International Journal on Artificial Intelligence Tools
Vol. 20, No. 4 (2011) 721–751

DOI: 10.1142/S0218213011000309

# A GRAPH-BASED APPROACH TO DETECT ABNORMAL SPATIAL POINTS AND REGIONS

CHANG-TIEN LU

*Computer Science Dept., Virginia Tech*
*7054 Haycock Road, Falls Church, VA 22043 USA*
*ctlu@vt.edu*

RAIMUNDO F. DOS SANTOS JR

*Spatial Data Management Lab, Virginia Tech*
*7054 Haycock Road, Falls Church, VA 22043 USA*
*rdossant@vt.edu*

XUTONG LIU

*Spatial Data Management Lab, Virginia Tech*
*7054 Haycock Road, Falls Church, VA 22043 USA*
*xutongl@vt.edu*

YUFENG KOU

*Spatial Data Management Lab, Virginia Tech*
*7054 Haycock Road, Falls Church, VA 22043 USA*
*ykou@vt.edu*

Received 8 June 2009
Revised 3 February 2011

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 differences of nonspatial attribute 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 three major advantages compared with other existing spatial outlier detection methods: accurate in detecting both point and region outliers, capable of avoiding false outliers, and capable of computing the local outlierness of an object within subgraphs. We present time complexity of the algorithms, and show experiments conducted on US housing and Census data to demonstrate the effectiveness of the proposed approaches.

*Keywords*: Spatial outliers; local outliers; KNN graphs; spatial anomalies.

## 1. Introduction

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 neighborhood.[30] In contrast to traditional outliers, spatial outliers are local anomalies that are extreme compared to their neighbors,[31] 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 termed "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".[3]

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, namely spatial and nonspatial attributes. Spatial attributes are used to determine the relationship of spatial neighborhood; 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 an 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 $kNN$ relationship. $k$ directed edges can be derived 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 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. In addition, we demonstrate the detection of spatial outliers using multiple attributes of the same data set. 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. Time complexity is also discussed. Experiments conducted on US Housing and Census data demonstrate the effectiveness of our proposed algorithms.

The rest of the paper is organized as follows. Section 2 discusses the related work in spatial outlier identification. Section 3 defines the problem and provides the motivation of our approaches. Section 4 presents two spatial outlier detection algorithms based on $kNN$ graph, and analyzes their time complexities. Section 5 illustrates the experimental design and results. Finally, we summarize our work in Section 6.

## 2. Related Work

Graph theory plays an important role in data mining, e.g., clustering and outlier detection. Karypis *et al.* proposed a hierarchical clustering method, Chameleon, based on graph partition and merge.[4] One of the key features of Chameleon is that it considers both the relative inter-connectivity and closeness to model cluster similarity. Since outlier detection is derived from clustering, most of the clustering methods can be used to detect outliers. Yu *et al.* proposed a graph-based method for numerical and categorical outlier detection.[5] This method calculates the ratio between the open $k$-walk distance from a point and the close $k$-walk distance for this point. A large ratio indicates a large degree of outlierness. In their experiments, the authors claim that this method is more accurate than the density-based method LOF,[6] and may outperform other methods that use the local notion of neighborhood to define outliers, such as LOCI.[39] One limitation of this method is that the similarity graph is a complete graph and may introduce extensive computation. In addition, both Karypis' method and Yu's approach do not separate the spatial attributes from non-spatial attributes and use the entire attribute space to compute the similarity. Therefore, they cannot be directly applied for spatial outlier detection.

Outliers have been extensively studied in the past decades and numerous methods have been developed, including distance-based,[8,33] cluster-based,[9,10] depth-based[11,12] and distribution-based methods.[13] Gao *et al.*,[1] for example, explore information networks and the links among data points to find outlying members. Network information is modeled as a graph indicating relationships among objects. While graph-based, this approach is more statistical in nature, and does not make use of $kNN$ neighbors. Rather, it utilizes Markov random fields in both continuous and text data to find data points that deviate from the normal data points. Kriegel *et al.* take a different path in the field of outlier detection by examining the directions of distance vectors.[2] The goal is to minimize distance computation and make use of angles among data points. In this manner, this approach becomes more suitable for high-dimensional data where objects tend to be equidistant. However, this method is non-spatial. Angiulli *et al.* describe a method to measure abnormality in data points with respect to overall population where those points reside.[4] This approach is also non-spatial, looking at the data from an attribute perspective to capture how much an element should be considered outlying or not. 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 applicable to accurately process spatial data. First, spatial data generally possess complex data formats, such as lines and polygons, which can be combined 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".[14]

To discover anomalies from spatial data, many methods have been proposed. Liu *et al.* utilize random walks in combination with bipartite graphs to detect spatial outliers.[7] Relevance scores are calculated to determine the likeness among objects, after which the top $k$ are retained as the most probable outliers. Similar to our approach, Janeja *et al.* also consider neighborhoods as a spatial technique of outlier detection.[9] Unlike our approach, however, the authors use autocorrelation and heterogeneity, not distance.

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.[15–18] A scatterplot[17] 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[18] normalizes attribute values against the neighborhood average of values. Other algorithms, such as $z$-value, perform statistical tests to discover local inconsistencies.[19,20] $z$-value is the normalized difference between a spatial object and the average of its spatial neighbors.[21] The absolute $z$-value can determine the outlierness of an object where higher $z$-values indicate higher likelihood that an object is a spatial outlier. Shekhar *et al.* provided a unified definition of spatial outlier detection and proved that this definition generalizes the existing algorithms, such as $z$-value, scatterplot, and Moran scatterplot.[21]

Lu *et al.* proposed algorithms to detect spatial outliers with multiple nonspatial attributes using Mahalanobis distance.[20] For detecting outlier with multiple attributes, traditional outlier detection approaches could not be properly used due to the sparsity of data objects in high dimensional space.[22] It has been shown that the distance between any pair of data points in a high dimensional space is so similar that either each data point or none can be viewed as an outlier if the concepts of proximity is used to define outliers.[23] As a result, using the traditional Euclidean distance function cannot effectively retrieve outliers in high dimensional data sets due to the averaging behavior of noisy and irrelevant dimensions. To address this problem, two categories of research work have been conducted. The first is to project high dimensional data to low dimensional data that have abnormally low local density.[22,24,25,32] The second approach is to redesign distance functions to accurately define the proximity relationship between data points.[23] All these multi-attribute outlier detection approaches deal with non-spatial attributes. For spatial outlier detection, there are two categories of attributes: spatial and non-spatial attributes. In detecting spatial outliers, spatial and non-spatial attributes should be considered separately. The spatial attributes are used to define the neighborhood relationship, while the non-spatial dimension is used to define the distance function.

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 weights when comparing nonspatial attributes.[26] Adam *et al.* proposed an algorithm which considers both spatial relationship and semantic relationship among neighbors.[23] Liu and Jezek proposed a method for detecting outliers in an irregularly-distributed spatial data set.[27] The outlierness of an object $o$ is measured by both the spatial interpolation residual and surface gradient of its neighborhood. Shekhar *et al.* proposed that spatial outliers are the spatial objects whose nonspatial attribute values are quite different from those of their spatial neighbors.[28] A number of algorithms have been developed to detect spatial anomalies in meteorological images, transportation systems, and contagious disease data. An outlier detection method to identify anomalies in transportation network is given by Shekhar *et al.*[28] Their methods are based on road network connectivity and temporal neighborhoods based on time series.

