19th IEEE International Conference on Tools with Artificial Intelligence

# Spatial Outlier Detection: A Graph-based Approach

Yufeng Kou, Chang-Tien Lu, Raimundo F. Dos Santos Jr.

Department of Computer Science
Virginia Polytechnic Institute and State University
7054 Haycock Road, Falls Church, VA 22043
[ykou, ctlu, rdossant]@vt.edu

**Abstract**

*Spatial outliers are the spatial objects whose nonspatial attribute values are quite different from those of their spatial neighbors. Identification of spatial outliers is an important task for data mining researchers and geographers. A number of algorithms have been developed to detect spatial anomalies in meteorological images, transportation systems, and contagious disease data. In this paper, we propose a set of graph-based algorithms to identify spatial outliers. Our method first constructs a graph based on k-nearest neighbor relationship in spatial domain, assigns the nonspatial attribute differences as edge weights, and continuously cuts high- weight edges to identify isolated points or regions that are much dissimilar to their neighboring objects. The proposed algorithms have two major advantages compared with the existing spatial outlier detection methods: accurate in detecting point outliers and capable of identifying region outliers. Experiments conducted on the US Housing data demonstrate the effectiveness of our proposed algorithms.*

## 1 Introduction

A spatial outlier is a spatially-referenced object whose nonspatial attribute values are significantly different from those of other spatially-referenced objects in its spatial neighborhood [16]. In contrast to traditional outliers, spatial outliers are local anomalies that are extreme compared to their neighbors [2], but do not necessarily deviate from the remainder of the whole data set. Informally, spatial outliers can be called "local outliers," because they focus on local differences, while traditional outliers can be called "global outliers," since they are based on global comparison. In certain situations, an outlier may not appear as a single spatial object but in the form of a group of adjoining objects, i.e., a region. The nonspatial attribute values of the spatial objects within this region are similar, but they are significantly different from the nonspatial attribute values of the objects surrounding this region. We name this type of outliers as "region outliers [22]."

The identification of spatial outliers can reveal hidden but valuable information in many applications. For example, it can help locate severe meteorological events, identify aberrant genes or tumor cells, discover highway congestion segments, pinpoint military targets in satellite images, determine potential locations of oil reservoirs, and detect water pollution incidents. For spatial outlier detection, the attribute space can be separated into two components, spatial and nonspatial attributes. Spatial attributes are used to determine the relationship of spatial neighborhood. Then nonspatial attributes are compared among spatial neighbors to identify local instabilities which break spatial autocorrelation and continuity.

The outlierness of a spatial object can be evaluated by comparing the nonspatial attribute values of this object with those of its $k$-nearest neighbors ($kNN$). The entire data set can be represented with a number of intra-connected subgraphs based on the $kNN$ relationships. $k$ directed edges can be drawn from each object to its $k$-nearest spatial neighbors, and the weight of an edge denotes the absolute nonspatial difference between two neighboring nodes. In this paper, we propose two algorithms to detect both point outliers and region outliers based on the $kNN$ graph. The proposed algorithms have two major advantages compared with the existing spatial outlier detection methods: accurate in detecting point outliers and capable of identifying region outliers.

The rest of the paper is organized as follows. Section 2 discusses the related work in spatial outlier identification; Section 3 defines the problem of spatial outlier detection and provides the motivation of our approaches; Section 4 presents two spatial outlier detection algorithms based on $kNN$ graph; Section 5 illustrates the experimental design and results; and finally Section 6 summarizes our work and points out future research directions.

## 2 Related Work

Outliers have been extensively studied in the past decades and numerous methods have been developed, including distance-based methods [5, 14], cluster-based methods [11, 21], depth-based methods [13, 15], and distribution-based methods [20]. Along with the fast development of geospatial information services and the wide usage of digital images, identification of outliers in spatial data has received more and more attention. Traditional outlier detection methods may not be efficiently applicable to spatial data. First, spatial data generally possess complex data formats such as lines and polygons which can be combined

1082-3409/07 $25.00 © 2007 IEEE
DOI 10.1109/ICTAI.2007.139

to form more complex objects. Second, spatial data often exhibit spatial autocorrelation where "everything is related to everything else, but nearby things are more related than distant things [19]."

To mine anomalies from spatial data, many methods have been proposed. Visualization methods illustrate the neighborhood distribution in a figure and identify particular points as spatial outliers. These methods include variogram cloud, pocket plot, scatterplot, and Moran-scatterplot [3, 4, 10, 12]. A Scatterplot [3] shows attribute values on the $X$-axis and the average of the attribute values in the neighborhood on the $Y$-axis. Nodes far away from the regression line are flagged as potential spatial outliers. A Moran scatterplot [10] normalizes attribute values against the neighborhood average of values. Other algorithms, such as $z$-value, perform statistical tests to discover local inconsistencies [8, 9]. $z$-value is the normalized difference between a spatial object and the average of its spatial neighbors [18]. The absolute $z$-value can determine the outlierness of an object where higher $z$-values indicate a higher likelihood that an object is a spatial outlier. Shekhar *et al.* provided a unified definition of spatial outlier detection and prove that this definition generalizes the existing algorithms such as z-value, Scatterplot, and Moran Scatterplot [18]. Lu *et al.* proposed algorithms to detect spatial outliers with multiple nonspatial attributes using Mahalanobis distance [9].

