A Distribution-Based Clustering Algorithm for Mining in Large Spatial Databases

Xiaowei Xu, Martin Ester, Hans-Peter Kriegel, Jörg Sander
University of Munich
Oettingenstr. 67
D-80538 München

Abstract

The problem of detecting clusters of points belonging to a spatial point process arises in many applications. In this paper, we introduce the new clustering algorithm DBCLASD (Distribution Based Clustering of LArge Spatial Databases) to discover clusters of this type. The results of experiments demonstrate that DBCLASD, contrary to partitioning algorithms such as CLARANS, discovers clusters of arbitrary shape. Furthermore, DBCLASD does not require any input parameters, in contrast to the clustering algorithm DBSCAN requiring two input parameters which may be difficult to provide for large databases. In terms of efficiency, DBCLASD is between CLARANS and DBSCAN, close to DBSCAN. Thus, the efficiency of DBCLASD on large spatial databases is very attractive when considering its nonparametric nature and its good quality for clusters of arbitrary shape.

1. Introduction

Increasingly large amounts of data obtained from satellite images, X-ray crystallography or other automatic equipment are stored in databases. Therefore, automated knowledge discovery becomes more and more important in databases. *Data mining* is a step in the process of knowledge discovery consisting of applying data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data [8]. Clustering, i.e. the grouping of objects of a database into meaningful subclasses, is one of the prominent data mining tasks.

Spatial Database Systems (SDBS) [9] are database systems for the management of spatial data such as points and polygons representing a part of the surface of the earth. In this paper, we consider the task of clustering in spatial databases, in particular the problem of detecting clusters of points which are distributed as a homogeneous Poisson point process restricted to a certain part of the space. This type of distribution is also called uniform distribution or random distribution. The clusters may have arbitrary shape. This problem arises in many applications, e.g. seismology (grouping earthquakes clustered along seismic faults), minefield detection (grouping mines in a minefield) and astronomy (grouping stars in galaxies) (see [1], [4] and [12]).

The application to large spatial databases raises the following requirements for clustering algorithms:

(1) Minimal number of input parameters, because appropriate values are often not known in advance for many applications. In the application of detecting minefields, e.g., we usually do not know the shape, the density and the number of clusters.

(2) Discovery of clusters with arbitrary shape, because the shape of clusters in spatial databases may be spherical, drawn-out, elongated etc. In geographical information systems the clusters, e.g., houses whose price falls within a given range, are usually shaped by the neighboring geographical objects such as rivers or parks.

(3) Good efficiency on large databases, i.e. on databases of significantly more than just a few thousand objects.

None of the well-known clustering algorithms fulfills the combination of these requirements. In this paper, we present the new clustering algorithm DBCLASD which is based on the assumption that the points inside a cluster are uniformly distributed. This is quite reasonable for many applications. The application of DBCLASD to earthquake catalogues shows that DBCLASD also works effectively on real databases where the data is not exactly uniformly distributed (see section 5.5). DBCLASD dynamically determines the appropriate number and shape of clusters for a database without requiring any input parameters. Finally, the new algorithm is efficient even for large databases.

The rest of the paper is organized as follows. Clustering algorithms have been developed and applied in different areas of computer science, and we discuss related work in section 2. In section 3, we present our notion of clusters which is based on the distribution of the distance to the nearest neighbors. Section 4 introduces the algorithm DBCLASD which discovers such clusters in a spatial database. In section 5, we report on an experimental evaluation of DBCLASD according to our major three requirements. A comparison with the well-known clustering algorithms CLARANS and DBSCAN is presented. Furthermore, we apply DBCLASD to a real database demonstrating its applicability to real world problems. Section 6 concludes with a summary and some directions for future research.

2. Related Work

One can distinguish two main types of clustering algorithms [10]: partitioning and hierarchical algorithms. *Partitioning algorithms* construct a partition of a database *D* of *n* objects into a set of *k* clusters. The partitioning algorithms typically start with an initial partition of *D* and then uses an iterative control strategy to optimize an objective function. Each cluster is represented by the gravity center of the cluster *(k-means algorithms*) or by one of the objects of the cluster located near its center (*k-medoid algorithms*). Consequently, partitioning algorithms use a two-step procedure. First, determine *k* representatives minimizing the objective function. Second, assign each object to the cluster with its representative "closest" to the considered object. The second step implies that a partition is equivalent to a voronoi diagram and each cluster is contained in one of the voronoi cells.

