
Data Mining and Knowledge Discovery 2, 169–194 (1998)

Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications

JÖRG SANDER sander@informatik.uni-muenchen.de
MARTIN ESTER ester@informatik.uni-muenchen.de
HANS-PETER KRIEGEL kriegel@informatik.uni-muenchen.de
XIAOWEI XU xwxu@informatik.uni-muenchen.de
Institute for Computer Science, University of Munich, Oettingenstr. 67, D-80538 München, Germany

Editor: Usama Fayyad

Received February 21, 1997; Revised December 1997 and March 1998; Accepted April 1998

Abstract. The clustering algorithm DBSCAN relies on a density-based notion of clusters and is designed to discover clusters of arbitrary shape as well as to distinguish noise. In this paper, we generalize this algorithm in two important directions. The generalized algorithm—called GDBSCAN—can cluster point objects as well as spatially extended objects according to both, their spatial and their nonspatial attributes. In addition, four applications using 2D points (astronomy), 3D points (biology), 5D points (earth science) and 2D polygons (geography) are presented, demonstrating the applicability of GDBSCAN to real-world problems.

Keywords: clustering algorithms, spatial databases, efficiency, applications

1. Introduction

Spatial Database Systems (SDBS) (Gueting 1994) are database systems for the management of spatial data, i.e., point objects or spatially extended objects in a 2D or 3D space or in some high-dimensional vector space. While a lot of research has been conducted on knowledge discovery in relational databases in the last years, only a few methods for knowledge discovery in spatial databases have been proposed in the literature. Knowledge discovery becomes more and more important in spatial databases since increasingly large amount of data obtained from satellite images, X-ray crystallography or other automatic equipment are stored in spatial databases.

Data mining is a step in the KDD process consisting of the application of data analysis and discovery algorithms that, under acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data (Fayyad et al., 1996). Clustering, i.e., grouping the objects of a database into meaningful subclasses, is one of the major data mining methods (Matheus et al., 1993). There has been a lot of research on clustering algorithms for decades but the application to large spatial databases introduces the following new requirements:

1. Minimal requirements of domain knowledge to determine the input parameters, because appropriate values are often not known in advance when dealing with large databases.

2. Discovery of clusters with arbitrary shape, because the shape of clusters in spatial databases may be nonconvex, spherical, drawn-out, linear, elongated etc.
3. Good efficiency on large databases, i.e., on databases of significantly more than just a few thousand objects.

Ester et al. (1996) present the density-based clustering algorithm DBSCAN. For each point of a cluster its *Eps*-neighborhood for some given $Eps > 0$ has to contain at least a minimum number of points, i.e., the "density" in the *Eps*-neighborhood of points has to exceed some threshold. DBSCAN meets the above requirements in the following sense: first, DBSCAN requires only one input parameter and supports the user in determining an appropriate value for it. Second, it discovers clusters of arbitrary shape and can distinguish noise, and third, using spatial access methods, DBSCAN is efficient even for large spatial databases.

In this paper, we present the algorithm GDBSCAN generalizing DBSCAN in two important ways. First, we can use any notion of a neighborhood of an object if the definition of the neighborhood is based on a binary predicate which is symmetric and reflexive. For example, when clustering polygons, the neighborhood may be defined by the intersect predicate. Second, instead of simply counting the objects in the neighborhood of an object, we can use other measures, e.g., considering the nonspatial attributes such as the average income of a city, to define the "cardinality" of that neighborhood. Thus, the generalized GDBSCAN algorithm can cluster point objects as well as spatially extended objects according to both, their spatial and their nonspatial attributes. Furthermore, we present four applications using 2D points (astronomy), 3D points (biology), 5D points (earth science) and 2D polygons (geography) demonstrating the applicability of GDBSCAN to real-world problems.

The rest of the paper is organized as follows. We discuss well-known clustering algorithms in Section 2 evaluating them according to the above requirements. In Section 3, we present our density-based notion of clusters and Section 4 introduces the algorithm GDBSCAN which discovers such clusters in a spatial database. In Section 5, we present an analytical as well as an experimental evaluation of the effectiveness and efficiency of GDBSCAN. Furthermore, a comparison with the well-known clustering algorithms CLARANS and BIRCH is performed. In Section 6, four applications of GDBSCAN are discussed and Section 7 concludes with a summary and some directions for future research.

2. Related work on clustering algorithms

Two main types of clustering algorithms can be distinguished (Kaufman and Rousseeuw, 1990): 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 use an iterative control strategy to optimize an objective function. Each cluster is represented by the gravity center of the cluster (*k-means algorithms*) (MacQueen, 1967) or by one of the objects of the cluster located near its center (*k-medoid algorithms*) (Vinod, 1969). Consequently, a partition is equivalent to a voronoi diagram and each cluster is contained in one of the voronoi polygons. Thus, the shape of all clusters found by a partitioning algorithm is convex (Kaufman and Rousseeuw, 1990), which is very restrictive for many applications.

Ng and Han (1994) explore partitioning algorithms for mining in spatial databases. An algorithm called CLARANS (Clustering Large Applications based on RANdomized Search) is introduced which is an improved k-medoid method restricting the huge search space using two additional user-supplied parameters. Compared to former k-medoid algorithms, CLARANS is more effective and efficient. Our experimental evaluation indicates that the runtime of a single call of CLARANS is close to quadratic (see Table 1, Section 5). Consequently, it is possible to run CLARANS efficiently on databases of some thousands of objects, but not for really large n. Methods to determine the "natural" number k_{nat} of clusters in a database are also discussed in Ng and Han (1994). They propose to run CLARANS once for each k from 2 to n. For each of the discovered clusterings the silhouette coefficient (Kaufman and Rousseeuw, 1990) is calculated, and finally, the clustering with the maximum silhouette coefficient is chosen as the "natural" clustering. Obviously, this approach is very expensive for large databases, because it implies $O(n)$ calls of CLARANS.

Hierarchical algorithms create a hierarchical decomposition of a database 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. The dendrogram can either be created from the leaves up to the root (*agglomerative approach*) or from the root down to the leaves (*divisive approach*) by merging or dividing clusters at each step. 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, e.g., the critical distance D_{min} between all the clusters of D. Alternatively, an appropriate level in the dendrogram has to be selected manually after the creation of the whole dendrogram.

The *single-link method* is a commonly used agglomerative hierarchical clustering method. Different algorithms for the single-link method have been suggested (e.g., Sibson, 1973; Jain and Dubes, 1988; Hattori and Torii, 1993). Here we have only described the basic idea. The single-link method starts with the disjoint clustering obtained by placing every object in a unique cluster. In every step the two closest clusters in the current clustering are merged until all points are in one cluster. The runtime of algorithms which construct the single-link hierarchy depends on the technique for retrieving nearest neighbors. Without any spatial index support (see Section 5.1 for a brief introduction into spatial access methods) for nearest neighbor queries, the runtime complexity of single-link algorithms is $O(n^2)$. This runtime can be significantly improved when using multidimensional hash- or tree-based index structures (see Murtagh, 1983).

Unfortunately, the runtime of most of the above algorithms is very high on large databases. Therefore, some focusing techniques have been proposed to increase the efficiency of clustering algorithms: (Ester et al., 1995) presents an R*-tree-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.

In Zhang et al. (1997), compact descriptions of subclusters, i.e., Clustering Features (CF), are incrementally computed containing the number of points, the linear sum and the square sum of all points in the cluster. The CF-values are organized in a balanced tree. In the first phase, BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) performs a

linear scan of all data points and builds a *CF-tree*, i.e., a balanced tree with branching factor B and threshold T. A nonleaf node represents a cluster consisting of all the subclusters represented by its entries. A leaf node has to contain at most L entries and the diameter of each entry in a leaf node has to be less than T. A point is inserted by inserting the corresponding CF-value into the closest leaf of the tree. If an entry in the leaf can absorb the new point without violating the threshold condition, then the CF-values for this entry are updated, otherwise a new entry in the leaf node is created. In an optional phase 2, the CF-tree can be further reduced until a desired number of leaf nodes is reached. In phase 3, an arbitrary clustering algorithm, e.g., CLARANS, is used to cluster the CF-values of the leaf nodes of the CF-tree.