Spatial data have various formats and semantics. Kou *et al.* developed spatial weighted outlier detection algorithms which use properties such as center distance and common border length as weight when comparing nonspatial attributes [6]. Adam *et al.* proposed an algorithm which considers both spatial relationship and semantic relationship among neighbors [1]. Liu and Jezek proposed a method for detecting outliers in an irregularly-distributed spatial data set [7]. The outlierness of an object $o$ is measured by both the spatial interpolation residual and surface gradient of its neighborhood. Shekhar *et al.* proposed an outlier detection method to identify anomalies in transportation network [17]. Their methods are based on road network connectivity and temporal neighborhoods based on time series.

## 3 Problem Formulation and Motivation

In this section, we formally define the problem and provide the motivation of the graph-based algorithms.

### 3.1 Problem Definition of Spatial Outlier Detection

We formalize the spatial outlier detection problem as follows.

**Given:**

- **X** is a set of spatial objects $\{x_1, x_2, \ldots, x_n\}$ with single or multiple nonspatial attributes, where $x_i \in \Re^d$.
- **k** is an integer denoting the number of adjacent data objects which form the neighborhood relationship. Every object $x_i$ has k neighbor objects based on its spatial location, denoted as $NN_k(x_i)$.
- **Y** is a set of attribute values $\{y_1, y_2, \ldots, y_n\}$, where $y_i$ is the nonsptial attribute value of $x_i$.
- **m** is the number of outliers to be identified; generally $m \ll n$.

**Objective:**

- Design a mapping function $f : (X, Y, k) \longrightarrow G(X, E)$. $G(X, E)$ is a graph, where each node is an object in $X$ and $E$ is the set of edges. The edge between two nodes reflects the difference of their non-spatial attributes.
- Find a partition strategy which continuously segments graph $G(X, E)$, to obtain a set $Z$ of m data objects where $Z \subset X$ and for $\forall x_i \in Z$, $x_i$ is disconnected with all other objects.

### 3.2 Motivation

Existing spatial outlier detection algorithms have several deficiencies. **First**, As described in [8], if an object has exceptionally large or small nonspatial attribute values, it will have negative impact on its spatial neighbors, such that some of them may be falsely marked as outliers and the "true" outliers may be overlooked. Figure 1 shows an example spatial data set, where x and y coordinates denote the 2D spatial locations and z coordinate represents the value of nonspatial attributes. In this data set, four spatial outliers are expected, $S1$, $S2$, $S3$, and $S4$, because their nonspatial attribute values are much larger or smaller than their spatial neighbors. Table 1 shows the outlier detection results of several algorithms on the data set in Figure 1, including $z$-value algorithm, Moran-scatterplot, and scatterplot. We can observe that none of the three can accurately identify all four "true" outliers. Moreover, they identify a "false" outlier, $E1$, which is in fact a normal point. $E1$ is erroneously detected because the nonspatial attribute value of its neighbor $S1$ is extremely high (200), and therefore dominates the neighborhood average of $E1$. Figure 2(a) and Figure 2(b) demonstrate scatterplot and Moran scatterplot of the example data set.

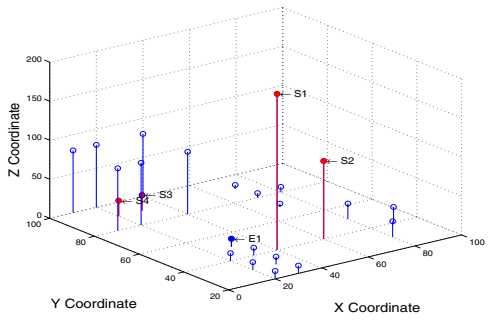
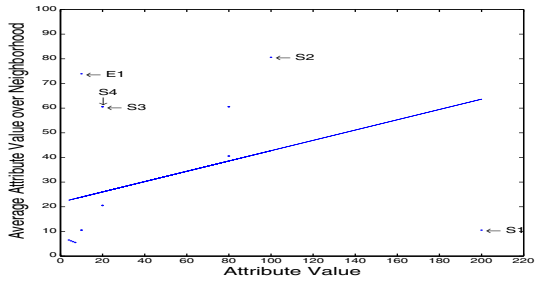



Figure 1: An example data set, where X and Y coordinates denote the spatial locations and Z coordinate represents the value of nonspatial attribute.

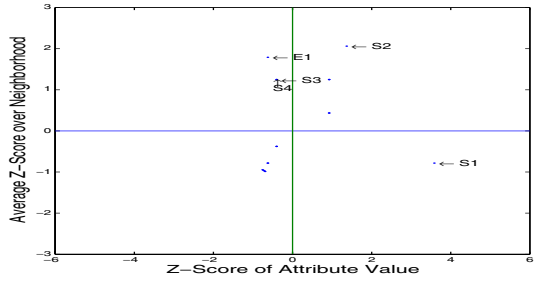
| Rank | Methods | | | |
|---|---|---|---|---|
| | Scatter-plot | Moran Scatterplot | $z$ Alg. | Graph-based |
| 1 | S1 | S1 | S1 | S1 |
| 2 | E1 | E1 | E1 | S2 |
| 3 | S2 | S3 | S3 | (S3,S4) |

Table 1: Top three spatial outliers detected by scatterplot, Moran scatterplot, $z$-value, and graph-based method.