## 3. Problem Formulation and Motivation

Graph-partition has been applied to identify data clusters. However, the existing graph-based clustering methods focus on locating large clusters and view the single points or small clusters as noises. These methods can be directly applied to find outliers, but their input/output parameters and internal cluster evaluation are not optimized for outlier detection. Thus, the efficiency and effectiveness cannot be assured. For example, most of the clustering methods try to generate balanced partition, which avoids small clusters. On the contrary, the objective of outlier detection is to find single points or small clusters. Thus, complex comparisons among clusters are not necessary. In addition, most of them are designed for non-spatial outlier detection, which does not separate spatial attributes and non-spatial attributes and treat them equally. In this section, we formally define the problem and provide the motivation for our graph-based approach.

### 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 $\{x_1, x_2, \ldots, x_z\}$ 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 $\{y_1, y_2, \ldots, y_z\}$, where $y_i$ is the nonspatial 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 Ref. 19, if an object has exceptionally large or small nonspatial attribute values, it will have a 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 a 2D space of 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 non-spatial attribute values are much larger or smaller than their spatial neighbors. Table 1 shows the outlier detection results of several algorithms on this data set in, including $z$-value algorithm, Moran scatterplot, and scatterplot. We can observe that none 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),

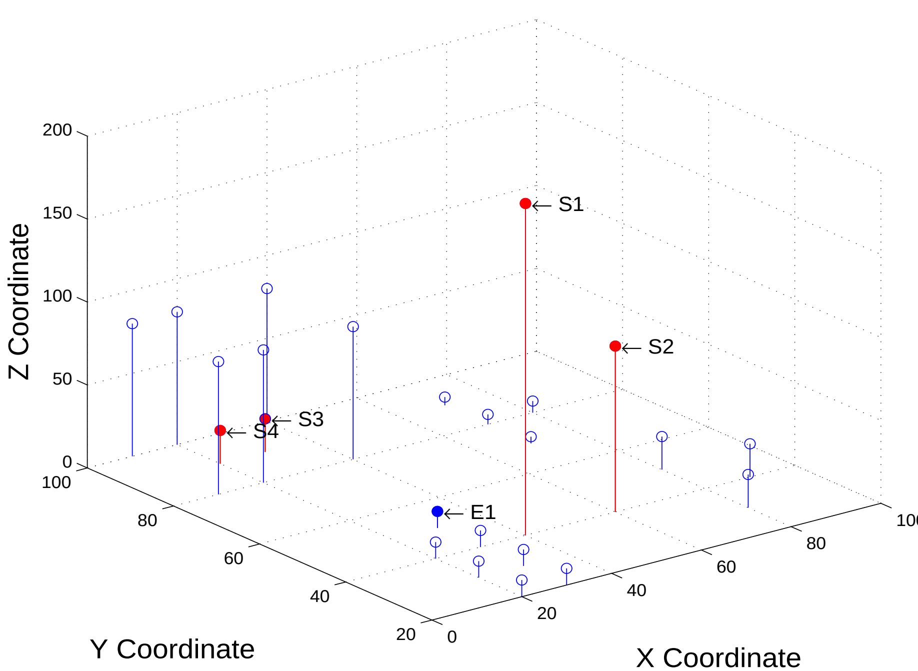



Fig. 1. An example data set, where X and Y coordinates denote the spatial locations and Z coordinate represents the value of nonspatial attribute.

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

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

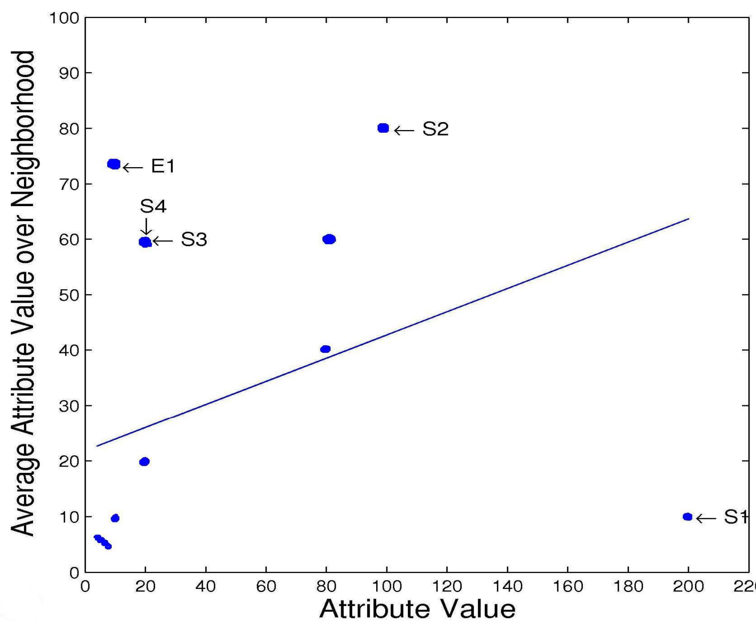



Fig. 2. The scatterplot of the data set shown in Figure 1.

and therefore dominates the neighborhood average of $E1$. Figure 2 and Figure 3 demonstrate scatterplot and Moran scatterplot of the example data set.

In the scatterplot of Figure 2, the points far from the regression line are marked as outliers. In the Moran scatterplot, 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.

**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, none of scatterplot, Moran scatterplot, or $z$-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

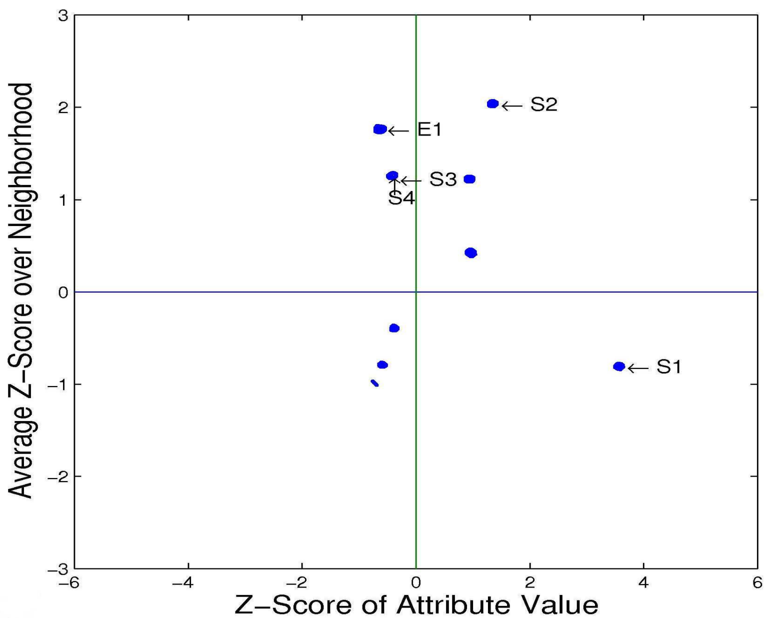



Fig. 3. The Moran scatterplot of the data set shown in Figure 1.

speed, air pressure, and water vapor density are much different from its surrounding areas.

