Polygonization of Point Clusters through Cluster Boundary Extraction for Geographical Data Mining

Ickjai Lee and Vladimir Estivill-Castro

School of Electrical Engineering and Computer Science, The University of Newcastle, NSW 2308, Australia, ijlee@cs.newcastle.edu.au, vlad@cs.newcastle.edu.au

Abstract

Interpretability and usability of clustering results are of fundamental importance. A linear time method for transforming point clusters into polygons is explored. This method automatically translates a point data layer into a space filling layer where clusters are identified as some of the resulting regions. The method is based on robustly identifying cluster boundaries in point data. The cluster polygonization process analyses the distribution of intra-cluster edges and the distribution of inter-cluster edges in Delaunay Triangulations. It approximates shapes of clusters and suggests polygons of clusters. The method can then be applied to display choropleth maps of point data without a reference map or to identify associations in the spatial dimension for geographical data mining.
Keywords: clustering, Delaunay triangulation, geographical data mining, cluster boundaries, cluster polygonization

1 Introduction

Geographic Information Systems (GIS) analyse real world phenomena with many data layers. Each layer captures some unique feature. Fast data gathering process results in data-rich environments with many layers that exceed the capability of human analysis (Miller and Han, 2001). Thus, sophisticated geographical data mining tools become necessary for handling hundreds of different themes that may contain thousands of data points. Among many other data mining techniques, clustering is one of the most popular and frequently used approaches for finding undetected or unexpected patterns of spatial concentrations residing in large databases. That is, clustering provides answers for “where” and suggests leads into “why” for post-clustering explorations. Thus, clustering is seen as a starting point

for a series of knowledge discovery processes using extensive spatial databases. However, post-clustering processes have attracted less attention than clustering itself despite their importance.

Identifying shapes (boundaries) of clusters is an intuitive way of reasoning about clusters. If shapes of clusters of a set P of points match with a particular feature data, then clusters exhibit a high association with the feature. Generally however, it is difficult to extract the shape and as yet, there is no method that can be considered as an absolute winner since most approaches are application-dependent (Okabe *et al*., 1999). The convex hull $CH(P)$ of P is one of the simplest methods to extract a shape. Since $CH(P)$ is the smallest convex set containing P, it is an appropriate choice. In addition, it is unique for P. This is useful for data mining where parameter tuning is laborious and time consuming. It is too crude a method however, to capture shape details about P unless the points are all arranged in a convex shape (in which , the vertices of $CH(P)$ coincide with P, *but do not help in summarising P).*). The α-shape (Edelsbrunner *et al*., 1983) (this is a generalisation of $CH(P)$) overcomes the crudeness of $CH(P)$. The value of the real number α controls the desired level of detail. The set of α values leads to a whole family of shapes capturing from crude to fine shapes of P. Boundary Shape Matching (BSM) (Knorr *et al*., 1997) uses the α-shape to explore associations among layers. However, several problems arise when the α-shape is used for detecting spatial cluster boundaries in data-rich environments.

1. It is difficult to determine the best value α that produces neither too crude nor too fine a shape (Knorr *et al*., 1997).
2. Several trial-and-error steps for tuning the value of α are necessary. In data-rich environments, this is laborious and time consuming.
3. If P contains i clusters, then we need to tune α for as many as i times to find the best shape of each cluster. Since clusters in P require different values for α, the independent manipulation of values of α for clusters belonging to the same layer not only increases required time but incorporates human bias.
4. The α-shape of a cluster $c \subset P$ is not affected by the distribution of points $p \in P\backslash c$. Clusters are truly the result of the characteristics in the entire distribution sampled by P. Thus, although a point $p \in P\backslash c$ does not belong to the cluster c, it has some effect on the shape of c.

In this paper, we propose an automatic boundary extraction process for clusters in point data and extend it to cluster polygonization. In contrast to the α-shape approach, our approach minimises the need for parameter tuning. Instead, it derives the shape of a cluster c from both the spread of points and the spread of points in $P\backslash c$. The proposed argument-free approach is well-suited for data-rich environments. It approximates shapes of clusters using the Delaunay Triangulation. Once the Delaunay Triangulation is constructed, the boundary extraction process requires $O(n)$ time. The algorithm can be used as post-processing for Short-Long clustering (Estivill-Castro and Lee, 2000). Short-Long

clustering is based on the Delaunay Triangulation as an underlying graph and detects various types of spatially aggregated concentrations.

The remainder of this paper is organised as follows. Section 2 details our boundary extraction and polygonization processes and analyses their complexity. We discuss experimental results with both synthetic and real datasets in Section 3. Section 4 provides three applications of cluster polygonization. Finally, the last section draws concluding remarks.