In Figure 2(a), the points far from the regression line are marked as outliers. In Figure 2(b), the points located in the upper left and lower right quadrants are identified as outliers. Both scatterplot and moran-scatterplot mistakenly detect $E1$ as an outlier.




(a) Scatterplot




(b) Moran-Scatterplot

Figure 2: The Scatterplot and Moran-scatterplot of the data set shown in Figure 1.

**Second**, most of the existing algorithms focus on evaluating the outlierness of single object and are not suitable for detecting region outliers. If a group of outlying objects cluster together and have similar nonspatial attribute values, they might be erroneously marked as normal points. As shown in Table 1, neither scatterplot, Moran-scatterplot, nor $z$-value approach can identify the region outlier (S3,S4). In many applications, spatial outliers appear in the form of connected components, or regions. For example, a hurricane zone consists of a group of points whose wind speed, air pressure, and water vapor density are much different from the surrounding areas.

**Third**, some existing methods such as $z$-value approach and Moran-scatterplot identify spatial outliers by calculating the normalized non-spatial attribute difference between each object and the average of its spatial neighbors. This normalization is across the entire data set, which may not be appropriate for certain conditions. For example, if the data set consists of a number of spatial clusters, the objects in the same cluster are spatially correlated to each other and the objects in different clusters have no direct correlations. Therefore, normalization based on the entire data set may not be suitable. One potential solution for this issue is to first identify the object clusters in the data set, and then consider statistical characteristics of each cluster individually to detect outliers. To address the above deficiencies, we propose a set of graph-based algorithms to accurately identify both point outliers and region outliers. Based on the example data set in Figure 1, we first construct a $kNN$ graph as shown in Figure 3. Each circle represents an object whose nonspatial attribute value is marked within the circle. An arrow pointing from a node $x_1$ to another node $x_2$ represents that $x_2$ is one of $x_1$'s $k$ nearest neighbors. If $x_1$ and $x_2$ are connected by a double directed edge, they are $k$-nearest neighbors of each other. The edge weight is the absolute difference between the nonspatial attribute values of two neighboring nodes. For example, the weight of the edge connecting $S1$ and $S2$ is $100(200 - 100)$. Note that there are 3 subgraphs in the $kNN$ graph. To make the edges comparable among different subgraphs, a standardization process can be performed for each subgraph independently. The proposed graph-based algorithms then continuously cut the longest edge, i.e., the edge with the largest weight until certain number of points or connected regions have been isolated. The isolated points will be marked as point outliers since their nonspatial attribute values are much different from those of their neighbors. The regions will be marked as region outliers once the verification procedure validates that their degrees of outlierness are beyond thresholds.

Figure 4 shows the edge-cut result of Figure 3, where three outliers are identified, $S1$, $S2$, and ($S3$, $S4$). One of them consists of two spatial objects, forming a region outlier. ($S3$, $S4$) is labelled as a region outlier because they have been isolated from other points and their nonspatial attribute values are similar to each other but dissimilar to those of their neighbors. A region is marked as "isolated" when there are no edges going out from each point in that region. The neighbors of a region consist of all points which have directed edges going out of that region. Table 1 shows that the graph-based method can detect all expected spatial outliers and is capable of identifying the outlier region. In summary, the graph-based algorithms have the following advantages compared with the existing spatial outlier detection methods. (1) Detect both point and region outliers. Based on graph connectivity, neighboring points with similar nonspatial attribute values can be grouped conveniently; (2) Avoid identifying "false" outliers and correctly locate "true" outliers; (3) More "local". We only consider the outlierness of an object within a subgraph where this object belongs to, instead of normalizing data across the entire data set.

## 4 Algorithm

In this section, we propose two graph-based algorithms. The first algorithm $POD$ is designed for point outlier detection, and the second algorithm $ROD$ is to identify region outliers.

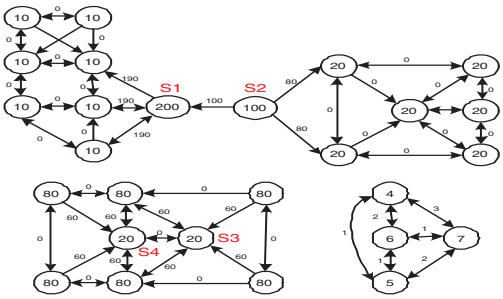



Figure 3: The $kNN$ based graph representation of a small data set,$k = 3$.

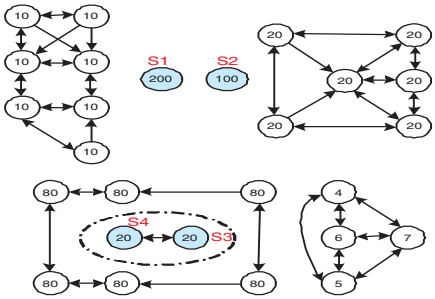



Figure 4: The result of graph-based outlier detection: 2 point outliers and 1 region outlier are identified.

Computational complexities of both algorithms are examined as well.