Ng and Han [13] explore partitioning algorithms for KDD in spatial databases. An algorithm called CLARANS (Clustering Large Applications based on RANdomized Search) is introduced which is an improved *k*-medoid method. Compared to former *k*-medoid algorithms, CLARANS is more effective and more efficient.

Hierarchical algorithms create a hierarchical decomposition of *D*. The hierarchical decomposition is represented by a *dendrogram*, a tree that iteratively splits *D* into smaller subsets until each subset consists of only one object. In such a hierarchy, each level of the tree represents a clustering of *D*. In contrast to partitioning algorithms, hierarchical algorithms do not need *k* as an input parameter. However, a *termination condition* has to be defined indicating when the merge or division process should be terminated.

In recent years, it has been found that probability model based cluster analysis can be useful in discovering clusters from noisy data (see [3] and [4]). The clusters and noise are represented by a mixture model. Hierarchical clustering then partitions the points between the clusters and noise. Both methods, however, are somewhat restricted when applied for class identification in spatial databases: [3] assumes the clusters to have a specific shape and [4] removes noise only from the database which restricts it to be a preprocessing step for further cluster analysis.

The run time of most of the above algorithms is too inefficient on large databases. Therefore, some focusing techniques have been proposed to increase the efficiency of clustering algorithms: [7] presents an R*-tree [2] based focusing technique (1) creating a sample of the database that is drawn from each R*-tree data page and (2) applying the clustering algorithm only to that sample. BIRCH [14] is a CF-tree, a hierarchical data structure designed for clustering, based multiphase clustering method. First, the database is scanned to build an initial in-memory CF-tree. Second, an arbitrary clustering algorithm is used to cluster the leaf nodes of the CF-tree. Both, the R*-tree based focusing technique and the CF-tree based technique perform basically a preprocessing for clustering and can be used for any clustering algorithm.

DBSCAN (Density Based Spatial Clustering of Applications with Noise) [6] relies on a density-based notion of clusters which is designed to discover clusters of arbitrary shape in spatial databases with noise. A cluster is defined as a maximal set of density-connected points, i.e. the Eps-neighborhood of every point in a cluster contains at least MinPts many points.

3. A Notion of Clusters Based on the Distance Distribution

Consider the problem of detecting surface-laid minefields on the basis of an image from a reconnaissance aircraft. After processing, such an image is reduced to a set of points, some of which may be mines, and some of which may be noise, such as other metal objects or rocks. The aim of the analysis is to determine whether or not minefields are present, and where they are. A typical minefield database is shown in figure 1. Since actual minefields data were not made available to the public, the minefields data in this paper were simulated according to specifications developed at the Naval Coastal System Station, Panama City, Florida, to represent minefield data encountered in practice [12].

Visually, two minefields can be easily discovered in figure 1 as clusters. We observe that the distances to the nearest neighbors for points inside the region of cluster 1 are typically smaller than the distances to the nearest neighbors for points outside, which are still in the neighborhood of cluster 1. The same holds for cluster 2. We conjecture that each cluster has a characteristic probability distribution of the distance to the nearest neighbors. If this characteristic distribution of a cluster has been discovered, it can be used to decide whether a neighboring point should be accepted as a member of the cluster or not.

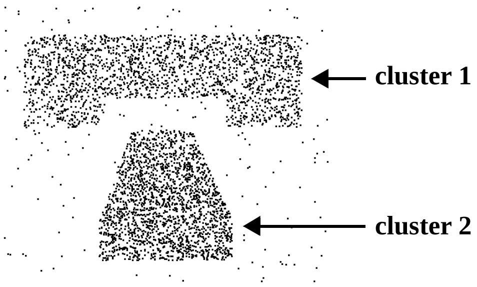


figure 1: Sample minefield database

3.1 Some Definitions

In the following, we introduce some basic definitions for our notion of clusters.

Definition 3.1 (nearest neighbor of a point and nearest neighbor distance) Let *q* be a query point and *S* be a set of points. Then the *nearest neighbor of q* in *S*, denoted by $NN_S(q)$, *is* a point *p* in $S - \{q\}$ which has the minimum distance to *q*. The distance from *q* to its nearest neighbor in *S* is called the *nearest neighbor distance of q*, $NNdist_S(q)$ for short.

Definition 3.2 (nearest neighbor distance set of a set of points) Let *S* be a set of points and e_i be the elements of *S*. The *nearest neighbor distance set of S,* denoted by $NNdistSet(S)$, or *distance set* for short, is the multi-set of all values $NNdist_S(e_i)$.

