Clustering Spatial Data in the Presence of Obstacles: a Density-Based Approach

Osmar R. Zaïane
University of Alberta
Database Laboratory
Edmonton, Canada
zaiane@cs.ualberta.ca

Chi-Hoon Lee
University of Alberta
Database Laboratory
Edmonton, Canada
chihoon@cs.ualberta.ca

Abstract

*Clustering spatial data is a well-known problem that has been extensively studied. Grouping similar data in large 2-dimensional spaces to find hidden patterns or meaningful sub-groups has many applications such as satellite imagery, geographic information systems, medical image analysis, marketing, computer visions, etc. Although many methods have been proposed in the literature, very few have considered physical obstacles that may have significant consequences on the effectiveness of the clustering. Taking into account these constraints during the clustering process is costly and the modeling of the constraints is paramount for good performance. In this paper, we investigate the problem of clustering in the presence of constraints such as physical obstacles and introduce a new approach to model these constraints using polygons. We also propose a strategy to prune the search space and reduce the number of polygons to test during clustering. We devise a density-based clustering algorithm, DBCluC, which takes advantage of our constraint modeling to efficiently cluster data objects while considering all physical constraints. The algorithm can detect clusters of arbitrary shape and is insensitive to noise, the input order, and the difficulty of constraints. Its average running complexity is O(N*log*N) where N is the number of data points.*

1. Introduction

Unsupervised classification of objects into groups such that the similarity of objects in a group is maximized while the similarity between objects of different groups is minimized, is an interesting problem that has attracted the attention of statisticians for many years because of the numerous potential applications. Recently, we are witnessing a resurgence of interest in new clustering techniques in the data mining community, and many effective and efficient methods have been proposed in the machine learning and data mining literature. The rapid increase in digitized spatial data availability has prompted considerable research in what is known as spatial data mining [19]. Clustering analysis for data in a 2-dimensional space is considered spatial data mining and has applications in geographic information systems, pattern recognition, medical imaging, marketing analysis, weather forecasting, etc. Clustering in spatial data has been an active research area and most of the research has focused on effectiveness and scalability. As reported in surveys on data clustering [11, 14] clustering methods can be classified into Partitioning approaches [17, 21, 5, 22, 31], Hierarchical methods [26, 32, 10, 15], Density based algorithms [8, 2, 13], Probabilistic techniques [6], Graph theoretic [31], Fuzzy methods [4, 23], Grid-based algorithms [30, 25, 1], and Model based approaches [24, 18]. As pointed out earlier, these techniques have focused on the performance in terms of effectiveness and efficiency for large databases. However, almost none of them have taken into account constraints that may be present in the data or constraints on the clustering. These constraints have significant influence on the results of the clustering process of large spatial data. In medical imaging, for example, while 2 points could be close together according to a distance measure they should be restrained from being clustered together due to physical or biological constraints. In a GIS application studying the movement of pedestrians to identify optimal bank machine placements, for example, the presence of a highway hinders the movement of pedestrians and should be considered as an obstacle. To the best of our knowledge, only two clustering algorithms for clustering spatial data in the presence of constraints have been proposed very recently: COD-CLARANS [28] based on a partitioning approach, and AUTOCLUST+ [9] based on a graph partitioning approach. [29] introduces the taxonomy of constraints for clustering: Constraints on individual objects; Obstacle objects as constraints; Clustering parameters as constraints; and Constraints imposed on each individual cluster. Its primary discussion has been focused on SQL aggregate and existential constraints. Those constraints have

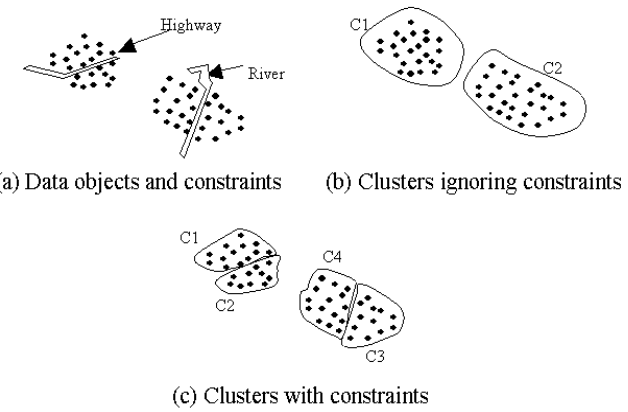


Figure 1. Clustering data objects with constraints

considerable effect on clustering a database, taking into account the capacity of involved resource. COD-CLARANS [28] and AUTOCLUST+[9] propose algorithms to solve the problem of clustering in the presence of physical obstacles to cross such as rivers, mountain ranges, or highways, etc. [28] defines obstacles by building visibility graphs to find the shortest distance among data objects in the presence of obstacles. The graph has edges between data points that are visible to each other and the edge is eliminated if an obstacle obstructs the "visibility". The visibility graph is expensive to build and is considered as pre-processed in [28], thus making the approach look performing and scaling well since the pre-processing is taken out of the performance evaluation. [9] builds a Delaunay structure to cluster data points considering obstacles, a more scalable and efficient data structure. However, it is expensive to construct and is not flexible to combine a different kind of constraints. As an example, Figure 1 shows clustering data objects in relation to their neighbours as well as the physical constraints. Ignoring the constraints leads to incorrect interpretation of the correlation among data points. Figure 1(b) shows two clusters. The correct grouping (Figure 1(c)) visually generates four clusters, four clusters due to highway and river constraints, and one naturally grouped cluster.

