Interestingness Hotspot Discovery in Spatial Datasets Using a Graph-Based Approach

Fatih Akdag(✉) and Christoph F. Eick

Department of Computer Science, University of Houston, Houston, USA
{fatihak,ceick}@cs.uh.edu

Abstract. This paper proposes a novel methodology for discovering interestingness hotspots in spatial datasets using a graph-based algorithm. We define interestingness hotspots as contiguous regions in space which are interesting based on a domain expert's notion of interestingness captured by an interestingness function. In our recent work, we proposed a computational framework which discovers interestingness hotspots in *gridded* datasets using a 3-step approach which consists of seeding, hotspot growing and post-processing steps. In this work, we extend our framework to discover hotspots in any given spatial dataset. We propose a methodology which firstly creates a neighborhood graph for the given dataset and then identifies seed regions in the graph using the interestingness measure. Next, we grow interestingness hotspots from seed regions by adding neighboring nodes, maximizing the given interestingness function. Finally after all interestingness hotspots are identified, we create a polygon model for each hotspot using an approach that uses Voronoi tessellations and the convex hull of the objects belonging to the hotspot. The proposed methodology is evaluated in a case study for a 2-dimensional earthquake dataset in which we find interestingness hotspots based on variance and correlation interestingness functions.

Keywords: Spatial data mining · Interestingness hotspot · Interestingness function · Hotspot discovery · Graph-based hotspot growing algorithm · Spatial polygon models

1 Introduction

Interestingness hotspots are contiguous regions in space which are interesting based on a domain expert's notion of interestingness which is captured in an interestingness function. Typically, spatial clustering algorithms have been used to find interestingness hotspot in spatial datasets using interestingness functions; however, in our previous works [1,2] we proposed an alternative, non-clustering approach to obtain interestingness hotspots in gridded datasets, which grows interestingness hotspots from seed hotspots. In this paper, we extend our framework to discover hotspots in any given spatial dataset, relying on a graph-based framework. In gridded datasets, determining the contiguity of region is trivial: grid cells are neighboring if they share an edge. However,

P. Perner (Ed.): MLDM 2016, LNAI 9729, pp. 530–544, 2016.
DOI: 10.1007/978-3-319-41920-6_42

contiguity or neighborhood relation is not well-defined for point based datasets, and an approach is needed to define neighborhood relation between points. In this paper, we propose an approach which employs Gabriel graphs [3] as neighborhood graphs of the spatial objects in the dataset. Next, we introduce a graph-based hotspot discovery algorithm that identifies contiguous, interestingness hotspots in Gabriel graphs, maximizing a plug-in interesting function. Furthermore, we generate polygon models that describe the scope of each hotspot based on Voronoi diagram.

Finding hotspots in a dataset maximizing an interestingness function that is based on a domain expert's notion of interestingness allows domain experts to discover regions with interesting patterns in the data. For example, Miller et al. [4] identifies regions with strong association of internet ad performance and demographic data to be used for geo-targeted advertising. In our previous work [1], we identify regions with high correlation of air pollutants and Ozone levels in an air pollution dataset which can be used to find associations between air pollutants and Ozone levels.

The proposed graph-based hotspot discovery framework can be summarized as follows:

1. We propose a methodology for finding hotspots in spatial datasets that consists of 4 steps: 1) building neighborhood graph 2) finding hotspot seeds 3) growing hotspot seeds 4) generating polygon models.
2. We propose methods for creating a neighborhood relation between spatial objects.
3. A heap-based hotspot growing algorithm is proposed to find interestingness hotspots using the neighborhood graph in spatial datasets.
4. We propose an approach to generate a polygon model for two-dimensional hotspots based on Voronoi diagram.
5. The proposed interestingness hotspot discovery framework is evaluated in a case study involving a two-dimensional earthquake dataset.

The rest of the paper is organized as follows. In Section 2, we compare the existing methods for discovering hotspots in spatial datasets. Section 3 introduces our framework and Section 4 provides a detailed discussion of our methodology. We present the experimental evaluation in Section 5, and Section 6 concludes the paper.

2 Related Work

Spatial scan statistics introduced by Kulldorff [5] is the most widely used hotspot discovery tool. It searches spatial circular regions occurring within a certain time interval and can obtain circular hotspots by growing circles from a point of origin by increasing the radius of the circle. However, our framework is quite different as we compute hotspots based on a given interestingness measure rather than using distance-based features of the regions.

