Efficiently Mining Regional Outliers in Spatial Data

Richard Frank, Wen Jin, Martin Ester

School of Computing Science
Simon Fraser University
Burnaby B.C., Canada V5A 1S6
{rfrank, wjin, ester}@cs.sfu.ca

Abstract. With the increasing availability of spatial data in many applications, spatial clustering and outlier detection has received a lot of attention in the database and data mining community. As a very prominent method, the spatial scan statistic finds a region that deviates (most) significantly from the entire dataset. In this paper, we introduce the novel problem of mining regional outliers in spatial data. A spatial regional outlier is a rectangular region which contains an outlying object such that the deviation between the non-spatial attribute value of this object and the aggregate value of this attribute over all objects in the region is maximized. Compared to the spatial scan statistic, which targets global outliers, our task aims at local spatial outliers. We introduce two greedy algorithms for mining regional outliers, growing regions by extending them by at least one neighboring object per iteration, choosing the extension which leads to the largest increase of the objective function. Our experimental evaluation on synthetic datasets and a real dataset demonstrates the meaningfulness of this new type of outliers and the greatly superior efficiency of the proposed algorithms.

Keywords: Data mining, Spatial outliers, Efficient algorithms, Delaunay-triangulation

1 Introduction

Spatial data is being collected, made available and used more and more both for research and commercial purposes. For the automatic analysis of such data, spatial data mining methods have received a lot of attention in the database and data mining community [21] [27], in particular methods for spatial clustering and outlier detection. Spatial clustering aims at partitioning the spatial region into sub-regions with high intra-region similarity and inter-region difference [13], the goal of spatial outlier detection is to find objects inconsistent with their spatial neighbours even though they may not be significantly different from the entire set of objects [27]. Hotspot analysis is related to both of the above tasks, attempting to discover spatial regions with densities or attribute values that are significantly different from the whole dataset.

Important applications are the discovery of disease outbreaks, of crime hotspots or of acts of bio-terrorism.

The goal of the spatial scan statistic [16] is to find such hotspots and various efficient methods for it have been proposed in the literature [10] [20]. Just as non-spatial outliers can be categorized into global [17] and local [7] outliers, so can spatial outliers. The spatial scan statistic is a global method, targeting a region that deviates (most) significantly from the entire dataset. In many applications however, users are interested in finding local outliers, spatial regions that enclose exceptional knowledge because they contain objects within the region that are outliers relative to the region itself. For example, according to *Tips & Traps when buying a home* [14], if someone wishes to multiply their chances for making money when reselling a home they have to buy an inexpensive house in the most expensive neighbourhood they can afford. As the neighbourhood appreciates over time, the least inexpensive property will appreciate more than its neighbours, relative to its price; the reverse is also true and purchasing an attractive house in a bad neighbourhood will not yield a good investment [30]. By searching for local spatial outliers in property data, it would be possible to find the least expensive properties within the best neighbourhoods, which could become promising investment opportunities.

In this paper we introduce the problem of mining regional outliers in spatial data. A spatial regional outlier is defined as a (rectangular) region with a spatial object in this region such that the value of the objective function, which measures the degree of outlierness of the object within the region, is maximized. For example, in our motivating real estate application, the object would be one (expensive) property, and the rectangle would be the equivalent to its (inexpensive) neighborhood. In a crime dataset, rectangles (neighbourhoods) could be found that have, for example, locations with very different crime rates than the neighbourhood.

A Naïve algorithm enumerating all possible rectangles has a runtime complexity of $O(n^4)$ which does not scale to large datasets as they appear in many practical applications. Therefore, greedy algorithms are presented which take advantage of the implicit neighbourhood relationships between objects and the different algorithms use different neighbourhood definitions to prune the search-space. In our methods however, the rectangles are "grown" from a seed and at each iteration are extended to include the object which causes the largest increase in the objective function.

The main contributions of our work are as follows:

1) We introduce the novel problem of mining regional outliers in spatial data.
2) We propose two greedy algorithms for efficiently mining such outliers in large datasets, reducing runtime from $O(n^4)$ to $O(n^2)$ compared to the Naïve algorithm.
3) Our extensive experimental evaluation on both synthetic and real datasets demonstrates the meaningfulness of spatial regional outliers and the efficiency of the proposed algorithms.

The rest of the paper is organized as follows. In Section 2, we discuss related work. Section 3 introduces our problem definition and presents the corresponding algorithms, including a comparative analysis of these algorithms. In Section 4 we report the results of our experiments on both synthetic and real datasets; Section 5 concludes the paper with a summary and outlook on future work.

2 Related Work

There are three parts of the literature related to our study: outlier detection, spatial scan statistics, and the use of Voronoi/Delaunay calculations. We survey them below.

(Outlier detection) Outliers can be defined as "data objects that appear inconsistent with respect to the remainder of the database" [5]. Outlier detection methods include *distribution-based* using standard statistical distributions, *depth-based* which map data objects into an *m*-dimensional information space and *distance-based approaches* which calculate the proportion of database objects that are a specified distance from a target object [13].

Statistical approaches to outlier detection [5] include distribution-based and depth-based methods. However, the first method has the problem that it must assume the dataset owns some probability distribution even though it is difficult to know the underlying data distribution. The second method is not efficient for high dimensions.

The concept of global distance-based outliers is first introduced in [17], which defines an object p being an outlier, if at most n objects are within distance d of p. It generalizes the notion of statistical outlier tests, but the running time of the proposed method is still exponential to the number of dimensions. Ramaswamy et al. [25] use the distance of the k^{th}-nearest neighbor to rank outliers and give an efficient algorithm for mining top-n global outliers. Bay and Schwabacher [8] improved the method in [17] by using the block nested loop strategy with pruning and randomization techniques. Recently, Tao et al. [29] presented a disk-resident algorithm of finding global outliers with a linear I/O cost. Shekhar et al. [27] studied spatial outliers, which refer to spatially referenced objects whose non-spatial attributes are significantly inconsistent with its neighbors, even though they may not be significantly different from the entire objects.