**Third**, some existing methods such as $z$-value approaches 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 correlation. Therefore, a normalization process based on the entire data set may not be appropriate. One potential solution for this issue is to first identify the clusters of objects 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 identify both point and region outliers. Based on the example in Figure 1, we first construct a $kNN$ graph, as shown in Figure 4. Note that at this stage, the $k$ number of neighbors can be limited by a user-defined threshold. This application parameter prevents objects that are considered "too distant" to be part of the same subgraph, leaving it free to become part of a more suitable cluster. The threshold is helpful for sparse data sets, and may possibly be disregarded when each object is densely surrounded by many nearby neighbors. 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

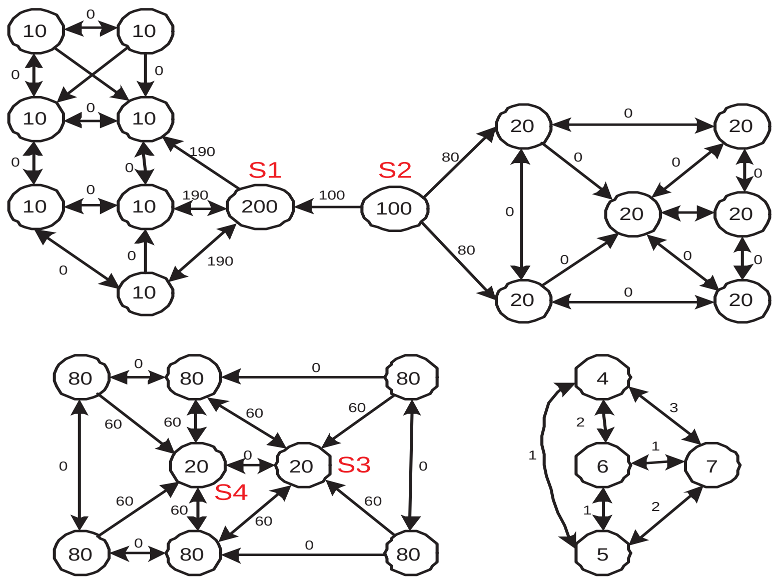



Fig. 4. The $kNN$-based graph representation of a small data set, $k = 3$.

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 ($i.e.$, $200 - 100$). 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 separately. 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) are 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 after the verification procedure validates that their degrees of outlierness are beyond thresholds.

Figure 5 shows the edge-cut result of Figure 4, 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. Note that under other examples, it is possible that a situation may arise where a disconnected subgraph is not found. This situation implies that all neighbors are extremely alike, i.e., their non-spatial attributes are very similar. In this case, no edge cutting will occur. To address this problem, the user should reconsider at what point edge values between nodes can be cut. Increasing or decreasing this parameter as a threshold should

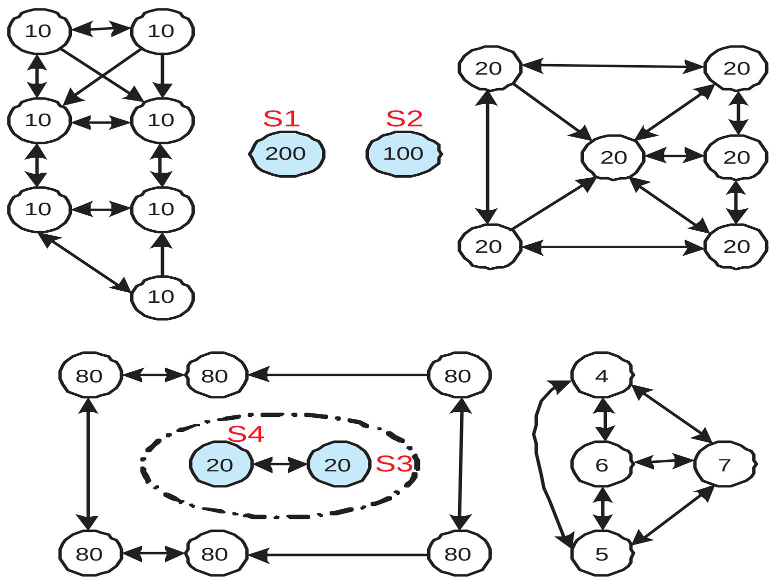



Fig. 5. The result of graph-based outlier detection: 2 point outliers and 1 region outlier are identified.

effectively permit outliers to be found, if any do exist. However, this is a judgment call left to the domain expert, not the algorithm. Table 1 shows that the graph-based method can accurately detect all expected spatial outliers and is capable of identifying the outlier region, (S3, S4). In summary, the graph-based algorithms have the following advantages compared with other 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) Consider and compute 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. The computational complexities of both algorithms are also examined.

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 neighbor ($kNN$) 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 attribute

| | | Nbr 1 | | | | Nbr k | | | |
|---|---|---|---|---|---|---|---|---|---|
| ID | val | Nbr_1 | Diff_1 | Flag_1 | ⋯ | Nbr_k | Diff_k | Flag_k | cID |
| 1 | 200 | 3 | 190 | 0 | ⋯ | 5 | 190 | 0 | 1 |
| 2 | 100 | 1 | 100 | 0 | ⋯ | 7 | 80 | 0 | 1 |
| ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ | ⋯ |

Fig. 6. The data structure for graph representation.

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 a 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 respective statistical characteristics in identifying spatial outliers.

The data structure for the graph is shown in Figure 6. 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 is cut and $flag_j = 0$ represents that $x_i$ and $x_j$ are connected); *cID* refers to the subgraph which $x_i$ belongs to. The neighbor list is sorted by their edge weights in descending order, *i.e.*, $diff_1 \geq diff_2 \geq \cdots \geq diff_k$. 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 6 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 constructed from the nearest neighbor table and used for extracting connected regions.

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*

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 procedures like dimension reduction or data standardization. $k$ is the number of neighbors. $m$ is the number of expected outliers. Generally, $m$ should not be greater than 5% of $n$, assuming the nonspatial attribute follows a normal distribution with a confidence interval of 95%. This section introduces an algorithm to detect single point outliers based on a $kNN$ graph. We assume that all $k(x_i)$ are equal to a fixed number $k$. (The algorithm can be conveniently generalized by replacing the fixed $k$ with a dynamic $k(x_i)$.)

For each data object, the first step is to identify its $k$ nearest neighbors (line 3). 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 (line 5). $G$ 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 for each subgraph (line 7). 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 (line 9). To partition the graph, a "longest-edge-cut" strategy is employed and the partition will be conducted with multiple iterations (line 11). In each iteration, the longest edge $\overrightarrow{x_i, x_j}$ (the edge with the highest weight) is dequeued from *edgeQueue* and cut (lines 12-14). Here, cutting an edge is equivalent 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 (line 16). If true, $o$ is marked as a spatial outlier (line 17) and inserted into the outlier set $O_s$ (line 19). Note that it is possible to return an empty set (i.e., no outliers found). Finally, when all $m$ spatial outliers have been identified, the partition loop is terminated and $O_s$ is returned. The ranking of outliers is determined by the 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 nonspatial 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

**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 ≤ 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) ≤ 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);
```

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 (line 5). Then a priority queue *edgeQueue* is constructed to rank all edges in descending order based on their weights (line 7). Next, the algorithm continuously selects the longest edge from the *edgeQueue* and cuts it if the weight of this edge is larger than threshold $T_{EDGE}$ (lines 9-12). After cutting all edges with weight above $T_{EDGE}$, a function *findCandidateRegions*() is called to detect candidate region outliers, *regions* (line 14). Each of these candidates is a connected

**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 ≤ 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);
```

region which contains less than $k$ objects. *regions* is only a set of candidate outliers, and each region in this set needs further verification. The verification includes three steps: (1) for each candidate *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*[$i$], *region*[$i$] is not likely to be a region outlier since outliers usually refers to a small set of irregular objects compared with large amount of regular objects; (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 nonspatial attribute value difference between *region*[$i$] and $nbrs$. If the difference is less than the threshold $T_{DIFF}$, *regions*[$i$] is not an outlier (lines 15-19). Once *regions*[$i$] has been verified as a region outlier through the above procedures, it will be inserted into $O_s$ (line 21).