There are also spatial clustering algorithms which can be used for computing spatial hotspots. Varlaro et al. [6] describe the goal of spatial clustering "*to group nearby sites and form clusters of homogeneous regions…Only similar sites (transitively)*

connected in the discrete spatial structure may be clustered together". However, in a clustering approach all hotspots are obtained in a single run of the clustering and the obtained clusters are always disjoint in contrast to hotspot growing approaches. DBSCAN [7] has been used and extended by many for performing spatial clustering. Another popular clustering algorithm SNN [8] (Shared Nearest Neighbor) uses the sharing of objects in k-nearest neighbor lists to assess the similarity of spatial object which enables the algorithm to identify clusters of varying densities. Wang et al. compare different spatial clustering approaches [9].

Most clustering algorithms compute clusters based on only the distance information. A new group of clustering algorithms has been introduced in the literature that find contiguous clusters by maximizing plug-in interestingness functions similar to the approach used in this paper. These algorithms are capable of considering non-spatial attributes in objective functions that drive the clustering process. Clusters are computed maximizing the sum of the rewards for each cluster based on a cluster interestingness function. CLEVER [10] is a k-medoids-style clustering algorithm which exchanges cluster representatives as long as the overall reward grows, whereas MOSAIC [11] is an agglomerative clustering algorithm which starts with a large number of small clusters, and then merges neighboring clusters as long as merging increases the overall interestingness. We use an algorithm similar to MOSAIC in our framework for reducing the number of hotspot seeds by merging seed regions if merging them increases the interestingness.

3 Interestingness Hotspot Discovery Framework

In this section, we describe the framework for discovering hotspots in spatial datasets using interestingness functions.

Interestingness hotspots are contiguous areas in space for which an interestingness function i assigns a reward $w \geq 0$, indicating "news-worthy" regional associations. Our goal is to mine spatial patterns for performance attributes in a predefined space. The scope of an interestingness hotspot is a contiguous spatial region for which the association is valid and validity is assessed using interestingness functions. Formally, we assume a spatial dataset O is given in which objects $o \in O$ are characterized by a set of performance attributes P, a set of spatial attributes S, a set of continuous attributes M, which provide meta data under which the performance attributes P are analyzed in the spatial space. Moreover, we assume that we have an interestingness measure $i: 2^O \rightarrow \{0\} \cup \Re^+$ that assesses the interestingness of subsets of the objects in O by assigning rewards to a particular cluster H. Moreover, we assume a spatial neighboring relationship $N \subseteq O \times O$ is given that describes which objects belonging to O are neighbors. N is usually computed using spatial attributes S of objects in O. Finally, we assume an interestingness threshold θ is given that defines which patterns are interesting.

In this research we find interestingness hotspots $H \subset O$; where H is an interestingness hotspot with respect to *i,* if the following 2 conditions are met:

1. $i(H) \geq \theta$
2. H is contiguous with respect to a neighborhood relation N; that is, for each pair of objects (o,v) with $o,v \in H$, there has to be a path from o to v that traverses

neighboring objects (w.r.t. N) belonging to H. In summary, interestingness hotspots H are contiguous regions in space that are interesting (i(H) ≥ θ).

The most simplistic interestingness i_p measure we can think of is one that directly uses the value of a single performance attribute p, which is defined as follows:

$$i_p(\mathrm{H}) = \frac{\sum_{h \in H} h.p}{|H|} \tag{1}$$

where H⊆O is an interestingness hotspot, |H| denotes cardinality of H and h.p denotes the value for attribute p for cell h in H.

Another interestingness function considers the correlation of two performance attributes p_1 and p_2; the corresponding interestingness function $i_{\mathrm{corr}(p1,p2)}$ is defined as follows:

$$i_{corr(p1,p2)}(H) = \begin{Bmatrix} 0, & if\ |correl(H,p_1,p_2)| < \theta \\ |correl(H,p_1,p_2)| - \theta, & otherwise \end{Bmatrix} \tag{2}$$

where 0 < θ < 1 is the interestingness threshold, and $correl(H,p_1,p_2)$ is the correlation of attributes p_1 and p_2 with respect to the objects belonging to hotspot H. This interestingness function is used to find regions in a dataset where the performance attributes p_1 and p_2 are correlated and therefore allows for the identification of regional correlation patterns in spatial datasets which is needed for commercial applications, such as geo-targeting.

Finally, variance interestingness function considers the variance of a performance attribute p and we define the corresponding interestingness function $i_{\mathrm{var}\,(p)}$ as follows:

$$i_{var(p)}(H) = \begin{Bmatrix} \theta - variance(H,\mathrm{p}), & if\ variance(H,p) < \theta \\ 0, & otherwise \end{Bmatrix} \tag{3}$$

where θ > 0 is the variance threshold, and $variance(H,p)$ is the variance of an attribute p with respect to the objects that form hotspot H. This interestingness function is used to find regions in a dataset where the performance attribute p does not change significantly. The obtained hotspots can be used to generate maps for the performance attribute and for generating prediction models for the performance attribute—similar to regression trees.