The efficiency of BIRCH is similar to the R*-tree-based focusing technique cited above. Experiments with synthetic data sets show that the clustering quality using BIRCH in combination with CLARANS is even higher than the quality obtained by using CLARANS alone.

3. Density-connected sets

In Section 3.1, we present "density-connected sets" which are a significant generalization of "density-based clusters" (see Ester et al., 1996), and indicate some important specializations of density-connected sets in Section 3.2 illustrating the high expressiveness of this concept.

In the following, we assume a spatial database D to be a finite set of objects characterized by spatial and nonspatial attributes. The spatial attributes may represent, e.g., points or spatially extended objects such as polygons in some d-dimensional space S. The nonspatial attributes of an object in D may represent additional properties of a spatial object, e.g., the unemployment rate for a community represented by a polygon in a geographic information system.

3.1. A generalized definition of density-based clusters

The key idea of a density-based cluster is that for each point of a cluster its *Eps*-neighborhood for some given $Eps > 0$ has to contain at least a minimum number of points, i.e., the "density" in the *Eps*-neighborhood of points has to exceed some threshold (Ester et al., 1996). This idea is illustrated by the sample sets of points depicted in figure 1. In these examples, we can easily and unambiguously detect clusters of points and noise points not belonging to any of those clusters, mainly because we have a typical density of points inside the clusters which is considerably higher than outside of the clusters. Furthermore, the density within the areas of noise is lower than the density in any of the clusters.

This idea of "density-based clusters" can be generalized in two important ways. First, we can use any notion of a neighborhood instead of an *Eps*-neighborhood if the definition of the neighborhood is based on a binary predicate which is symmetric and reflexive. Second, instead of simply counting the objects in a neighborhood of an object we can as well use other measures to define the "cardinality" of that neighborhood.

Figure 1. Sample databases.

Definition 1. (*neighborhood* of an object) Let *NPred* be a binary predicate on D which is reflexive and symmetric, i.e., for all $p, q \in D$: *NPred*(p, p) and, if *NPred*(p, q) then *NPred*(q, p). Then the *NPred*-neighborhood of an object $o \in D$ is defined as $N_{NPred}(o) = \{o' \in D \mid NPred(o, o')\}$.

The definition of a cluster in Ester et al. (1996) is restricted to the special case of a distance-based neighborhood, i.e., $N_{Eps}(o) = \{o' \in D \mid |o - o'| \leq Eps\}$. A distance-based neighborhood is a natural notion of a neighborhood for point objects, but if clustering spatially extended objects such as a set of polygons of largely differing sizes it may be more appropriate to use neighborhood predicates like *intersects* or *meets* for finding clusters of polygons.

Although in many applications the neighborhood predicate is defined by using only spatial properties of the objects, the formalism is in no way restricted to purely spatial neighborhoods. We can as well use nonspatial attributes and combine them with spatial properties of objects to derive a neighborhood predicate (see Section 6.4).

Another way to take into account the nonspatial attributes of objects is as a kind of "weight" when calculating the "cardinality" of the neighborhood of an object. To keep things as simple as possible, let us not introduce a weight function operating on objects, but a *weighted cardinality* function *wCard* for sets of objects. The "weight" of a single object o can then be expressed by the weighted cardinality of the singleton containing o, i.e., *wCard*$(\{o\})$.

Definition 2. (*MinWeight* of a set of objects) Let *wCard* be a function from the powerset of the Database D into the nonnegative Real Numbers, $wCard: 2^D \rightarrow \Re^{\geq 0}$ and *MinCard* be a positive real number. Then, the predicate *MinWeight* for a set S of objects is defined to be true iff $wCard(S) \geq MinCard$.

The expression $wCard(S) \geq MinCard$ generalizes the condition $|N_{Eps}(o)| \geq MinPts$ in the definition of density-based clusters where *cardinality* is just a special case of a *wCard* function. There are numerous possibilities to define $wCard(S)$ for subsets of the database D. Simply summing up the values of some nonspatial attribute for the objects in S is another example of a *wCard* function. For example, if we want to cluster objects represented by polygons and if the size of the objects should be considered to influence the "density" in

the data space, then the area of the polygons could be used as a weight for these objects. A further possibility is to sum up a value derived from several nonspatial attributes, e.g., by specifying ranges for some nonspatial attribute values of the objects (i.e., a selection condition). We can realize the clustering of only a subset of the database D by attaching a weight of 1 to objects that satisfy the selection condition and a weight of 0 to all other objects. Note that by using nonspatial attributes as a weight for objects one can "induce" different densities, even if the objects are equally distributed in the space of the spatial attributes. Also note that by means of the *wCard* function the combination of a clustering with a selection on the database is possible, i.e., performing a selection "on the fly" while clustering the database. This may be more efficient than performing the selection first under certain circumstances because GDBSCAN can use existing spatial index structures (see Section 5.1).

We can now define density-connected sets,analogously to the definition of density-based clusters in Ester et al. (1996), in a straightforward way (see also Ester et al., 1997).

A naive approach could require for each object in a density-connected set that the weighted cardinality of the *NPred*-neighborhood of that object has at least a value *MinCard*. However, this approach fails because there may be two kinds of objects in a density-connected set, objects inside (*core object*) and objects "on the border" of the density-connected set (*border objects*). In general, an *NPred*-neighborhood of a border object has a significantly lower *wCard* than an *NPred*-neighborhood of a core object. Therefore, we would have to set the value *MinCard* to a relatively low value in order to include all objects belonging to the same density-connected set. This value, however, will not be characteristic for the respective density-connected set—particularly in the presence of noise objects. Therefore, for every object p in a density-connected set C there must be an object q in C so that p is inside of the *NPred*-neighborhood of q and the weighted cardinality *wCard* of $NPred(q)$ is at least *MinCard*. We also require the objects of the set C to be somehow "connected" to each other. This idea is elaborated in the following definitions and illustrated by 2D point objects, using a distance-based neighborhood for the points, and cardinality as *wCard* function.

Definition 3. (directly density-reachable) An object p is *directly density-reachable* from an object q with respect to *NPred*, *MinWeight* if

(1) $p \in N_{NPred}(q)$ and
(2) $MinWeight(N_{NPred}(q)) = true$ (core object condition).

Obviously, directly density-reachable is symmetric for pairs of core objects. In general, however, it is not symmetric if one core object and one border object are involved. Figure 2 shows the asymmetric case.

Figure 2. Core objects and border objects.

Figure 3. Density-reachability and density-connectivity.

Definition 4. (density-reachable) An object p is *density-reachable* from an object q with respect to *NPred* and *MinWeight* if there is a chain of objects $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that for all $i = 1, \ldots, n-1 : p_{i+1}$ is directly density-reachable from p_i with respect to *NPred* and *MinWeight*.

Density-reachability is a canonical extension of direct density-reachability. This relation is transitive, but it is not symmetric. Figure 3 depicts the relations of some sample objects and, in particular, the asymmetric case. Although not symmetric in general, it is obvious that density-reachability is symmetric for core objects because a chain from q to p can be reversed if also p is a core object.

Two border objects of the same density-connected set C are possibly not density-reachable from each other because the core objects condition might not hold for both of them. However, for a density-connected set C we require that there must be a core object in C from which both border objects are density-reachable. Therefore, we introduce the notion of density-connectivity which covers this relation of border objects.

Definition 5. (density-connected) An object p is *density-connected* to an object q with respect to *NPred*, *MinWeight* if there is an object o such that both, p and q are density-reachable from o with respect to *NPred*, *MinWeight*.

Density-connectivity is a symmetric relation. For density-reachable objects, the relation of density-connectivity is also reflexive (cf. figure 3).

Now, a density-connected set is defined to be a set of density-connected objects which is maximal with respect to density-reachability.

Definition 6. (density-connected set) A *density-connected set* C with respect to *NPred*, *MinWeight* in D is a nonempty subset of D satisfying the following conditions:

(1) Maximality: For all $p, q \in D$: if $p \in C$ and q is density-reachable from p with respect to *NPred*, *MinWeight*, then $q \in C$.
(2) Connectivity: For all $p, q \in C$: p is density-connected to q with respect to *NPred*, *MinWeight*.

Note that a density-connected set C with respect to *NPred* and *MinWeight* contains at least one core object: since C contains at least one object p, p must be density-connected to itself via some object o (which may be equal to p). Thus, at least o has to satisfy the core object condition. Consequently, the *NPred*-Neighborhood of o has to satisfy *MinWeight*.