The algorithm we propose in a 2-dimensional planar space, DBCluC (Density-Based Clustering with Constraints, pronounced DB-clu-see), is based on DBSCAN [8] a density-based clustering algorithm that clearly outperforms the effectiveness and efficiency of CLARANS [21], the algorithm used for COD-CLARANS. The performance is better not only in terms of time complexity, but also in terms of clustering performance; detection of natural cluster shapes and noise sensitivity.

In this paper, we also introduce a new idea for modeling constraints using simple polygons. Note that there are two classes of polygons with respect to mathematical context, as illustrated in the geometry lieterature [12, 27]: a simple polygon and a crossing polygon (see Obstacle Modeling). Polygons can represent obstacles of arbitrary shapes, lengths, thickness, etc. Given the potential large number of edges representing all the polygons in the space to cluster, we have reduced the size of the search space by introducing the idea of representing polygons with a minimum number of lines that preserve the connectivity and disconnectivity between points in space.

The remainder of the paper is organized as follows: In Section 2, we briefly introduce the notions of reachability and connectivity needed in the expansion process of DBSCAN [8] since the same reachability idea is adopted in our algorithm, and present the motivating concepts significant to this study. In Section 3, we show how we model the constraints, obstacles, and illustrate how the edges of the polygons are reduced to improve performance. The main clustering algorithm that considers constraints during the clustering is introduced with its complexity analysis in Section 4. Section 5 shows the performance of this algorithm and its clustering results. Finally, Section 6 concludes this study with some discussion of future work.

2. Background Concepts

While COD-CLARANS is based on the partitioning approach of CLARANS, which adopts Euclidean distances and considers only clusters with spherical shapes, we have selected the density-based idea behind DBSCAN that expands the neighbourhood of a point based on a fixed minimum number of points reachable within an area of a given radius. Unlike the partitioning approach, the density-based method we adopted does not require the a priori knowledge of the number of clusters in the data. In the following section, we present the important concepts of DBSCAN and define the important notions that motivate our clustering algorithm.

2.1. DBSCAN

DBSCAN is a clustering algorithm with two parameters, *Eps* and *MinPts*, utilizing the density notion that involves correlation between a data point and its neighbours. In order for data points to be grouped, there must be at least a minimum number of points called MinPts in Eps-neighbourhood, N_{Eps} *(p)*, from a data point *p*, given a radius *Eps*. Its intuition has focused on detecting natural clusters among data objects, while discriminating noises (outliers) from clusters. In DBSCAN, the following definitions are denoted.

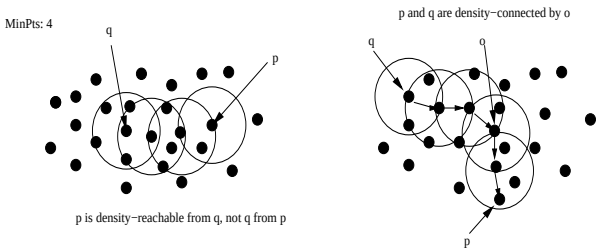


Figure 2. Density-reachable and Density-connected

Definition 1. (Directly density-reachable) A point *p* is directly density-reachable from a point q with respect to *Eps*, *MinPts* if

(1) $p \in N_{Eps}(q)$

(2) $|N_{Eps}(p)| \geq$ MinPts, where $|N_{Eps}(p)|$ denotes the number of points in the circle of radius Eps and centre p.

Definition 1 is important to understand the difference between core points and border points, since border points have the less number of Eps-neighbourhood than core points do.

Definition 2. (Density-reachable) A point p is density-rea- chable from a point q with respect to Eps and MinPts, if there is a chain of points p_1, .., p_n, p_1 = q, , p_n = p such that p_{i+1} is directly density-reachable from p_i.

Definition 3. (Density-connected) A point p is density-connected to a point q with respect to Eps and MinPts, if there is a point o such that both, p and q are density-reachable from o with respect to Eps and MinPts.

Definition 4. (Cluster) Let D be a database of points. A cluster C with respect to Eps and MinPts is a non-empty subset of D satisfying the following conditions:

(1) Maximality: $\forall$ p, q if $p \in C$ and q is density-reachable from p with respect to Eps and MinPts, then $q \in C$.

(2) Connectivity: $\forall$ p, q $\in$ C, p is density-connected to q with respect to Eps and MinPts.

Definition 5. (Noise) Let p $\in$ a database D. p is noise, if $p \notin C_i$, where C_i is i^{th} cluster, $0 \leq i \leq k$, k is the number of clusters in D.

Figure 2 illustrates the above definitions. The detailed figures and discussion are found in [8].

Once the two parameters *Eps* and *MinPts* are defined, DBSCAN starts to group data points from an arbitrary point. If a cluster cannot be expanded with respect to the density reachable and density-connected definitions, it starts grouping data points for another cluster. This procedure is iterated until there is no data point to be expanded and all data points in the dataset clustered or labelled as a noise. One major issue with DBSCAN, as presented in [8], is the problem of high dimensionality of data objects when looking for range queries to quickly identify points in the neighbourhood of another point. In [8] the authors indexed data objects using the R*tree [3], but this structure has a dimensionality limitation of 16 dimensions with respect to efficiency, as reported in [7]. While we are only dealing with 2 dimensions for spatial data, we use SR-Tree structures [16] instead in our implementation of DBCluC, which allows us to do efficient range queries for neighbourhood data points.