The proposed hotspot discovery framework firstly identifies some small regions with high interestingness as seed regions, and then grow these seed regions by adding neighboring objects which increase the reward most when added.

4 Methodology

In this section, we describe the graph-based interestingness hotspot discovery algorithm which works in 4 phases:

1) Building a neighborhood graph
2) Finding "small" hotspot seed regions with high interestingness
3) Finding hotspots by growing the hotspot from the seed regions
4) Generating a polygon model for the hotspots found in Phase 3

4.1 Building a Neighborhood Graph

As interestingness hotspots are defined as contiguous regions in space, a neighborhood relation has to be defined for the spatial objects in the dataset that indicates which objects are neighboring. Various neighborhood graphs have been proposed in the literature. The most popular graphs include: Delaunay triangulation, Gabriel graphs, relative neighborhood graphs, Euclidian minimum spanning trees and Beta skeletons. Fig. 1 shows comparison of 4 popular graph types for a dog shaped point set. As seen in the figure, the Delaunay triangulation (DT), in general, contains many edges between distant points in the dataset which are non-intuitive as they capture irrelevant relationships; therefore it is not a good choice for a neighborhood graph. On the other hand, minimum spanning tree and relative neighborhood graph contain only a small amount of connections between points and many close points are not connected, losing important relationships. In contrast, Gabriel graphs strike a good balance: many edges between distant points in the DT are eliminated, yet edges between close points are preserved. Thus, we use Gabriel graphs to identify neighboring objects in spatial datasets. For a more detailed discussion of various neighborhood graphs we refer to [12, 13].

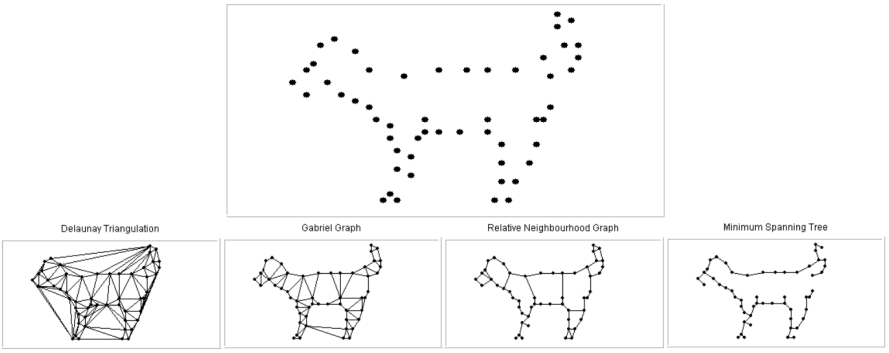


Fig. 1. Popular proximity graph types for a dog shaped dataset

Building Gabriel Graphs: Gabriel graphs are defined by its authors [3] as "Any two localities [points in the plane] A,B, are said to be contiguous iff all other localities are outside the A-B circle, that is, the circle on whose circumference A and B are at opposite points. In other words, two localities A and B are contiguous, unless there exists some other locality C such that in the triangle ABC the angle subtended at C is of 90 degree or more." Any points P and Q in a dataset are adjacent in the Gabriel graph if P and Q are distinct and the closed disc *D*, of which the line segment PQ is a diameter, contains no other points. Unlike Delaunay graph, Gabriel graphs generalize to higher dimensions, with the empty disks replaced by empty closed balls. For 2-dimensional data, the Gabriel graph can be computed from the Delaunay graph in O(n) time [12], in a total of O(*n*log*n*) complexity. For higher dimensional data, Gabriel graph can be computed in $O(n^3)$ time by brute force.

4.2 Finding Small Hotspot Seeds with High Interestingness

Once we create the Gabriel graph for the spatial dataset, in this phase, we identify the small, contiguous subgraphs with high interestingness, which we call "hotspot seed regions" and grow these seed regions in Phase 3 to obtain larger hotspots. In order to identify seed regions, we visit each vertex in the graph, and create a region consisting of the vertex and all of its neighbors. The interestingness value for each region is calculated by applying the plugin interestingness function on the set of objects in the region. A "seed interestingness threshold" is used to determine which of these regions can be used to grow larger hotspots.