The following lemmas are important for validating the correctness of our clustering algorithm. Intuitively, they state the following. Given *NPred* and *MinWeight*, we can discover a density-connected set in a two-step approach. First, choose an arbitrary object from the database satisfying the core object condition as a seed. Second, retrieve all objects that are density-reachable from the seed obtaining the density-connected set containing the seed.

Lemma 1. *Let p be an object in D and MinWeight$(N_{NPred}(p)) = true$. Then the set $O = \{o \in D \mid o$ is density-reachable from p with respect to NPred, MinWeight$\}$ is a density-connected set with respect to NPred, MinWeight.*

Proof:

(1) O is nonempty: p is a core object by assumption. Therefore p is density-reachable from p. Then p is in O.
(2) Maximality: Let $q_1 \in O$ and q_2 be density-reachable from q_1 with respect to *NPred, MinWeight*. Since q_1 is density-reachable from p and density-reachability is transitive with respect to *NPred, MinWeight*, it follows that q_2 is also density-reachable from p with respect to *NPred, MinWeight*. Hence, $q_2 \in O$.
(3) Connectivity: All objects in O are density-connected via object p. □

Furthermore, a density-connected set C with respect to *NPred, MinWeight* is uniquely determined by *any* of its core objects, i.e., each object in C is density-reachable from any of the core objects of C and, therefore, a density-connected set C contains exactly the objects which are density-reachable from an arbitrary core object of C.

Lemma 2. *Let C be a density-connected set with respect to NPred, MinWeight and let p be any object in C with MinWeight $(N_{NPred}(p)) = true$. Then C equals to the set $O = \{o \in D \mid o$ is density-reachable from p with respect to NPred, MinWeight$\}$.*

Proof:

(1) $O \subseteq C$ by definition of O.
(2) $C \subseteq O$: Let $q \in C$. Since also $p \in C$ and C is a density-connected set, there is an object $o \in C$ such that p and q are density-connected via o, i.e., both p and q are density-reachable from o. Because both p and o are core objects, it follows that also o is density-reachable from p (symmetry for core objects). With transitivity of density-reachability with respect to *NPred, MinWeight* it follows that q is density-reachable from p. Then $q \in O$. □

Let us now define a clustering *CL* of a database D with respect to *NPred* and *MinWeight* as the set of all density-connected sets with respect to *NPred* and *MinWeight* in D, i.e., all clusters from a clustering *CL* are density-connected sets with regard to the same "parameters" *NPred* and *MinWeight*. *Noise* will then be defined relative to a given clustering *CL* of D, simply as the set of objects in D not belonging to any of the clusters of *CL*.

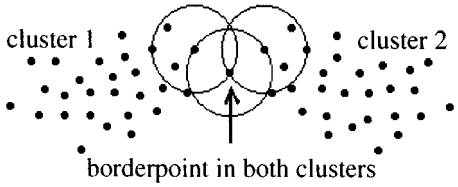

Figure 4. Overlap of two clustures for $MinCard = 4$.

Definition 7. (clustering) A clustering CL of D with respect to *NPred*, *MinWeight* is a set of density-connected sets with respect to *NPred*, *MinWeight* in D, $CL = \{C_1, \ldots, C_k\}$, such that for all C the following holds: if C is a density-connected set with respect to *NPred*, *MinWeight* in D, then $C \in CL$.

Definition 8. (noise) Let $CL = \{C_1, \ldots, C_k\}$ be a clustering of the database D with respect to *NPred*, *MinWeight*. Then we define the *noise* in D as the set of objects in the database D not belonging to any density-connected set C_i, i.e., $noise_{CL} = D \setminus (C_1 \cup \cdots \cup C_k)$.

There are other possibilities to define a clustering based on Definition 6. But this simple notion of a clustering has the nice property that two clusters can at most overlap in objects which are border objects in both clusters. Figure 4 illustrates the overlap of two clusters using cardinality and $MinCard = 4$.

Lemma 3. *Let CL be a clustering of D with respect to NPred, MinWeight. If $C_1, C_2 \in CL$ and $C_1 \neq C_2$, then for all $p \in C_1 \cap C_2$ it holds that p is not a core object, i.e., $wCard(NPred(p)) < MinCard$.*

Proof: Since *NPred* and *MinWeight* are the same for all clusters in CL it follows that if $p \in C_1 \cap C_2$ would be a core object for C_1, then p would also be a core object for C_2. But then it follows from Lemma 2 that $C_1 = C_2$, in contradiction to the assumption. Hence, p is not a core object. □

3.2. Important specializations

The first specialization of a density-connected set obviously is a density-based cluster as defined in Ester et al. (1996):

- *NPred*: "distance $\leq Eps$", *wCard*: cardinality, $MinWeight(N)$: $|N| \geq MinPts$. Specializing this instance further yields a description of a *level* in the *single-link hierarchy* determined by a "critical distance" $D_{\min} = Eps$ (Sibson, 1973):
- *NPred*: "distance $\leq$ *NN-dist*", *wCard*: cardinality, $MinWeight(N)$: $|N| \geq 2$, every point p in the set $noise_{CL}$ must be considered as a single cluster.

Figure 5. Different specializations of density-connected sets.

Note that if cardinality is used and $MinCard \leq 3$ there exists *no* overlap between the clusters of a clustering *CL*. But then, the well-known "*single-link effect*" can occur, i.e., if there is a chain of points between two clusters where the distance of each point in the chain to the neighboring point in the chain is less than *Eps* then the two clusters will not be separated. Higher values for *MinCard* will significantly weaken this effect and even for regular distributions where the *k-distance* values may not differ from the 1-*distance* values for almost all points, a clustering according to Definitions 7 and 8, in general, is not equivalent to a level in the single-link hierarchy.

A further specialization of density-connected sets allows the clustering of spatially extended objects such as polygons:

- *NPred*: "*intersects*" or "*meets*", *wCard*: sum of areas, $MinWeight(N)$: sum of areas $\geq MinArea$

There are also specializations equivalent to simple forms of region growing (Niemann, 1990), i.e., only local criteria for expanding a region can be defined by the weighted cardinality function. For instance, the neighborhood may be simply given by the neighboring cells in a grid and the weighted cardinality function may be some aggregation of the nonspatial attribute values.

- *NPred*: "*neighbor*", $MinWeight(N)$: *aggr*(nonspatial values) $\geq$ *threshold*
  While region growing algorithms are highly specialized to pixels, density-connected sets can be defined for any data types.

Figure 5 illustrates some specializations of density-connected sets.

4. GDBSCAN: Generalized density-based spatial clustering of applications with noise

In Section 4.1, we present the algorithm GDBSCAN (Generalized Density Based Spatial Clustering of Applications with Noise) which is designed to discover the density-connected sets and the noise in a spatial database. To apply the algorithm, we have to know the *NPred*-neighborhood, *MinCard* and the *wCard* function. In Section 4.2, the issue of determining these "parameters" is discussed and a simple and effective heuristic to determine *Eps* and *MinCard* for *Eps*-neighborhoods combined with cardinality as *wCard* function is presented.

```
GDBSCAN (SetOfObjects, NPred, MinCard, wCard)
 // SetOfObjects is UNCLASSIFIED
 ClusterId := nextId(NOISE);
 FOR i FROM 1 TO SetOfObjects.size DO
  Object := SetOfObjects.get(i);
  IF Object.ClId = UNCLASSIFIED THEN
   IF ExpandCluster(SetOfObjects,Object,ClusterId,NPred,MinCard,wCard) THEN
     ClusterId:=nextId(ClusterId)
   END IF
  END IF
 END FOR
END; // GDBSCAN
```

Figure 6. Algorithm GDBSCAN.

4.1. The algorithm

To find a density-connected set, GDBSCAN starts with an arbitrary object p and retrieves all objects density-reachable from p with respect to *NPred* and *MinWeight*. If p is a core object, this procedure yields a density-connected set with respect to *NPred* and *MinWeight* (see Lemmas 1 and 2). If p is not a core object, no objects are density-reachable from p, and p is assigned to NOISE. This procedure is iteratively applied to each object p which has not yet been classified. Thus, a clustering and the noise according to Definitions 7 and 8 are detected.