2.2. Motivating Concepts

As briefly mentioned in the introduction, the existence of constraints may not only affect the efficiency but also the accuracy of the clustering. This leads us to investigate means to overcome these two problems while considering constraints. In the following, we characterize these physical constraints by introducing some definitions. We assume that obstacles are given in the form of polygons with rigid edges.

Definition 6. (Obstacle) An obstacle is a polygon denoted by P(V, E) where V is a set of k points from an obstacle: V = $\{v_1, v_2, v_3, \ldots, v_k\}$ and E is a set of k line segments: E = $\{e_1, e_2, e_3, \ldots, e_k\}$ where e_i is a line segment joining v_i and v_{i+1} ,$1 \leq i \leq k$, i+1=1 if i+1>k. There are two classes of obstacles: *convex* and *concave*. The distinction is important as we shall see later.

There are two classes of obstacles: *convex* and *concave*. The distinction is important as we shall see later.

Definition 7. (Visibility) Visibility is a relation between two data points, if the line segment drawn from one point to the other is not intersected with a polygon P (V, E) representing a given obstacle. Given a set D of n data points D =$\{d_1, d_2, d_3, \ldots, d_n\}$, a line segment *l* joining d_i and d_j where d_i, $d_j \in$ D, i$\neq$j, i and j $\in$ [1..n], and a line segment $e_k \in$ E, If $\nexists$ a point p that is an intersection point between two line segments *l* and e_k , then d_i is visible to d_j.

Definition 8. (Visible Space) Given a set D of n data points D=$\{d_1, d_2, d_3, \ldots, d_n\}$, A visible space is a set S of k points S=$\{s_1, s_2, s_3, \ldots, s_k\}$ such that $\forall$ s_i,$s_j \in$ S, s_i and s_j are visible to each other, while s_k is not visible to $s'_k in$ S' , where S' is a visible space such that S' $\cap$ S = $\emptyset$, S' and S $\subseteq$ D, i$\neq$j, and i and j $\in$ [1..n].

Before defining the problem of clustering data points in the presence of obstacles, we need to redefine a cluster (Definition 4) that conforms to Definition 7.

Definition 9. (Cluster) Given a set D of n data points D={d_1, d_2, d_3, ..., d_n}, a cluster is a set C of c points C={ c_1, c_2, c_3, ..., c_c }, satisfying the following conditions, where C $\subseteq$ D, i $\neq$ j, and i and j $\in$ [1..n]. Let D be a database of points. A cluster C with respect to Eps and MinPts is a non-empty subset of D satisfying the following conditions:

(1) Maximality: $\forall$ d_i, d_j $\in$ D, if d_i $\in$ C and d_j is density-reachable from d_i with respect to Eps and MinPts, then d_j $\in$ C.

(2) Connectivity.$\forall$ c_i, c_j $\in$ C, c_i is density-connected to c_j with respect to Eps and MinPts.

(3) $\forall$ c_i, c_j $\in$ C, c_i and c_j are visible to each other.

3. Modeling Constraints

We have defined the problem of clustering with constraints in the previous section. Considering the efficiency of the process of clustering, it is essential to model the constraints efficiently to limit the impact of the constraints on the cost-effectiveness of the clustering algorithm. As mentioned in Section 1, we have opted to model the physical constraints with polygons. However, with a large number of obstacles, we would have a large number of edges to test for the division of visibility spaces (Definition 8). We present herein a scheme to model polygons that minimizes the number of edges to take into consideration.

3.1. Obstacle Modeling

Many research areas such as spatial data mining, computational geometry, computer graphics, robot navigations, etc, have considered obstacles as polygons. The performance of the algorithms used is dependent on the size of inputs (i.e. the number of polygon edges). For instance, for finding a shortest path between a starting point and a destination in the presence of obstructions, it is required to evaluate obstacles to minimize a tour distance. Understanding connectivity among given data points is also necessary to evaluate obstacles. [28] and [9] discuss the clustering problem in the presence of obstacles, but [9] does not explicitly explain how to model obstacles but simply uses a Delaunay graph to model the whole data space. Even though [28] models obstacles using a visibility graph, the authors did not present how the types of obstacles are specified, since the classification of an obstacle type improve the searchin cost in [28]. In addition, the visibility graph is assumed to be given and never considered in the complexity analysis or execution time.

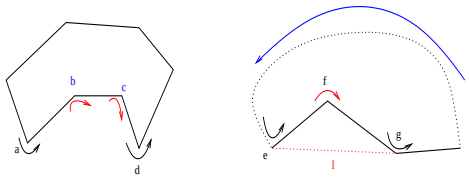
While we model obstacles with polygons, a polygon is represented with a minimum set of line segments, called obstruction lines, such that the definition of visibility (Definition 7) is not compromised. This minimum set of lines is smaller than the number of edges in the polygon. This in turn reduces the input size and enhances the searching task. The obstruction lines in a polygon depend upon the type of polygon: convex or concave. Note that an obstacle creates a certain number of visible spaces along with the number of convex points. Before we discuss the idea to convert a given polygon into a smaller number of line segments (obstruction lines), we need to test if a given polygon is convex or concave. If any point of the polygon is categorized as concave, the polygon is said to be concave. It is convex otherwise. The convexity test is fundamental in composing the obstruction lines. The problem then consists in determining whether polygon points are either concave or convex. The convex point test and the multifaceted process to determine the obstruction lines, representing a given polygon, is given in details in [20]. For the sake of brevity and lack of space, we only present the essential idea herein.