Breunig et al. introduced the concept of local outlier, a kind of density-based outlier, which assigns each data a local outlier factor LOF of being an outlier depending on their neighborhood [7]. The outlier factors can be computed very efficiently if some multi-dimensional index structures such as R-tree and X-tree [6] are employed. A top-n based local outliers mining algorithm which uses distance bound of micro-cluster to estimate the density, was presented in [15]. Lazarevic and Kumar [18] proposed a local outlier detection algorithm with a technique called "feature bagging".

Some clustering algorithms like DBSCAN [9] consider identifying outliers, but only to the point of ensuring that they do not interfere with the clustering process. Further, outliers are only by-products of clustering algorithms.

(Spatial Scan Statistics) The task of spatial scan statistics, which computes the maximum discrepancy region by scanning a set of circular regions with different radius in the spatial space [16], has received much attention in the data mining community. Authors in [10] proposed a greedy method to find a sub-region R of the input domain S for which the mean value of R is as large as possible. Neill and Moore [20] aimed to find a 2-dimensional rectangular region or square region with highest density given an $n \times n$ grid of rectangles/squares. As an extension work, assuming a uniform, multidimensional grid of bivariate data, where each cell of the grid has a count C_i and a baseline B_i, they aim to find spatial regions (d-dimensional rectangles) where the C_i are significantly higher than expected given B_i [21]. Agarwal et al.[2]

studied the problem of largest discrepancy region in a domain, and present a new exact algorithm, which has the same asymptotic running time as the algorithm of Neill and Moore [20], but with much simpler implementation. Authors in [3] present algorithms for maximizing statistical discrepancy functions over the space of axis-parallel rectangles with provable approximation guarantees, both additive and relative. Their methods apply to any convex discrepancy function.

(Voronoi Diagrams) With spatial datasets, Voronoi diagrams and Delaunay triangulations represent the spatial relationships between the objects [1], [12]. The dataset is partitioned into regions, called Voronoi cells, containing all the points that are closest to the object in the Voronoi cell [19]. For point data, the bisector segments will be the perpendicular bisectors of neighbouring pairs of sites while for spatially extended data, the borders will be circular arcs or arcs of parabolas [19].

Without a loss in the usefulness the distance measure could be exchanged for any distance measure, for example, the Manhattan distance or the distance covered by visiting *k* shops, as illustrated in [23]. [12] defined the distance measure as 'furthest-distance' indicating regions of *least* influence. [24] applied Voronoi diagrams to measure flow in population samples and then model profitability of destination points (i.e.: stores). Voronoi diagrams are also used to describe the internal structure of objects in [19] while [11] uses it for two applications: a) the catchment area of each object and b) denoting the largest polluter of the object in the Voronoi cell.

The dual of the Voronoi diagram, the Delaunay triangulation, is the structure that is the result of connecting all objects with neighbouring Voronoi cells (Fig. 1). [19] defines the Delaunay triangulation of a set of points S as *"a partition of the convex hull of S into polytopical regions whose vertices are the points in S. The convex hull of the nearest neighbour set of a Voronoi vertex v is called the Delaunay cell of v."*

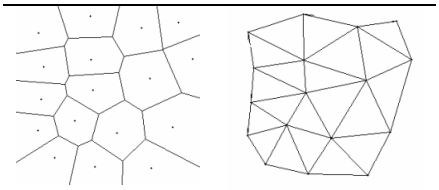


Fig. 1. Voronoi Diagram and corresponding Delaunay triangulation

3 Mining Regional Outliers in Spatial Data

In this section, we introduce the problem of mining regional outliers in spatial data (Section 3.1). We present a Naïve algorithm that enumerates all possible solutions to search for the best regional outlier (Section 3.2). Section 3.3 proposes a greedy algorithm, Global Neighbourhood Algorithm (GNA), which at each iteration considers adding to the region the object which causes the largest increase in the objective function. Section 3.4 introduces another greedy algorithm, Local Neighbourhood Algorithm (LNA), which prunes the search-space even further by only considering objects which are direct neighbours. The most expensive operation of the greedy algorithms is the calculation of the objective function for each rectangle evaluated, which is efficiently supported by a method of caching (Section 3.5).

3.1 Problem definition

According to Tobler's First Law of Geography *'everything is related to everything else; but that near things are more related than those far apart'* [28]. Hence it is expected that within the spatial data relationships exist between objects that are near each other, but not necessarily between those that are far apart. The existence of the relationships causes neighbourhoods to be formed within the spatial data. The neighbourhoods can exhibit spatial trends which illustrate the correlation of one or more non-spatial attributes and the distance away from a central object [26]. A spatial outlier is an object which is inconsistent with its spatial neighbours even if the non-spatial values are normal for the rest of the objects of the same class. Due to this, non-spatial outlier detection methods cannot work accurately without somehow taking into account the spatial location [13].

Fig. 2 shows a representation of a small part of the BC Assessment dataset, consisting of properties with spatial attributes (street-address and object polygons) and non-spatial attributes (various values, for example the total value of the property and building). The street-addresses are converted to a longitude/latitude and used for outlier detection. In the following, we formally define the problem.

Definition 1: A spatial dataset D is a set of objects $P \in D$ with 2 dimensional coordinates (X,Y) and at least one non-spatial descriptive attribute value v.

We find rectangular regions, which contain the most deviating outlier given the values of all the objects in the region. This is equivalent to finding the most expensive, or cheapest, building in a neighbourhood for example. For each region that is considered, an objective function f calculates and assigns to the region a value which indicates the degree to which the region is an outlier. This is then maximized across the entire dataset to determine the best region with the largest outlier value. The proposed approach and algorithms work equally well for regions of other shapes, but for cities with grid-like road-networks, rectangular blocks are appropriate.

Definition 2.: Given a spatial dataset D. A region R is an axis-parallel rectangular area $R=(P_L,P_U)$, $P_L, P_U \in \mathbb{R}^2$, where P_L denotes the lower-left vertex (X_L,Y_L) and P_U the upper-right vertex (X_U,Y_U). For every edge of R, there exists an object $P \in D$ that lies on the edge. More precisely, for each pair of neighbouring vertices (X_i,Y_i) and

Fig. 2. Sample of the BC Assessment dataset

(X_j, Y_j) of R there is a $\lambda \in \mathbb{R}$, such that $(X_i, Y_i) + \lambda\left((X_i, Y_i) - (X_j, Y_j)\right) = P$, $\lambda \in [0,1]$. The set of objects in R is given by $S_R = \{P \in D, P = (X,Y) \mid X_L \le X \le X_U \wedge Y_L \le Y \le Y_U)\}$. The complimentary set is given by $S_{\overline{R}} = D - S_R$.