4.2.1 Merging Seed Regions

The seeding phase finds many regions with high interestingness which can be used to grow hotspots in in the next phase. However, the case study reported in [1] shows that many seed regions grow to the same hotspot. This is not surprising as large hotspots usually have smaller sub-regions with high interestingness which were identified as seeds. Thus, we try to eliminate some of these seeds before growing by merging them. We use the following method to reduce the number of seed regions grown:

1) Find neighboring seed regions
2) Create a neighborhood graph of seed regions where each seed region is a node. Seed regions are neighbors if they share at least 1 node. Create an edge between nodes in this graph if the corresponding seed regions are neighbors *and* the union of these regions yields a region with an acceptable reward value. A merge threshold is used to assess if the union of two seed regions is acceptable. Weight of the edge is the reward gain when the two regions connected to the edge is merged.
3) Merge the seed regions connected to the edge with the highest weight.
4) Update seed neighborhood graph after the merge operation: Create an edge between the new node and neighbors of the merged nodes using the same procedure.
5) Continue merging seed regions as long as there are nodes to be merged in the graph.

We use a merge threshold μ to define if the union of seed regions is acceptable. If the reward of the new region is higher than the total reward of merged regions multiplied by μ, then the merge is acceptable:

$$\text{merge}(s_1, s_2) \text{ if } R(\cup(s_1,s_2)) > (R(s_1) + R(s_2)) * \mu \quad (4)$$

where R(si) represents the reward of seed region s_i. Fig. 2 gives a shortened version of seed processing algorithm explained above and Fig. 3 depicts pseudocode for the seed merge procedure. We assign the reward gain which is calculated by $R(\cup(s_1,s_2)) - (R(s_1) + R(s_2))$ as the weight of an edge. We keep all edges to be processed in a max-heap where the edge with the highest weight (reward gain) is the root of the heap tree and processed first.

```
Create an neighborhood graph G of all seed regions
Create a max-Heap H of all edges using edge weights as priority
while H has elements
     nextEdge = H.dequeue()
     if graph contains both nodes connected by nextEdge then
          Merge(nextEdge)
     end if
end while
```

Fig. 2. Pseudocode for seed processing algorithm

```
Procedure Merge (Edge e)
   Set s1 = e.source, s2 = e.target
   Merge s1 and s2 by adding all elements in s1 and s2
   in a newregion snew
   Add snew into G as a new vertex
   Remove e from G
   foreach neighbor si of s1 connected by edge ei
       if R(U(si,snew)) >  (R(si) + R(snew)) * μ then
            Create an edge enew connecting nodes si and snew
            enew.weight = R(U(si,snew))  -  (R(si) + R(snew))
            G.AddEdge(enew)
           G.RemoveEdge(ei)
      end if
   end foreach
   foreach neighbor sj of s2 connected by edge ej
       if R(U(sj,snew)) >  (R(sj) + R(snew)) * μ then
           Create an edge enew connecting nodes sj and snew
           enew.weight = R(U(sj,snew))  -  (R(sj) + R(snew))
           G.AddEdge(enew)
           G.RemoveEdge(ej)
      end if
   end foreach
   Remove s1 and s2 from the graph G
end procedure
```

Fig. 3. Pseudocode for Seed Merge procedure

4.3 Hotspot Growing Phase

In this section, we describe the heap-based hotspot growing algorithm. In hotspot growing phase, we search the best neighbor among all neighbors in each step, and after each time we add a new neighbor we do this search again. Searching for the best neighbor each time increases the complexity of hotspot growing algorithm. Instead, when new neighbors are encountered as a result of growing the hotspot, we assign each new neighbor a fitness value by evaluating the reward gain in case the neighbor

is added to the region. We use a max-heap data structure to keep the list of neighbors where the neighbor with the highest fitness value is the root of the heap tree. We add each neighbor into the heap using the fitness value as the priority. Fig. 4 gives the algorithm for each step of hotspot growing algorithm. We continue growing the region as long as there are more neighbors to be added and the interestingness of the region is higher than the interestingness threshold.

```
1:  Procedure AddNextNeighbor(region)
2:    set bestNeighbor = Heap.dequeue()
3:    add bestNeighbor to region
4:    set newReward = CalculateReward(region)
5:    foreach neighbor n of bestNeighbor
6:       if n is not in region and n is not in the neighbors list
         then
7:          set fitness = CalculateFitness(region, n)
8:          Heap.enqueue(n,fitness)
9:       end if
10:   end foreach
11:   if newReward > region.alltimeBestReward then
12:      set region.alltimeBestReward = newReward
13:      set region's overallBestGridCells =region.currentGridCells
14:   end if
15: end procedure
```

Fig. 4. Pseudocode for heap-based hotspot growing algorithm

The runtime complexity of the heap-based hotspot growing algorithm is O(nlogn) as a total of O(logn) time is spent in each step where n is the number of objects in the hotspot. We refer to [2] for more details about the runtime complexity calculation. We also implemented incremental calculation of correlation and variance interestingness function to calculate the new reward in O(1) time. Details of incremental calculation is also available in [2]. Moreover, we grow seeds in parallel using a parallel processing framework which is also discussed in [2].