3.2 The Statistic Model for our Cluster Definition

In the following, we analyze the probability distribution of the nearest neighbor distances of a cluster. This analysis is based on the assumption that the points inside of a cluster are uniformly distributed, i.e. the points of a cluster are distributed as a homogeneous Poisson point process restricted to a certain part of the data space. However, all points of the database need not be uniformly distributed. This assumption seems to be reasonable for many applications, e.g. for the detection of minefields using reconnaissance aircraft images and for the detection of geological faults from an earthquake record (see [4], [11] and [12]).

We want to determine the probability distribution of the nearest neighbor distances. Let the *N* points be uniformly distributed over a data space R with volume $Vol(R)$. We can imagine that these *N* points "fall" independently into the data space R, such that the probability of one of these *N* points falling into a subspace S of R with volume $Vol(S)$ is equal to $Vol(S)/Vol(R)$. The probability that the nearest neighbor distance *D* from any query point *q* to its nearest neighbor in the space R is greater than some *x* is therefore equal to the probability that none of the *N* points is located inside of a hypersphere around *q* with radius *x*, denoted by $SP(q, x)$:

$$P(D > x) = (1 - Vol(SP(q, x))/Vol(R))^N$$

Consequently, the probability that *D* is not greater than *x* is:

$$\begin{aligned} P(D \le x) &= 1 - P(D > x) \\ &= 1 - (1 - Vol(SP(q, x))/Vol(R))^N \end{aligned}$$

In 2-dimensional space, the distribution function is therefore:

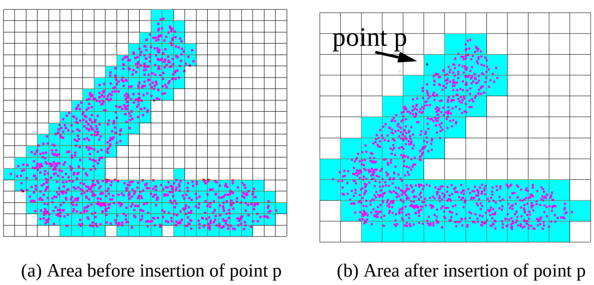
$$\begin{aligned} F(x) &= P(D \le x) \\ &= 1 - (1 - \pi x^2/Vol(R))^N \qquad \text{(i)} \end{aligned}$$

From (i), we know that the distribution function has two parameters *N* and $Vol(R)$. While it is straightforward to determine *N*, it is not obvious how to calculate $Vol(R)$ of a point set which may have arbitrary shape.

3.3 The Computation of the Area of a Cluster

Strictly speaking, there is no area of a set of points *S*. Therefore, we assign some *approximation* to the set *S* and calculate the area of that approximation. First, the shape of the approximation should be as similar as possible to the intuitive shape of the cluster. Second, the approximation should be connected, i.e. it should be one polygon which may contain holes.

We use a grid based approach for determining the approximating polygons. The key problem is the choice of a value for the grid length which is appropriate for some given set of points. If the chosen grid length is too large, the shape of the cluster is poorly approximated. On the other hand, if the chosen grid length is too small, the approximation may be split into several disconnected polygons. Figure 2 illustrates the influence of the grid length on the area of the approximation. We define the grid length such that the insertion of a distant point *p* into the cluster yields a large increase of the area of the cluster implying that the distance distribution in this area no longer fits the expected distance distribution. We set the grid length for a set of points S as the maximum element of *NNDistSet(S).* An occupied grid cell is a grid cell containing at least one of the



(a) Area before insertion of point p (b) Area after insertion of point p

figure 2: The influence of the grid length on the area of the approximation

points of the set and we define the approximation of set S as the union of all occupied grid cells. Based on the calculation of the area of a cluster, figure 3 compares the expected and the observed distance distributions for cluster 1 from figure 1.

Figure 3 shows a good fit between the expected and the observed distance distribution. Formally, we use the concept of the χ^2–test [5] to derive that the observed distance distribution fits the expected distance distribution.

To conclude the above discussion, we state our definition of a cluster based on the distribution of the nearest neighbor distance set as follows:

Definition 3.3 (cluster) Let *DB* be a set of points. A *cluster C* is a non-empty subset of *DB* with the following properties:

(1) *NNDistSet(C)* has the expected distribution with a required confidence level.

(2) *C* is *maximal*, i.e. each extension of C by neighboring points does not fulfill condition (1). (maximality).

(3) *C* is *connected*, i.e. for each pair of points (a,b) of the cluster there is a path of occupied grid cells connecting a and b (connectivity).