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. While such methods can be useful in determining threshold values such as $T_{DIFF}$, $T_{SIM}$, and $T_{EDGE}$, this can also be a judgment call based on what is appropriate for the application in question. A medical application, for instance, may consider a 5% discrepancy between two attributes to be excessive. On the other hand, a real estate application may accept values of up to 30%. Thus, a domain expert may be the best option to determine appropriate thresholds.

### 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 the 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.

Another aspect that may influence running time is the neighborhood size. In the current literature, there are not many algorithms that can perform efficiently when the data set is large and high-dimensional. In this case, scalability becomes an important factor in the determination of neighborhood size. When small neighborhoods are chosen, the general tendency will be for a higher number of subgraphs to be created. Increasing the neighborhood, i.e., less clusters overall, will often decrease the subgraph partitions. There's no single clear-cut way to determine which approach is best. Since edge cutting cannot be evaluated ahead of time, neighborhood size may be left to the domain expert's discretion as an application parameter. Under high data dimensionality (i.e., multiple attributes are considered), the algorithm tends to scale more poorly. When $n$ objects are considered, each $kNN$ query requires $O(n^2)$ when many data attributes are included in the analysis. Thus, most approaches limit the number of attributes that are taken into consideration, or perform dimensionality reduction as a preprocessing stage.

### 4.4. *Adapting POD to multi-attributes*

We can adapt the $POD$ algorithm to detect multi-attribute spatial outliers. For spatial data with multiple non-spatial attributes, the $POD$ algorithm is the same except that a different dissimilarity measure will be used to derive edge weights. We apply Mahalanobis distance on the $POD$ algorithm, serving as the edge weights between pairs of neighbors. Mahalanobis distance takes into account the measured attributes, along with average, variance and covariance. Compared with Euclidean distance, Mahalanobis distance is more accurate to represent the difference between two vectors. Note that Mahalanobis distance treats each attribute as equally important. As such, if the application in question ranks attributes in any sort of priority, it is important to pre-process the attributes in a such a way that less desirable attributes (if any) are punished to a lower degree of influence.

Suppose $q$ attribute values ($q \geq 1$) are measured from $x$, a spatial object in the data set. Given a set of spatial points $X = \{x_1, x_2, \ldots, x_n\}$, an attribute function $f$ is defined as a map from $X$ to $R^q$ (the $q$ dimensional Euclidean space) such that for each spatial point $x$, $f(x)$ equals the attribute vector $y$. This can be written as

$$\begin{aligned} y_i &= f(x_i) \\ &= (f_1(x_i), f_2(x_i), \ldots, f_q(x_i))^T \\ &= (y_{i1}, y_{i2}, \ldots, y_{iq})^T \end{aligned}$$

for $i = 1, 2, \ldots, n$. For a given integer $k$, let $NN_k(x_i)$ denote the $k$ nearest neighbors of point $x_i$. In order to detect spatial outliers, we compare all values of $y$ measured at $x$ with the corresponding quantities from the neighbors of $x$.

For $n$ spatially-referenced objects $x_1, \ldots, x_n$, the sample mean of their non-spatial attributes is calculated as

$$\mu_s = \frac{1}{n} \sum_{i=1}^{n} y_i$$

and sample variance-covariance matrix is

$$\sum_s = \frac{1}{n-1} \sum_{i=1}^{n} [y_i - \mu_s][y_i - \mu_s]^T .$$

Mahalanobis distance between object $x_i$ and object $x_j$ is defined as

$$d(x_i, x_j) = \sqrt{(f(x_i) - f(x_j))^T \sum_s^{-1} (f(x_i) - f(x_j))} .$$

$d(x_i, x_j)$ can be used as the weight of edge $\overrightarrow{x_i, x_j}$. The edge with the largest Mahalanobis distance shows the largest difference between two end nodes, thus being cut first. The computation of the Mahalanobis distance adds $O(q^2 * n)$ to the POD algorithm. So the total time complexity of POD for processing multiple attributes is $O(q^2 * n + nlogn) \approx O(nlogn)$, assuming $q \ll n$.

## 5. Experiment

In this section, we discuss our experiments on two real data sets: Fair Market Rents data provided by the $PDR - DHUD$ (Policy Development and Research, U.S. Department of Housing and Urban Development), and U.S. Census Bureau demographics, to validate the effectiveness of our proposed algorithms. The experiment design is introduced in Section 5.1. Section 5.2 provides analysis and detailed examples related to the various results.

### 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 three parts: point and region outlier detection using U.S. Housing data, and multiple-attribute outlier detection using U.S. Census data. At the algorithm level, we use a grid-based indexing scheme while maintaining the grid directory in memory. Disk-based approaches may also be used. However, time complexity will tend to increase since disk I/O must be taken into consideration. In our experiments, memory-based indexing proved efficient for our data sets.

### 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 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 a 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 Moran scatterplot algorithm. However, Dorchester Co. is not very outlying if we take a closer look. Figure 7 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 classified as a spatial outlier. Dorchester 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 8, the one-bed rent in Dukes Co. is 941, which is much higher than its 7 neighbors (difference $> 150$) except Nantucket 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. not to be detected by other three algorithms. However, it is an authentic outlier, since its attribute value is much different from most of its neighboring counties.

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

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

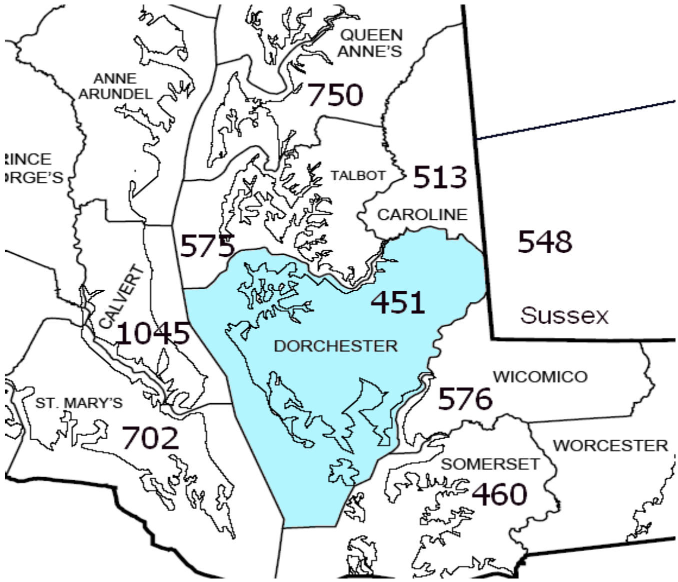



Fig. 7. Dorchester Co.(MD) detected by Moran scatterplot algorithm based on the one-bedroom rent data in 2005.