2 Extracting Cluster Boundaries

Clustering quality is of central concern when clustering is used in mining for associations. Although several spatial clustering methods have been proposed within the data mining and the GIS disciplines, most peak-inference clustering methods are based on global thresholds (Ester *et al*., 1996; Openshaw, 1987). Thus, these approaches fail to detect some spatially interesting groups.

Fig. 1 illustrates one such example. This example exhibits regions with data concentration in arbitrary shapes. These regions can be interpreted as clusters of different densities. In particular, a region ("8-like" shape) surrounding two high density regions ("ball-like" shapes) shows higher density in the study region. However, peak-inference clustering may be unable to detect this distinction without careful and laborious tuning of the global thresholds. This is because the "8-like" shape is nearby the spherical clusters. When considered as clusters, the intra-cluster distance of the sparse "8-like" shape is greater than the inter-cluster distance between the "8-like" shape and the high density "ball-like" regions. Thus, peak-inference clustering methods report three high density clusters (2 balls and the helix region at the bottom) when global parameters are set for high density regions (refer to Fig. 1(a)). Alternatively, the parameters may be set for low density regions, and in such settings the output is two clusters: one large cluster (for the merging of the "8-like" region and the two "ball-like" regions) and one high density cluster for the helix region below (refer to Fig. 1(b)). The analyst would have to contrast these two results to infer the "8-like" region that is also of interest, and only after consuming time on parameter tuning and experimentation. Fig. 1(a) and (b) demonstrate this effect with DBSCAN (Ester *et al*., 1996). The DBSCAN approach requires two global thresholds *MinPts* and *Eps*. The value of *Eps* defines a spherical neighbourhood for each point. Fixing the value of *MinPts* to 4 and then tuning the other threshold *Eps* as originally suggested for the method, we see that DBSCAN detects three high density clusters with *Eps* = 50, but leaves the sparse "8-like" region undetected as depicted in Fig. 1(a). As we increase the value of *Eps*, some points within the "8-like" are merged with the "ball-like" clusters, breaking the "8-like" shape into many meaningless sub-clusters. Finally, when the value of *Eps* is large enough so all the points within the "8-like" shape are in the same cluster, then the "8-like" and the "ball-like" shapes are all merged into a single cluster as shown in Fig. 1(b). Since this unsatisfactory behaviour occurs with this simple visual example, we cannot expect peak-

inference clustering to be effective in scenarios that are more complex. Clustering methods by partitioning (Ng and Han, 1994) utilise some objective function to find spatial groups. Although these methods are suitable for facility location

