M. Knorr, Raymond T. Ng, and V. Tucakov. Distance-Based Outliers: Algorithms and Applications. *VLDB Journal*, 8(3-4):237–253, 2000.

[16] Anselin Luc. Exploratory Spatial Data Analysis and Geographic Information Systems. In M. Painho, editor, *New Tools for Spatial Analysis*, pages 45–54, 1994.

[17] Anselin Luc. Local Indicators of Spatial Association: LISA. *Geographical Analysis*, 27(2):93–115, 1995.

[18] Minnesota Department of Transportation Traffic Management Center. http://www.dot.state.mn.us/tmc/tmchome.htm.

[19] 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.

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

[21] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, volume 29, pages 427–438. ACM, 2000.

[22] I. Ruts and P. Rousseeuw. Computing Depth Contours of Bivariate Point Clouds. In *Computational Statistics and Data Analysis, 23:153–168*, 1996.

[23] S. Shekhar and S. Chawla. *A Tour of Spatial Databases*. Prentice Hall, 2002.

[24] S. Shekhar, S. Chawla, S. Ravada, A. Fetterer, X. Liu, and C.T. Lu. Spatial Databases: Accomplishments and Research Needs. *IEEE Transactions on Knowledge and Data Engineering*, 11(1):45–55, 1999.

[25] S. Shekhar and D-R. Liu. CCAM: A Connectivity-Clustered Access Method for Aggregate Queries on Transportation Networks. *IEEE Transactions on Knowledge and Data Engineering*, 9(1):102–119, January 1997.

[26] S. Shekhar, C.T. Lu, X. Tan, S. Chawla, and R. R. Vatsavai. Map Cubes: A Visualization Tool for Spatial Data Warehouses. In *Geographic Data Mining and Knowledge Discovery, Harvey Miller and Jiawei Han (eds.)*. Taylor and Francis, 2001.

[27] Michael F. Worboys. *GIS - A Computing Perspective*. Taylor and Francis, 1995.

[28] D. Yu, G. Sheikholeslami, and A. Zhang. Find-Out: 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.

A Characterizing the Distribution of the Statistic

Theorem 2 *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:

For a spatially referenced object x_1, let X_1 be a random variable from the normally distributed attribute function $f(x) \sim N(\mu, \sigma^2)$, where μ is the mean and σ is the standard deviation.

$x_{11}, x_{12}, \ldots, x_{1k}$ are k neighbors of x_1. Attribute variables $X_{11}, X_{12}, \ldots, X_{1k}$ of objects $x_{11}, x_{12}, \ldots, x_{1k}$ are normally distributed from $N(\mu_{1i}, \sigma_{1i}^2), 1 \le i \le k$ respectively. Now let us consider two conditions of neighborhoods as follows:

(1)Assume i.i.d. in the local neighborhood window

$$(X_1, X_{11}, \ldots, X_{1k})^T \sim N\left(\begin{pmatrix} \mu_1 \\ \mu_{11} \\ . \\ \mu_{1k} \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & 0 & . & 0 \\ 0 & \sigma_{11}^2 & . & 0 \\ 0 & 0 & . & 0 \\ 0 & 0 & 0 & \sigma_{1k}^2 \end{pmatrix}\right)$$

(2)Consider the spatial correlation in the local neighborhood window

$$(X_1, X_{11}, \ldots, X_{1k})^T \sim N\left(\begin{pmatrix} \mu_1 \\ \mu_{11} \\ . \\ \mu_{1k} \end{pmatrix}, \begin{pmatrix} \sigma_1^2 & \sigma_1\sigma_{11}\rho_{X_1,X_{11}} & . & . \\ . & \sigma_{11}^2 & . & . \\ . & . & . & . \\ . & . & . & \sigma_{1k}^2 \end{pmatrix}\right)$$

Based on the definition of neighborhood, for each spatially referenced object x, the average attribute values $E_{y \in N(x)}(f(y))$ of $x's$ k neighbors can be derived from $f(x)$. Since the attribute function $f(x)$ is normally distributed and an average of normal variables is also normally distributed[4], the average attribute values $E_{y \in N(x)}(f(y))$ over neighbors is also a normal distribution for a fixed cardinality neighborhood.

We apply a normal linear transformation to derive $S(x)$, and the transformation guarantees the normal distribution [4]. Let $\bar{X}_k$ be a random variable of $E_{y \in N(x)}$

$$S(x) = X_1 - \bar{X}_k = (1, -\tfrac{1}{k}, -\tfrac{1}{k}, \ldots, -\tfrac{1}{k}) \begin{pmatrix} X_1 \\ X_{11} \\ . \\ X_{1k} \end{pmatrix} \sim N(A\mu_k, A\Sigma A'),$$

where $A = (1, -\frac{1}{k}, -\frac{1}{k}, \ldots, -\frac{1}{k})$, $\mu_k = \begin{pmatrix} \mu_1 \\ \mu_{11} \\ . \\ \mu_{1k} \end{pmatrix}$, $\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_1\sigma_{11}\rho_{X_1,X_{11}} & . & . \\ . & \sigma_{11}^2 & . & . \\ . & . & . & . \\ . & . & . & \sigma_{1k}^2 \end{pmatrix}$