In figure 6, we present a basic version of GDBSCAN omitting details of data types and generation of additional information about clusters.

SetOfObjects is either the whole database or a discovered cluster from a previous run. NPred and MinCard are the global density parameters and wCard is a pointer to a function wCard(Objects) that returns the weighted cardinality of the set Objects. ClusterIds are from an ordered and countable datatype (e.g., implemented by Integers) where UNCLASSIFIED $<$ NOISE $<$ "other Ids", and each object is marked with a ClusterId Object.ClId. The function nextId(ClusterId) returns the successor of ClusterId in the ordering of the datatype (e.g., implemented as Id $:=$ Id $+ 1$). The function SetOfObjects.get(i) returns the ith element of SetOfObjects. In figure 7, function ExpandCluster constructing a density-connected set for a core object Object is presented in more detail.

A call of SetOfObjects.neighborhood(Object, NPred) returns the *NPred*-neighborhood of Object in SetOfPoints as a list of objects. Obviously the efficiency of the above algorithm depends on the efficiency of the neighborhood query because such a query is performed exactly once for each object in SetOfObjects which satisfies the selection condition. The performance of GDBSCAN is discussed in detail in Section 5, where we see that neighborhood predicates based on spatial proximity like distance predicates or intersection can be evaluated very efficiently by using spatial index structures.

The ClusterId of some objects p are to be marked NOISE because $wCard(NPred(p)) <$ *MinCard* may be changed later if they are density-reachable from some other object of the

```
ExpandCluster(SetOfObjects, Object, ClId, NPred, MinCard, wCard): Boolean;
 IF wCard({Object}) ≤ 0 THEN  // point not in selection
  SetOfObjects.changeClId(Object,UNCLASSIFIED);
  RETURN False;
 END IF
 seeds:=SetOfObjects.neighborhood(Object,NPred);
 IF wCard(seeds) < MinCard THEN // no core object
  SetOfObjects.changeClId(Object,NOISE);
  RETURN False;
 END IF
 // still here? Object is a core object
 SetOfObjects.changeClIds(seeds,ClId);
 seeds.delete(Object);
 WHILE seeds ≠ Empty DO
  currentObject := seeds.first();
  result := SetOfObjects.neighborhood(currentObject, NPred);
  IF wCard(result) ≥ MinCard THEN
   FOR i FROM 1 TO result.size DO
    P := result.get(i);
    IF wCard({P}) > 0 AND P.ClId IN {UNCLASSIFIED, NOISE} THEN
     IF P.ClId = UNCLASSIFIED THEN
      seeds.append(P);
     END IF;
     SetOfObjects.changeClId(P,ClId);
    END IF; // wCard > 0 and UNCLASSIFIED or NOISE
   END FOR;
  END IF; // wCard ≥ MinCard
  seeds.delete(currentObject);
 END WHILE; // seeds ≠ Empty
 RETURN True;
END; // ExpandCluster
```

Figure 7. Function ExpandCluster.

database. This may happen only for border objects of a cluster. Those objects are then not added to the seeds-list because we already know that an object with a ClusterId of NOISE is not a core object, i.e., no other objects are density-reachable from them.

If two clusters C_1 and C_2 are very close to each other, it might happen that some object p belongs to both C_1 and C_2. Then p must be a border object in both clusters according to Lemma 3. In this case, object p will only be assigned to the cluster discovered first. Except from these rare situations, the result of GDBSCAN is independent of the order in which the objects of the database are visited due to Lemmas 1 and 2.

There may be reasons to apply a postprocessing to the clustering obtained by GDBSCAN. According to Definition 7, each set of objects having *MinWeight* is a density-connected set. In some applications (see, e.g., Section 6.1), however, density-connected sets of

this minimum size are too small to be accepted as clusters. Furthermore, GDBSCAN produces clusters and noise. But for some applications a non-noise class label for each object is required. For this purpose, we can reassign each noise object and each object of a rejected cluster to the closest of the accepted clusters. This postprocessing requires just a simple scan over the whole database without much computation, in particular, no region queries are necessary. Therefore, such postprocessing does not increase the runtime complexity of GDBSCAN.

To conclude this section, we introduce the algorithm DBSCAN (Ester et al., 1996) as an important specialization of GDBSCAN.

Definition 9. (DBSCAN) DBSCAN is a specialization of the algorithm GDBSCAN using the following parameters: *NPred*: "distance $\leq$ *Eps*", *wCard*: cardinality, *MinWeight*(N): $|N| \geq$ *MinCarol*.

4.2. Determining the parameters for GDBSCAN

GDBSCAN requires a neighborhood predicate *NPred*, a weight function *wCard* and a minimum weight *MinCard*. The concrete parameters we are going to use, depends on the goal of the application. In some applications there may be a natural way to provide values without any further parameter determination. In other cases, we may only know the *type* of neighborhood that we want to use, e.g., a distance-based neighborhood for the clustering of point objects. In these cases we have to use a heuristic to determine the appropriate parameters.

In this section, we present a simple heuristic which is effective in many cases to determine the parameters *Eps* and *MinCard* for DBSCAN (cf. Definition 9), which is the most important specialization of GDBSCAN. DBSCAN uses a distance-based neighborhood "distance less or equal than *Eps*" and cardinality as the *wCard* function. Thus, we have to determine appropriate values for *Eps* and *MinCard*. The density parameters of the "thinnest", i.e., least dense, cluster in the database are good candidates for these global values specifying the lowest density, which is not considered to be noise.

For a given $k \geq 1$ we define a function *k-distance*, mapping each object to the distance from its kth nearest neighbor. When sorting the objects of the database in descending order of their *k-distance* values, the plot of this function gives some hints concerning the density distribution in the database. We call this plot the *sorted k-distance plot* (see, e.g., figure 8). If we choose an arbitrary object p, set the parameter *Eps* to *k-distance*(p) and set the parameter *MinCard* to $k+1$, all objects with an equal or smaller *k-distance* value will be core objects, because there are at least $k+1$ objects in an *Eps*-neighborhood of an object p if *Eps* is set to *k-distance*(p). If we can now find a *threshold object* with the maximum *k-distance* value in the "thinnest" cluster of D, we would obtain the desired parameter values. Therefore, we have to answer the following questions. (1) Which value of k is appropriate? (2) How can we determine a threshold object p?

Let us discuss the value k first, assuming it is possible to set the appropriate value for *Eps*. The smaller we choose the value for k, the lower are the computational costs to calculate the *k-distance* values and the smaller is the corresponding value for *Eps* in general. But a small change of k for an object p, in general, only results in a small change of *k-distance*(p).

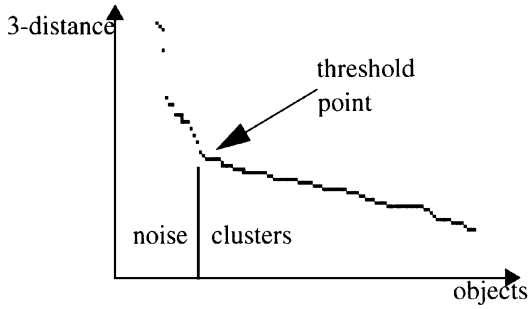

Figure 8. Sorted 3-distance plot for sample database 3.

Furthermore, our experiments indicate that the *k-distance* plots for "reasonable" k (e.g., $1 \leq k \leq 10$ in 2D space) do not significantly differ in shape and that also the results of DBSCAN for the corresponding parameter pairs (k, *Eps*) do not differ very much. Therefore, the choice of k is not very crucial for the algorithm. We can even fix the value for k (with respect to the dimension of the dataspace), eliminating the parameter *MinCard* for DBSCAN. Considering only the computational cost, we would like to set k as small as possible. On the other hand, if we set $k = 1$, the *k-distance* value for an object p will be the distance to the nearest neighbor of p and the "single-link effect" can occur. To weaken this effect, we must choose a value for $k > 1$.

We propose to set k to 2^**dimension* $- 1$. Our experiments indicate that this value works well for databases D where each point occurs only once, i.e., if D is really a *set* of points. Thus in the following, if not stated otherwise, k is set to this value, and the value for *MinCard* is fixed according to the above strategy (*MinCard* $= k + 1$, e.g., *MinCard* $= 4$ in 2D space).