The major step of spatial outlier detection is to compare the nonspatial values between each object and its spatial neighbors. Thus, a graph $G(X, E)$ can be created based on the $k$-nearest neighbors ($kNN$) relationship. Each node represents an object $x_i$, and $k$ edges direct from $x_i$ to each of its $k$-nearest neighbors. The weight of an edge represents the dissimilarity between two neighboring points. For single (nonspatial) attribute data set, we can use the absolute difference of the nonspatial attribute as the weight. For a data set with multiple (nonspatial) attributes, we can use the Mahalanobis distance between two nodes as the weight. The outgoing degree of each node is $k$, the number of neighbors. Note that the graph may consist of a set of subgraphs which are not connected to each other. We call each subgraph as a cluster and consider their different statistical characteristics in spatial outlier detection.

The data structure for the graph is shown in Figure 5. Each node (object) $x_i$ is represented by a tuple with $3k + 3$ elements, where $x_i$=$\langle\, id, val, nbr_1, diff_1, flag_1, \ldots, nbr_k, diff_k, flag_k, cID \,\rangle$. $id$ denotes the unique identification of a node; $val$ is the nonspatial attribute value of $x_i$; $nbr_j(1 \leq j \leq k)$ represents the $j$-th neighbor of $x_i$; $diff_j$ stands for the weight of the edge $\overrightarrow{x_i, x_j}$; $flag_j$ indicates the status of edge $\overrightarrow{x_i, x_j}$ ($flag_j = 1$ denotes that the edge has been cut and $flag_j = 0$ represents that $x_i$ and $x_j$ are still connected); $cID$ refers to the subgraph which $x_i$ belongs to. The neighbor list is sorted by their edge weights

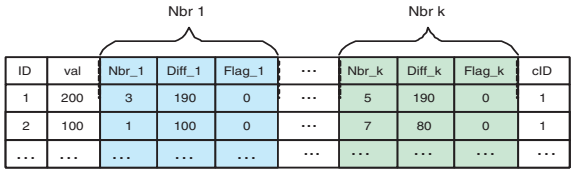



Figure 5: The data structure for graph representation.

in descending order, *i.e.*, $diff_1 \geq diff_2 \geq \ldots \geq diff_k$. The reverse $k$ nearest neighbors($rKNN$) of each object also need to be stored. The $rKNN$ of an object $x_i$ is a set of objects whose $k$ nearest neighbors contains $x_i$. We can use a data structure similar to Figure 5 to store the $rKNN$ of each node. The only difference is that the number of objects in $rKNN$ may not be fixed as $k$, so the tuple for each node is a variant-length array. The reverse nearest neighbor table can be contructed from the nearest neighbor table and used for extracting connected regions from the graph.

In addition, a priority queue $EdgeQueue$ is created to maintain all edges, which are sorted based on their weights in descending order. An array $clusterArr$ contains a set of subgraphs. Each subgraph is represented as a four element tuple $cluster_i$, where $cluster_i$=<$cID$, $mean$, $std$, $size$>. $cID$ is the unique identification of a subgraph; $mean$ is the average nonspatial attribute value of all objects in the subgraph; $std$ is the standard deviation of nonspatial attribute values in this subgraph; and $size$ is the number of objects in the subgraph. These statistics can be used for normalizing edge weights in each subgraph.

### 4.1 Algorithm 1: Point Outlier Detection

This section introduces an algorithm to detect single point outliers based on $kNN$ graph. We assume that all $k(x_i)$ are equal to a fixed number $k$. (The algorithm can be easily generalized by replacing the fixed $k$ by a dynamic $k(x_i)$.)

The proposed algorithm has four input parameters. $X$ is a set of $n$ objects containing spatial attributes such as location, boundary, and area. The non-spatial attributes are contained in another set $Y$. In many applications, these nonspatial attribute values can be the results of preprocessing procedure like dimension reduction or data standardization. $k$ is the number of neighbors. $m$ is the number of requested outliers. Generally, $m$ should not be greater than 5% of $n$, assuming the nonspatial attribute follows normal distribution with confidence interval of 95%.

For each data object, the first step is to identify its $k$ nearest neighbors. Calculating the Euclidean distance between the centers of two objects is the most commonly used approach. Next, based on the $k$-nearest neighbor set $NN_k$, a graph $G$ is constructed. $G$ may contain multiple subgraphs which are disconnected with each other. Since different subgraphs may have distinct characteristics, the nonspatial attribute values need to be standardized first within each subgraph. In this way, the edge weights are comparable among different subgraphs. Next, a priority queue $edgeQueue$ is created to arrange all edges in descending order based on their weights. To partition

**Algorithm 1** : Point Outlier Detection (POD)

**Input**:
$X$ is the set of n spatial objects;
$Y$ is the set of nonspatial attribute values for $X$;
$k$ is the number of neighbors;
$m$ is the number of requested spatial outliers;
**Output**:
$O_s$ is a set to store detected spatial outliers

for(i=1; i $\leq n$ ; i++) {
/* *calculate the neighborhood relationship* */
$NN_k(x_i)$ = GetNeighbors($X$, $x_i$); }
/* Generate a graph $G$ based on $NN_k$ */
$G$ = createGraph($X$, $Y$, $NN_k$);
/* standardize the attribute values by subgraph */
$G$ = standardize($G$);
/* create the edge priority queue */
$EdgeQueue$ = createEdgeQueue($G$);
$O_s = empty$; /* initialize $O_s$ */
while ( $sizeOf(O_s) \leq m$ ) {
edge = DeQueue($EdgeQueue$); /* select the longest edge */
/* cut this edge and return the starting node */
o = CutEdge($G, edge$);
/* check if o has become an isolated point*/
if ( isIsolated($o$) ) {
MarkOutlier($O_s$, $o$); } }
/* output the outliers */
Output($O_s$);