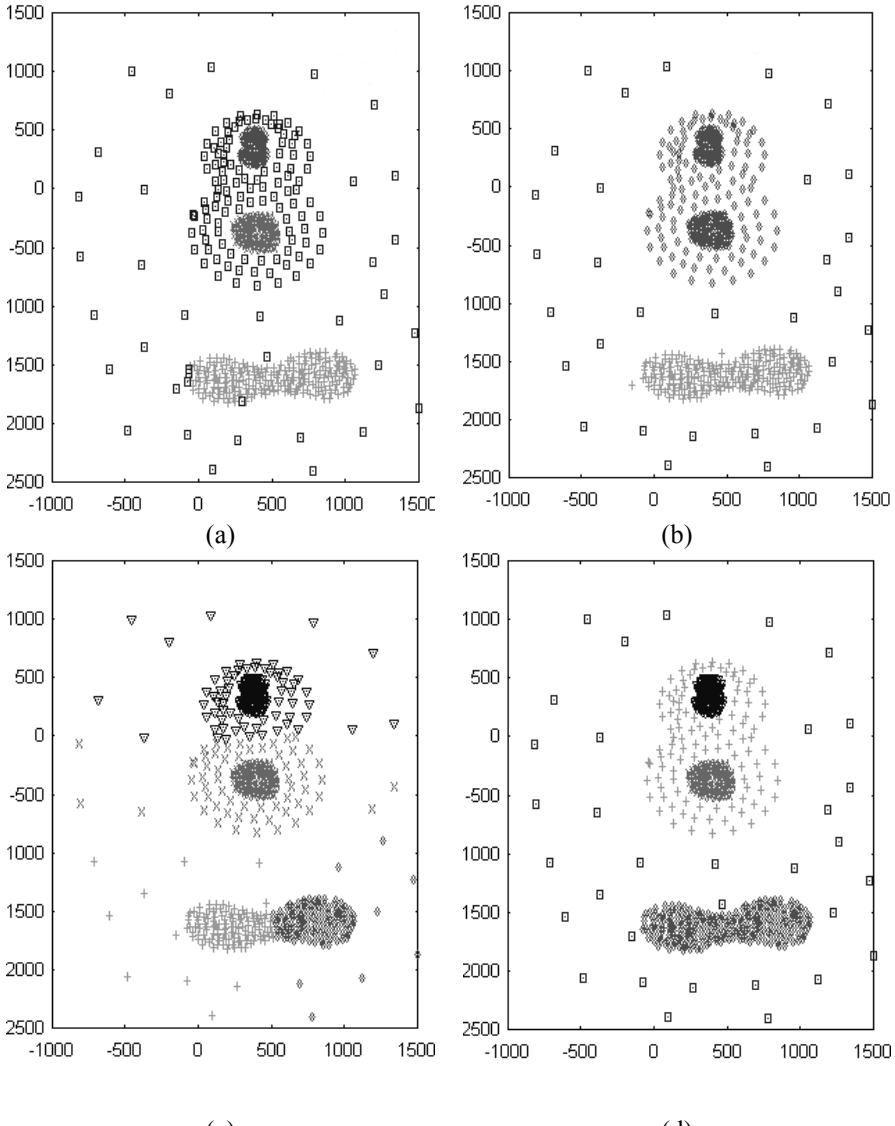


Fig. 1. Clustering with different approaches with 600 data points (Cluster-1: θ, Cluster-2: +, Cluster-3: ×, Cluster-4: ∇, Noise: ∃), **(a)** DBSCAN with *Eps* = 50 and *MinPts* = 4 (3 clusters), **(b)** DBSCAN with *Eps* = 150 and *MinPts* = 4 (2 clusters), **(c)** Partitioning clustering approaches (4 clusters), **(d)** Short-Long clustering (4 clusters)

problems, their results are convex clusters. Thus, they are less informative for cluster reasoning. Fig. 1(c) shows 4 clusters detected by medoid-basedpartitioning. Not only is the "8-like" region assigned to two clusters, but there seems to be a degree of heterogeneity in the density of the proposed clusters.

Recently, the Short-Long criteria for edge analysis in proximity graphs result in a clustering method (Estivill-Castro and Lee, 2000) overcoming drawbacks of traditional clustering methods. This approach detects possible boundaries of clusters, and as such, it is able to identify various types of spatial concentrations. Fig. 1(d) demonstrates that the Short-Long criteria find the four regions of similar concentration, despite the complexity of the "8-like" shape around the balls. Clustering for massive datasets should remain efficient and effective while minimising the number of user-supplied arguments for clustering. Efficient clustering algorithms that demand tuning of several user-supplied arguments for their best result remain unsuitable for mining vast amounts of data. Finding best values for arguments is expensive in terms of time efficiency since this necessitates several trial-and-error steps or pre-processing. Short-Long clustering is a solid candidate for data mining and post-clustering analysis, since it needs $O(n\log n)$ time, produces quality clusters and requires minimum tuning of only one parameter. For these reasons, Short-Long clustering is used as a basis for our post-clustering analysis in this paper. However, our method for polygonization extends in a straightforward manner to other clustering approaches.

2.1 The Short-Long Clustering Criteria

Short-Long clustering belongs to the family of graph-based clustering since it is based on the Delaunay Triangulation $DT(P)$ of P. In graph-based clustering, we first build a proximity graph G. The graph has node-set $N(G)$ and edge-set $E(G)$. An undirected edge $e \in E(G)$ has associated with it a set of two nodes $\{v, w\}$ while a directed edge has associated with it a pair of nodes $(v, w) \in N(G) \times N(G)$. Here, nodes represent data points and edges connect pairs of nodes to model spatial proximity, interaction or adjacency. Clustering operations are then performed on the graph. Graph-based clustering removes edges connecting points in different clusters based on a certain criterion function and computes connected components of the remaining graph $G' \subset G$ to represent clusters.

For every node $v_i \in N(DT(P))$, Short-Long clustering classifies incident edges into globally short, long or other edges. Then, it removes globally long edges when analysis reveals they are inter-cluster links and eliminates short edges and other edges when analysis reveals they are links (chains or bridges) between clusters. The notions of $ShortEdges(v_i)$, $LongEdges(v_i)$ and $OtherEdges(v_i)$ for edges incident to a node v_i are defined as follows.

$$ShortEdges(v_i) = \{ e_{vi,vj} : | e_{vi,vj}| < LocalMean(v_i)\ \ m \times MeanStDev(P)\}, \quad (1)$$

$$LongEdges(v_i) = \{ e_{vi,vj} : | e_{vi,vj}| > LocalMean(v_i) + m \times MeanStDev(P)\}, \quad (2)$$

$$OtherEdges(v_i) = AdjEdges(v_i) - (LongEdges(v_i) \cup ShortEdges(v_i)), \quad (3)$$

where $AdjEdges(v_i)$ denotes the set of edges incident to v_i in $DT(P)$, $LocalMean(v_i)$ is the mean length of $AdjEdges(v_i)$, $MeanStDev(P)$ is the average of the standard deviation of lengths of edges in $AdjEdges(v_i)$, and m is a control value. Criteria for $ShortEdges(v_i)$, $LongEdges(v_i)$ and $OtherEdges(v_i)$ are not static, but rather dynamic over the study region, since $LocalMean(v_i)$ varies with v_i. This dynamic nature of the Short-Long criteria overcomes the static nature of peak-inference clustering and partition-based clustering methods.