Standardization : $\frac{(X_1 - \bar{X}_k) - A\mu_k}{\sqrt{A\Sigma A'}} \sim N(0,1)$

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

B Aggregate Function

B.1 Distributive Aggregate Function

An aggregate function F is called distributive if there exists a function G such that the value of F for a data set can be computed by applying a G function to the value of F in each partition of the whole data set. In most cases, $F = G$. For example, in Figure 18, the computation of distributive aggregate functions for a two-dimensional matrix M, $F(M_{ij}) = G(F(C_j)) = G(F(R_i))$, where M_{ij} represents the elements of a two-dimensional matrix, C_j denotes each column of the matrix, and R_i denotes each row of the matrix. Consider the aggregate functions Min and $Count$. In the first example, $F = Min$ and $G = Min$, since $Min(M_{ij}) = Min(Min(C_j)) = Min(Min(R_i))$. In the second example, $F = Count$, $G = Sum$, since $Count(M_{ij}) = Sum(Count(C_j)) = Sum(Count(R_i))$. Other distributive aggregate functions include Max and Sum. Note that "null" valued elements are ignored in computing aggregate functions.

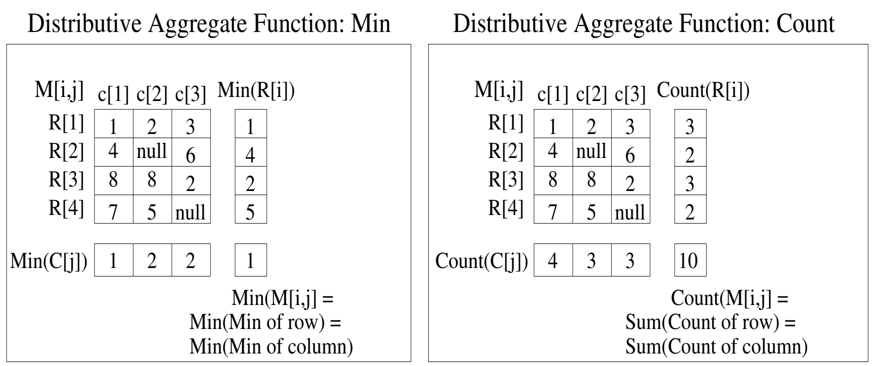


Figure 18: Computation of distributive aggregate functions

B.2 Algebraic Aggregate Function

An aggregate function F is algebraic if F of a data set can be computed using a fixed number of sub-aggregates from each partition of the data set. *Average*, *variance*, *standard deviation*, $maxN$, $minN$ are all algebraic aggregate functions. In Figure 19, for example, the computations of *average* and *variance* for a two-dimensional matrix M are shown. The average of elements in the data set M can be computed from *sum* and *count* values of the one dimensional sub-matrix (e.g., rows or columns). The *variance* can be derived from the *count*, *sum*(i.e. $\sum_i X_i$), and *sum of sq*(i.e. $\sum_i X_i^2$) of rows or columns. Similar techniques apply to other algebraic functions.

B.3 Holistic Aggregate Function

An aggregate function F is called holistic if the value of F for a data set cannot be computed using a constant number of sub-aggregates from each partition of the data set.

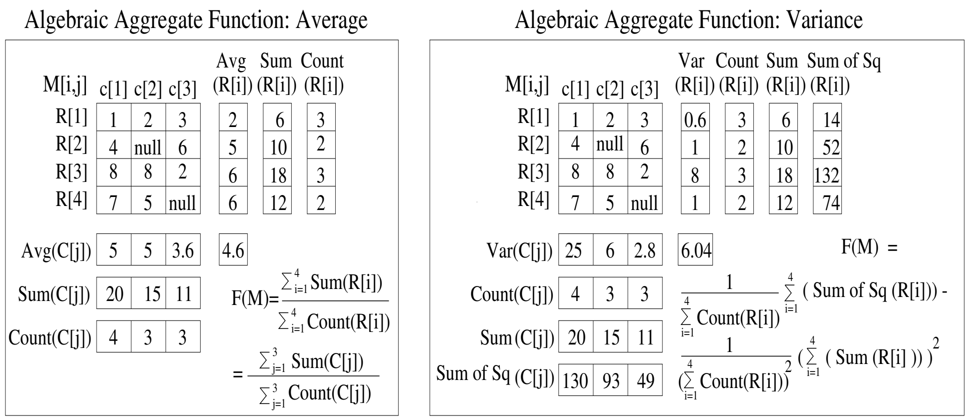


Figure 19: Computation of algebraic aggregate functions

To compute the value of F, we need to access the whole data set. Examples of holistic function include $median$, $mostFrequent$, and $rank$.