the graph, a "longest-edge-cut" strategy is employed and the partition will be conducted with multiple iterations. In each iteration, the longest edge $\overrightarrow{x_i, x_j}$ (the edge with the highest weight) is dequeued from $edgeQueue$ and cut. Here, cutting an edge is equal to set the $flag_j$ of the node $x_i$ as "1." The $CutEdge$ function returns the starting node of the edge. After each cutting operation the algorithm will check if this starting node $o$ has become isolated, i.e., no outgoing edges. If true, $o$ is marked as a spatial outlier and inserted into the outlier set $O_s$. Finally, when all $m$ spatial outliers are identified, the partition loop stops and $O_s$ is output. The ranking of outliers is determined by their identification sequence.

### 4.2 Algorithm 2: Region Outlier Detection

One major benefit of the graph-based method is that it can detect region outliers. A region outlier consists of a group of adjoining spatial objects whose nonspatial attributes are similar to each other but quite different from the surrounding objects of this group. In a partitioned graph, a connected component can be deemed as a region outlier if the variance in the component is small and the variance between the component and its neighborhood is large. Algorithm 2 is designed especially for discovering region outliers. For simplicity, we call it $ROD$ algorithm. $ROD$ has 6 input parameters: $X$ contains the spatial attributes of the data set; $Y$ contains the nonsptial attribute values of the data set; $k$ is the number of neighbors; $T_{EDGE}$ denotes the threshold for edge-cutting; $T_{SIM}$ represents the threshold for evaluating the evenness within a region; $T_{DIFF}$ is the threshold to measure the difference between a region and its neighbors. The detailed algorithm is described as follows. Similar to Algorithm 1, a graph first needs to be created based on the $kNN$ relationship. Then a priority queue $edgeQueue$ is to rank all edges in descending order based on their weights. Next, the algorithm continuously selects the longest edge from the $edgeQueue$ and cuts it if the weight of this edge is larger than threshold $T_{EDGE}$. After cutting all edges with weight above $T_{EDGE}$, a function is called to detect the candidate region outliers, $regions$. Each of these candidates is a connected region which contains less than $k$ objects. $regions$ is only a candidate set, and each region in this set needs further verification. The verification includes three steps: (1) for a region $region[i]$, determine the set of its spatial neighbors, $nbrs$, which contains all the distinct k-nearest neighbors of each point in $regions[i]$. If the size of $nbrs$ is smaller than the size of $regions[i]$, then $region[i]$ is not a region outlier; (2) check the evenness of nonspatial attribute values in $regions[i]$. If the degree of evenness is smaller than the threshold $T_{SIM}$, $regions[i]$ should not be identified as an outlier; (3) compute the nonsptial attribute value difference between $region[i]$ and $nbrs$.

**Algorithm 2** : Region Outlier Detection (ROD)

**Input**:
$X$ is a set of n spatial objects;
$Y$ is the set of nonspatial attribute values for $X$;
$k$ is the number of neighbors;
$T_{EDGE}$ is the threshold for the edge weights to be cut;
$T_{SIM}$ is the threshold of within region similarity;
$T_{DIFF}$ is the threshold of between-region difference;
**Output**:
$O_s$ is a set of region outliers

for(i=1; i $\leq n$ ; i++) {
/* *calculate the neighborhood relationship* */
$NN_k(x_i)$ = GetNeighbors($X$, $x_i$); }
/* Generate a $kNN$ graph $G$ */
$G$ = createGraph($X$, $Y$, $NN_k$);
/* create a priority queue based on edge weight */
$EdgeQueue$ = createEdgeQueue($G$);
$edge$ = DeQueue($EdgeQueue$); //select the longest edge
while ( $weightOf(edge) > T_{EDGE}$ ) {
$G$ = CutEdge($G, edge$); // cut edge in the graph
/* select the longest edge */
$edge$ = DeQueue($EdgeQueue$); }
/* find all regions with size less than k*/
$regions$ = findCandidateRegions($G$);
for(i=1; $i < sizeOf(regions)$ ; i++) {
if ( $regionEvenness(regions[i]) > T_{SIM}$ ) {
$nbrs$ = getNbrOfRegion($G, regions[i]$);
if ( $computeDiff(regions[i], nbrs > T_{DIFF}$) {
markOutlier($O_s$, $regions[i]$); } } }
/* output the outliers */
Output($O_s$);

If the difference is less than the threshold $T_{DIFF}$, $regions[i]$ is not an outlier. Once $regions[i]$ has been verified as a region outlier through the above procedures, it will be inserted into $O_s$.