4.4 Generating Polygon Models for Hotspots

In this phase, we present a method to create polygon models for 2-dimensional hotspots. Polygons serve an important role in the analysis of spatial data as they can be used as higher order representations for spatial clusters. Computationally it is much cheaper to perform certain calculations on polygons than on sets of objects. Moreover, relationships and change analysis between spatial clusters can be studied more efficiently and quantitatively by representing each spatial cluster as a polygon. Furthermore, many database systems support operations on polygons, increasing the importance of polygons as models for spatial data.

We use the Voronoi diagram for the spatial dataset as the basis for creating a polygon model for hotspots. Each point in the hotspot will either be in a Voronoi polygon, or if the point is on the convex hull of the dataset, it will not be enclosed by a Voronoi polygon (in which case it will be in an unbounded Voronoi region). In this case, we propose enclosing such points in a polygon by intersecting the convex hull of the

dataset with the Voronoi edges. Moreover, some points will not lie on the convex hull, but they will be enclosed by a polygon which crosses the convex hull. Such points and their Voronoi cells usually lie on the edges of the dataset and their Voronoi polygons are quite large, beyond the convex hull. In this case, we intersect such Voronoi polygons with the convex hull to obtain more compact hotspots. Once we create the Voronoi diagram and the convex hull of the dataset, we propose the following algorithm for creating a polygon model for a spatial hotspot:

1. Find the Voronoi polygons or edges for each point in the hotspot.
2. For each point P in the hotspot:

    a. If P is in a closed Voronoi cell (Voronoi polygon), check if it crosses with the convex hull:

        i. If the convex hull does not cross Voronoi polygon, then add this polygon into the polygon model for the hotspot.

        ii. Else if the convex hull crosses the Voronoi polygon, then the convex hull splits this polygon into 2 polygons. In this case, the point will be inside one of these polygons. Add the polygon with the point into the polygon model.

    b. If the point is not in a Voronoi polygon: find the intersection of the Voronoi edges around the point and the convex hull. The intersection will create a polygon; add this polygon into the polygon model.

This method will create polygons for all points in the hotspot and merge them. Since all points in the hotspot are connected, the union of all polygons will create one large polygon model for the hotspot.

5 Experimental Evaluation

We tested our methodology in a case study involving an earthquake dataset containing all earthquakes of magnitude 6.0 or higher in Japan and Korea region from January 1st 2000 to January 1st 2016. The dataset contains latitude, longitude, depth and magnitude of 236 earthquakes in the region. The data was downloaded from USGS (United States Geological Survey) website [14] for latitudes from 29.091 to 45.841 and longitudes from -234.756 to -210.41. Fig. 5 shows the earthquakes on a map. In the first experiment, we find hotspots with very high correlations of earthquake depth and magnitude. Next, we find hotspots where variance of earthquake depth is very low.

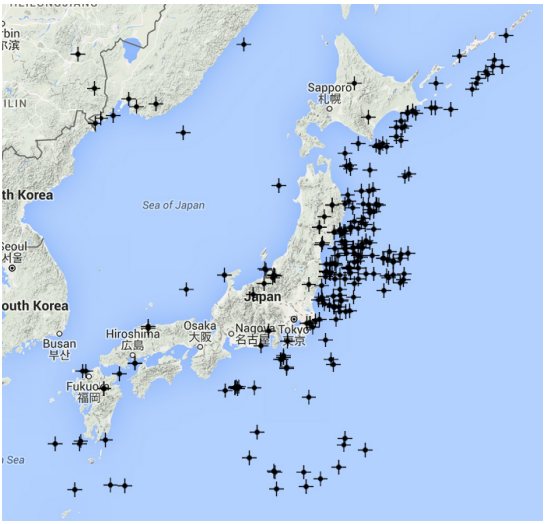


Fig. 5. Earthquake dataset visualized on a map

Following reward function is used for evaluating the quality of a region Ri:

$$\varphi(R_i) = interestingness(R_i) \times size(R_i)^{\beta} \quad (5)$$

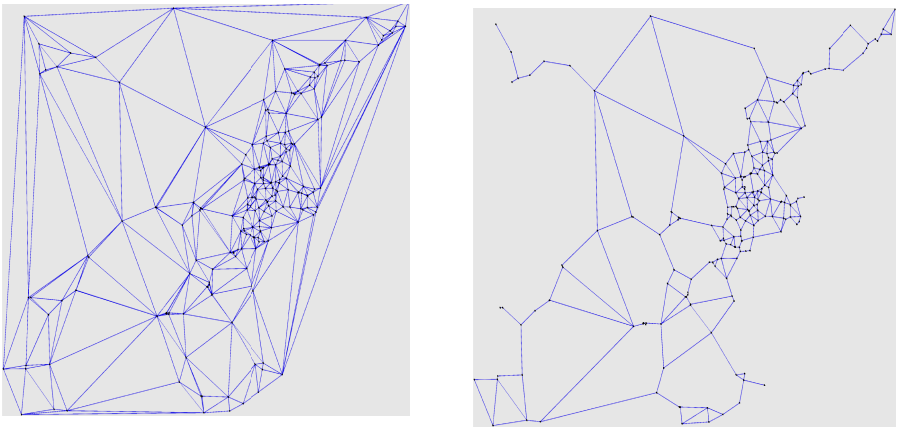
where β is a real number determining the preference for larger regions. In the case study, we set β to 1.01 as we prefer smaller regions with high interestingness to larger regions with low interestingness.

5.1 Finding High Correlation Hotspots

In this case study, we find hotspots in the area in which depth and magnitude of the earthquake is highly correlated. We use the correlation interestingness function defined in (2) and set 0.75 as the interestingness threshold.

Step 1: Building the Neighborhood Graph: Figure 6 shows the Delaunay and Gabriel graphs for this dataset. As seen on Figure 6a, Delaunay graph contains too many long edges which connect very distant objects in the dataset. Many of those edges are removed in the Gabriel graph. This provides additional evidence for our choice to use Gabriel graph to determine the neighborhood of the objects. In case it is desired, our framework allows using Delaunay graphs too. However, Delaunay graphs cannot be used for higher dimensions.

Step 2: Identifying Hotspot Seeds: We used 0.95 as the seed threshold to identify regions with very high correlation of depth and magnitude. The correlation of these variables in the dataset is 0.029, which is very low. Out of 235 regions evaluated in the dataset (1 region around each object), 33 regions had an absolute correlation value greater than 0.95. After applying seed merge operation, we obtained 30 seed regions which were grown in the next phase.



a) Delaunay Graph b) Gabriel Graph

Fig. 6. Delaunay and Gabriel graphs for the earthquake dataset

Step 3: Growing Seed Regions: Out of 30 seed regions grown, 29 of them had high positive correlation (greater than 0.75) and only 1 region had very high negative correlation (-0.93). Average positive correlation was 0.86. 3 seeds grew to the hotspots which were already discovered so they were deleted.

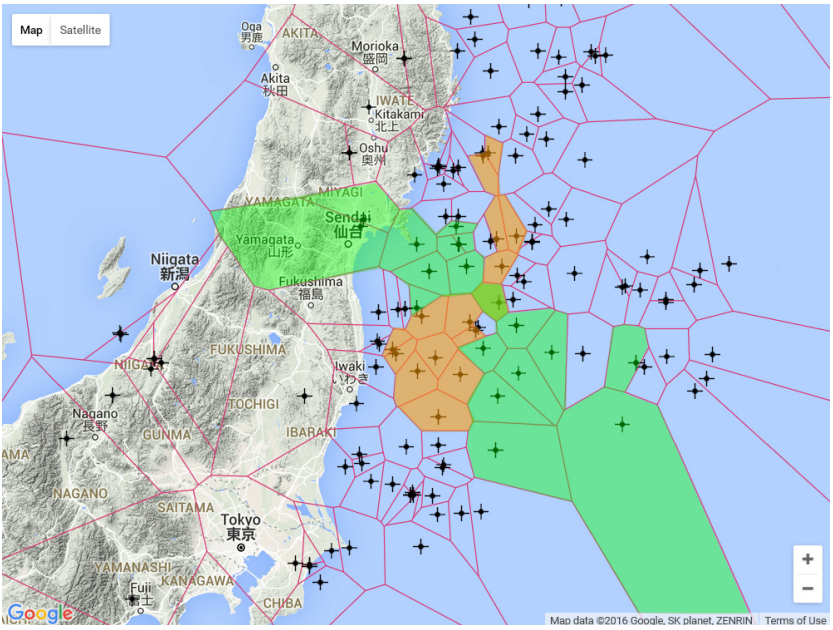
Step 4: Creating Polygon Models for Hotspots: In this step, we create polygon models for hotspots by merging the Voronoi cells of each vertex in the hotspot. As many hotspots share objects, it is not feasible to visualize all hotspots, thus we will only visualize two sample hotspots.