4. The Algorithm DBCLASD

Having defined our notion of a cluster, we now design an algorithm to efficiently discover clusters of this type. We call this algorithm *Distribution Based Clustering of LArge Spatial Databases (DBCLASD)*. DBCLASD is an incremental algorithm, i.e. the assignment of a point to a cluster is based only on the points processed so far without considering the whole cluster or even the whole database. DBCLASD incrementally augments an initial cluster by its neighboring points as long as the nearest neighbor distance set of the resulting cluster still fits the expected distance distribution.

A *candidate* is a point not yet belonging to the current cluster which has to be checked for possible membership in this cluster. The generation of the candidates is discussed in section 4.1. The procedure of testing the candidates is the subject of section 4.2. While the incremental approach is crucial for the efficiency of DBCLASD on large databases, it implies an inherent dependency of the discovered clustering from the order of generating and testing candidates from the database. DBCLASD, however, incorporates two important features minimizing this dependency which are outlined in section 4.2. To conclude this section, the complete algorithm of DBCLASD is presented in section 4.3.

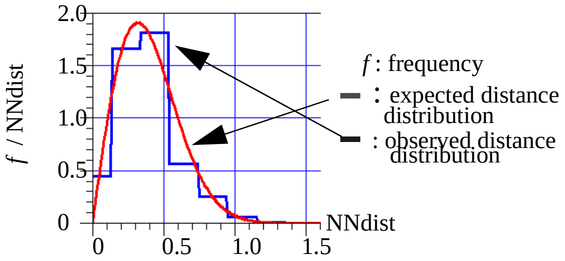


figure 3: The expected and the observed distance distributions for cluster 1 from figure 1

4.1 Generating Candidates

The set of candidates of a cluster is constructed using region queries, which can be efficiently supported using spatial access methods (*SAM*), e.g. R*-tree. A *region query* returns all objects of the database intersecting the specified query region, e.g. a circle. For each new member *p* of the current cluster *C*, we retrieve new candidates using a circle query with a suitable radius *m*. This radius is chosen such that for no point of the cluster a larger distance to the nearest neighbor is to be expected. A larger *m* would lead to more candidates for the χ^2–test and decrease the efficiency of the algorithm. The calculation of *m*, again, is based on the model of uniformly distributed points inside of a cluster *C*. Let *A* be the area of *C* and *N* be the number of its elements. A necessary condition for *m* is:

$$N \times P(NNdist_C(p) > m) < 1$$

and applying formula (i) from section 3.2 we obtains

$$(1 - \pi m^2 / A)^N < 1 / N$$

Consequently, we require

$$m > \sqrt{A / \pi \cdot (1 - 1 / N^{1/N})} \qquad \text{(ii)}$$

When inserting a new point *p* into cluster *C*, a circle query with center *p* and radius *m* is performed and the resulting points are considered as new candidates.

4.2 Testing Candidates

The incremental approach of DBCLASD implies an inherent dependency of the discovered clustering from the order of generating and testing candidates. While the distance set of the whole cluster might fit the expected distance distribution, this does not necessarily hold for all subsets of this cluster. Thus, the order of testing the candidates is crucial. Candidates which are not accepted by the test when considered the first time are called *unsuccessful candidates*. To minimize the dependency on the order of testing, DBCLASD incorporates two important features:

(1) Unsuccessful candidates are not discarded but tried again later.

(2) Points already assigned to some cluster may switch to another cluster later.

In the following, we discuss these features in detail. Unsuccessful candidates are not discarded but stored. When all candidates of the current cluster have been processed, the unsuccessful candidates of that cluster are considered again. In many cases, they will now fit the distance distribution of the augmented cluster. Figure 4, e.g., depicts the history of the χ^2 values obtained for each of the candidates to be checked for sample cluster 1 from figure 1. For several of the candidates, a χ^2value is obtained which is significantly higher than the threshold value implying that this candidate is not assigned to the cluster. When performing the χ^2 test for the distance set of the whole cluster, however, a χ^2 value significantly smaller than the threshold is obtained indicating that the distance set of the cluster indeed fits the expected distance distribution.

Even if none of the unsuccessful candidates separately passes the test, it may be possible that some larger subset of the set of unsuccessful candidates fits the distance distribution of the current cluster. So far, such subsets would yield a cluster of their own conflicting with our definition 4 of a cluster (maximality). To avoid such erroneous splits of clusters, DBCLASD tries to merge neighboring clusters whenever a candidate is generated already assigned to some other cluster. During the generation of candidates DBCLASD does not check whether a candidate already has been assigned to some other cluster or not. While this approach yields an elegant way of merging clusters, clearly it is computationally expensive. It is possible that some point may be switched to a different cluster several times before it is assigned to its final cluster. On the average case, however, the number of reassignments of some given point tends to be reasonably small. The algorithm terminates because of the following properties. First, each point of the database is chosen at most once as a starting point for the generation of a new cluster. Second, the generation of a single cluster terminates because if no more candidates exist the unsuccessful candidates are not considered again if none of them fits the current cluster.