To determine the parameter *Eps* for DBSCAN, we have to know an object in the "thinnest" cluster of the database with a high *k-distance* value for that cluster. Figure 8 shows a sorted *k-distance* plot for sample database 3 which is very typical for databases where the density of clusters and noise are significantly different. Our experiments indicate that the threshold object is an object near the first "valley" of the sorted *k-distance* plot (see figure 8). All objects with a higher *k-distance* value (to the left of the threshold) will then be noise, all other objects (to the right of the threshold) will be assigned to some cluster.

In general, it is very difficult to detect the first "valley" automatically, but it is relatively simple for a user to recognize this valley in a graphical representation. Additionally, if the user can estimate the percentage x of noise, a proposal for the threshold object can be derived, because we know that most of the noise objects have a higher *k-distance* value than objects of clusters. The *k-distance* values of noise objects are located on the left of the *k-distance* plot, so that we can simply select an object after x percent of the sorted *k-distance* plot.

There is always a *range* of values for the parameter *Eps* that yield the same clustering because not all objects of the "thinnest" cluster need to be core objects. They may also belong to the cluster if they are only density-reachable. Furthermore, the *Eps* value may be larger than needed if the clusters are well separated and the density of noise is clearly lower than the density of the thinnest cluster. Thus, the robustness of the parameter determination,

i.e., the width of the range of appropriate *Eps* values, depends on the application. However, in general, the width of this range is broad enough to allow the parameters to be determined in a sorted *k-distance* plot for only a very small sample of the whole database (1–10%).

To summarize, we propose the following interactive approach for determining the parameters for DBSCAN.

- The user gives a value for k (default is $k = 2^*dimension - 1$).
- The system computes and displays the *k-distance* plot for a sample of the database.
- The user selects an object as the threshold object and the *k-distance* value of this object is used as the *Eps* value; *MinCard* is set to $k + 1$ (if the user can estimate the percentage of noise, the system can derive a proposal for the threshold object from it).

Obviously, the shape of the sorted *k-distance* plot and hence the effectiveness of the proposed heuristic depends on the distribution of the k-nearest neighbor distances. For example, the plot looks more "stairs-like" if the objects are distributed more regularly within each cluster, or the first "valley" is less clear if the densities of the clusters differ not much from the density of noise (which also means that the clusters are not well separated). Then, knowing the approximate percentage of noise in the data may be very helpful.

Although for some applications it may be difficult to determine the correct parameters, we want to point out that the parameters may be reused in different but similar applications, e.g., if the different datasets are produced by a similar process. We see in Section 6 that there are even applications where the appropriate parameter values for GDBSCAN can be derived analytically (e.g., Section 6.2), or a natural notion of a neighborhood for the application exists, which does not require any further parameters (e.g., *intersects* for polygons).

5. Performance evaluation

In this section, we evaluate the performance of GDBSCAN. In Section 5.1, we discuss the performance of GDBSCAN with respect to the underlying spatial index structure. In Section 5.2, an experimental evaluation of DBSCAN and a comparison with the well-known clustering algorithms CLARANS and BIRCH is presented.

5.1. Analytical evaluation

The runtime of GDBSCAN obviously is $O(n *$ runtime of a neighborhood query): n objects are visited and exactly one neighborhood query is performed for each of them. The number of neighborhood queries cannot be reduced since a ClusterId for each object is required. Thus, the overall runtime depends on the performance of the neighborhood query. Fortunately, the most interesting neighborhood predicates are based on spatial proximity—like distance predicates or intersection—which can be efficiently supported by spatial index structures. Such index structures are assumed to be available in a SDBS for efficient processing of several types of spatial queries (Brinkhoff et al., 1994).

In the following, we introduce a typical spatial index, the R*-tree (Beckmann et al., 1990). The *R*-tree* (see figure 9) generalizes the one-dimensional B-tree to d-dimensional

Table 1. Runtime complexity of GDBSCAN.

Runtime complexity of	A single neighborhood query	The GDBSCAN algorithm
Without index	$O(n)$	$O(n^2)$
With spatial index	$O(\log n)$	$O(n * \log n)$
With direct access	$O(1)$	$O(n)$

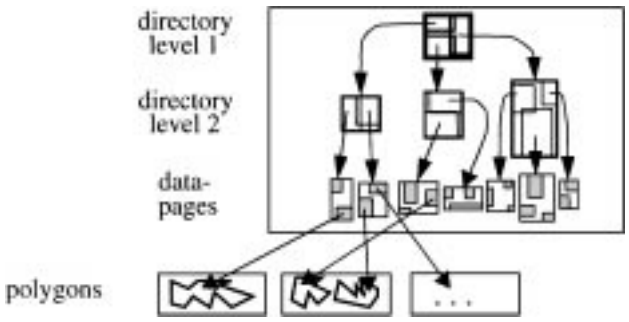

Figure 9. Structure of an R*-tree.

data spaces, specifically an R*-tree manages k-dimensional hyperrectangles instead of one-dimensional keys. An R*-tree may organize extended objects such as polygons using *minimum bounding rectangles* (MBR) as approximations as well as point objects as a special case of rectangles. The leaves store the MBRs of the data objects and a pointer to the exact geometry of the polygons. Internal nodes store a sequence of pairs consisting of a rectangle and a pointer to a child node. These rectangles are the MBRs of all data or directory rectangles stored in the subtree having the referenced child node as its root. To answer a region query, starting from the root, the set of rectangles intersecting the query region is determined and then their referenced child nodes are searched until the data pages are reached.

The height of an R*-tree is $O(\log n)$ for a database of n objects in the worst case and a query with a "small" query region has to traverse only a limited number of paths in the R*- tree. Since most *NPred*-neighborhoods are expected to be small compared to the size of the whole database, the average runtime complexity of a single neighborhood query is $O(\log n)$.

Table 1 lists the runtime complexity of GDBSCAN with respect to the underlying spatial index structure. Without any index support, the runtime of GDBSCAN is $O(n^2)$. This does not scale well with the size of the database. But, if we use a spatial index, the runtime is reduced to $O(n \log n)$. If we have a direct access to the *NPred*-neighborhood, e.g., if the objects are organized in a grid, the runtime is further reduced to $O(n)$.

5.2. Experimental evaluation

We have implemented GDBSCAN in C++ based on an implementation of the R*-tree (Beckmann et al., 1990). All experiments were run on HP 735/100 workstations. In order

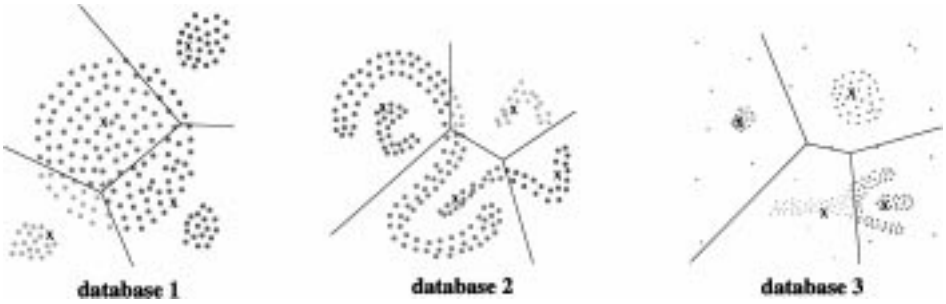

Figure 10. Clusterings discovered by CLARANS.

to allow a comparison with CLARANS and BIRCH—both use a distance-based neighborhood definition—we evaluated the specialized DBSCAN algorithm (cf. Definition 9). For an evaluation of the effectivity of the more general GDBSCAN, see the applications in Section 6.

To compare DBSCAN with CLARANS in terms of effectiveness (accuracy), we use the three synthetic sample databases which are depicted in figure 1. Since DBSCAN and CLARANS are clustering algorithms of different types, they have no common quantitative measure of the classification accuracy. Therefore, we evaluate the accuracy of both algorithms by visual inspection. In sample database 1, there are four ball-shaped clusters of significantly differing sizes. Sample database 2 contains four clusters of nonconvex shape. In sample database 3, there are four clusters of different shape and size with additional noise. To show the results of both clustering algorithms, we visualize each cluster by a different color. To give CLARANS some advantage, we set the parameter k (number of clusters) to 4 for these sample databases. The clusterings discovered by CLARANS are depicted in figure 10.