Figure 7 shows two hotspots discovered. As seen in the figure, the hotspots share an object in the middle. This makes sense, as that object's attributes are positively correlated with objects in the orange region and negatively correlated with objects in the green region.

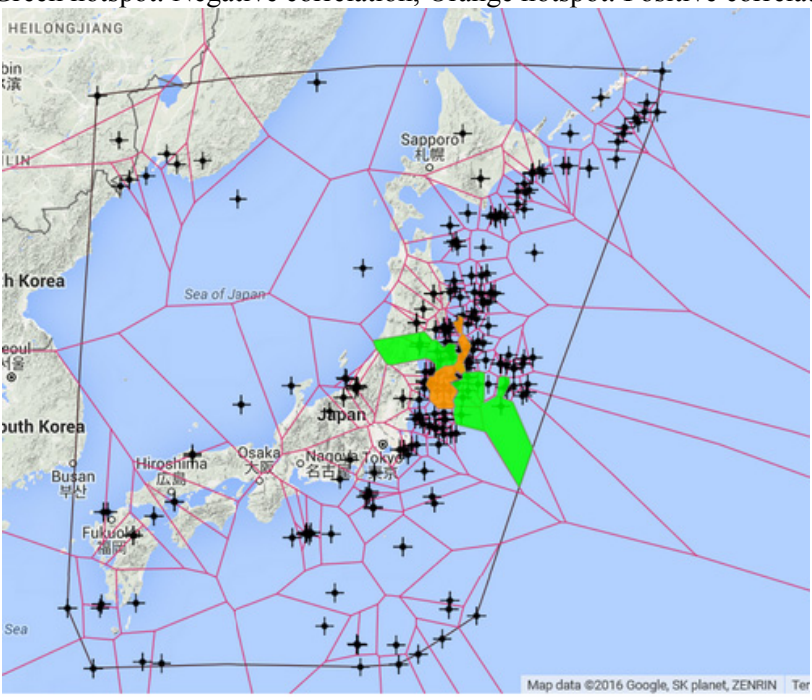
When creating the polygon model for the green colored hotspot, there was one Voronoi polygon crossing the convex hull (which is the cell on the bottom right corner in Fig 7a and that polygon was split into two parts, and the polygon with the point was added to the polygon model.

5.2 Finding Low Variation Hotspots

In this case study, we find low variation hotspots in the same geographic area in which the variance of the depth of the earthquakes is lower than 5. We used the variance interestingness function defined in (3) and set 5 as the interestingness threshold. We used 3 as the seed interestingness threshold to find small regions with variance less than 3. There were 10 seed regions in the dataset. 2 of the seed regions were merged and the resulting 9 seed regions were grown. While growing these seed regions, 2 of them grew to the already discovered hotspots so they were deleted. Table I lists the remaining 7 hotspots. Average hotspot size (number of earthquakes in the region) is 5.57. Figure 8 depicts the 3 non-overlapping low variation hotspots (hotspot 2, 6 and 7) generated.



a) Green hotspot: Negative correlation, Orange hotspot: Positive correlation



b) Hotspots on a map with smaller scale (black lines are the convex hull edges)

Fig. 7. Two hotspots and their location on maps with different scales

Table 1. Listing of discovered low variation hotspots

hotspot	size	variance
1	4	4.25
2	4	3.9
3	7	2.41
4	4	2.21
5	8	1.79
6	8	1.48
7	4	0.32