2.2 Cluster Polygonization Algorithm

Receiving a set of points with cluster identifier for each point is not enough for post-clustering analysis. We need to obtain the boundaries of those clusters. There are many possibilities for where the boundary might be, and following is a fast and accurate way of resolving this problem. The following technique is considered to be better than computing convex hulls or α-shapes for mining large spatial databases. In the following, the assumption is made that a dataset P is provided and each point $p \in P$ is labelled with a cluster identifier $ID(p)$. This is the input to the entire process. An illustration of a cluster polygonization process appears in Fig. 2.

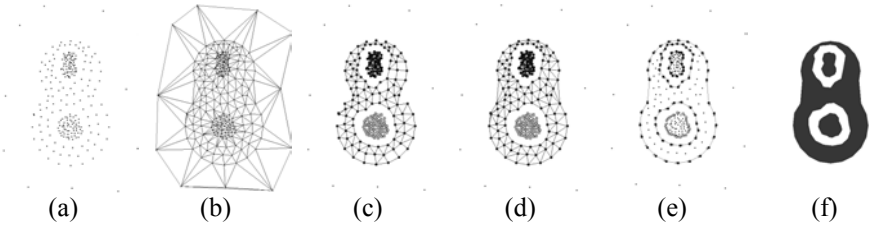


Fig. 2. Polygonization through cluster boundary extraction ($||P||$ = 250), **(a)** A set P of points, **(b)** $DT(P)$, **(c)** 3 clusters found by Short-Long clustering, **(d)** Intra-cluster edges after Phase 1 (inter-cluster edges are not shown), **(e)** After Phase 2 (the boundary extraction process), **(f)** After Phase 3 (the polygonization process)

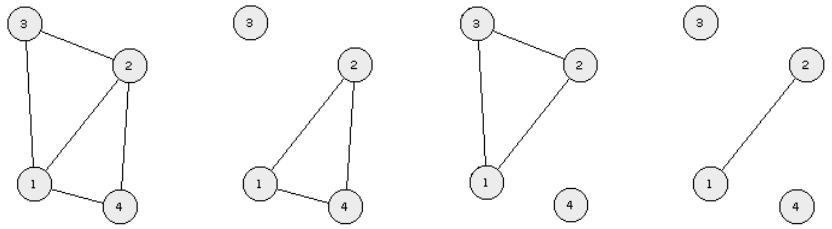
Fig. 2(c) depicts clusters after Short-Long clustering. Fig. 2(d) displays only intra-cluster edges that approximate shapes of clusters (note that, inter-cluster edges are ignored).. Directed cluster boundaries that represent cluster regions are illustrated in Fig. 2(e). Sets of cyclic lists of edges, that are in counterclockwise ordering, are external cluster boundaries and definine inner areas as cluster regions, while sets of cyclic lists of edges, that are in clockwise ordering, are internal cluster boundaries defining holes within corresponding cluster regions. Fig. 2(f) depicts derived cluster regions. Details of the cluster polygonization process are as follows. Phase 0 will compute the Delaunay Triangulation $DT(P)$ of P (this phase is omitted when using Short-Long clustering). Since the Delaunay Triangulation is a planar graph embedded in the plane, in what follows we may

refer to points $p \in P$ as nodes, and to edges $e = \{p_a, p_b\}$ as sides of the Delaunay triangles. Phase 1 labels each Delaunay edge in $DT(P)$ with either intra-cluster edge or inter-cluster edge based on cluster identifiers provided by a clustering method (in this case Short-Long clustering). Intra-cluster edges are those that expand endpoints with the same cluster identifier while inter-cluster edges are those that connect endpoints in different clusters. That is, two endpoints of intra-cluster edge $e = \{p_a, p_b\}$ satisfy $ID(p_a)$ equals to $ID(p_b)$. Phase 2 extracts and orients boundary edges. Each intra-cluster edge e is analysed. Because we have a triangulation (a planar subdivision into triangles), each edge in $DT(P)$ belongs to at least one Delaunay triangle or at most to two in $DT(P)$. For each intra-cluster edge e after Phase 1, the intra-cluster edge analysis proceeds as follows. Let the edge $e = \{p_a, p_b\}$ be an intra-cluster edge in $DT(P)$.