There are two essential issues in setting the $ROD$ algorithm: how to evaluate the evenness within a region and how to measure the difference between a region and its neighborhood. The evenness of a region can be evaluated by "Coefficient of Variation" (dividing standard deviation by mean), by "Inter-Quartile Range" (difference between the first and third Quartiles), or by "MinMax Difference" ($\frac{Max-Min}{Mean}$). The variation between two regions can be measured using $diff = \frac{|m_r - m_{nbr}|}{(m_r + m_{nbr})/2}$, where $m_r$ is the median nonspatial attribute value of a region $r$ and $m_{nbr}$ is the median nonspatial attribute value of $r$'s neighbors. A high $diff$ value represents large difference between a region and its neighbors.

### 4.3 Time Complexity

In the $POD$ algorithm, a $k$ nearest neighbor ($kNN$) query is issued for each spatial point. It will take $O(n)$ to perform $kNN$ query for $n$ objects if a grid-based indexing is used and the grid directory resides in memory. The cost of generating a $kNN$ graph is $O(kn)$, and edge weight standardization takes $O(kn)$ computation. The priority queue generation has time complexity of $O((kn)log(kn))$ based on heap sort. The number of edge cuts is indeterministic, so we assume there are at most $n$ edge-cuts. The cost of checking "isolated" status of a point is $O(k)$, thus the cost for checking all edge cuts is $O(kn)$. In summary, assuming $n \gg k$ and $n \gg m$, the total time complexity of $POD$ algorithm is $O(n)$ + $O(kn)$ + $O(kn)$ + $O((kn)log(kn))$ + $O(kn) \approx O(nlogn)$ for grid-base indexing. The computation cost is primarily determined by priority queue generation.

For the $ROD$ algorithm, the time complexity of $kNN$ computation, graph generation, and edge weight priority queue creation is the same as that of the $POD$ algorithm. For edge cutting cost, we can assume the total number of edge-cuts is $n$, leading to complexity of $O(n)$. In addition, it takes $O(n)$ to identify candidate regions. The verification procedure costs $O(mk)$, where $m$ is the number of detected region outliers and $m \ll n$. Finally, the total time complexity of the $ROD$ algorithm is $O(n)$ + $O(kn)$ + $O((kn)log(kn))$ + $O(n)$ + $O(n)$ + $O(mk) \approx O(nlogn)$, which is also primarily determined by priority queue generation.

## 5 Experiment

In this section, we discuss our experiments on a real data set, Fair Market Rents data provided by the $PDR - DHUD$ (Policy Development and Research, U.S. Department of Housing and Urban Development), to validate the effectiveness of our proposed algorithms. Experimental design is introduced first, followed by result analysis.

### 5.1 Experiment Design

The Fair Market Rent data contain the 50th percentile rents for fiscal year 2005 at county level. This data include the rental prices for efficiencies, one-bedroom apartments, two-bedroom apartments, three-bedroom apartments, and four-bedroom apartments in 3000+ counties across the United States. The proposed algorithms can help administrative personnel identify and then investigate outlier counties whose rental prices are much different from their neighboring counties. Apartment renters can also explore the rent data to find a place where the rent is abnormal.

The location of each county is determined by the longitude and latitude of its center. The proposed algorithms facilitate the discovery of abnormal rent rates by comparing reports from neighboring counties. The number of neighbors was chosen to be $k = 8$, which represents eight different directions from the centering county: East, West, North, South, Northeast, Northwest, Southeast, and Southwest. The distance between a county and its neighbors is the Euclidean distance between their centers. The experiments contain two parts: point outlier detection and region outlier detection. For the point outlier detection methods, top ten outliers will be identified from 3095 counties.

### 5.2 Result Analysis

#### 5.2.1 Point outlier Detection

We applied four algorithms to the apartment rent data set, including $z$-algorithm, Scatterplot, Moran-scatterplot, and the proposed graph-based point outlier detection algorithm $POD$. Table 2 shows the top ten outlier counties based on one-bedroom rent in year 2005. The number in parenthesis denotes the average rent for one-bedroom apartments in the given county. $POD$ algorithm identifies seven common counties as does the $z$ algorithm, which is a commonly used approach.

$POD$ algorithm can avoid identifying "false" outliers. For example, Dorchester Co.(MD) is identified as the 7th outlier by MoranScatterplot algorithm. However, Dorchester Co. is not very outlying if we take a closer look. Figure 6 illustrates the rental prices of Dorchester Co. (451) and its 8 neighbors (575,576,513,1045,460,750,548,702). The number in each county denotes the average rent in US dollar. Generally, the rent difference within 140 dollar is viewed as normal. More than half of the neighboring counties have normal rent difference with Dorchester Co., so Dorchester Co. should not be identified as a spatial outlier. Dochester Co. is identified by Moran-scatterplot mainly because it has a neighbor, Calvert Co.(MD), which has a very high rent (1045) and significantly raises the average rent of the neighborhood. The $POD$ algorithm does not have this problem, because the edge-cutting makes each neighboring county contribute equally to the outlierness of the centering county. In addition, $POD$ algorithm can detect outliers that are not identified by other three methods. For example, Dukes Co.(MA) is ranked as the sixth outlier by $POD$ algorithm. As shown in Figure 7, the one-bed rent in Dukes Co. is 941, which is much higher than its 7 neighbors (difference $> 150$) except Nantucket