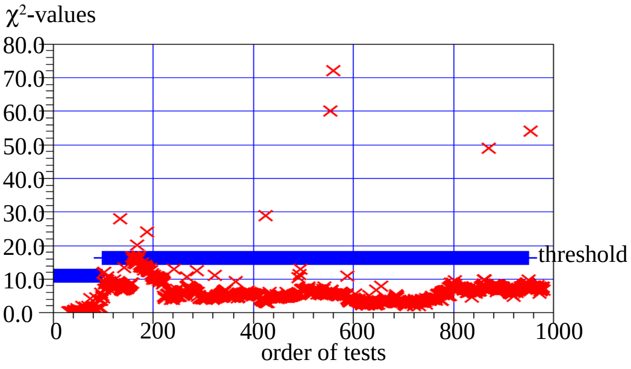


figure 4: History of the χ^2-values for cluster 1 from figure 1

The test of a candidate is performed as follows. First, the current cluster is augmented by the candidate. Then, we use the χ^2–test to verify the hypothesis that the nearest neighbor distance set of the augmented cluster still fits the expected distance distribution.

4.3 The Algorithm

In the following, we present the algorithm DBCLASD in a pseudo code notation. Major functions are written in italic font and are explained below. Also, important details are numbered and commented below.

```
procedure DBCLASD (database db)
  initialize the points of db as being assigned to no cluster;
  initialize an empty list of candidates;
  initialize an empty list of unsuccessful candidates;
  initialize an empty set of processed points;
  for each point p of the database db do
    if p has not yet been assigned to some cluster then
      create a new cluster C and insert p into C;
      reinitialize all data structures for cluster C;
(1)   expand cluster C by 29 neighboring points;
      for each point p1 of the cluster C do
(2)     answers := retrieve_neighborhood(C,p1);
(3)     update_candidates(C,answers);
      end for each point p1 of the cluster C;
      expand_cluster (C);
    end if p has not yet been assigned to some cluster;
  end for each point of the database;
```

(1) The χ^2–test can only be applied to clusters with a minimum size of 30. Therefore, the current cluster is expanded to the size of 30 using k nearest neighbor queries without applying the χ^2–test.

(2) The list of answers is sorted in ascending order of the distances to point *p1*.

(3) Each element of the list of answers not yet processed is inserted into the list of candidates.

```
procedure expand_cluster (cluster C)
  change := TRUE;
  while change do
    change := FALSE;
    while the candidate list is not empty do
      remove the first point p from the candidate list
        and assign it to cluster C;
      if distance set of C still has the expected distribution then
        answers:= retrieve_neighborhood(C,p);
        update_candidates(C,answers);
        change := TRUE;
      else
        remove p from the cluster C;
        insert p into the list of unsuccessful candidates;
      end if distance set still has the expected distribution;
    end while the candidate list is not empty;
    list of candidates := list of unsuccessful candidates;
  end while change;
```

```
listOfPoints **procedure** retrieve_neighborhood(cluster C;point p);
    calculate the radius m according to (ii) in section 4.1;
    **return** result of circle query with center p and radius m;

**procedure** update_candidates(cluster C; listOfPoints points);
    **for** each point in points **do**
        **if** point is **not** in the set of processed points **then**
            insert point at the tail of the list of candidates;
            insert point into the set of processed points;
        **end if** point is not in the set of processed points;
    **end for** each point in points;
```

5. Experimental Evaluation

We evaluate DBCLASD according to the three major requirements for clustering algorithms on large spatial databases as stated in section 1. We compare DBCLASD with the clustering algorithms CLARANS and DBSCAN in terms of effectivity and efficiency. Furthermore, we apply DBCLASD to a real database from an earthquake catalog. The evaluation is based on an implementation of DBCLASD in C++ using the R*-tree. All experiments were run on HP 735 workstations.