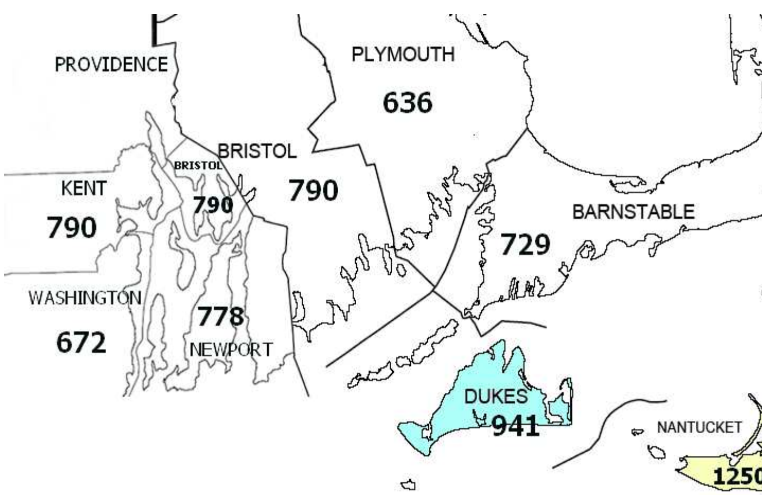



Fig. 8. Dukes Co.(MA) identified by *POD* algorithm based on the one-bedroom rent data in 2005.

### 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

Table 3. Two region outliers detected by *ROD* based on 2-bedroom rental prices in 2005.

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

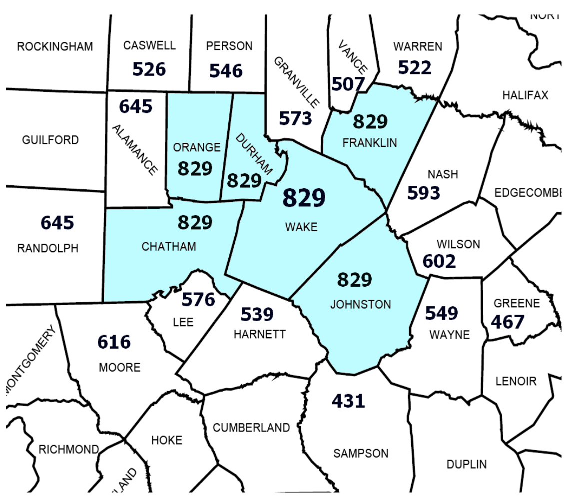



Fig. 9. A region outlier detected by *ROD* algorithm based on two-bedroom rent data in 2005.

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 9 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

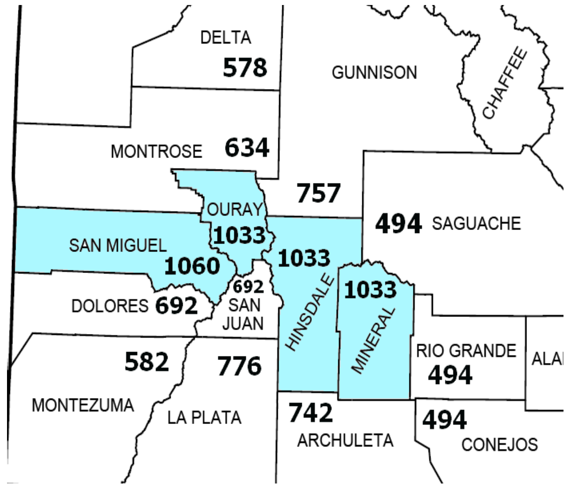



Fig. 10. A region outlier detected by *ROD* algorithm based on two-bedroom rent data in 2005.

from 431 to 645), whose rents are much lower than the outlier region. Figure 10 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.

### 5.2.3. *Experiments on multiple attributes*

We used a real U.S. Census data set to evaluate POD based on multiple attributes. This data is a rich collection of American demographics from the U.S. Census Bureau.[29] It contains detailed statistics on population, race and ethnicity, age and sex, education, employment, income, poverty, housing, and many other attributes for each of the following different levels of geography: (1) the United State and major regions of the country; (2) each state and metropolitan area; (3) all 3000+ counties in the United States; (4) municipalities, census tracts, and block groups. The location of each county is determined by one or more polygons consisting of hundreds of longitude and latitude pairs. The neighborhoods were chosen to be dynamic, i.e., the neighborhood of a county was chosen to be the set of adjacent counties.

We performed experiments with the following 11 attributes: population in 2001, population percent change from April 1 2000 to July 1 2001, population percent change from 1990 to 2000, percentage of persons under 5 years old in 2000, percentage of persons under 18 years old in 2000, percentage of persons over 65 years old in 2000, percentage of females, percentage of White persons, percentage of Black persons, percentage of Asian persons, and percentage of American Indian persons.

Different attributes may have different magnitudes. For example, population of a county is usually more than 10,000, but the percentage of population change is mostly less than 1. Since our approach considers the differences between attributes, we standardized the values for each attribute, thus to prevent the domination of one over others. The experiment results are shown in Table 4.

Table 4 show the top 10 counties most likely to be spatial outliers. As can be seen, Los Angeles Co. (CA) is selected as the top spatial outlier because it generated the largest Mahalanobis distance. Specifically, the largest distance mainly comes from the contribution of the corresponding attribute population (standardized value 32.16), compared with its neighboring counties, i.e., Ventura, Orange, Kern, and San Bernardino. The second spatial outlier, Cook Co(IL), is identified due to its high change in population less than 5 years old (standardized value 0.82), compared with its neighboring counties, i.e., Du Page, McHenry, Kane, Lake, and Will. The third and fourth spatial outliers, Shannon Co. and Todd Co. have high percentage of Indian population (standardized Value 14.49 and 13.15, respectively) compared with the Indian population of their neighboring counties. The remaining five counties were detected as spatial outliers because the total contributions to the Mahalanobis distance from various attribute values are significant. This shows that the algorithms are effective in detecting spatial outliers.

### 5.3. *Simulations*

This section presents simulations to compare the performance between POD and other related Spatial Outlier Detection (SOD) methods. The experimental study followed the standard statistical approach for evaluating the performance of 5 SOD methods.

#### Simulation Settings

Data Set: The simulation data were generated based on a standard statistical model[35] with the decomposition form:

$$Z(s) = \beta + \omega(s) + \epsilon(s)$$

where $\beta \sim N(0,1)$, $\omega(s)$ refers to a Gaussian random field with covariogram model $C(h;\theta)$, and $\epsilon(s)$ refers to measurement error or white noise variation. We considered a popular exponential covariogram model. The exponential model is defined as

$$C(h;b,c) = \begin{cases} b & \text{if } x \geq 0 \\ b\left(1 - exp\left(-\dfrac{h}{c}\right)\right) & \text{if } 0 < h \leq c \\ 0 & \text{if } h > c \end{cases}$$

Table 4. The top 10 spatial outliers candidates detected by POD algorithm using standardized multiple attributes.