| Rank | Methods | | | |
|---|---|---|---|---|
| | $z$ Alg. | Scatterplot | MoranScatterplot | $POD$ Alg. |
| 1 | Nantucket Co.,MA(1250) | Nantucket Co.,MA(1250) | Blaine Co.,ID(801) | Blaine Co.,ID(801) |
| 2 | Pitkin Co.,CO(1095) | Caroline Co.,VA(495) | Teton Co.,WY(748) | Nantucket Co.,MA(1250) |
| 3 | Frederick Co.,MD(1045) | Blaine Co.,ID(801) | Elbert Co.,CO(444) | Teton Co.,WY(748) |
| 4 | Suffolk Co.,MA(1120) | Teton Co.,WY(748) | Surry Co.,VA(446) | Summit Co.,UT(901) |
| 5 | Blaine Co.,ID(801) | Elbert Co.,CO(444) | La Paz Co.,AZ(471) | Suffolk Co.,MA(1120) |
| 6 | Summit Co.,UT(901) | Plymouth Co.,MA(636) | Moffat Co.,CO(435) | **Dukes Co.,MA(941)** |
| 7 | Teton Co.,WY(748) | Worcester Co.,MA(549) | **Dorchester Co.,MD(451)** | Fairfield Co.,CT(1239) |
| 8 | Ventura Co.,CA(1093) | Summit Co.,UT(901) | Sumter Co.,FL(415) | Dane Co.,WI(660) |
| 9 | Fairfield Co.,CT(1239) | Barnstable Co.,MA(729) | Polk Co.,WI(465) | Centre Co.,PA(610) |
| 10 | Clarke Co.,VA(956) | Pitkin Co.,CO(1095) | Polk Co.,GA(414) | Pitkin Co.,CO(1095) |

Table 2: Top 10 spatial outliers detected by $z$-value, scatterplot, Moran scatterplot, and graph-partition algorithms based on 1-bedroom rent.

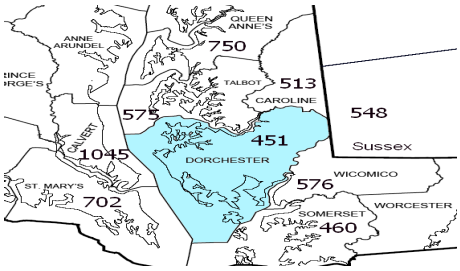



Figure 6: Dorchester Co.(MD) is detected by Moran-scatterplot algorithm based on one-bedroom rent data in 2005.

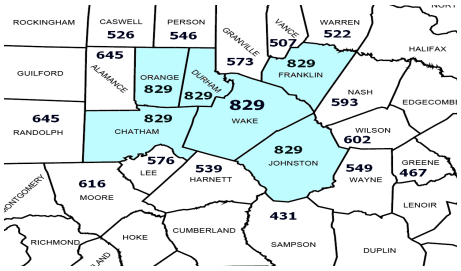



Figure 8: A region outlier detected by $ROD$ algorithm based on two-bedroom rent data in 2005.

Co.(MA). Nantucket Co. has been identified as the 2nd outlier which has very high attribute value, 1250. If the arithmetic average is used to represent the overall characteristics of Dukes Co.'s neighbors in other three algorithms, Nantucket Co. will significantly increase the neighborhood average. Therefore, the difference between Dukes Co. and its neighborhood average will be small, thus causing Dukes Co. to not being detected by other three algorithms. However, it is a "true" outlier, since its attribute value is much different from most of its neighboring counties.

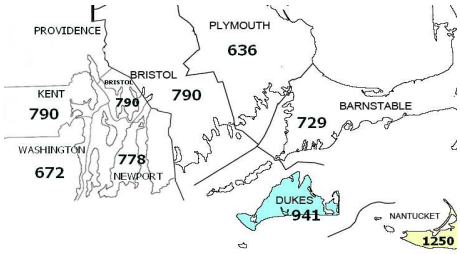



Figure 7: Dukes Co.(MA) identified by $POD$ algorithm based on one-bedroom rent data in 2005.

#### 5.2.2 Region Outlier Detection

We also applied $ROD$ algorithm to the 2-bedroom rent data in 2005. The evenness within a region is evaluated by "MinMax" difference, which is calculated by $\frac{Max-Min}{Mean}$. The threshold $T_{SIM}$ was chosen to be $0.15$, which means the difference between the minimum and the maximum nonspatial attribute values should be less than 15% of the average value. The dissimilarity between a region and its neighbor set is measured with $diff = \frac{|m_r - m_{nbr}|}{(m_r + m_{nbr})/2}$. The threshold $T_{DIFF}$ was set to be $0.4$, which means that a region will be identified as an outlier if the difference with its neighbors is more than 40% of the average of both sets. Table 3 shows two identified region outliers: one consists of 6 counties and the other consists of 4 counties.

Figure 8 shows the first outlier region with 6 counties having the same rent values 829 (marked in light gray). This region has 15 neighboring counties(rent ranging from 431 to 645), whose rents are much lower than the outlier region. Figure 9 presents the second outlier region with 4 counties, whose average two-bedroom rents are 1060, 1033, 1033, and 1033 respectively. This region has 11 neighboring counties whose rents are significantly lower than the four counties within the region. The existing algorithms are not capable of detecting a group of counties as outliers. In fact, many counties in the region cannot be detected by point outlier detection methods, because their non-spatial attribute values are similar to most of their neighbors.