For DBSCAN, the parameter *Eps* was set, giving a noise percentage of 0% for sample databases 1 and 2, and 10% for sample database 3, respectively. The clusterings discovered by DBSCAN are depicted in figure 11.

DBSCAN discovers all clusters (according to Definition 7) and detects the noise points (according to Definition 8) from all sample databases. CLARANS, however, splits clusters if they are relatively large or if they are close to some other cluster. Furthermore, CLARANS has no explicit notion of noise. Instead, all points are assigned to their closest medoid.

To test the efficiency of DBSCAN and CLARANS, we use the SEQUOIA 2000 benchmark data. The SEQUOIA 2000 benchmark database (Stonebraker et al., 1993) uses real data sets that are typical for Earth Science tasks. There are four types of data in the database: raster data, point data, polygon data and directed graph data. The point data set contains 62,584 Californian names of landmarks, extracted from the US Geological Survey's Geographic Names Information System, together with their location. The point data set occupies about 2.1 MB. Since the runtime of CLARANS on the whole data set is very high, we have extracted a series of subsets of the SEQUIOA 2000 point data set containing 2–20% representatives of the whole set. The runtime comparison of DBSCAN and CLARANS on these databases is shown in Table 2. The results of our experiments show

Table 2. Comparison of runtime (s).

Number of points	1252	2503	3910	5213	6256	7820	8937	10426	12512	62584
DBSCAN	3	7	11	16	18	25	28	33	42	233
CLARANS	758	3,026	6,845	11,745	18,029	29,826	39,265	60,540	80,638	???

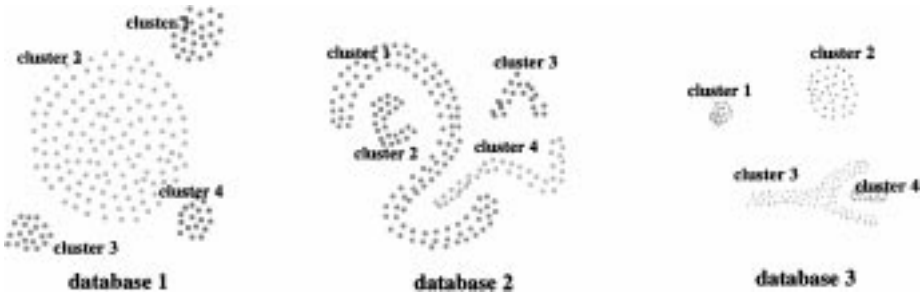

Figure 11. Clusterings discovered by DBSCAN.

that the runtime of DBSCAN is almost linear to the number of points. The runtime of CLARANS, however, is close to quadratic to the number of points. The results show that DBSCAN outperforms CLARANS by a factor of between 250 and 1,900 which grows with increasing size of the database.

Since we found it rather difficult to set the parameters of BIRCH appropriately for the SEQUIOA 2000 point data, we used the test data sets 1, 2 and 3 introduced by Zhang et al. (1997) to compare DBSCAN with BIRCH. The used implementation of BIRCH—using CLARANS in phase 3—was provided by its authors. The runtime of DBSCAN was 1.8, 1.8 and 12.0 times the runtime of BIRCH. Note, however, that in general the same restrictions with respect to clusters of arbitrary shape apply to BIRCH as they apply to CLARANS. Furthermore, the clustering features—on which BIRCH is based—can be only defined in a Euclidean vector space implying a limited applicability of BIRCH compared to DBSCAN (and compared to CLARANS).

6. Applications

In this section, we present four typical applications of GDBSCAN. In the first application we cluster a spectral space (5D points) created from satellite images in different spectral channels which is a common task in remote sensing image analysis. The second application comes from molecular biology. The points on a protein surface (3D points) are clustered to extract regions with special properties. To find such regions is a subtask for the problem of protein-protein docking. The third application uses astronomical image data (2D points) showing the intensity on the sky at different radio wavelengths. The task of clustering is to detect celestial sources from these images. The last application is the detection of spatial trends in a geographic information system. GDBSCAN is used to cluster 2D polygons creating so-called influence regions which are used as input for trend detection.

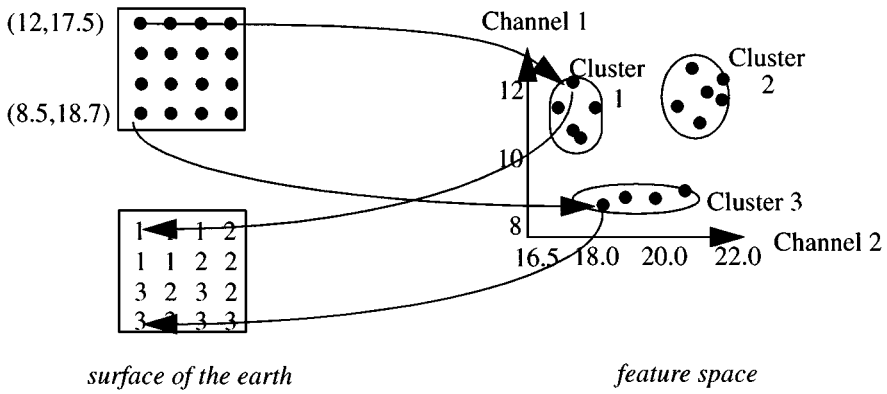

Figure 12. Relation between 2D image and feature space.

6.1. Application 1: earth science (5D points)

In this application, we use a five-dimensional feature space obtained from several satellite images of a region on the surface of the earth covering California. These images are taken from the raster data of the SEQUOIA 2000 Storage Benchmark. After some preprocessing, five images containing 1,024,000 intensity values (8 bit pixels) for five different spectral channels for the same region were combined. Thus, each point on the surface, corresponding to an earth surface area of $1{,}000 \times 1{,}000$ m, is represented by a five-dimensional vector.

Finding clusters in such feature spaces is a common task in remote sensing digital image analysis (e.g., Richards, 1983) for the creation of thematic maps in geographic information systems. The assumption is that feature vectors for points of the same type of underground on the earth are forming groups in the high-dimensional feature space (see figure 12 illustrating the case of 2D raster images).

Application 1 has two characteristics that were not present in the synthetic databases used in Section 5.2. First, the coordinates of points can only be integer values between 0 and 255 in each dimension. Second, many of the raster points have exactly the same features, i.e., are represented by the same five-dimensional feature vector. Only about 600,000 of the 1,024,000 feature vectors are different from each other.

We used "distance$(X, Y) < 1.42$" as *NPred*(X, Y). The neighborhoods are very small due to the first characteristic of the application, e.g., for about 15% of the points the distance to the ninth nearest neighbor is 0. We used cardinality as *wCard* function and set *MinCard* to 20 to take into account the second characteristic of the data.

There are several reasons to apply a postprocessing to improve the clustering result of GDBSCAN. First, GDBSCAN only ensures that a cluster contains at least *MinCard* points (using cardinality as *wCard* function), but a minimum size of 20 points is too small for this application, especially because many points have the same coordinates. Therefore, we accepted only the clusters containing more than 200 points. This value seems arbitrary but a minimum size can be chosen reasonably after the size of all clusters is known. Second, GDBSCAN produces clusters and noise. But for this application a non-noise class label for

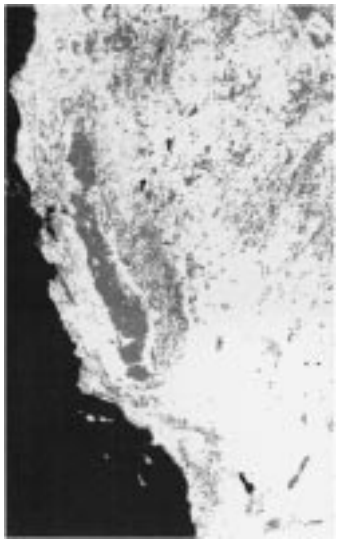

Figure 13. Visualization of the clustering result for the SEQUIOA 2000 raster data.

each raster point is required. Therefore, we reassigned each noise point and each point of a rejected cluster to the closest of the accepted clusters. We obtained 9 clusters with sizes ranging from 598,863 to 2,016 points.

To visualize the result, each cluster was coded by a different color. Then each point in the image of the surface of the earth was colored according to the identificator of the cluster containing the corresponding five-dimensional vector. The resulting image is shown in figure 13. A high degree of correspondence between the obtained image and a physical map of California can easily be seen. A detailed discussion of this correspondence is beyond the scope of this paper.