- Simple case: The intra-cluster edge $e = \{p_a, p_b\}$ is in only one Delaunay triangle (external Delaunay edge or hull edge (Okabe *et al*., 1999)).
    1. Sub-case: Triangle in a cluster --- The third node p_c of the triangle $<p_a, p_b, p_c>$ has the same cluster identifier as nodes p_a and p_b. Then, edge e is labelled as a boundary edge and oriented (because $DT(P)$ is a planar embedding) such that p_c is on its left (the interior of the triangle) and the exterior is on its right. Fig. 3(a) illustrates this sub-case.
    2. Sub-case: No triangle --- The third node p_c of the triangle $<p_a, p_b, p_c>$ does not have the same cluster identifier as p_a and p_b. Then, e is not labelled as a boundary edge (it will be removed). That is, e is not placed in the output of this phase because there is no area (region or triangle) of the cluster of p_a and p_b delimited by e. Fig. 3(b) illustrates this sub-case.
- Complex case: The edge $e = \{p_a, p_b\}$ is shared by two Delaunay triangles with opposing nodes p_c and p_d (internal Delaunay edge (Okabe *et al*., 1999)).
    1. Sub-case: Two triangles in a cluster --- The third nodes p_c and p_d of triangles $<p_a, p_b, p_c>$ and $<p_a, p_b, p_d>$ are both in the same connected component (cluster) as p_a and p_b. It follows that e is removed and it will not be included in the output as a boundary edge since clearly it is inside the quadrilateral $<p_a, p_c, p_b, p_d>$. Of course, his quadrilateral is inside the cluster. Fig. 4(a) illustrates this subcase.
    2. Sub-case: Triangle in a cluster --- One of the third nodes p_c or p_d is in the cluster of p_a and p_b but the other is not. Without loss of generality, we assume p_c is in the cluster of p_a and p_b (recall that $ID(p_a) = ID(p_b)$) while p_d is not (i.e. $ID(p_d) \neq ID(p_a)$). Then, edge e is a boundary edge since the triangle $<p_a, p_b, p_c>$ is in the cluster but the triangle $<p_a, p_b, p_d>$ is not. Then, the boundary edge e is oriented (because $DT(P)$ is a planar embedding) in such a way that p_c is on its left (the interior) and p_d (the exterior) is on its right. Fig. 4(b) and (c) illustrate this sub-case.
    3. Sub-case: No triangle --- Both, p_c and p_d are not in the same cluster as p_a and p_b. Again e is removed. There is no area of the cluster of p_a and p_b, thus, e can not be in the boundary. Fig. 4(d) illustrates this sub-case.

(a) (b)

Fig. 3. The two sub-cases of the simple case with $p_a = 1$, $p_b = 2$ and $p_c = 3$ (only intra-cluster edges are drawn), **(a)** triangle in a cluster, **(b)** no triangle



(a) (b) (c) (d)

Fig. 4. The sub-cases of the complex case. Here $p_a = 1$, $p_b = 2$, $p_c = 3$ and $p_d = 4$ (only intra-cluster edges are shown), **(a)** illustration of the first complex sub-case, two triangles in a cluster, **(b)** and **(c)** are the second sub-case triangle in a cluster, **(d)** no triangle.

After the intra-cluster edge analysis, the output of this phase is a set of lists of oriented edges in no particular order (see Fig. 2(e)). Phase 3 (polygonization) constructs a list of edges such that traversing the list of edges corresponds to navigating along the boundary of a cluster. Because of the orientation of the previous phase, the interior of the polygon will be on the left while the exterior will be on the right. The list will be circular, eventually returning to the same node. A polygon with holes will be made of several of these lists. We now prove that cluster polygonization is always possible from the output of Phase 2.

Lemma 2.2.1 *For every node p_a attached to an oriented edge e in the output of Phase 2, the following holds.*

- The degree of p_a is even.
- The *in*degree of p_a equals the *out*degree of p_a.
- It is possible to alternate the incoming edges and the outgoing edges either clockwise or counterclockwise in the planar embedding representing the output of Phase 2.