5.1 Choice of Comparison Partners

In the following, we explain our choice of algorithms for comparison with DBCLASD. So far, most clustering algorithms have been designed for relatively small data sets. Recently, some new algorithms have been proposed with the goal of applicability to large spatial databases. To our best knowledge, CLARANS is the first clustering algorithm developed for spatial databases. DBSCAN is designed to discover clusters of arbitrary shape in spatial databases with noise. BIRCH is a CF-tree based multi-phase clustering method. Both, the R*-tree based focusing technique and the CF-tree based technique perform basically a preprocessing for clustering and can be used for any clustering algorithm including DBCLASD. Therefore, we compare the performance of DBCLASD with CLARANS and DBSCAN without any preprocessing.

5.2 Discovery of Clusters With Arbitrary Shape

Clusters in spatial databases may be of arbitrary shape, e.g. spherical, drawn-out, linear, elongated etc. Furthermore, the databases may contain noise.

We will use visualization to evaluate the quality of the clusterings obtained by the different algorithms. In order to create readable visualizations without using color, in these experiments we used small databases. Due to space limitation, we only present the results from one typical database which was generated as follows: (1) draw three polygons of different shape (one of them with a hole) for three clusters. (2) generate 1000, 500, and 500 uniformly distributed points in each polygon respectively. Then, inserting 400 noise points (about 17% of the database) into the database which is depicted in figure 5.

The clustering result of DBCLASD on this database is shown in Figure 5. Different clusters are depicted using different symbols and noise is represented using crosses.

This result shows that DBCLASD assigns nearly all points to the “correct” clusters, i.e. as they were generated. Only very few points close to the borders of a cluster are assigned to “wrong” clusters. The clustering result of DBSCAN is rather similar to the results of DBCLASD and therefore it is not depicted.

CLARANS splits up existing clusters and even merges parts of different clusters. Thus, CLARANS, like other partitioning algorithms, cannot be used to detect clusters of arbitrary shapes.

5.3 Number of Input Parameters

It is very difficult for a user to determine suitable values for the input parameters of clustering methods when the database is large. In the following, we discuss these problems for CLARANS and DBSCAN.

Ng and Han discusses methods to determine the “natural” number k_{nat} of clusters in a database. It proposes to run CLARANS once for all possible k_{nat}, from 2 to n, where n is the total number of objects in the database. For each of the discovered clustering the *silhouette coefficient* which is a numerical value indicating the quality of the clustering, is calculated, and finally, the clustering with the maximum silhouette coefficient is chosen as the “natural” clustering. Unfortunately, the run time of this approach is prohibitive for large n, because it implies $n - 1$ calls of CLARANS. Furthermore, if the shape of the clusters is not convex, then there exists no k_{nat} for CLARANS.

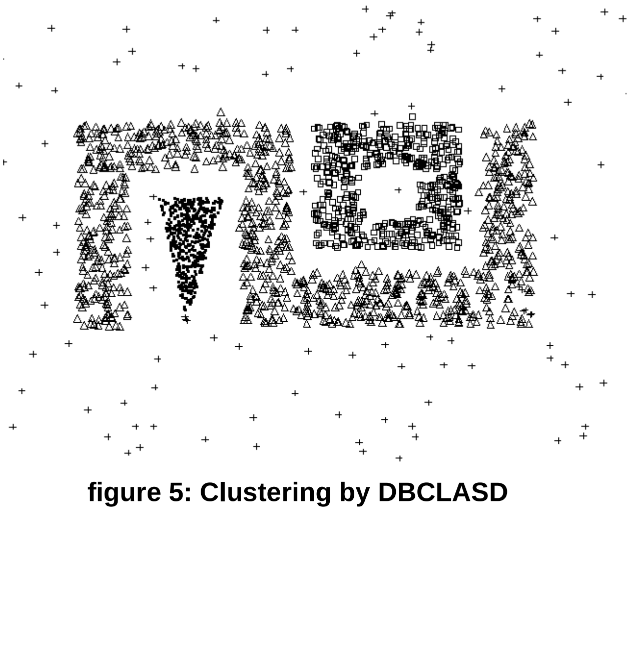


figure 5: Clustering by DBCLASD

DBSCAN requires two parameters. A simple heuristic is also proposed for DBSCAN which is effective in many cases to determine the parameters Eps and MinPts of the “thinnest” cluster in the database. The heuristic helps the user to manually choose Eps and MinPts through the visualization of the *k-dist graph*, i.e. the distance to the k-th nearest neighbor for the points in the database. However, this heuristic has also following drawbacks. First, a k-nearest neighbor query is required for each point which is inefficient for very large databases. Second, the visualization of k-dist graphs is limited to relatively small databases due to the limited resolution of the screen. This limitation can be overcome by restricting the k-dist graph to a sample of the database but then the accuracy of the parameter estimation may decrease significantly.