Definition 3.: Given a spatial dataset D and a region R. The value of the region under consideration, $V_R \in \mathbb{R}$, is the maximum value of applying the objective function to R and any object $P \in S_R$.

The objective function has to compare the range of values within the region against the value of a certain individual object in the same region. This is done by aggregating the attribute values of all objects in the region and comparing the individual against the aggregate value. For example, some alternate objective functions are shown below, where v_i is the value of object i which is in the region R:

- (average of all objects in R) – (lowest value of an object):
$$AVG_{i=1}^{|R|}(v_i) - MIN_{i=1}^{|R|}(v_i) \quad (1)$$
- (average of all above-average objects) – (lowest value of an object):
$$AVG_{i=1, V_i > AVG_{i=1}^{|R|}(v_i)}^{|R|}(v_i) - MIN_{i=1}^{|R|}(v_i) \quad (2)$$
- (average of above-average objects) – (average of below-average objects):
$$AVG_{i=1, V_i > AVG_{i=1}^{|R|}(v_i)}^{|R|}(v_i) - AVG_{i=1, V_i < AVG_{i=1}^{|R|}(v_i)}^{|R|}(v_i) \quad (3)$$
- (average of top-k highest) – (average of top-k lowest):
$$AVG_{i=1, i \in TOP(k)}^{|R|}(v_i) - AVG_{i=1, i \in BOTTOM(k)}^{|R|}(v_i) \quad (4)$$
- ABS[(average of all objects in R) – (value of most extreme object)]:
$$MAX[AVG_{i=1}^{|R|}(v_i) - MIN_{i=1}^{|R|}(v_i), MAX_{i=1}^{|R|}(v_i) - AVG_{i=1}^{|R|}(v_i)] \quad (5)$$

Definition 4.: Given a dataset D and an objective function f, a Spatial Regional Outlier is the region R such that the value of R is maximum over all R. The top-k Spatial Regional Outliers for D are the top-k regions with the k-highest f values over the entire dataset.

Although any of the objective functions above could be used in our problem definition, we used function (5) in our experimental evaluation. It is an intuitive and understandable objective function that would find the largest deviation from the average value. The result would be, for example, the least/most expensive property in the most/least expensive neighbourhood.

3.2 Naïve algorithm

The Naïve algorithm enumerates all possible rectangles defined by at most four objects from the dataset, with one object lying on at least one edge of the rectangle. Assume four objects would make up a rectangle: $P_1=(X_1,Y_1)$, $P_2=(X_2,Y_2)$, $P_3=(X_3,Y_3)$, $P_4=(X_4,Y_4)$; the left bottom vertex of the rectangle can be given by:

$$(X_L, Y_L) = (MIN(X_1, X_2, X_3, X_4), MIN(Y_1, Y_2, Y_3, Y_4))$$

and the right upper vertex given by

$$(X_U, Y_U) = (MAX(X_1, X_2, X_3, X_4), MAX(Y_1, Y_2, Y_3, Y_4)).$$

```
INPUT:dataset D,eval function, k
OUTPUT: TOP-k outlier regions

Algorithm Naïve(D, f, k)
 V_R = 0, V_R' = 0, TOP-k = {}
 For P_1 ∈ D
  For P_2 ∈ D
   For P_3 ∈ D
    For P_4 ∈ D
    (X_L, Y_L) = (MIN(X_1, X_2, X_3, X_4), MIN(Y_1, Y_2, Y_3, Y_4))
    (X_U, Y_U) = (MAX(X_1, X_2, X_3, X_4), MAX(Y_1, Y_2, Y_3, Y_4))
     R←rectangle({X_L, Y_L}, {X_U, Y_U})
     V_R←GetRectangleValue(R, f)
     If V_R>MIN(V_R) for R∈TOP-k
      Add R to TOP-k
 Return TOP-k
```

Fig. 3. Naïve algorithm. Function *f* is a user-defined objective function

```
INPUT: region R, eval function
OUTPUT: value of rectangle

Method GetRectangleValue(R,f)
 Get bin # R would be placed in
 If R already in bin
  V_R ← retrieve R from cache
 else
  retrieve objects S_R in R
  V_R ← value of S_R using f
  add {R, V_R} to cache
 Return V_R
```

Fig. 4. The function that calculates, and caches, the values of each region *R*

The naïve algorithm has a run-time complexity of $O(n^4)$, given that there are *n* objects in the dataset. This the runtime complexity applies both to the worst case and the average case. The pseudo-code for this approach is given in Fig. 3 with Fig. 4 showing how the regions are evaluated. Fig. 5 shows a sample execution of the naïve algorithm.

3.3 Greedy Global Neighbourhood Algorithm

The naïve algorithm is too inefficient for large datasets. We propose to improve the efficiency, although at the expense of guaranteeing optimality, through an iterative greedy algorithm. Using any of the objects as seed, we iteratively grow the rectangle by including one or more object at a time, always choosing the one that leads to the highest increase of the objective function. In the Global Neighbourhood Algorithm (GNA), each object is defined to be a 'neighbour' to each other object, hence the entire dataset consists of a 'global neighbourhood'. At each iteration the current region *R* is extended by adding to *R* the object which yields the largest increase in *f* until there are no objects in $S_{\overline{R}}$ that increase the value of *f*. For the pseudo-code of algorithm GNA see Fig. 6 and Fig. 7.

Given a starting set of objects S_R in region *R*, which in the initial iteration only contains a single object called the seed, the greedy algorithm considers all objects in $S_{\overline{R}}$ as possible extensions to *R*. During each iteration, a locally optimal object O', yielding the highest V_R from $S_{\overline{R}}$, is selected after which O' is added to S_R (Fig. 7).