Proof. Because we started from a triangulation and the output of Phase 2 is edges who belong to one and exactly one triangle in the cluster, the output of Phase 2 is equivalent to a union of disjoint triangles (the triangles share edges but not their interiors). Note that, when Phase 2 removes intra-cluster edges, it performs the union of two regions that are the union of triangles and thus the new

region is the union of triangles. The point p_a must belong to a triangle in the union because it is attached to an oriented edge e. Let T be the sequence of triangles clockwise around the point p_a. This sequence T of triangles corresponds to a sequence of edges E. These edges are all incident to p_a with e oriented. Without loss of generality, assume e is an incoming edge and that the sequence $E = <e, e_1,\ldots, e_k>$ is the sequence of edges in the triangulation incident to p_a when travelling clockwise around p_a. Because e is incoming, the triangle determined by $<e, p_a, e_1>$ must be exterior. If e_1 is oriented, then it must be outgoing and the triangle $<e_1, p_a, e_2>$ must be interior. If e_1 is not oriented, it is because the next triangle $<e_1, p_a, e_2>$ is exterior. Continuing in this way, we see that the triangles in T alternate between a few that are interior and a few that are exterior. In any case, when the triangles swap from interior to exterior, the edge incident to p_a is incoming and when they swap back the edge must be outgoing. The sequence of triangles around node $p_{a,,}$ however is finite, so when completing the clockwise traversal we see that the lemma is satisfied.

The previous lemma proves that the graph after Phase 2 is Eulerian and it is now a matter of traversing the graph within the planar embedding to extract the circular (cyclic) lists of edges (equivalently nodes) that constitute the polygons.

2.3 Polygonization Requires Linear Time

For a set P of two dimensional points, the number of edges in $DT(P)$ is linear in P (Okabe *et al*., 1999). Thus, storing $DT(P)$ requires linear space. To compute the connected components of a graph is linear in the sum of both number of edges and number of nodes. Thus, Phase 1 requires $O(n)$ time where n is the number of nodes in $DT(P)$. The boundary extraction process in Phase 2 tests if every intra-cluster edge e is a boundary edge. It performs constant work for each edge. Thus, this is linear as well. Therefore, polygonization of clusters requires linear time for Short-Long clustering. However, if other clustering methods do not use the Delaunay Triangulation as an underlying proximity graph, then we need to compute $DT(P)$, which requires $O(n\log n)$ time.

3. Performance Evaluation

We present results from experiments with synthetic datasets and real datasets that illustrate the robustness of our approach. In Short-Long clustering, we use $m = 1$ as the default control value for all datasets. In addition, in all experiments, clusters whose sizes are less than 1% of the total number of points are considered as noise.

(a) (b) (c) (d)

Fig. 5. Synthetic dataset I (||*P*|| = 8000), **(a)** points with *DT*(*P*), **(b)** 6 clusters, (c) cluster boundaries, **(d)** cluster regions

Fig. 6. Synthetic dataset II (||*P*|| = 8000), **(a)** points with *DT*(*P*), **(b)** 8 clusters, **(c)** cluster boundaries, **(d)** cluster regions

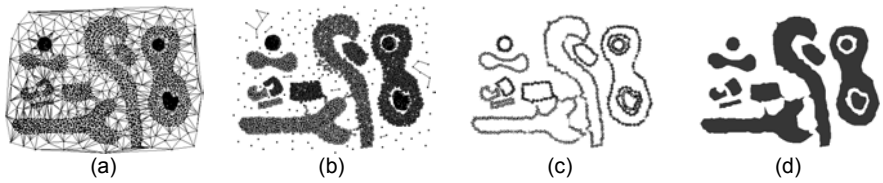


Fig. 7. Synthetic dataset III (||*P*|| = 8000), **(a)** points with *DT*(*P*), **(b)** 12 clusters, **(c)** cluster boundaries, **(d)** cluster regions

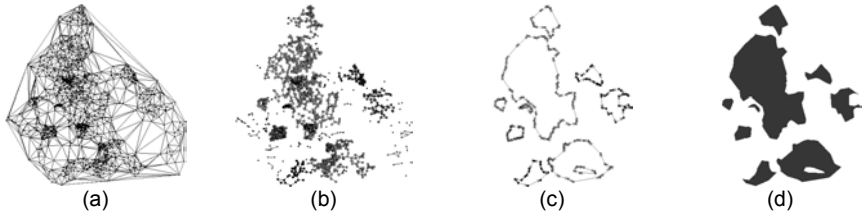


Fig. 8. Real Dataset representing sexual offences (||*P*|| = 1695), **(a)** points with *DT*(*P*), **(b)** 9 clusters, **(c)** cluster boundaries, **(d)** cluster regions