| Rank | County | ≤ 5% | ≤ 18% | ≥ 65% | Female | White | Black | Asian | Indian | Pop 2001 | 1990–2000 Change | 2000–2001 Change |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Los Angeles(CA) | 1.28 | 0.78 | −1.24 | 0.06 | 2.24 | 0.06 | 7.08 | −0.12 | 32.16 | −0.23 | 0.37 |
| 2 | Cook(IL) | 0.82 | 0.16 | −0.75 | 0.58 | −1.77 | 1.18 | 2.56 | −0.20 | 17.71 | −0.36 | −0.42 |
| 3 | Shannon(SD) | 4.27 | 6.21 | −2.43 | −0.19 | −5.00 | −0.60 | −0.49 | 14.49 | −0.26 | 0.92 | 0.98 |
| 4 | Todd(SD) | 5.29 | 5.80 | −2.19 | 0.01 | −4.49 | −0.60 | −0.42 | 13.15 | −0.27 | −0.16 | 0.94 |
| 5 | Buffalo(SD) | 3.99 | 4.95 | −2.02 | −0.92 | −4.26 | −0.60 | −0.49 | 12.52 | −0.30 | 0.27 | −0.60 |
| 6 | Menominee(WI) | 2.96 | 4.20 | −1.53 | 0.11 | −4.56 | −0.60 | −0.49 | 13.41 | −0.29 | 0.05 | 0.42 |
| 7 | Sioux(ND) | 3.89 | 4.64 | −2.24 | −0.76 | −4.39 | −0.60 | −0.49 | 12.99 | −0.29 | −0.22 | 0.04 |
| 8 | Harris(TX) | 1.84 | 1.10 | −1.80 | −0.14 | −1.62 | 0.66 | 2.75 | −0.19 | 11.35 | 0.59 | 0.65 |
| 9 | Maricopa(AZ) | 1.47 | 0.47 | −0.75 | −0.24 | −0.46 | −0.35 | 0.91 | 0.03 | 10.45 | 2.10 | 1.69 |
| 10 | Dewey(SD) | 2.68 | 4.20 | −1.58 | 0.32 | −3.78 | −0.60 | −0.42 | 11.36 | −0.28 | −0.18 | 0.42 |

Table 5. Parameter settings.

| Variable | Settings |
|---|---|
| $N$ | $N \in 100, 200$. Randomly generate $n$ spatial locations $s_i$ ($i \in [1, N]$) in the range $[0, 25] \times [0, 25]$ |
| $b$ $c$ | $b = 5$; $c = 5, 15, 25$ |
| $\beta$ | $\beta_1\, N(0, 1)$ and $\beta_i = 0$, $i = 2, \ldots, 5$ |
| $\sigma_0$, $\sigma_C$ | $\sigma_0^2 = 2, 10$; $\sigma_C^2 = 20$ |
| $\alpha$ | $\alpha = 0.05, 0.10, 0.15$ |
| $K$ | $K = 5, 10$ |

where $h$ refers to the spatial distance between two sample objects $s_i$ and $s_j$, the parameter $b$ refers to a constant variance for each $Z(s)$, and $c$ refers to a valid distance range for nontrivial dependence (or covariance). For the white noise component, we employed the following standard model:[36]

$$\epsilon(s) \sim \begin{cases} N(0, \sigma_0^2) & \text{with probability } 1 - \alpha \\ N(0, \sigma_C^2) & \text{with probability } \alpha \end{cases}$$

There are three related parameters $\sigma_0^2$, $\sigma_C^2$ and $\alpha \cdot \sigma_0^2$ is the variance of a normal white noise, $\sigma_C^2$ is the variance of contaminated error that generates outliers, and $\alpha$ is used to control the number of outliers. Note that it is possible that the distribution $N(0, \sigma_C^2)$ generates some normal white noises. All true outliers must be only identified based on standard statistical test by calculating the conditional mean and standard deviation for each observation.[35] In the simulations, we tested several representative settings for each parameter, which are summarized in Table 5.

**Outlier detection methods:** We compared our methods with other common local-based SOD methods, including $Z$-test,[20] Scatterplot,[17] MoranScatterplot,[37] and SLOM-test.[38] Our proposed methods are identified as POD. The implementations of all existing methods are based on their published algorithm descriptions.

**Performance metric:** We tested the performance of all methods for every combination of parameter setting in Table 5. For each specific combination, we ran the experiments ten times and then calculated the mean of accuracy for each method. To compare the accuracies of each method, we used the standard ROC curves.

As shown in Table , POD outperforms other approaches. They identify the true outliers (like Blaine(ID), Fairfield(CT), Summit(UT), etc). Compared with the other 4 methods, Z-value tends to miss some true outliers, like St. Mary's(MD) and FairField(CT), etc. The four accompanying mini tables give the rental prices of the county and its neighbors. As we can see, the rents of some neighbors (such as Calvert(1045) and Charles(1045)) are high while others (Westmoreland(496), Top 10 spatial outliers with single attribute detected by five different approaches.

| | **Z-Value** | **SLOM** | **ScaPlot** | **M-ScaPlot** | **POD** |
|---|---|---|---|---|---|
| 1 | Nantucket(MA) | Blaine(ID) | KingGeorge(VA) | Blaine(ID) | Blaine(ID) |
| 2 | Pitkin(CO) | Teton(WY) | Plymouth(MA) | Teton(WY) | Teton(WY) |
| 3 | Summit(UT) | Lubbock(TX) | Blaine(ID) | Elbert(CO) | Summit(UT) |
| 4 | Orange(CA) | Summit(UT) | Caroline(VA) | Surry(VA) | Suffolk(MA) |
| 5 | Blaine(ID) | Pennington(SD) | Howard(MD) | LaPaz(AZ) | Coconino(AZ) |
| 6 | Clarke(VA) | Hughes(SD) | Kern(CA) | Kanabec(MN) | Fairfield(CT) |
| 7 | Suffolk(MA) | Dane(WI) | Teton(WY) | Dorchester(MD) | Nantucket(MA) |
| 8 | Frederick(MD) | Boone(MO) | Summit(UT) | Sumter(FL) | Dane(WI) |
| 9 | Coconino(AZ) | Yellowstone(MT) | SanJoaquin(CA) | Blanco(TX) | Pitkin(CO) |
| 10 | Ventura(CA) | Codington(SD) | Worcester(MA) | Sussex(VA) | Eagle(O) |

Current and Mean B with a forward polarity.

| County | Rent |
|---|---|
| St. Mary's(MD) | 702 |
| Calvert(MD) | 1045 |
| Westmoreland(VA) | 496 |
| Richmond(VA) | 496 |
| Charles(MD) | 1045 |
| Northumberland(VA) | 496 |
| Essex(VA) | 496 |
| King(VA) | 611 |
| Lancaster(VA) | 496 |
| Dorchester(MD) | 451 |
| Caroline(VA) | 496 |

Current and Mean B with a reversed polarity.