The addition of an object O causes *R* to expand, and since *R* is limited to a rectangular shape, it will also add all intermediate objects to S_R between the original *R* and O. For example, given the shaded region *R* in Fig. 8 the extension adding the

circled object would create a much larger rectangle which includes all intermediary objects. Hence when calculating the value of the objective function for a potential extension, the entire rectangle needs to be constructed and all objects falling into it considered.

Analysis

Algorithm GNA has a worst-case runtime complexity of $O(n^3)$, and expected runtime of $O(n^2)$. For the worst-case scenario assume that at each iteration a single object is added into S_R. Given a seed-object, the first iteration will consider all (n-1) objects in $S_{\overline{R}}$, the second iteration will consider (n-2) objects, the third iteration (n-3), etc, with the last iteration considering a single object. Since each iteration only adds a single object to S_R, thus there are a total of n iterations, each considering on average $n/2$ objects and this is repeated for all n seed-objects. Thus the worst-case runtime is $O(n^3)$.

For the proof of the $O(n^2)$ expected run-time, we first analyze the runtime complexity per seed object. Assume that with each iteration p percent of the data are additionally covered. Hence, at iteration (1), n-1 objects are considered, of those pn are added to S_R, and removed from $S_{\overline{R}}$. With iteration (2), n-pn-1 objects are left and

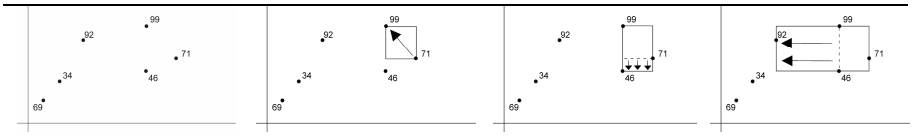


Fig. 5. Sample rectangles created by the Naïve algorithm

```
INPUT:dataset D,eval function,k
OUTPUT: TOP-k outlier regions

Algorithm GNA(D, f, k)
 for each seed-object O in D
  VR=0, SR = {O}, TOP-k={}
  loop
   R ← bounding region for SR
   O'←FindBestExtensionG(D,R,f)
   SR'← add O' to SR
   R'← bounding region for SR'
   VR'←GetRectangleValue(R',f)
   If VR' > VR then
    VR = VR', SR ← add O' to SR
   Else exit
   If VR>MIN(VR) for R∈TOP-k
    Add R to TOP-k
 Return TOP-k
```

Fig. 6. Pseudo-code of the GNA algorithm

```
INPUT: D, R, f
OUTPUT: object with highest f

Method
FindBestExtensionG(D,R,f)
 SR ← objects in R
 S_R̄ ← D - SR
 for object O in S_R̄
  R' ← expand R to include O
  VR←GetRectangleValue(R',f)
  if VR > VR' then
   VR = VR', O' = O
 Return O'
```

Fig. 7. Method to find best extension

considered, with *pn* added to S_R. At iteration (i) *n-(i-1)pn-1* are considered. Hence the expected number of rectangles, *T*, that are considered until iteration *i* is:

$$T = (n-1) + (n-pn-1) + (n-2pn-1) + ...(n-(i-1)pn-1)$$

$$T = in - i - \sum_{j=1}^{i}[(j-1)np] = in - i - np\sum_{j=1}^{i}(j-1)$$

Since at each iteration *pn* objects are removed, hence there are at most $i = \frac{1}{p}$ iterations. Hence,

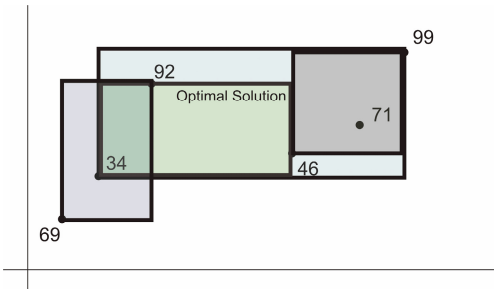
$$T = \frac{n}{p} - \frac{1}{p} - \frac{n}{p}\sum_{j=1}^{i}(j-1) = \frac{n}{p} - \frac{1}{p} - \frac{i^2 n}{p} = \frac{n}{p} - \frac{1}{p} - \frac{n}{p^3} \rightarrow O(n)$$

Since there are *n* seed-objects, hence the expected runtime of GNA is $O(n^2)$.

The drawback to the GNA algorithm is that it cannot guarantee finding the optimal solution. This is best illustrated with a counter-example (Fig. 9). In this case, the optimal solution consists of the set of three objects with attribute values {34, 46, 92}. By definition, starting at any object not in this set cannot yield the optimal value. Starting at any of the objects in the set, the greedy algorithm chooses a local-optimal object which happens not to be in the solution-set and hence any further extension can never be optimal.

The reason for the inability of the greedy algorithm to find the optimal solution is because: Starting from any object in the optimal solution, two objects have to be

Fig. 8. Original region and its extension, optimal object for extension is circled.



Optimal solution:
Object-values:{34, 92, 46}
Region-value: 34.66

Solution of GNA:
Seed-object is object 34
{34, 92, 46, 71, 99} → V_R = 34.4
Seed-object is object 92
{92, 34, 69} → V_R=31
Seed-object is object 46
{46, 71, 99} → V_R=27

Fig. 9. Algorithm GNA cannot guarantee optimality.

added to the region simultaneously in order for the optimal solution to be found. The greedy algorithm is only able to consider adding one object at a time. Adding two objects at a time would require considering pairs of objects at each iteration, yielding an $O(n^4)$ algorithm. However, even this algorithm would not be able to find rectangles where 3 objects have to be added simultaneously.

3.4 Greedy Local Neighbourhood Algorithm

Algorithm GNA, at every iteration, attempts to extend the region R by considering all other objects in $S_{\overline{R}}$. This means R could grow arbitrarily large during any given extension. An alternative approach is to limit to number of possible extensions considered in the iterations which would also allow R to grow at a much more controlled pace. A uniform and consistent way of accomplishing this would be to allow only locally neighbouring objects of R to be considered for extension. Since each iteration only extends R locally, hence the greedy Local Neighbourhood Algorithm (LNA) has to consider a smaller number of objects at each iteration.