3.1.1. The Convexity test: Turning Directional Approach

The "Turning Directional" approach has been introduced by [27]. The intution of the Turning directioanl approach is to evaluate the convexity of polygons via the definition of a polygon that is mathematically defined. [27] classifies a polygon as either a simple polygon or a crossing polygon. A simple polygon is a polygon such that every edge in the polygon is not intersected with other edges that exist in the polygon. A crossing polygon is a polygon such that there is an edge that is intersected with other edges that exist in the polygon. Therefore, crossing polygons are not determined as convex. Note that this paper considers a simple polygon and a crossing polygon, as the former is a dominant object in spatial applications and the latter is simply dealt with set of simple polygon.

Now we claim that a polygon is a convex polygon, if and only if all points from the polygon make a same directional turn. The claim can be easily proved . Suppose a polygon *P* does not follow the claim. It then is obvious that *P* is not a convex. As desrcribed in Figure 3 (a), points *b* and *c* are concave verteces that make *P* a concave. Let *P* be a convex polygon. Then all possible line segment that join two non-consecutive points from *P* should be interior to *P*. Hence the vertex *f* must be pulled out at least up to the line *l* in order for *P* to be a convex. If the vertex *f* lies on the line *l*, then a convex is composed erasing one vertex. Note that a different shape of a polygon is drawn, if *f* lies over the line *l*.

In order to test a turning direction for 3 consecutive verteces, the sign of the triangle area of 3 points is exam-



(a) A concave (b) A polygon

Figure 3. Turning examples in polygons

ined via a determinant. As a result, the sign of the determninant evaluates the turning direction either a clockwise or a counterclockwise. Note that we assume all points in a polygon are enumerated in an order either clockwise or a counterclockwise. Hence, we can easily identify a type of a polygon as well as a type of each vertex from the polygon in a linear time *O(n)*, where *n* is the number of points in a polygon.

3.1.2. Polygon Reduction

In order to generate obstruction lines for a given polygon, we first need to identify the type of the polygon. convex or concave. Notice that the number of obstruction lines depends on a polygon type. The convex point test on a point from a given polygon is based on testing whether the turning direction of all points is in a same way. Note that the Convex Hull algorithm in the graph theory literature is not applicable in this context. For instance, the polygon on the bottom of Figure 4 could be identified as a concave, whereas the Convex Hull algorithms can not recognize a type of a point. It is critical for Polygon Reduction algorithm to label a type of a point from a polygon. Once we categorize a polygon into a convex or a concave, we find the obstruction lines to replace the initial line segments from the polygon as we make sure the visibility spaces are not divided or merged. It is clear for a convex polygon to have the same number of visible spaces (Definition 8) as the number of its convex points since each line segment from the convex point blocks visibility of its neighbour visible spaces. In contrast, a concave point does not create two visible spaces, but does a visible space that is created by its nearest convex point. Accordingly, a convex point creates two adjacent visible spaces. We observe the fact that two adjacent line segments sharing a convex vertex in a polygon are interchangeable with two line segments such that one of which obstructs visibility in a dimension between two neighbour visible spaces that are created by the convex vertex and the other impedes visibility between two neighbour visible spaces and the rest of visible spaces created by the polygon. As a consequence, the initial polygon is to be represented as a lose-less set of primitive edges with respect to visibility. Now, it is sufficient to discover a set of obstruction lines

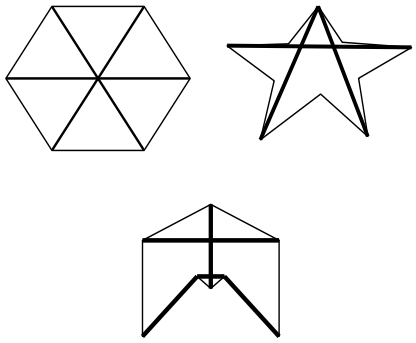


Figure 4. Simple polygons and generated obstruction lines

that impedes visible spaces for every convex vertex from a polygon minimizing the number of obstruction lines. It is obvious that we reduce *n* lines for a convex to $\lceil \frac{n}{2} \rceil$ obstruction lines. Due to a bi-partition method that divides a set of convex points into two sets by an enumeration order, it is straightforward to compose a set of obstruction lines by joining two points from each partition. In addition, the bi-partition method achieves the lose-less reduction of a polygon.

It is not, however, trivial for a concave due to its characteristic, whereas an obstruction edge is easily drawn between two convex vertex in a convex. It is required for a concave to check a possible obstruction line drawn by the bi-paritition method is intersected with a line segment from the concave polygon and is exterior to the polygon, which is a *"non-admissible"* obstruction line. If a possible obstruction line is non-admissible to a given concave polygon, then it is to be replaced with a set of obstruction lines that might be line segments from the concave or are interior to and not intersected with the concave polygon, which is a set of *"admissible"* obstruction lines. In order to construct a set of obstruction edges from a concave, as a possible obstruction edge candidate is not admissible, we employ a modified single-source shortest path algorithm [20] converting concave *P(V, E)* into a weighted graph whose edges are all possible obstruction lines for each point in the concave and whose weight is the distance between a source and a destination. The distance on a path $\overrightarrow{vw}$ between two points *v* and *w* is not *Euclidean*, but the number of points on the path $\overrightarrow{vw}$. Note the source and the destination vertex are one of two end points from possible obstruction edge candidates. The detailed study is found in [20]. These measures should ensure that the generated number of obstruction lines is less than the original number of line segments from a polygon.