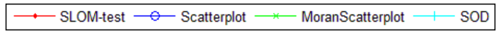
| County | Rent |
|---|---|
| Rockingham(NH) | 750 |
| Strafford(NH) | 648 |
| Essex(MA) | 878 |
| Hillsborough(NH) | 605 |
| Middlesex(MA) | 884 |
| Suffolk(MA) | 1120 |
| York(ME) | 577 |
| Merrimack(NH) | 624 |
| Belknap(NH) | 592 |
| Norfolk(MA) | 914 |
| Carroll(NH) | 564 |

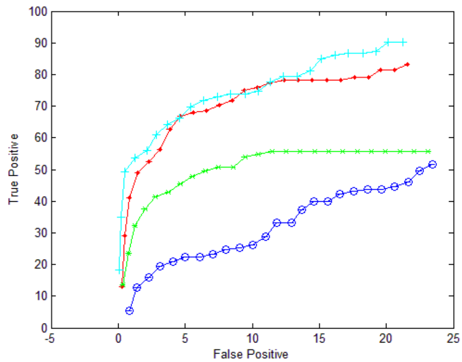
Current and Mean B with a forward polarity.

| County | Rent |
|---|---|
| San Benito(CA) | 824 |
| Monterey(CA) | 1045 |
| Santa(CA) | 1111 |
| Merced(CA) | 536 |
| Santa(CA) | 1107 |
| Stanislaus(CA) | 645 |
| San(CA) | 635 |
| Alameda(CA) | 1132 |
| Madera(CA) | 556 |
| San(CA) | 1305 |
| Fresno(CA) | 556 |

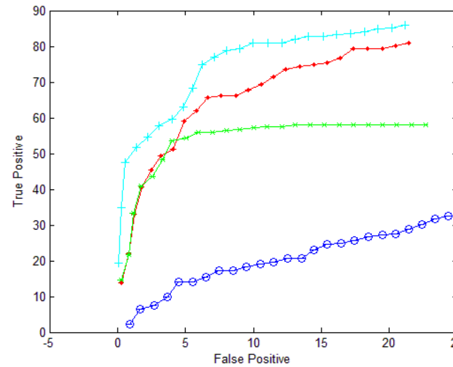
Current and Mean B with a reversed polarity.

| County | Rent |
|---|---|
| Yellowstone(MT) | 452 |
| Musselshell(MT) | 398 |
| Carbon(MT) | 405 |
| Golden(MT) | 398 |
| Stillwater(MT) | 398 |
| Big(MT) | 398 |
| Petroleum(MT) | 398 |
| Treasure(MT) | 398 |
| Big(MT) | 417 |
| Park(MT) | 428 |
| Sweet(MT) | 398 |

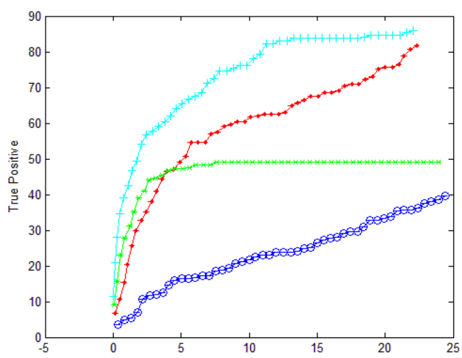


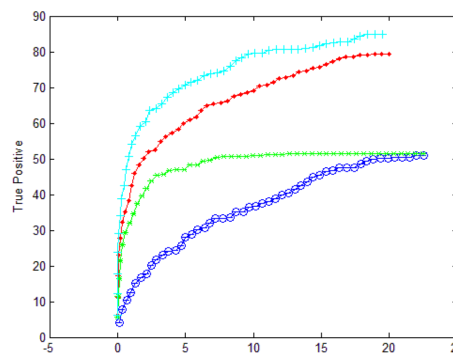

(a) $\alpha = 0.1$, $\sigma_0^2 = 2$, $c = 15$, $K = 5$, $N = 100$



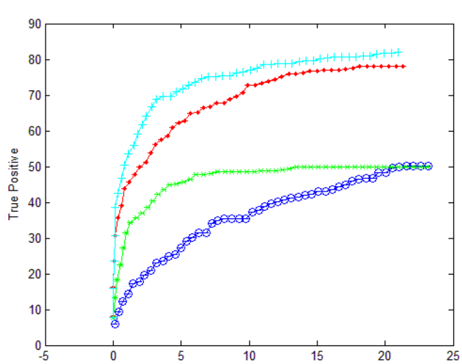

(b) $\alpha = 0.1$, $\sigma_0^2 = 2$, $c = 15$, $K = 10$, $N = 100$



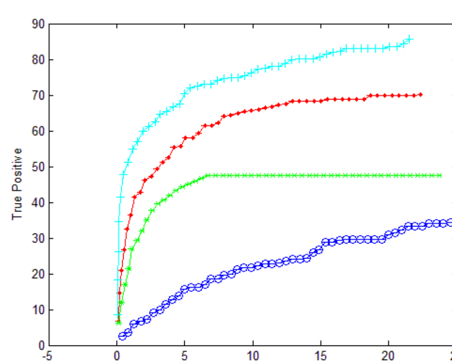

(c) $\alpha = 0.15$, $\sigma_0^2 = 2$, $c = 15$, $K = 5$, $N = 100$




(d) $\alpha = 0.15$, $\sigma_0^2 = 2$, $c = 15$, $K = 10$, $N = 100$




(e) $\alpha = 0.15$, $\sigma_0^2 = 2$, $c = 15$, $K = 5$, $N = 200$




(f) $\alpha = 0.15$, $\sigma_0^2 = 2$, $c = 15$, $K = 10$, $N = 200$

Fig. 11. (color online) ROC Curves. Same settings ($n = 100$, $b = 5$, $c = 5$).

Richmond(196), Northumberland(496), etc) are lower. Intuitively, the rent in St. Mary's is very different from those of its neighbors. However, such outlying behavior cannot be detected by Z-Value, SLOM, scatterplot and Moran scatterplot. This is due to their intrinsic properties when identifying the outlying behavior. For example, Z-Value identifies the outliers by normalizing the difference between a spatial object and the average of its spatial neighbors. Moran scatterplot detects the spatial outliers by normalizing the attribute values against the average values of the corresponding neighborhood. Averaging the rents of the neighbors neutralizes such significant differences. POD based methods can also avoid identifying the false outliers while SLOM, Scatterplot, and Moran-Scatterplot not only miss some true outliers, but incorrectly recognize some not very outlying points as true outliers. For example, Yellowstone(MT) by SLOM approach and Dorchester(MD) by Moran-Scatterplot approach. Take county Dorchester(MD) as an example. Most of its neighbors have somewhat similar values. Therefore, it should not be identified as a spatial outlier. It is identified as an outlier by Moran-Scatterplot approach mainly because of Calvert(MD).

**Detection Accuracy:** We compared the outlier detection accuracies of different methods based on different combinations of parameter settings as shown in Table 5. Six representative results are displayed in each one of the graphs, showing the detection of true and false positives. Obviously, SOD (aka POD in our approach) has significant performance increases. POD achieved 15-20% improvement over SLOM methods, 30-40% over Moran-Scatterplot method and about 50% over Scatterplot method. Meanwhile, Z-value test has also very impressive performance on the simulation. Z-value is under the null hypothesis stating that the data fits a normal distribution. It computes the mean and standard deviation of the entire dataset to compute the outlierness for each object. Since our simulation data is just generated from standard normalized distribution, there is no doubt that Z-value is one of the most appropriate methods for the simulation data. Some of the ROC curves derived from POD-based methods have somewhat similar trend with that of Z-value method. In a sense, POD based approach can accurately detect the outliers in the dataset with normal distribution although they do not make such hypothesis. When being utilized into a real dataset with unknown distribution, Z-value may not show such good performance since many datasets do not conform to normal distribution.

## 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. Moreover, we extend our approach to include multiple attributes based on Mahalanobis Distance. 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 large/small values on their neighbors. Thus the graph-based approach can detect legitimate outliers ignored by other methods and avoid identifying "false" outliers.The experiments conducted on the US House Rent data and U.S. Census data validated the effectiveness of the proposed methods.