Definition 5.: Let the set of all objects be divided into 3 subsets. The objects in R are still called S_R with the complimentary set $S_{\overline{R}}$ now being split into two subsets: those objects neighbouring objects in R are denoted by $S_{\overline{RN}}$, and those not neighbouring R are denoted by $S_{\overline{RN}}$, i.e.: $S_{\overline{RN}} = D - S_R - S_{\overline{RN}}$. The neighbourhood relationships are established via the dual of the Voronoi diagram, the Delaunay triangulation.

According to [4], a non-random dataset's Voronoi structure has complexity $O(n)$ in 2 dimensions and more specifically, on average, in a random or non-random dataset there are 6 Voronoi neighbours [22] for each object. Given that multiple objects in R could share the same neighbour and that some objects on the inside of R will only have neighbours that are also in R, hence the number of actual neighbours to R is much less than $6*|R|$.

Given any region R, only objects that are direct neighbours to at least one object in

Fig. 10. Region under consideration, S_R in dark, with its neighbors, $S_{\overline{RN}}$ in light-shading. Everything outside S_R and $S_{\overline{RN}}$ is called $S_{\overline{RN}}$.

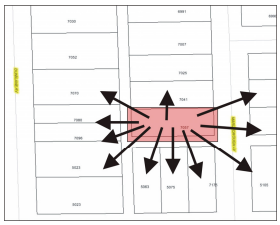


Fig. 11. A single object in S_R is related to many objects in $S_{\overline{RN}}$.

```
INPUT:dataset D,eval function,k
OUTPUT: TOP-k outlier regions

Algorithm LNA(D, f, k)
 for each seed-object O in D
  V_R=0, S_R = {O}, TOP-k = {}
  loop
   R ← bounding region for S_R
   O'←FindBestExtensionL(D,R,f)
   S_R'← add O' to S_R
   R'← bounding region for S_R'
   V_R'←GetRectangleValue(R',f)
   If V_R' > V_R then
    V_R = V_R', S_R ← add O' to S_R
   If V_R>MIN(V_R) for R∈TOP-k
    Add R to TOP-k
 Return TOP-k
```

Fig. 12. Pseudo-code for the LNA algorithm

```
INPUT: region R, eval function
OUTPUT: object with highest f

Method FindBestExtensionL(D,R,f)
 S_R̄N ←neighbours to objects in R
 for object O in S_R̄N
  R' ← expand R to include O
  V_R ← GetRectangleValue(R',f)
  if V_R > V_R' then
   V_R'= V_R, O' = O
 Return O' corresponding to V_R'
```

Fig. 13. Method to find best extension.

R would be considered as possible extensions. For example, given the original region in Fig. 8, this approach would only consider the objects highlighted in Fig. 10 as further extensions instead of all other objects in the dataset. Each object could have multiple neighbours and the union of neighbours of S_R is $S_{\bar{R}N}$ (Fig. 11). The algorithm is close to the GNA, but the set of objects analyzed at each stage is much smaller. The pseudo-code is presented in Fig. 12 and Fig. 13.

Analysis

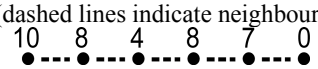
The worst-case runtime for this approach is also $O(n^3)$ because there are at most n objects considered at each iteration, and assuming that a single object is added to R at each iteration, there will be n iterations. The algorithm also analyzes each of the n seed-objects independently and hence the worst-case run-time is also $O(n^3)$. The expected runtime complexity is similar to the GNA, but since at each iteration only the local neighbourhood is analyzed rather than the global neighbourhood, hence the constant ratio is much smaller.

Optimality is not guaranteed with this approach either, since it prunes the search space even more strictly than GNA. LNA restricts the extension-search to only the local neighbourhood, but if the optimal solution contains objects which are not

Fig. 14. Example dataset where LNA will not find the optimal solution (neighhourhood relationships are dashed).

connected according to the Delaunay triangulation (i.e.: not direct neighbours), then this approach will not find them. As a counter-example assume the dataset contains 6 objects (Fig. 14) with the optimal solution being disconnected. With this approach, that region could not be discovered since the objects in the region do not share neighbourhood relationships (shown as dashes).

This method might also not find the optimal solution if the data is connected. For example, the 6 data-objects (dashed lines indicate neighbourhood relationships)



will not yield an optimal solution using the LNA. Starting at any of the objects in subset {10-8-4-8} will lead to a local-optimal solution of "10-8-4-8", while starting with any of the objects in {8-7-0} will lead to a local-optimal solution of "8-7-0". Although this algorithm theoretically cannot guarantee finding the optimal solution, in practice it often finds target outliers that are very close to the optimal. For a discussion on the optimality of our approaches based on experiments, see Section 4.3.

3.5 Rectangle Caching

One of the most expensive operations within the algorithm was the calculation of V_R for each region R that had to be evaluated. One solution to this is to keep a record of all the regions that have been evaluated and if the same region is being re-calculated then use the stored value.

Each region can be described by two pairs of (X,Y) (or longitude/latitude) coordinates, an (X,Y) pair for the lower-left and upper-right corners of the region. Let this pair be represented by the 4 numbers $\{X_1,Y_1,X_2,Y_2\}$, for example {-130.495, 49.58, -130.694, 49.70}. Since the numbers are arbitrarily large (and possibly negative), hence representing this as a 4D matrix will not be possible. All 4 coordinates however can be binned by normalizing to an integer value between 1 and 10 after which they can be represented as a 4D matrix or a tree. For example, $\{X_1,Y_1,X_2,Y_2\}$ could become bin number {5, 4, 6, 8}. There will be multiple $\{X_1,Y_1,X_2,Y_2\}$ regions that will map into the same bin, but for any given $\{X_1,Y_1,X_2,Y_2\}$ only the bin it is mapped into would have to be scanned to see if it has already been evaluated. If it exists in the bin then retrieve the value, otherwise evaluate the region and place it into the bin. By modifying the number of bins, it is possible to significantly influence the number of comparisons that must be done, depending on the dataset. This caching is shown in Fig. 4.