Fig. 5 and Fig. 6 illustrate our approach applied to two datasets of the CHAMELEON's benchmark suite (Karypis *et al*., 1999). Cluster boundaries are successfully derived. Fig. 5(c) depicts boundaries for the first dataset and Fig. 6(c) for the second set, respectively. Boundaries reveal shapes of clusters more easily than human inspection of the dataset or of the clustered data. Cluster regions depicted in Fig. 5(d) and Fig. 6(d) reveal holes within clusters. Note that, visual inspection is insufficient to find holes in the regions of clusters, when presented simply as clusters (Fig. 5(b) and Fig. 6(b)). However, the holes become visually apparent with polygonization (Fig. 5(d) and Fig. 6(d)). Fig. 7 depicts a dataset containing many heterogeneous clusters: small and large clusters, non-convex clusters, clusters with heterogeneous densities, clusters inside clusters and clusters linked by multiple bridges. Cluster boundaries and regions shown in Fig. 7(c) and Fig. 7(d) illustrate the robustness of our approach. Real datasets representing geographical phenomena are more complex than synthetic datasets, thus it is more difficult to extract cluster boundaries. A real dataset shown in Fig. 8 indicates

locations of sexual offences that occurred in 1997 around urban areas of Queensland,Australia. A large cluster is spread around Brisbane, the capital city of Queensland and several crime concentrations are discovered around the suburbs of Brisbane. Although shapes of clusters are heterogeneous, irregular and complex, our approach provides cluster boundaries for crime cluster regions. Altogether, these experimental results demonstrate that our approach effectively approximates shapes of clusters.

4 Applications

4.1 Choropleth Mapping with Cluster Boundaries

One of the difficulties of presenting point data is to manage privacy. Another problem with point data is that theoretically, the probability of an event at a point is zero. To resolve the privacy problem or to make inferences on regions rather than locations, point data are often aggregated based on polygonal maps. For example, one way of presenting discrete point data is to use choropleth maps. These maps display point data related to a specific topic with respect to a boundary map such as political or administrative area map. The display of a choropleth map fills polygons of the reference areas with colours or gray tones according to densities (the number of points per unit of area). One key aspect about choropleth mapping is that its visual presentation is dependent on the base map used for plotting. Dent (1999) warns that distributions of continuous geographical phenomena are not governed by political or administrative subdivisions. Thus, choropleth mapping of points with political or administrative layers is less informative. Using the set of cluster regions of P as a basic area map overcomes the problem of traditional choropleth maps, since boundaries of cluster regions are derived from the distribution of P. Thus, it minimises artificial constraints on choropleth mappings, which is important in exploratory spatial analysis.

(a) (b) (c) (d)

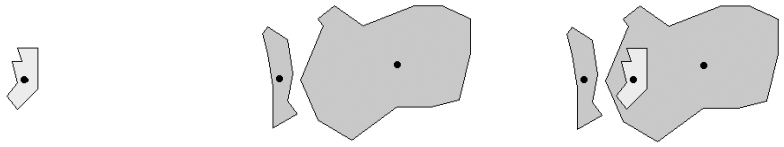
Fig. 9. Choropleth maps with cluster regions, **(a)** synthetic dataset I, **(b)** synthetic dataset II, **(c)** Synthetic Dataset III, **(d)** real dataset

Fig. 9 displays choropleth maps for the datasets discussed in Section 3. Here, the densest cluster is in black (RGB(0,0,0)) and the sparest cluster is in light gray (RGB(200,200,200)). Intermediate clusters are shaded with gray tones in

proportion to their respective densities. Visual inspection of the original datasets (see for example Fig. 5) is insufficient to report relatively dense or sparse clusters. In Fig. 5, one can hardly see any difference in density. The datasets seem too large for such visual inspection. However, the choropleth map in Fig. 9(a) clearly indicates that two clusters on the left-hand side are relatively dense. A pattern that shows density decreasing from left to right is now clearly visible. By contrast, Fig. 9(b) indicates that density among clusters does not have a global pattern. The denser clusters of synthetic dataset II are randomly mixed with other clusters. Since this spread is not easily recognised from Fig. 6, the corresponding choropleth map is more informative. Fig. 9(c) shows that clusters are either high density or low density. Thus, we can easily find three high concentrations. Fig. 9(d) reveals that most densities of clusters are about the same although there are substantial discrepancies in the cluster sizes.

4.2 Cluster Associations

Clustering for data mining is to summarise the distribution of *P* in an effort to suggest a manageable number of patterns of concentrations for further explorations. Thus, clusters are representatives of the phenomena recorded with locations in *P* and suggesting interesting groups. Several approaches (Knorr *et al*., 1997; Estivill-Castro and Murray, 1998) have been proposed to detect associations among geographical layers using clusters in data-rich environments. Boundaries of clusters and representatives of clusters (medoids or means) are the most popular candidates for reasoning with clusters. However, these candidates are summaries. They constitute only partial information about clusters, thus we need special care when we use these information when mining for associations.