## References

1. J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun and J. Han, On community outliers and their efficient detection in information networks, in *Proc. 16th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, Washington, DC, USA, pp. 813-822, 2010.
2. H. Kriegel, M. Schubert and A. Zimek, Angle-based outlier detection in high-dimensional data, in *Proc. 14th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, Washington, DC, USA, pp. 444-452, 2008.
3. J. Zhao, C. T. Lu and Y. Kou, Detecting region outliers in meteorological data, in *Proc. 11th ACM Intl. Symposium on Advances in Geographic Information Systems*, pp. 49-55, New Orleans, LA, USA, 2003.
4. F. Angiulli, F. Fassetti and L. Palopoli, Detecting outlying properties of exceptional objects, in *ACM Transactions on Database Systems*, Vol. 34, Number 1, Article 7, 2009.
5. J. Xu Yu, W. Qian, H. Lu and A. Zhou, Finding centric local outliers in categorical/numerical spaces, in *Journal of Knowledge and Information Systems*, Springer-Verlag, New York, NY, Vol. 9, Number 3, pp. 309-338, 2006.
6. M. M. Breunig, H. P. Kriegel, R. T. Ng and J. Sander, LOF: Identifying density-based local outliers, in *Proc. 2000 ACM SIGMOD Intl. Conf On Management of Data*, ACM Press, Dallas, Texas, USA, pp. 93-104, 2000.
7. X. Liu, C. T. Lu and F. Chen, Spatial outlier detection: Random walk based approaches, in *Proc. 18th ACM SIGSPATIAL Intl. Conf. on Advances in Geographic Information Systems*, San Jose, CA, USA, pp. 370-379, 2010.
8. S. Ramaswamy, R. Rastogi and K. Shim, Efficient algorithms for mining outliers from large data sets, in *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, Dallas, TX, USA, pp. 427-438, 2000.
9. V. Janeja and V. Atluri, Spatial Outlier detection in heterogeneous neighborhoods, in *Journal of Intelligent Data Analysis*, Vol. 13, Number 1, pp. 86-107, 2009.
10. R. T. Ng and J. Han, Efficient and effective clustering methods for spatial data mining, in *Proc. 20th Intl. Conf. on Very Large Data Bases*, San Francisco, USA, pp. 144-155, 1994.
11. F. P. Preparata and M. I. Shamos, *Computational Geometry — An Introduction*, Springer, New York, NY, USA, 1985.
12. I. Ruts and P. Rousseeuw, Computing depth contours of bivariate point clouds, in *Computational Statistics and Data Analysis*, Vol. 32, pp. 153-168, 1996.
13. K. Yamanishi, J. I. Takeuchi, G. Williams and P. Milne, On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms, in *Journal of Data Mining and Knowledge Discovery*, Kluwer, Hingham, MA, USA, Vol. 8, Number 3, pp. 275-300, 2004.
14. W. Tobler, A Computer movie simulating urban growth in the detroit region, in *Journal of Economic Geography*, Vol. 46, pp. 234-240, 1970.

15. 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, in *The American Statistician*, Vol. 45, pp. 234-242, 1991.
16. Y. Panatier, *Variowin. Software for Spatial Data Analysis in 2D*, Springer-Verlag, New York, NY, USA, 1996.
17. R. P. Haining, *Spatial Data Analysis in the Social and Environmental Sciences*, Cambridge University Press, New York, NY, USA, 1993.
18. Anselic Luc, Local Indicators of Spatial Association: *LISA*, in *Geographical Analysis*, Vol. 27, Number 2, pp. 93–115, 1995.
19. C.T. Lu, D. Chen and Y. Kou, Algorithms for spatial outlier detection, in *Proc. 3rd IEEE Intl. Conf. on Data Mining*, Melbourne, FL, USA, pp. 597–600, 2003.
20. C. T. Lu, D. Chen and Y. Kou, Detecting spatial outliers with multiple attributes, in *Proc. 15th Intl. Conf. on Tools with Artificial Intelligence*, Sacramento, CA, USA, pp. 122–128, 2003.
21. S. Shekhar, C. T. Lu and P. Zhang, A unified approach to spatial outliers detection, in *GeoInformatica*, An International Journal on Advances of Computer Science for Geographic Information System, Vol. 7, Number 2, pp. 122–128, 2003.
22. C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu and J. S. Park, Fast algorithms for projected clustering, in *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, Philadelphia, PA, USA, pp. 61–72, 1999.
23. N. R. Adam, V. P. Janeja, and V. Atluri, Neighborhood-Based detection of anomalies in high-dimensional spatio-temporal sensor datasets, in *Proc. ACM symposium on Applied computing*, Nicosia, Cyprus, pp. 576–583, 2004.
24. C. C. Aggarwal, Redesigning distance functions and distance-based applications for high dimensional data, in *SIGMOD Record*, Vol. 30, Number 1, 2001.
25. S. Berchtold, C. Bohm and H. P. Kriegel, The pyramid-technique: Towards breaking the curse of dimensionality, in *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, Seattle, WA, USA, pp. 142–153, 1998.
26. Y. Kou, C. T. Lu and D. Chen, Spatial weighted outlier detection, in *Proc. Sixth SIAM Int. Conf. on Data Mining*, Bethesda, MD, USA, pp. 614–618, 2006.
27. 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, in *International Journal of Geographical Information Science*, Vol. 15, Number 8, pp. 721–741, 2001.
28. S. Shekhar, C. T. Lu and P. Zhang, Detecting graph-based spatial outliers: Algorithms and applications (A summary of results), in *Proc. 7th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, pp. 371–376, 2001.
29. U.S. Census Bureau, U.S. Department of Commerce, http://www.census.gov/.
30. S. Shekhar and S. Chawla, *A Tour of Spatial Databases*, Prentice Hall, New Jersey, USA, 2002.
31. A. Cerioli and M. Riani, The ordering of spatial data and the detection of multiple outliers, in *Journal of Computational and Graphical Statistics*, Vol. 8, Number 2, pp. 239–258, 1999.
32. G. Karypis, E. H. Han and V. Kumar, Chameleon: Hierarchical clustering using dynamic modeling, in *IEEE Computer*, Vol. 32, Number 8, pp. 68–75, 1999.
33. E. M. Knorr and R. T. Ng, Algorithms for mining distance-based outliers in large datasets, in *Proc. 24th Intl. Conf. on Very Large Data Bases*, San Francisco, CA, USA, pp. 392–403, 1998.
34. T. Zhang, R. Ramakrishnan and M. Livny, BIRCH: An efficient data clustering method for very large databases, in *Proc. 1996 ACM SIGMOD Intl. Conf. on Management of Data*, Montreal, Canada, pp. 103–114, 1996.

35. O. Schabenberger and C. A. Gotway, *Statistical Methods for Spatial Data Analysis*, Chapman and Hall/CRC, 2005.
36. N. A. C. Cressie, *Statistics for Spatial Data*, Wiley, 1993.
37. L. Anselin, Local indicators of spatial association, in *LISA. Geographical Analysis*, Vol. 27, pp. 93–115, 1995.
38. P. Sun, S. Chawla, On local spatial outliers, in *Proc. Intl. Conf. on Data Mining*, Washington, DC, pp. 209–216, 2004.
39. S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos, LOCI: Fast outlier detection using the local correlation integral, in *Proc. Intl. Conf. on Data Engineering*, Bangalore, India, pp. 315–326, 2003.