4 Experimental Evaluation

The experiments were run on both synthetic and real data. The synthetic data was generated using a uniform distribution of n objects and was used to compare the three approaches in order to evaluate the efficiency of each algorithm. This is presented in Section 4.1. In order to find neighbourhoods where an individual property is most different from the neighbourhood, experiments (shown in Section 4.2) were also run

on the BC Assessment Authority dataset. The dataset consists of 667,734 properties, and includes the location and assessed values of each property in BC, Canada.

All experiments were performed on an Intel Core2Duo 6300 @ 2.5GHz with 2GB of RAM. The implementation did not take advantage of multi-threading. We searched for the top 5 outlier regions in each dataset and used a bin-size of 500 for our rectangle-cache. The neighbourhood relationships were calculated via a call to MatLab, which is treated as a black-box.

4.1 Synthetic Datasets

Different size datasets were generated randomly with both uniformly distributed (X,Y) coordinates and descriptive attribute values. The data-size was doubled for each consecutive run. The results are shown in Fig. 15. The runtime for the naïve algorithm quickly becomes prohibitive, running a small dataset of 160 objects took 2.5 hours to process and each iteration increased the number of rectangles evaluated as well as runtime by approximately a factor of 16. The runtime for the GNA was much better, but it also quickly became prohibitive as anything above 1000 records already required hours to run. With the GNA, the number of rectangular regions that must be evaluated also increased exponentially. The LNA approach however had a super-linear runtime and was able to process datasets larger than 20,000. This was due to the much smaller neighbourhood it had to evaluate at each iteration.

Since the naïve algorithm become infeasible even with a small dataset, the effect on the run-time as a result of doubling the dataset was investigated and is presented in Fig. 16. 'Number of Doublings' of 0 corresponds to a dataset size of 10. It is clear in these results that the naïve complexity approximately increases by a factor of 16 when the dataset is doubled: $O(n^4)$. It becomes prohibitive after only 4 doublings (dataset size of 160). The GNA is two factors better at $O(n^2)$ but it still became prohibitive after 7 doublings (corresponding to a dataset size of 1280). The LNA algorithm was surprisingly much better because a doubling of the dataset led to a straight doubling of the run-time implying that this is actually an $O(n)$ algorithm; it became prohibitive after 11 doublings (dataset size of 20480).

4.2 Real dataset

The experiments were run on the 2005 dataset of the British Columbia Assessment Authority (BCAA). The dataset consists of the street-address, assessed property and

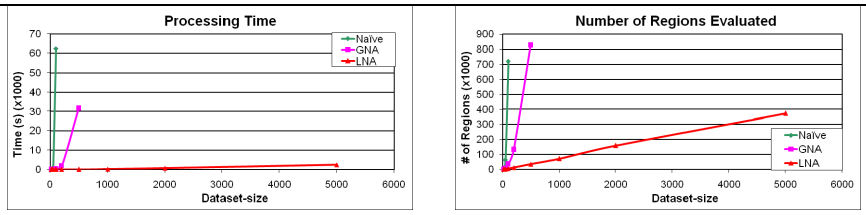


Fig. 15. Run-time results and number of regions vs. dataset size

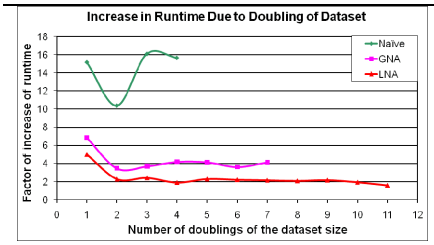


Fig. 16. Factor of increase in runtime as a result of a doubling the dataset.

Datasize	City	Type of Property
13	Vancouver	Shopping Centre - Neighbourhood
23	Vancouver	2 Acres Or More - Single Family Dwelling
33	Vancouver	Printing & Publishing Industry
46	Vancouver	Shopping Centre - Regional
70	Vancouver	Multi-Family - Apartment & Row Housing
152	Burnaby	Office Building (Primary Use)
220	Vancouver	Churches & Bible Schools
483	Vancouver	Store(S) And Offices
1462	New West	Single Family Dwelling With Basement Suite
3122	Langley	Single Family Dwelling
4794	North Van	Single Family Dwelling

Fig. 17. Property types and cities used

building values of all taxable properties (homes, businesses, etc) within the borders of British Columbia. Each plot was also categorized into 191 types of properties. The original dataset consisted of 667,734 such records. 15,268 of those properties had no specified street-address since they were classified as 'vacant' and hence are not assigned addresses. As a preprocessing step, the addresses were converted into a latitude and longitude coordinate value, a process known as geo-coding. This was accomplished with the use of Microsoft MapPoint 2006 (MMP). The locations were determined to within 10 decimal places. After geo-coding was complete, it was found that 27,331 addresses existed in the source data but not on the street-network of MMP, hence were discarded, leaving 625,135 entries for outlier detection. Through the geo-coding process, each address was mapped to a single (X,Y) point. Note that our problem definition requires spatial objects that are points, not polygons.

Selecting a single large dataset and creating samples of different sizes from within that set to evaluate our performance would not at all have yielded correct results. For example, creating equal sized non-overlapping random samples from the set of 'Stores and Offices' in Vancouver resulted in value-ranges in one subset of ($180,400 → $1,795,000) while another subset had a value-range of ($186,600 → $10,977,000) while a third subset had a range of ($215,300 → $42,848,000). Performing analysis on these subsets would clearly have been impacted by the sampling.

Considering all types of properties together also would not yield relevant results. For example, Simon Fraser University, with an assessed value of $468million, would immediately be flagged as an outlier since it is mainly surrounded by residential property with values between $100,000 and $500,000. Hence outlier detection is performed only within each type of property. Different types of properties were extracted from the BCAA dataset to create the data that our experiments would be run on. The criteria for the different datasets used for experimentation is shown in Fig. 17. All algorithms were run on each dataset unless the runtime was infeasible.

As can be seen in Fig. 18, the run-time of both the GNA and LNA is still significantly better than the naïve algorithm, with the LNA significantly out-performing both. Whereas the naïve was only able to process a data-size of 70 properties (of type 'Multi-Family - Garden Apartment & Row Housing') in a reasonable time due to its $O(n^4)$ behaviour, the GNA was able to process 220 ('Churches & Bible Schools') and was roughly $O(n^3)$. With LNA however we were

able to get results for a data-size of 4794 properties (of type Single Family Dwelling) since it still behaved near-linearly.