## 6 Conclusion

In this paper, we propose two spatial outlier detection algorithms based on $kNN$ graph: one to detect point outliers and another to identify region outliers. The construction of $kNN$ graphs makes it possible to connect adjacent points to a region if their nonspatial attribute values are similar. Therefore, the ability to identify region outliers distinguishes our method from existing spatial outlier detection algorithms. In addition, the edge cut strategy can reduce the negative impact of objects with very

| Outliers | Counties in the region | Neighboring counties |
|---|---|---|
| 1 | Wake,NC(829) Orange,NC(829) Johnston,NC(829) Franklin,NC(829) Durham,NC(829) Chatham,NC(829) | Wilson,NC(602) Wayne,NC(549) Warren,NC(522) Vance,NC(507) Sampson,NC(431) Randolph,NC(645) Person,NC(546) Nash,NC(593) Moore,NC(616) Lee,NC(576) Harnett,NC(539) Greene,NC(467) Granville,NC(573) Caswell,NC(526) Alamance,NC(645) |
| 2 | Hinsdale,CO(1033) Ouray,CO(1033) Mineral,CO(1033) San Miguel,CO(1060) | San Juan,CO(692) Saguache,CO(494) Rio Grande,CO(494) Montrose,CO(634) La Plata,CO(776) Gunnison,CO(757) Dolores,CO(692) Delta,CO(578) Conejos,CO(494) Archuleta,CO(742) Montezuma,CO(582) |

Table 3: 2 region outliers detected by $ROD$ Alg. based on two-bedroom rent data in 2005.

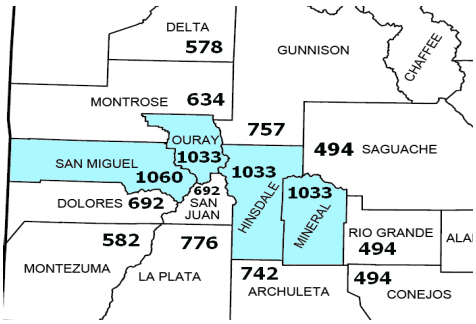



Figure 9: A region outlier detected by $ROD$ algorithm based on two-bedroom rent data in 2005.

large/small values on their neighbors. Thus the graph-based approach can detect "true" outliers ignored by other methods and avoid identifying "false" outliers. The experimental results on the US House Rent data set validate the effectiveness of the proposed methods. This paper focuses on single nonspatial attribute outlier detection. We are working on multi-attribute algorithms based on $kNN$ graphs and further result will be reported in the near future.

## References


[1] N. R. Adam, V. P. Janeja, and V. Atluri. Neighborhood-based detection of anomalies in high-dimensional spatio-temporal sensor datasets. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 576–583, Nicosia, Cyprus, 2004.

[2] A. Cerioli and M. Riani. The ordering of spatial data and the detection of multiple outliers. *Journal of Computational and Graphical Statistics*, 8(2):239–258, June 1999.

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

[4] J. Haslett, R. Brandley, P. Craig, A. Unwin, and G. Wills. Dynamic Graphics for Exploring Spatial Data With Application to Locating Global and Local Anomalies. *The American Statistician*, 45:234–242, 1991.

[5] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 392–403, San Francisco, CA, USA, 1998.

[6] Y. Kou, C.-T. Lu, and D. Chen. Spatial weighted outlier detection. In *Proceedings of the Sixth SIAM International Conference on Data Mining*, pages 614–618, Bethesda, Maryland, USA, 2006.

[7] H. Liu, K. C. Jezek, and M. E. O'Kelly. Detecting outliers in irregularly distributed spatial data sets by locally adaptive and robust statistical analysis and gis. *International Journal of Geographical Information Science*, 15(8):721–741, 2001.

[8] C.-T. Lu, D. Chen, and Y. Kou. Algorithms for spatial outlier detection. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 597–600, 2003.

[9] C.-T. Lu, D. Chen, and Y. Kou. Detecting spatial outliers with multiple attributes. In *Proceedings of the 15th International Conference on Tools with Artificial Intelligence*, pages 122–128, Sacramento, California, United States, November 3-5 2003.

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

[11] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 144–155, San Francisco, CA, USA, 1994.

[12] Y. Panatier. *Variowin. Software For Spatial Data Analysis in 2D*. New York: Springer-Verlag, 1996.

[13] F. P. Preparata and M. I. Shamos. *Computational Geometry - An Introduction.* Springer, 1985.

[14] 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*, pages 427–438, 2000.

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

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

[17] S. Shekhar, C.-T. Lu, and P. Zhang. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *Proceedings of the seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 371–376, San Francisco, California, 2001.

[18] S. Shekhar, C.-T. Lu, and P. Zhang. A Unified Approach to Spatial Outliers Detection. *GeoInformatica, An International Journal on Advances of Computer Science for Geographic Information System*, 7(2):139–166, June 2003.

[19] W. Tobler. A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46:234–240, 1970.

[20] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne. Online unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, 8(3):275–300, 2004.

[21] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Quebec, Canada, 1996.

[22] J. Zhao, C.-T. Lu, and Y. Kou. Detecting region outliers in meteorological data. In *Proceedings of the 11th ACM International Symposium on Advances in Geographic Information Systems*, pages 49–55, New Orleans, Louisiana, USA, 2003.