To conclude, providing correct parameter values for clustering algorithms is a problem for which no general solution exists. DBCLASD, however, detects clusters of arbitrary shape without requiring any input parameters.

5.4 Efficiency

In the following, we compare DBCLASD with CLARANS and DBSCAN with respect to efficiency on synthetic databases. Since the runtime of CLARANS is very large, we had to restrict this comparison to relatively small test databases with the number of points ranging from 5,000 to 25,000. The run times for DBCLASD, DBSCAN and CLARANS on these test databases are listed in table 1.

Table 1: Run Time Comparison (sec)

Number of Points	5000	10000	15000	20000	25000
DBCLASD	77	159	229	348	457
DBSCAN	51	83	134	183	223
CLARANS	5031	22666	64714	138555	250790

The run time of DBCLASD is roughly twice the run time of DBSCAN, while DBCLASD outperforms CLARANS by a factor of at least 60.

We generated seven large synthetic test databases with 5000, 25000, 100000, 200000, 300000, 400000 and 500000 points to test the efficiency and scalability of DBCLASD and DBSCAN wrt. increasing number of points. The run times are plotted in figure 6 which shows that both, DBSCAN and DBCLASD, have a good scalability w.r.t. the size of the databases. The run time of DBCLASD is roughly three times the run time of DBSCAN.

5.5 Application to an Earthquake Database

We consider the problem of detecting seismic faults based on an earthquake catalog. The idea is that earthquake epicenters occur along seismically active faults, and are measured with some error. So, over time, observed earthquake epicenters should be clustered along such faults. We considered an earthquake catalog recorded over a 40,000 km^2 region of the central coast ranges in California from 1962 to 1981 [11]. The left part of figure 7 shows the locations of the earthquakes recorded in the database.

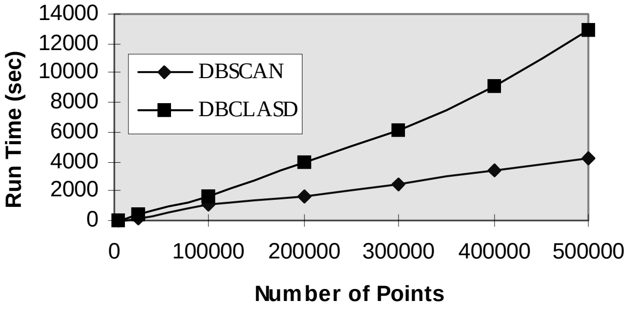


figure 6: Scalability wrt. Increasing Number of points

The right part of figure 7 shows the clustering obtained by DBCLASD. The algorithm detects 4 clusters corresponding to the 2 main seismic faults without requiring any input parameters. This example illustrates that DBCLASD also works effectively on real databases where the data are not as strictly uniformly distributed as the synthetic data.

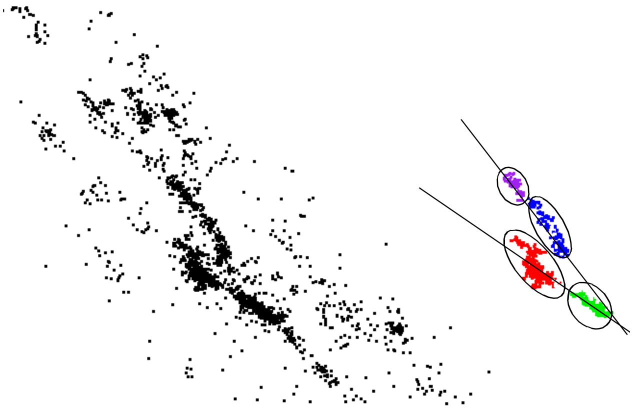


figure 7: California earthquake database

6. Conclusions

The application of clustering algorithms to large spatial databases raises the following requirements: (1) minimal number of input parameters, (2) discovery of clusters with arbitrary shape and (3) efficiency on large databases. The well-known clustering algorithms offer no solution to the combination of these requirements.

In this paper, we introduce the new clustering algorithm DBCLASD which is designed to fulfil the combination of these requirements. Our notion of a cluster is based on the distance of the points of a cluster to their nearest neighbors and we analyze the expected distribution of these distances for a cluster. The analysis in this paper is based on the assumption that the points inside of a cluster are uniformly distributed. This assumption is quite reasonable for many applications. DBCLASD incrementally augments an initial cluster by its neighboring points as long as the nearest neighbor distance set of the resulting cluster still fits the expected distribution. The retrieval of neighboring points is based on region queries, which are efficiently supported by spatial access methods such as R*-trees.