The time required to process real data was significantly larger than with uniformly distributed data. This was due to one major difference between the uniformly distributed and real datasets. The real datasets included properties that shared the same address and hence geo-coded to identical coordinates, such as condominiums which could include hundreds of such residences. If all coordinates are unique then the neighbourhood relationship between them is straightforward: there is one neighbourhood relationship. However, between two neighbouring condominiums, let's say each with 100 properties, there will be 100*100=10,000 neighbourhood relationships significantly increasing the runtime since each will be evaluated.

The naïve method could only evaluate 12 regions/second while the LNA was able to evaluate 150 regions/second, hence although the number of regions is small for the naïve it was expensive compared to the LNA. This was due to our use of the cache (section 3.5); with the naïve, no region was evaluated twice whereas lots of duplicate regions were not evaluated with the LNA due to the cache, saving considerable time.

Fig. 19 shows three regional outliers found in the BCAA dataset. The 'Single-Dwelling' region identified had an average property value of $452,263 but had a single property in it worth over $1million and was the largest regional outlier identified. One of the properties within the 'Duplex' regional outlier (rectangle A, Fig. 19) had a value of $959,000, more than $200,000 more than the next most-expensive property in the neighbourhood where the average price was $587,052. Another outlier region of the same type (rectangle B) had a single property valued at $552,000 in a region with average values of $392,337.

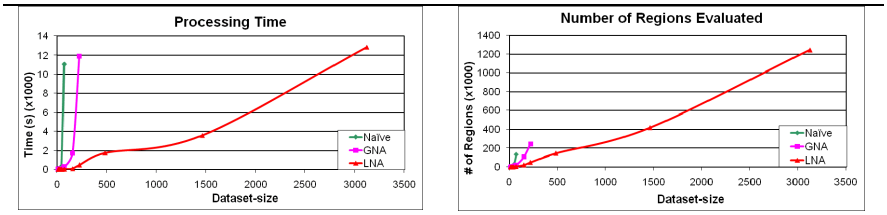


Fig. 18. Runtime and number of regions vs. dataset size

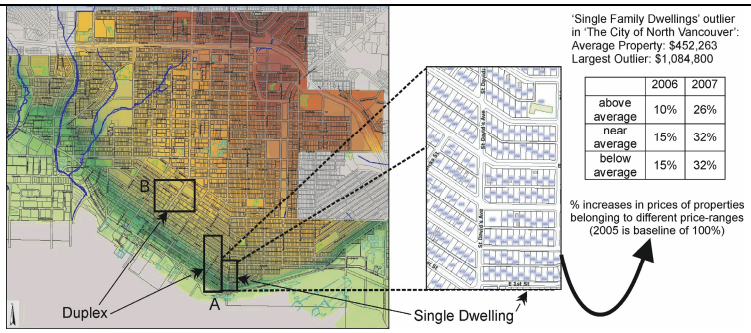


Fig. 19. Three spatial regional outliers in the BCAA dataset

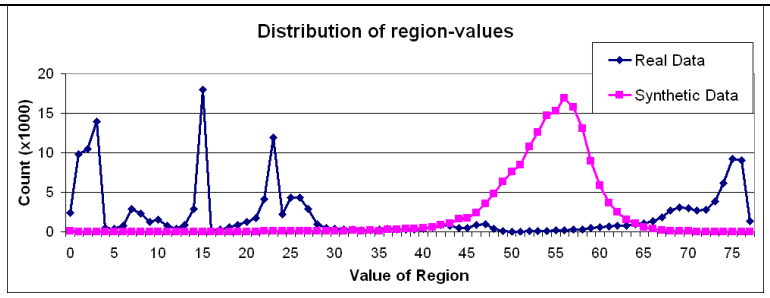


Fig. 20. Distribution of V_R both on real and synthetic data.

The test-dataset only included a single year of data, but by retrieving multi-year property assessment data from the City Of North Vancouver website[1], it was possible to further investigate. By taking 2005 as the baseline, it was possible to calculate percentage increases in some below, near and above,-average-priced properties for the 'Single-Dwelling' property type. Interestingly, the results indicated that the more expensive properties increased at a lower-rate than, while the below-average properties kept pace with, properties valued close to the average value within the region. This perfectly illustrates the significance and interestingness of our results: that purchasing an inexpensive property in a relatively expensive neighbourhood is a good investment.

4.3 Optimality

In order to test the goodness of the results of the two greedy algorithms, we tested how sub-optimal their results were compared to the naïve approach which finds the optimal solution. The naïve algorithm was run and the values of all the rectangles evaluated were stored then a distribution graph created. An example of such a distribution calculated from a uniformly distributed synthetic dataset, as well as a subset of the real dataset, is shown in Fig. 20. This figure illustrates that the bulk of the rectangles are clearly sub-optimal. In about half the cases, the greedy approaches were not able to find the global optimal solution, but they did consistently find local optima that were very close to the global optimal. In almost all cases where the global-optimal solution was not found there were less than 10 possible rectangles that yielded a better solution than the local-optima found by the greedy algorithm. The probability of randomly picking a better solution than found by the LNA is less than 0.0008%. This illustrates that the greedy approaches gain a significant improvement in runtime but sacrifice very little in optimality.

[1] http://www.cnv.org/?c=3&i=167

5 Conclusions

With the increasing availability of spatial data in many applications, methods for clustering and outlier detection in spatial data have received a lot of attention in the database and data mining community. In this paper, we have introduced the novel problem of mining regional outliers in spatial data. A spatial regional outlier is defined as a (rectangular) region which contains an outlying object such that the deviation between the non-spatial attribute value of this object and the aggregate value of this attribute over all objects in the region is maximized. In a real estate application, for example, these outliers could represent the least expensive properties within the best neighbourhoods, which could become promising investment opportunities. We have proposed two greedy algorithms for efficiently mining such outliers in large datasets, reducing the runtime from $O(n^4)$ to $O(n^2)$ compared to the Naïve algorithm that enumerates all possible rectangles. Our algorithms grow regions starting from a seed object and extending them by at least one neighboring object per iteration, always choosing the extension which leads to the largest increase of the objective function. An extensive experimental evaluation has been conducted, using synthetic datasets as well as the BC Assessment dataset. Our experimental results demonstrate the meaningfulness of spatial regional outliers. They also show that the proposed greedy algorithms scale much better to large datasets than the Naïve algorithm, while producing results that are close to the optimum solution.