4. Algorithm

Once we have modeled obstacles using the polygon reduction algorithm, DBCluC starts the clustering procedure from an arbitrary data point. Hence it is not sensitive to an order of the data input. The clustering procedure in DBCluC is quite similar to DBSCAN [8]. As illustrated in [8], the distance between clusters C_1 and C_2 is defined as a minimum distance between data objects in C_1 and C_2, respectively. Normally defined clusters that are not satisfied with Definition 9 or whose distance between clusters is larger than *Eps* are isolated.

A database is a set of data points to be clustered in Algorithm 1. Line 1 initiates the clustering procedure. In the course of clustering, Line 4 assigns a new cluster id for the next expandable cluster.

The ExpandCluster in Algorithm 2 may seem similar to the function in the DBSCAN. However, the distinction is that obstacles are considered in RetrieveNeighbours (Point, Eps, Obstacles) illustrated by Algorithm 3. Given a query point, neighbours of the query point are retrieved using SR-tree. In DBCluC, we have adopted the range neighbour query approach instead of the nearest neighbour query approach from SR-tree, since it is extremely difficult for the latter to expand a set of clusters if a density of data objects is high. Its average run time of a neighbour query is O(logN) where N is the number of data objects. Notice that the range search in SR-tree is very expensive, especially when the density is very high with a large database.

Once retrieving neighbours of a query point, it is trivial to evaluate visibilities between a query point and its neighbours. The visibility between two data objects in the presence of obstacles is computed using a line segment whose endpoints are the two data objects in question. If any line segment representing an obstacle is intersected with this line, then the two data points are not grouped together, since they are not visible to each other according to Definition 7. Those accepted neighbours defined as the SEED that are retrieved by RetrieveNeighbours of Algorithm 2 continue to expand a cluster from elements of the SEED, if the number of elements in the SEED is not less than MinPts. A data object is labeled by a proper cluster id, if retrieved neighbours are satisfied with the parameter *MinPts* discriminating outliers. Note that line 15 in Algorithm 2 does exclude a noise from being an element of the SEED in order to enhance query efficiency.

The "RESULT" in Algorithm 3 is a set of data objects that are neighbours of a given query objects. The elements in the RESULT is collected and the obstacles are evaluated by Alogorithm 3. The RESULT elements are constructed by removing data objects that are not visible each other because of the blockage of obstacles. This is performed by line 3 in Algorihtm 3. Notice that line 1 in Algorihtm 3 retrieves neighbours of a given query point using SR tree [16].

```
   Input    : Database and Obstalces
   Output   : A set of clusters
1  // Start clustering;
2  for Point ∈ Databse do
3  |  if ExpandCluster(Database,Point,ClusterId,Eps,MinPts,Obstacles)
   |  then
4  |  |  ClusterId = nextId(ClusterId);
   |  endif
   endfor
```

Algorithm 1: DBCluC

4.1. Complexity

As discussed before, the polygon reduction algorithm models obstacles by classifying an obstacle into convex or concave. Let n be the number of points of a polygon p, and n_{cc} and n_{cv} are the number of concave points and the number of convex points respectively with $n = n_{cc} + n_{cv}$. The convexity test for p requires $O(n)$. The polygon reduction algorithm requires a weighted graph [20] to replace a non-adimissible obstruction line segment with a set of admissible line segments. The complexity in the replacement is in worst case $O(n\log n)$ with an indexing scheme to search line segments. It in turn creates a set with E number of edges including a set of line segments that lies in P. Notice that the number of E is less than $\alpha * n$, where $\alpha \ll n$. The polygon reduction algorithm for P requires $O(n\log n + n_{cv}\, I\, E)$ in the worst case where I is the number of non-admissible line segments to be replaced, while the lower bound is $\Omega(n)$. Hence, the upper bound of the polygon reduction algorithm is depicted by $O(n_{cv}\ I\ n)$. When we evaluate a set of polygons and n_{cc} and I that are on average far smaller than n, the complexity of the polygon reduction is on average in the order of $O(n)$. Polygon Reduction Algorithm is a pre-processing phase that precedes the clustering. The complexity of the clustering algorithm alone, is in the order of $O(N \cdot logN \cdot L)$, where L is the number of obstruction lines generated by the polygon reduction algorithm, and N is the number of points in the database. The complexity can, however, be reduced to $O(N \cdot \log N)$, if we adopt an indexing method for obstacles. Currently, to check the visibility between two data points, all obstruction lines are tested. We could reduce the number of obstruction lines to be checked by evaluating only lines that traverse the point neighbourhood (see conclusions and future work).

```
Input  : Database, a data point Point, ClusterId,
         Eps, MinPts, and Obstacles
Output : True or False
SEED = RetrieveNeighbours(Point, Eps, Obstacles);
if size of seed is less than MinPts then
    Classify Point as NOISE;
    Return False;
endif
change clustered of all elements in SEED into ClusterId;
delete Point from SEED;
while SEED.SIZE < 0 do
    CurrentPoint = SEED.first();
    RESULT = RetrieveNeighbours(CurrentPoint,
    Eps, Obstacles);
    if RESULT.SIZE ≥ MinPts then
        for element ∈ RESULT do
            if element is UNCLASSFIED then
                put it into SEED;
                set its cluster id to ClusterId;
            endif
            if element is NOISE then
                set its cluster id to ClusterId;
            endif
        endfor
    endif
    delete CurrentPoint from SEED;
endw
Return True;
```