6.2. Application 2: molecular biology (3D points)

Proteins are biomolecules consisting of some hundreds to thousands of atoms. Their mode of operation lies in the interaction with other biomolecules, e.g., proteins, DNA or smaller partner molecules. These interactions are performed by the so-called *docking*, i.e., the process of connecting the partner molecules.

Molecular biologists point out that the geometry of the molecular surfaces at the interaction site plays an important role along with the physicochemical properties of the molecules. A necessary condition for protein-protein docking is the complementarity of the interaction site with respect to surface shape, electrostatic potential, hydrophobicity, etc. We use the crystallographically determined atom coordinates of proteins and protein complexes from the Brookhaven Protein Data Bank (Bernstein et al., 1977, Protein Data Bank, 1994) and derive for each protein a surface with some 10,000 equally distributed 3D points. For each point on the protein surface, several geometric and physicochemical features are computed.

The *solid angle* (*SA*), for example, (Connolly, 1986) is a geometric feature describing the degree of convexity or concavity of the surface in the neighborhood of the considered point.

A database system for protein-protein docking has to process queries for proteins with complementary surfaces. This search is performed at the level of surface *segments*, defined as a set of neighboring surface points with similar nonspatial attributes, e.g., with similar *SA* values. The segments should have a good correlation with the known docking sites of the proteins, i.e., a docking site on a protein surface should consist of a small number of segments. Therefore, finding a segmentation of protein surfaces is an important subtask for a protein docking DB. We applied GDBSCAN for this task.

We used a *wCard* function performing a simple selection on the *SA* values. The *SA* values are normalized in the interval [0, 1], such that high *SA* values indicate points on a *convex* and low *SA* values indicate points on a *concave* surface segment. To find the convex segments, we defined $wCard(S)$ as the number of points in S with *SA value* between 0.75 and 1.00. As the selection criterion for points on a concave surface segment, we used *SA values* between 0.00 and 0.65. The parameters *NPred* and *MinCard* were determined analytically. Since the surface points are equally distributed with a density of 5 points per $Å^2$, we used "distance$(X, Y) < 0.6$" as $NPred(X, Y)$ and set $MinCard = 5$. Note that if we would use these parameters together with cardinality, only a single cluster containing all points of the protein surface would be found. In applications with equally distributed points GDBSCAN can only find reasonable clusters if the *wCard* function is defined appropriately, i.e., the *wCard* function must "simulate" regions of different density. We searched for clusters covering at least 1% of the surface points of the protein. For example, for the protein 133DA consisting of 5,033 surface points, only clusters with a minimum size of 50 surface points were accepted. In this case 8 convex and 4 concave clusters (segments) were found by using the above parameter settings. Figure 14 depicts the clustering results of GDBSCAN for this protein. Note that some of the clusters are hidden in the visualization. GDBSCAN discovered the most significant convex and concave surface segments of the protein, which can easily be verified by visual inspection.

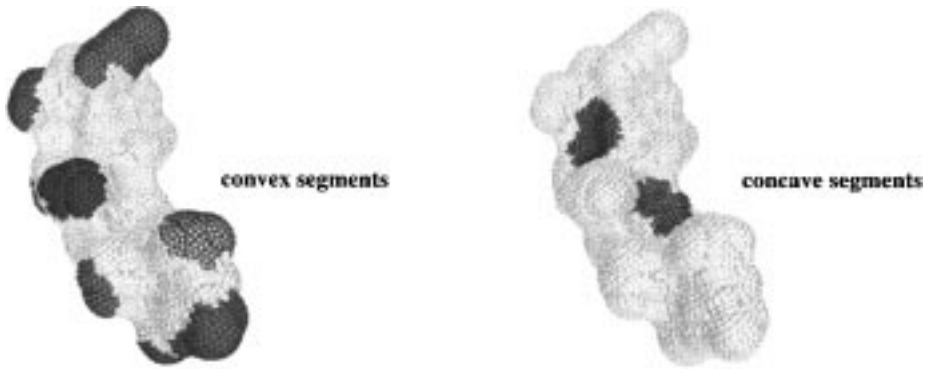

Figure 14. Visualization of the clustering results for protein 133DA.

6.3. *Application 3: astronomy (2D points)*

Surveys of the sky form an integral part of astronomy. Celestial sources detected in a survey are typically classified by the domain scientists; large surveys detect many objects and enable statistical studies of the objects in a given classification. Surveys may also reveal exotic or anomalous objects or previously unidentified classes of objects. A typical result of a survey is a two-dimensional grid of the intensity on the sky. The measured intensity is typically the sum of the emission from discrete sources, diffuse emission (e.g., from the atmosphere, interplanetary medium or interstellar medium), and noise contributed by the surveying instrument itself. Modern surveys are capable of producing thousands of images of the sky, consuming 10 GB–1 TB of storage space, and may contain 10^5 to 10^6 or more sources, e.g., (Reid et al., 1991; Becker et al., 1995).

Maximizing the yield from a survey requires an accurate yet efficient method of detecting sources. The traditional method of separating the discrete sources from the noise and other emissions is to require that the sources exceed a predefined threshold, e.g., 5σ, where σ is an estimate of the rms intensity in the image, e.g. (Becker et al. 1995). Recently, alternate methods, which utilize the expected statistics of the intensity (Zepka et al., 1994) or classifier systems (Weir et al., 1995), have been deployed.

An extreme example of a noisy image is shown on the left side of figure 16. The image shows the intensity, as measured by the Very Large Array, in a direction towards the Galactic center at a radio wavelength of 4,865 MHz. The image is dominated by a celestial source near the center, and the sidelobes which appear as radial spokes and are produced by the optics of the instrument. A second image of the same area at a slightly different wavelength was also given for this application. Because of its similarity to the first image, it is not depicted. The intensity values in the images range from -0.003084 to 0.040023 and from -0.003952 to 0.040509, respectively. We applied GDBSCAN using the same parameter settings for both images:

- *NPred*(X, Y) is "distance$(X, Y) < 1.42$", i.e., the neighborhood of a raster point is a 3×3 array of points.
- *wCard* calculates the sum of the weights of all 9 points of the neighborhood weighting each point by its intensity value.
- *MinCard* $= 0.045$ (i.e., the average intensity required for points in clusters $= 0.005$).

The resulting clusterings for both images are given in figure 15. For example, the brightest celestial source can easily be identified as the cluster in the center.

For the other clusters it is not so easy to verify that they are in fact celestial sources. The only way to confirm a weak source is to detect it again in different images, e.g., if it can be detected again by looking at it at slightly different frequencies. A source is required to appear at the same position, may be with a shift of a pixel or two, at all frequencies. Therefore, we extracted only the clusters which are present in *both* images, there are 20 of them. The result of this procedure is depicted on the right side of figure 16.

Figure 15. Clustering results for both images.

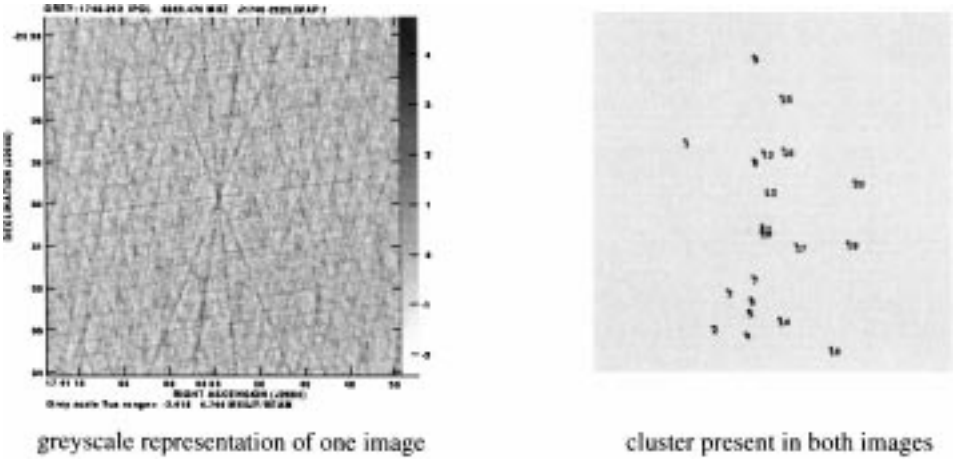

Figure 16. Visualization of the astronomy data and the potential sources found by DBSCAN.