There are several interesting directions for future research. In this paper we have considered only simple rectangular regions. This definition is very generic and widely applicable, but in certain applications other types of regions may be more appropriate. We plan to explore, for example, regions that are taking into account an underlying road-network or system of waterways. In the context of spatio-temporal data, it seems to be promising to investigate temporal aspects as well to find objects that have clearly deviated from their region over time, e.g. properties which have demonstrated a historic trend of outperforming their neighbourhoods.

REFERENCES

[1] F. Aurenhammer: Voronoi diagrams - a survey of a fundamental geometric data structure. ACM Computing Surveys, 23(3):345-405, September 1991.

[2] D. Agarwal, A. McGregor, J. M. Phillips, S. Venkatasubramanian and Z. Zhu: Spatial scan statistics: approximations and performance study. KDD 2006.

[3] D. Agarwal, J. M. Phillips and S. Venkatasubramanian: The hunting of the bump: on maximizing statistical discrepancy. Proc. 17th Ann. ACM-SIAM Symp. on Disc. Alg., pages 1137–1146, 2006.

[4] S. Bereg: Recent Developments and Open Problems in Voronoi Diagrams. 3rd International Symposium, ISVD '06.

[5] V. Barnett and T. Lewis: Outliers in Statistical Data. John Wiley & Sons, 1994.

[6] S. Berchtold, D. Keim, and H.-P. Kriegel: The X-tree: An efficient and robust access method for points and rectangles. In VLDB 1996.

[7] M. M. Breunig, H. P. Kriegel, R. T. Ng and J. Sander: LOF: Identifying Density-Based Local Outliers. SIGMOD 2000.

[8] S. D. Bay, M. Schwabacher: Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In KDD 2003.
[9] M. Ester, H. P. Kriegel, J. Sander and X. Xu: A Density-based Algorithm for Discovering Clusters in Large Spatial Databases. In KDD 1996.
[10] J. H. Friedman and N. I. Fisher: Bump Hunting in High-dimensional Data. Stat. and Comp., 9(2):123–143, April 1999.
[11] M. Graf, S. Winter, 2003: Netzwerk-Voronoi-Diagramme. Österreichische Zeitschrift für Vermessung und Geoinformation. 91(3): 166-174. ("Network Voronoi Diagrams", english translation available at www.sli.unimelb.edu.au/winter/pub.htm)
[12] K. Hoff, T. Culver, J. Keyser, M. Lin, D. Manocha: Fast computation of generalized voronoi diagrams using graphics hardware. Proceedings of ACM SIGGRAPH 1999, 1999.
[13] J. Han, M. Kamber, AKH. Tung: Spatial clustering methods in data mining: A survey. In H. Miller and J. Han, editors, Geographic Data Mining and Knowledge Discovery. Taylor and Francis, 2001
[14] R. Irwin: Tips and Traps When Buying a Home, Third Edition, McGraw-Hill. 2003
[15] W. Jin, A. K. H. Tung and J. W. Han: Mining Top-n Local Outliers in Large Databases. In KDD 2001.
[16] M. Kulldorff: A spatial scan statistic. Comm. in Stat.: Th. and Meth., 26:1481–1496, 1997.
[17] E. M. Knorr and R. T. Ng: Algorithms for Mining Distance-Based Outliers in Large Datasets. In VLDB 1998.
[18] A. Lazarevic and V. Kumar: Feature Bagging for Outlier Detection. In KDD 2005.
[19] N. Mayya, V.T. Rajan: Voronoi diagrams of polygons: A framework for shape representation. Journal of Mathematical Imaging and Vision, Volume 6, Issue 4, Dec 1996, Pages 355 – 378
[20] D. B. Neill and A. W. Moore. Rapid Detection of Significant Spatial clusters. In KDD, 2004.
[21] D. B. Neill, A. W. Moore, F. Pereira, and T. Mitchell: Detecting significant multidimensional spatial clusters. L.K. Saul, et al., eds. Adv. Neur. Info. Proc. Sys., 17:969–976, 2005.
[22] M. Naor, U. Wieder: Novel architectures for P2P applications: the continuous-discrete approach. ACM Symposium on Parallel Algorithms and Architectures 2003.
[23] T. Ohyama: Some Voronoi diagrams that consider consumer behavior analysis. Industrial Mathematics of the Japan Journal of Industrial and Applied Mathematics, July 2005
[24] T. Ohyama: Application of the Additively Weighted Voronoi Diagram to Flow Analysis. The 2nd International Symposium on Voronoi Diagrams in Science and Engineering, Seoul, Korea, Oct 2005
[25] S. Ramaswamy, R. Rastogi and K. Shim: Efficient Algorithms for Mining Outliers from Large Data Sets. In SIGMOD 2000.
[26] MY. Santos, LA. Amaral: Geo-spatial data mining in the analysis of a demographic database, Soft Computing - A Fusion of Foundations, Methodologies and Applications, Volume 9, Issue 5, pp 374-384, May 2005
[27] S. Shekhar, CT. Lu, P. Zhang: A unified approach to detection spatial outliers, GeoInformatica 7, pp 139-166. 2003
[28] W.R. Tobler: A computer movie simulating urban growth in the Detroit region. Economic Geography, Vol46, pp 234-240, 1970
[29] Y. F. Tao, X. K. Xiao and S. G. Zhou: Mining Distance-based Outliers from Large Databases in Any Metric Space. In KDD 2006.
[30] M.B. Weiss: The Everything Homebuying Book - All the Ins and Outs of Making the Biggest Purchase of Your Life (Paperback). Adams Media Corporation; 2nd edition (Feb 18 2003)