Algorithm 2: ExpandCluster

```
Input  : a data object Point, Eps, and Obstalces
Output : A set of data points
RESULT = getNeighbour(Point, Eps);
for element RESULT do
    if CheckVisibility_with(element, Obstacles) then
        RESULT.delete(element);
    endif
endfor
Return RESULT;
```

Algorithm 3: RetrieveNeighbours(Point, Eps, Obstacles)

5. Performance

In this section we evaluate the performance of the algorithm in terms of effectiveness and scalability. Although COD-CLARANS and AUTOCLUST+ discuss the clustering problem in the presence of obstacles, it is hard to compare quantitatively the performance with their respective approaches due to the difference in the datasets used. To realistically compare the algorithms, we ought to use the exact datasets with the same constraints. However, we did not have access to these datasets, even if they are synthetic datasets, we can not regenerated them . Yet it is known that density- based clustering algorithms such as DBSCAN [8] outperforms partitioning algorithms such as CLARANS [28] in terms of efficiency and effectivenss. It can be concluded that DBCluC would outperform COD-CLARANS with respect to the scalability and clustering quality. We have evaluated DBCluC by generating datasets with complex cluster shapes and by varying the size of data as well as the number and complexity of the physical constraints.

For the purpose of the experiments, we have generated synthetic datasets. We report three of them herein Dataset1, Dataset2, and Dataset3. Obstacles such as rivers, lakes, and highways are also simulated in these datasets. Dataset1 containing 434 data points with four obstacles is for illustration purposes. Figure 5 shows the 16 polygon line segments reduced to 8 obstruction lines. Since Dataset1 is sparse, it is primarily grouped into one cluster. Adding obstacles creates four distinct clusters (Figure 5(c)). Figures 6 and 7 illustrate the effectiveness of DBCluC in the presence of obstacles. For the convenience of comparison of clustering results, Figure 6 and 7 illustrate sequentially data points, and obstacles, before clustering (a); clustering results in the absence of constraints (b); clusters in the presence of obstacles (c). The red lines from obstacles in all datasets are the obstruction lines to replace initial polygons that are drawn in blue. Dataset2 has about 1063 data points with 4 obstacles. There are visually 6 clusters ignoring obstacles, as shown in Figure6 (b). Dataset2 represents the primary intuition of the problem we have investigated in this paper. The correct clustering shows 8 groups of data points. Dataset3 has 11775 data points with 6 obstacles that consist of 29 line segments. The initial 29 line segments from simulated obstacles are replaced with 15 obstruction lines.

We also conducted experiments varying the size of the dataset and the number of obstacles to demonstrate scalability of DBCluC. Figure 8 represents the execution time in seconds for eight datasets varying in size from 25K to 200K with an increment of 25K data points. The figure shows a good scalability. The execution time is almost linear to the number of data objects. Figure 9 presents the execution time in seconds for clustering 40K data objects but by varying the number of obstacles. The numbers in the X-axis represent

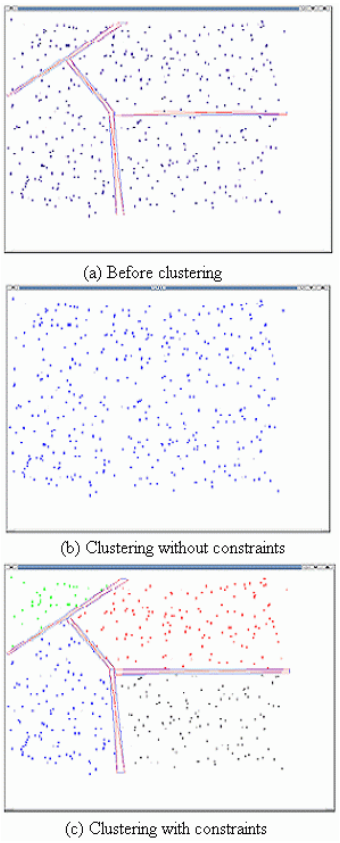


Figure 5. Clustering dataset Dataset1

the total number of polygon edges and the respective obstruction lines. Notice that our polygon reduction algorithm manages to reduce the number of lines by approximately half each time. The differential in the increase of the polygon edges is not constant. However, in the proportion of the increase in the polygons, the execution time is almost linear. Thus DBCluC is scalable for large databases with complicated obstacles in terms of size of the database and in terms of the number of constraints.

6. Conclusions

In this paper we have addressed the problem of clustering spatial data in the presence of physical constraints. The constraints we considered are not only obstacles such as rivers, highways, mountain ranges, etc. We have proposed a model for these constraints using polygons and have devised a method for reducing the edges of polygons representing obstacles by identifying a minimum set of line segments, called obstruction lines, that does not compromise the visibility spaces. The polygon reduction algorithm reduces the number of lines representing a polygon by half,