Experiments on both synthetic and real data demonstrate that DBCLASD, contrary to partitioning clustering algorithms such as CLARANS, discovers clusters of arbitrary density, shape, and size. Furthermore, DBCLASD discovers the intuitive clusters without requiring any input parameters, in contrast to the algorithm DBSCAN which needs two input parameters. In terms of efficiency, DBCLASD is between CLARANS and DBSCAN, close to DBSCAN. Thus, the efficiency of DBCLASD on large spatial databases is very attractive when considering its nonparametric nature and its good quality for clusters of arbitrary shape. Furthermore, we applied DBCLASD to a real database. The result indicates that DBCLASD also works effectively on real databases where the data is not exactly uniformly distributed.

Future research will have to consider the following issues. The analysis in this paper is based on the assumption that the points inside of a cluster are uniformly distributed. Other distributions of the points should be investigated. Furthermore, we will consider the case of unknown distributions and explore the use of algorithms from the area of stochastic approximation methods in order to estimate the unknown distribution.

The application of DBCLASD to high-dimensional feature spaces (e.g. in CAD databases or in multi-media databases) seems to be especially useful because in such spaces it is nearly impossible to manually provide appropriate values for the input parameters required by the well-known clustering algorithms. Such applications will be investigated in the future.

Acknowledgments

We thank Abhijit Dasgupta, University of Washington, for providing both, the earthquake data and the specification of the minefield data.

References

[1] Allard D. and Fraley C.:*”Non Parametric Maximum Likelihood Estimation of Features in Spatial Point Process Using Voronoi Tessellation”*, Journal of the American Statistical Association, to appear in December 1997. [Available at http://www.stat.washington.edu/tech.reports/tr293R.ps].

[2] Beckmann N., Kriegel H.-P., Schneider R., Seeger B.: *‘The R*-tree: An Efficient and Robust Access Method for Points and Rectangles’*, Proc. ACM SIGMOD Int. Conf. on Management of Data, Atlantic City, NJ, 1990, pp. 322-331.

[3] Banfield J. D. and Raftery A. E.: “*Model based Gaussian and non-Gaussian clustering*”, Biometrics 49, September 1993, pp. 803-821.

[4] Byers S. and Raftery A. E.: “*Nearest Neighbor Clutter Removal for Estimating Features in Spatial Point Processes*”, Technical Report No. 305, Department of Statistics, University of Washington. [Available at http://www.stat.washington.edu/tech.reports/tr295.ps].

[5] Devore J. L.: ‘*Probability and Statistics for Engineering and the Sciences*’, Duxbury Press, 1991.

[6] Ester M., Kriegel H.-P., Sander J., Xu X.: *“A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”*, Proc. 2cnd Int. Conf. on Knowledge Discovery and Data Mining, Portland, Oregon, 1996, AAAI Press, 1996.

[7] Ester M., Kriegel H.-P., Xu X.: “*Knowledge Discovery in Large Spatial Databases: Focusing Techniques for Efficient Class Identification*”, Proc. 4th Int. Symp. on Large Spatial Databases, Portland, ME, 1995, in: Lecture Notes in Computer Science, Vol. 951, Springer, 1995, pp.67-82.

[8] Fayyad U. M.,.J., Piatetsky-Shapiro G., Smyth P.: *“From Data Mining to Knowledge Discovery: An Overview”*, in: Advances in Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, 1996, pp. 1 - 34.

[9] Gueting R. H.: *“An Introduction to Spatial Database Systems”*, in: The VLDB Journal, Vol. 3, No. 4, October 1994, pp.357-399.

[10] Kaufman L., Rousseeuw P. J.: *“Finding Groups in Data: An Introduction to Cluster Analysis”*, John Wiley & Sons, 1990.

[11] McKenzie M., Miller R., and Uhrhammer R.: “*Bulletin of the Seismographic Stations*”, University of California, Berkeley. Vol. 53, No. 1-2.

[12] Muise R. and Smith C.: “*Nonparametric minefield detection and localization*”, Technical Report CSS-TM-591-91, Naval Surface Warfare Center, Coastal Systems Station.

[13] Ng R. T., Han J.: “*Efficient and Effective Clustering Methods for Spatial Data Mining”*, Proc. 20th Int. Conf. on Very Large Data Bases, Santiago, Chile, 1994, pp. 144-155.

[14] Zhang T., Ramakrishnan R., Linvy M.: *“BIRCH: An Efficient Data Clustering Method for Very Large Databases”*, Proc. ACM SIGMOD Int. Conf. on Management of Data, pp. 103-114.