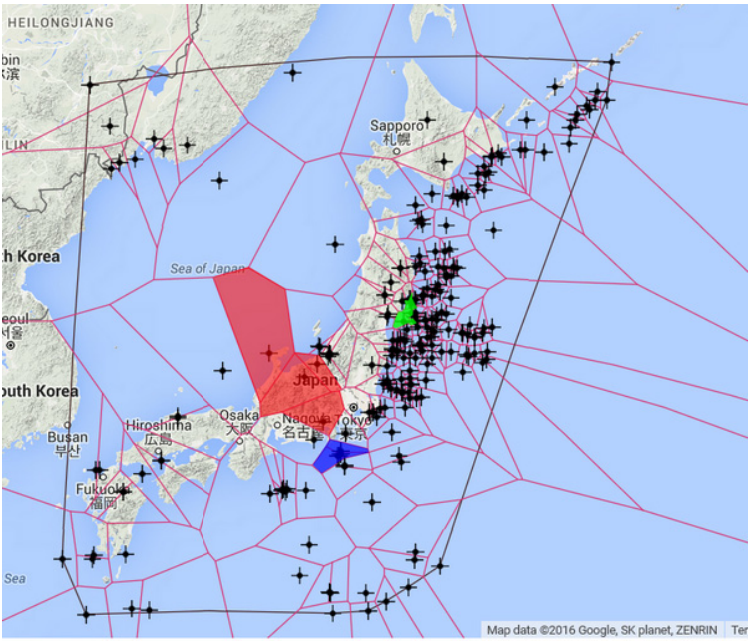


Fig. 8. Three low variation hotspots (2:blue, 6:green, 7:red) and their locations on a map

6 Conclusion

In this paper, we proposed a novel methodology for discovering interestingness hotspots in spatial datasets using a graph-based approach. The proposed methodology firstly creates a neighborhood graph for the given dataset using Gabriel graphs and then identifies small seed regions in the graph using a plugin interestingness measure. Interestingness hotspots are generated by growing seed regions by adding neighboring nodes, maximizing the given reward function. To the best of our knowledge, this is the only algorithm that grows hotspots from seed regions using a reward function. Furthermore, we create a polygon model for each hotspot by merging Voronoi cells

for each spatial object in the hotspot, identifying the scope of the identified also allowing for the quantitative assessment of the size of the hotspots and relationships to other objects in the spatial dataset.

The proposed methodology is evaluated in a case study for a 2-dimensional earthquake dataset in which we find interestingness hotspots based on variance and correlation interestingness functions. The methodology proved to be successful in finding hotspots based on plugin interestingness and reward functions. We plan to extend our framework for higher dimensional datasets in which we create higher dimensional Gabriel graphs and polygonal models.

Compared to clustering approaches we believe that our approach has more potential to compute "better", more interesting hotspots, as the clustering approach searches for all hotspots in parallel, being forced to make compromises, as switching one sub region from one to another cluster might increase the reward of one cluster but decrease the reward of the other cluster. We plan to compare our approach to clustering approaches in a future work.

References

1. Akdag, F., Davis, J.U., Eick, C.F.: A computational framework for finding interestingness hotspots in large spatio-temporal grids. In: Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, pp. 21–29. ACM, November 2014
2. Akdag, F., Eick, C.F.: An optimized interestingness hotspot discovery framework for large gridded spatio-temporal datasets. In: 2015 IEEE International Conference on Big Data (Big Data), pp. 2010–2019. IEEE, October 2015
3. Gabriel, K.R., Sokal, R.R.: A New Statistical Approach to Geographic Variation Analysis. Systematic Zoology **18**, 259–278 (1969)
4. Miller, R., Chen, C., Eick, C.F., Bagherjeiran, A.: A framework for spatial feature selection and scoping and its application to geo-targeting. In: 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services (ICSDM), pp. 26–31. IEEE (2011)
5. Kulldorff, M.: A spatial scan statistic. Communications in Statistics-Theory and methods **26**(6), 1481–1496 (1997)
6. Varlaro, A., Appice, A., Lanza, A., Malerba, D.: An ILP Approach to Spatial Clustering. Convegno Italiano di Logica Computazionale, Roma (2005)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd **96**(34), 226–231 (1996)
8. Ertöz, L., Steinbach, M., Kumar, V.: Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. SDM, pp. 47–58 (2003)
9. Wang, X., Hamilton, H.J.: A comparative study of two density-based spatial clustering algorithms for very large datasets. In: Kégl, B., Lee, H.-H. (eds.) Canadian AI 2005. LNCS (LNAI), vol. 3501, pp. 120–132. Springer, Heidelberg (2005)
10. Cao, Z., Wang, S., Forestier, G., Puissant, A., Eick, C.F.: Analyzing the composition of cities using spatial clustering. In: Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing, p. 14. ACM (2013)

11. Choo, J., Jiamthapthaksin, R., Chen, C.-S., Celepcikay, O.U., Giusti, C., Eick, C.F.: MOSAIC: a proximity graph approach for agglomerative clustering. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 231–240. Springer, Heidelberg (2007)
12. Matula, D.W., Sokal, R.R.: Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane. Geographical analysis **12**(3), 205–222 (1980)
13. Jaromczyk, J.W., Toussaint, G.T.: Relative neighborhood graphs and their relatives. Proceedings of the IEEE **80**(9), 1502–1517 (1992)
14. United States Geological Survey (USGS). http://earthquake.usgs.gov/earthquakes/search/