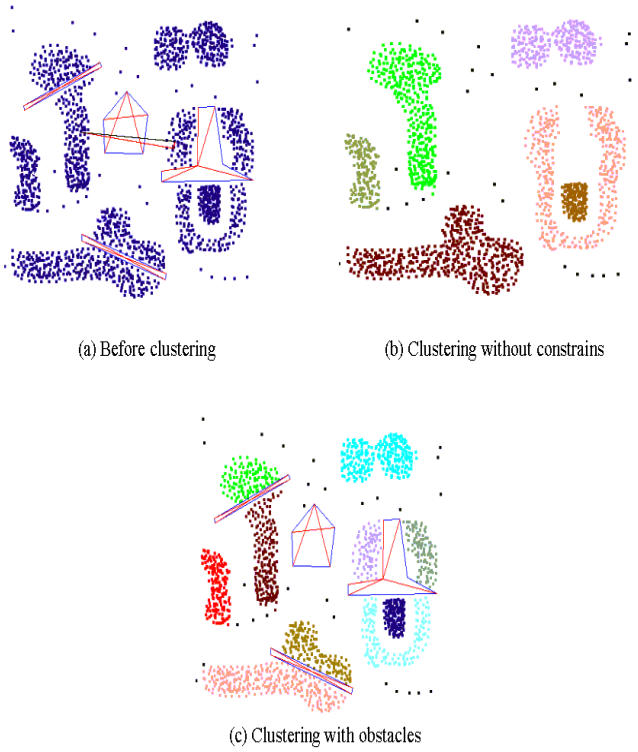


Figure 6. Clustering dataset Dataset2

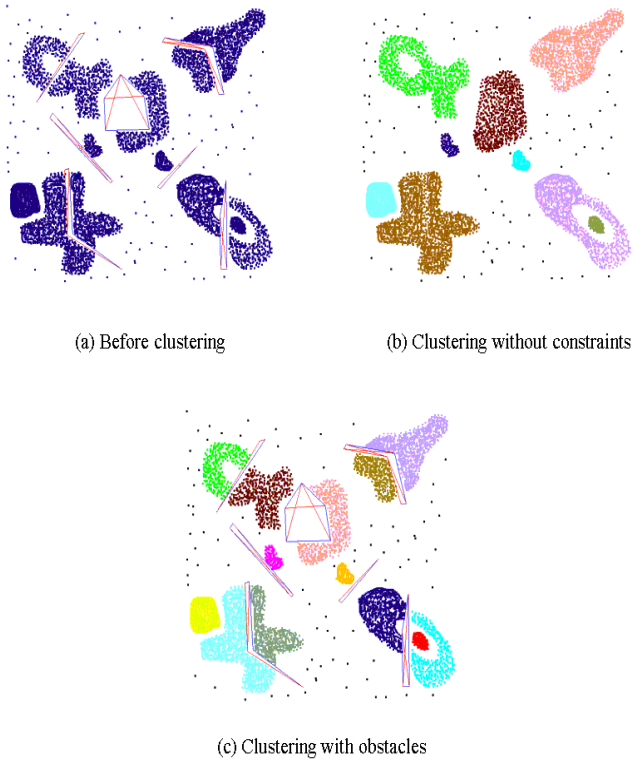


Figure 7. Clustering dataset Dataset3

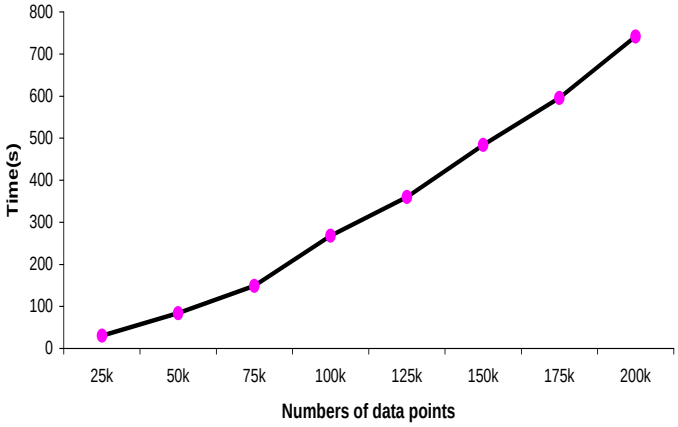


Figure 8. Algorithm Run Time by varying the number of data points

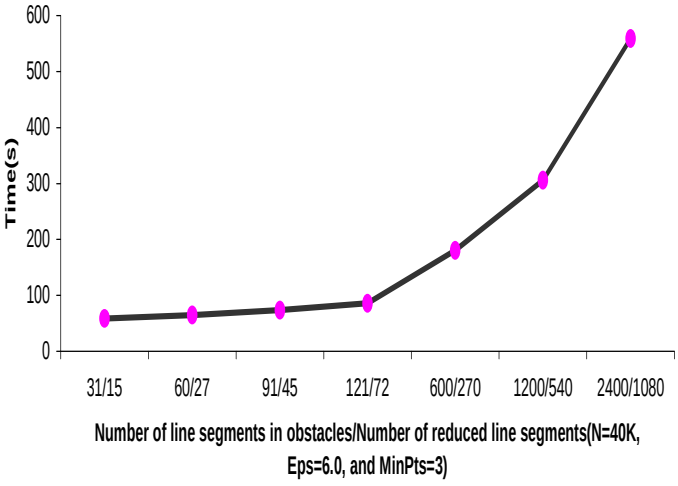


Figure 9. Algorithm Run Time by varying the number obstacles

and thus reduces the search space by half. We have also defined the concept of reachability in the context of obstacles and have used it in the designation of the clustering process. Finally, we have developed a density-based clustering algorithm, DBCluC, which takes constraints into account during the clustering process. Owing to the effectiveness of the density-based approach, DBCluC finds clusters of arbitrary shapes and sizes with minimum domain knowledge. In particular, it is not necessary to know the number of clusters to be discovered. In addition, experiments have shown scalability of DBCluC in terms of size of the database in number of data points as well as scalability in terms of number and complexity of physical constraints.