(a) (b) (c)

Fig. 10. Cluster associations, **(a)** layer 1, **(b)** layer 2, **(c)** overlay of layer 1 and layer 2

Fig. 10 illustrates the problem of using such partial information. Consider two layers shown in Fig. 10(a) and Fig. 10(b). Layer 1 displayed in Fig. 10(a) has a small cluster while the Layer 2 depicted in Fig. 10(b) has two clusters small and large. Black dots represent medoids of clusters. Although the small cluster in Layer 1 has a high association with the large cluster in Layer 2 (since the small cluster lies within the large cluster), traditional approaches fail to detect this association. Association analysis using boundary matching (Knorr *et al*., 1997) does not succeed in detecting this correlation, since boundaries of the small cluster in Layer 1 do not match with those of the large cluster in Layer 2. This aspect is shown in Fig. 10(c). Similarly, association analysis using medoids (Estivill-Castro

and Murray, 1998) is unable to report this association, since the medoid of the small cluster in Layer 1 is closer to the medoid of the small cluster in Layer 2 than that of the large cluster in Layer 2. We can detect this relationship by computing intersection area of cluster regions. The intuition of this approach is that two clusters exhibit a high association if most phenomena in Layer 1 take place where concentrations of phenomena in Layer 2 occur. The small cluster in Layer 1 intersects with the large cluster in Layer 2, while it does not intersect with the small cluster in Layer 2. Thus, it is more associated with the large cluster in Layer 2 rather than the small one.

5 Final Remarks

Clustering methods are becoming more sophisticated and effective. Post-clustering processes that seek to identify possible lures (positive associations) are now in demand. Detection of cluster boundaries is a natural choice for reasoning about clusters such as matching boundaries with various feature data, polygonizing clusters and mining for associations. We propose an automatic cluster boundary extraction method that is well-suited for data-rich environments. In this approach, shapes of clusters are governed by data, not by users. The automatic approach derives the shape of a cluster not only from the distribution of points in the cluster, but also from that of points in different clusters. Thus, this approach is more informative and less biased since it is data-driven, which is very important in exploratory spatial data analysis (Openshaw, 1987). The cluster boundary extraction has been extended to cluster polygonization that has potential as a tool for spatial analysis and mining. Choropleth mapping with cluster regions as a base map exhibits unbiased visualization. Further, intersecting polygonized cluster regions enables us to find positive associations among layers with ease. As a results a supporting application has been developed. The application is implemented in the C++ programming language using LEDA (Library of Efficient Data types and Algorithms) version 4.2. The application supports visualization of cluster boundaries, polygonized cluster regions and choropleth maps.

References

Dent BD (1999) Cartography: Thematic Map Design. McGraw-Hill, Boston.

Edelsbrunner H, Kirkpatrick D, Seidel R (1983) On the Shape of a Set of Points in the Plane. IEEE Transactions on Information Theory 29(4):551-559

Ester M, Kriegel H-P, Sander J, Xu X (1996) A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Simoudis E, Han J, Fayyad UM (eds) Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining. AAAI Press, pp 226-231

Estivill-Castro V, Lee I (2000) AUTOCLUST: Automatic Clustering via Boundary Extraction for Mining Massive Point-Data Sets. In: Abrahart J, Carlisle BH (eds) Proc. of the 5th Int. Conf. on Geocomputation.

Estivill-Castro V, Murray AT (1998) Discovering Associations in Spatial Data - An Efficient Medoid based Approach. In: Wu X, Ramamohanarao K, Korb KB (eds) Proc. of the 2nd Pacific-Asia Conf. on Knowledge Discovery and Data Mining. LNAI 1394, Springer, pp 110-121

Karypis G, Han E, Kumar V (1999) CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling. IEEE Computer: Special Issue on Data Analysis and Mining 32(8):68-75

Knorr EM, Ng RT, Shilvock DL (1997) Finding Boundary Shape Matching Relationships in Spatial Data. In: Scholl M, Voisard A (eds) Proc. of the 5th Int. Symposium on Spatial Databases. LNCS 1262, Springer, pp 29-46

Miller HJ, Han J (2001) Geographic Data Mining and Knowledge Discovery: An Overview. Taylor & Francis, New York

Ng RT, Han J (1994) Efficient and Effective Clustering Method for Spatial Data Mining. In: Bocca JB, Jarke M, Zaniolo C (eds) Proc. of the 20th Int. Conf. on Very Large Data Bases. Morgan Kaufmann, pp 144-155

Okabe A, Boots BN, Sugihara K, Chiu SN (1999) Spatial Tessellations: Concepts and Applications of Voronoi Diagrams. John Wiley & Sons, West Sussex

Openshaw S (1987) A Mark 1 Geographical Analysis Machine for the automated analysis of point data sets. Int. Journal of GIS 1(4):335-358