In the current implementation of DBCluC, obstacles are not indexed. This obliges a check of all obstruction lines before expanding the reachability of any point. While the number of line segments to test is reduced significantly thanks to the polygon reduction algorithm, this number can still be reduced with a better indexing of the obstruction lines. Indeed, it suffices to test only the lines traversing the neighbourhood of a data point to expand. However, since lines are represented by their end-points, and end-points of a close line can be relatively far, it is difficult to issue a range query for such lines. With a good indexing scheme of the obstruction lines, the complexity of the clustering algorithm can be reduced to $O(N \cdot \log N)$. Moreover, most of the execution time in the current implementation is spent in retrieving neighbours with range queries in the SR-tree structure indexing data points. SR-trees perform well for k- NN type of queries instead. For spatial databases, with an index structure optimized for range queries of 2- dimensional data objects, the run time of DBCluC could be dramatically improved.

We have addressed the problem of clustering in the presence of constraints such as physical obstacles but only in a 2-dimensional space. While the SR-tree structure allows us to cluster spaces of higher dimensionality, our model for constraints has not been tested at higher dimensionality and our polygon reduction algorithm is limited to two-dimensional plans. We believe that some consideration should be given to modeling constraints when clustering high dimensional spaces. We are currently investigating other constraints such as bridges and pedways that can invalidate obstacles such as rivers and highways at some given points or simply connect distant clusters. Moreover, operational constraints, not considered in this paper, have a key role with respect to the effectiveness of clustering results, even though they require expensive processing.

References

[1] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for

data mining applications. In *Proc. 1998 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'98)*, pages 94–105, 1998.

[2] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: ordering points to identify the clustering structure. In *ACM-SIGMOD Int. Conf. Management of Data (SIGMOD' 99)*, pages 49–60, 1999.

[3] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R* tree: An efficient and robust access method for points and rectangles. In *Proc. of the 1990 ACM SIGMOD Intl. Conf.*, pages 332–331, 1990.

[4] J. Bezdek and R. Hathaway. Numerical convergence and interpretation of the fuzzy c-shells clustering algorithm. *IEEE Transactions on Neural Networks*, 3(5):787–793, 1992.

[5] P. S. Bradley, U. M. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Knowledge Discovery and Data Mining*, pages 9–15, 1998.

[6] V. Brailovsky. A probabilistic approach to clustering. *Pattern Recognition Letters*, 12(4):193–198, 1991.

[7] N. Colossi and M. Nascimento. Benchmarking access structures for high-dimensional multimedia data. In *Proc. IEEE Intl. Conf. on Multimedia and Expo (ICME'2000)*, pages 1215–1218, 2000.

[8] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[9] V. Estivill-Castro and I. Lee. Autoclust+: Automatic clustering of point-data sets in the presence of obstacles. In *International Workshop on Temporal and Spatial and Spatio-Temporal Data Mining (TSDM2000)*, pages 133–146, 2000.

[10] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *ACM-SIGMOD International Conference Management of Data*, pages 73–84, 1998.

[11] J. Han and M. KamberK. *Data Mining: Concepts and Techniques*. Morgan Kaufman, 2000.

[12] P. S. Heckbert. *Graphics Gems(4)*. Academic Press, 1994.

[13] A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65, 1998.

[14] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[15] G. Karypis, E. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. In *IEEE Computer*, pages 68–75, 1999.

[16] N. Katayama and S. Satoh. The SR-tree: an index structure for high-dimensional nearest neighbor queries. In *Proc. of the 1997 ACM SIGMOD Intl. Conf.*, pages 369–380, 1997.

[17] L. Kaufman. Finding groups in data: an introduction to cluster analysis. In *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley, New York, 1990.

[18] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[19] K. Koperski, J. Adhikary, and J. Han. Spatial data mining: Progress and challenges survey paper, 1996.

[20] C.-H. Lee and O. R. Zaïane. Polygon reduction: An algorithm for minimum line representation for polygons. In *Submitted to 14th Canadian Conf. on Computational Geometry*, 2002.

[21] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. of VLDB Conf.*, pages 144–155, 1994.

[22] H. Ralambondrainy. A conceptual version of the k-means algorithm. *Pattern Recognition Letters*, 16(11):1147–1157, 1995.

[23] E. H. Ruspini. A new approach to clustering. *Information and Control*, 15(1):22–32, 1969.

[24] J. W. Shavlik and T. G. Dietterich. *Readings in Machine Learning*. Morgan Kaufmann, 1990.

[25] G. Sheikholeslami, S. Chatterjee, and A. Zhang. Wavecluster: A multi-resolution clustering approach for very large spatial databases, 1998.

[26] P. Sneath and R. Sokal. Numerical taxonomy, 1973.

[27] M. Stone. A mnemonic for areas of polygons. *AMER. MATH. MONTHLY*, 93:479–480, 1986.

[28] A. K. H. Tung, J. Hou, and J. Han. Spatial clustering in the presence of obstacles. In *Proc. 2001 Int. Conf. On Data Engineering(ICDE'01)*, 2001.

[29] A. K. H. Tung, R. T. Ng, L. V. S. Lakshmanan, and J. Han. Constraint-based clustering in large databases. In *ICDT*, pages 405–419, 2001.

[30] W. Wang, J. Yang, and R. R. Muntz. STING: A statistical information grid approach to spatial data mining. In *The VLDB Journal*, pages 186–195, 1997.

[31] C. Zahn. Graph-theoretical methods for detecting and describing gestalt clusters. In *IEEE Transactions on Computers*, pages 20:68–86, 1971.

[32] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM-SIGMOD International Conference Management of Data*, pages 103–114, June 1996.