6.4. Application 4: geography (2D polygons)

In the following, we present a simple method for detecting spatial trends based on GDBSCAN. GDBSCAN is used to extract density-connected sets of neighboring objects having a similar value of the nonspatial attribute(s). To define the similarity on an attribute, we partition its domain into a number of disjoint classes and consider the values in the same class as similar to each other. The sets with the highest or lowest attribute value(s) are most interesting and are called *influence regions*, i.e., the maximal neighborhood of a center having a similar value in the nonspatial attribute(s) as the center itself. Then, the resulting influence region is compared to the circular region representing the theoretical trend to obtain a possible deviation. Different methods may be used for this comparison. A *difference-based method* calculates the difference of both, the observed influence region

Figure 17. Approximating ellipsoid of the influence region of Ingolstadt.

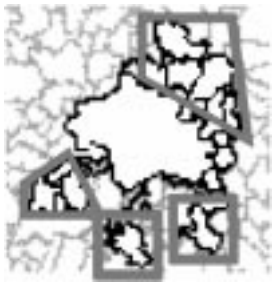

Figure 18. Difference between the observed and the theoretical influence region of Munich.

and the theoretical circular region, thus returning some region indicating the location of a possible deviation. An *approximation-based method* calculates the optimal approximating ellipsoid of the observed influence region. If the two main axes of the ellipsoid differ in length significantly, then the longer one is returned indicating the direction of a deviation.

GDBSCAN can be used to extract the influence regions from an SDBS. We define *NPred* (X, Y) as "intersect$(X, Y) \wedge$ attr-class$(X) =$ attr-class (Y)" and use cardinality as *wCard* function. Furthermore, we set *MinCard* to 2 in order to exclude sets of less than 2 objects.

To illustrate the results of this approach, we discuss some typical influence regions obtained by GDBSCAN. The influence region of Ingolstadt is elongated indicating a deviation in west-east direction caused by the river Danube traversing Ingolstadt in this direction. Figure 17 shows the approximating ellipsoid and the significantly longer main axis in west-east direction.

The influence region of Munich has four significant deviations from the theoretical region (NE, SW, S and SE). Figure 18 illustrates the difference between the observed influence region and the theoretical circular region. These areas coincide with the highways originating from Munich.

7. Conclusions

In this paper, we presented the clustering algorithm GDBSCAN generalizing the density-based algorithm DBSCAN (Ester et al., 1996) in two important ways. GDBSCAN can

cluster point objects as well as spatially extended objects according to both, their spatial and their nonspatial attributes. After a review of related work, the general concept of density-connected sets and an algorithm to discover them were introduced. A performance evaluation, analytical as well as experimental, showed the effectiveness and efficiency of GDBSCAN on large spatial databases. Furthermore, we presented four applications using 2D points (astronomy), 3D points (biology), 5D points (earth science) and 2D polygons (geography) demonstrating the applicability of GDBSCAN to real-world problems.

Future research will have to consider the following issues. First, heuristics to determine the parameters for GDBSCAN where *wCard* is different from the cardinality function should be developed. Second, GDBSCAN creates a one-level clustering. However, a hierarchical clustering may be more useful, in particular, if the appropriate input parameters cannot be estimated accurately. An extension of GDBSCAN to detect simultaneously a hierarchy of clusterings will be investigated.

Acknowledgments

We thank T. Joseph W. Lazio for making the astronomy data available to us and for his substantial help in understanding and modeling the astronomy application. We also thank Thomas Schmidt and Thomas Seidl for providing the protein data. We are grateful to Henning Brockfeld for introducing us into the application of mining in the area of economic geography.

References

Becker, R.H., White, R.L., and Helfand, D.J. 1995. The FIRST survey: Faint images of the radio sky at twenty centimeters. Astrophys. J., 450:559.

Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. 1990. 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, pp. 322–331.

Bernstein, F.C., Koetzle, T.F., Williams, G.J., Meyer, E.F., Brice, M.D., Rodgers, J.R., Kennard, O., Shimanovichi, T., and Tasumi, M. 1977. The protein data bank: A computer-based archival file for macromolecular structures. Journal of Molecular Biology, 112:535–542.

Brinkhoff, T., Kriegel, H.-P., Schneider, R., and Seeger, B. 1994. Multi-step processing of spatial joins. Proc. ACM SIGMOD Int. Conf. on Management of Data. Minneapolis, MN, pp. 197–208.

Connolly, M.L. 1986. Measurement of protein surface shape by solid angles. Journal of Molecular Graphics, 4(1):3–6.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, OR, pp. 226–231.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. 1997. Density-connected sets and their application for trend detection in spatial databases. Proc. 3rd Int. Conf. on Knowledge Discovery and Data Mining. Newport Beach, CA, pp. 10–15.

Ester, M., Kriegel, H.-P., and Xu, X. 1995. A database interface for clustering in large spatial databases. Proc. 1st Int. Conf. on Knowledge Discovery and Data Mining. Montreal, Canada, pp. 94–99.

Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. 1996. Knowledge discovery and data mining: Towards a unifying framework. Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining. Portland, OR, pp. 82–88.

Gueting, R.H. 1994. An introduction to spatial database systems. The VLDB Journal, 3(4):357–399.

Hattori, K. and Torii, Y. 1993. Effective algorithms for the nearest neighbor method in the clustering problem. Pattern Recognition, 26(5):741–746.

Jain, A.K. and Dubes, R.C. 1988. Algorithms for Clustering Data. New Jersey: Prentice Hall.
Kaufman, L. and Rousseeuw, P.J. 1990. Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley & Sons.
MacQueen, J. 1967. Some Methods for Classification and Analysis of Multivariate Observations. In 5th Berkeley Symp. Math. Statist. Prob., L. Le Cam and J. Neyman (Eds.), vol. 1, pp. 281–297.
Matheus, C.J., Chan, P.K., and Piatetsky-Shapiro, G. 1993. Systems for knowledge discovery in databases. IEEE Transactions on Knowledge and Data Engineering, 5(6):903–913.
Murtagh, F. 1983. A survey of recent advances in hierarchical clustering algorithms. The Computer Journal, 26(4):354–359.
Ng, R.T. and Han, J. 1994. Efficient and effective clustering methods for spatial data mining. Proc. 20th Int. Conf. on Very Large Data Bases. Santiago, Chile, pp. 144–155.
Niemann, H. 1990. Pattern Analysis and Understanding. Berlin: Springer-Verlag.
Protein Data Bank. 1994. Quarterly Newsletter 70. Brookhaven National Laboratory, Upton, NY.
Reid, I.N., et al. 1991. The second palomar sky survey. Publ. Astron. Soc. Pacific, 103:661.
Richards, A.J. 1983. Remote Sensing Digital Image Analysis. An Introduction. Berlin: Springer-Verlag.
Sibson, R. 1973. SLINK: An optimally efficient algorithm for the single-link cluster method. The Computer Journal, 16(1):30–34.
Stonebraker, M., Frew, J., Gardels, K., and Meredith, J. 1993. The SEQUOIA 2000 storage benchmark. Proc. ACM SIGMOD Int. Conf. on Management of Data. Washington, DC, pp. 2–11.
Vinod, H. 1969. Integer programming and the theory of grouping. J. Amer. Statist. Assoc., 64:506–517.
Weir, N., Fayyad, U.M., and Djorgovski, S. 1995. Automated star/galaxy classification for digitized POSS-II. Astron. J., 109:2401.
Zepka, A.F., Cordes, J.M., and Wasserman, I. 1994. Signal detection amid noise with known statistics. Astrophys. J., 427–438.
Zhang, T., Ramakrishnan, R., and Linvy, M. 1997. BIRCH: An efficient data clustering method for very large databases. Data Mining and Knowledge Discovery, 1(2):141–182.

Jörg Sander is a research and teaching assistant in the Institute for Computer Science at the University of Munich. His research interests include knowledge discovery and data mining, in particular clustering. He holds a MS degree in philosophy of science and received his MS degree in computer science in 1996 from the University of Munich.

Martin Ester is a research and teaching assistant in the Institute for Computer Science at the University of Munich. His research interests include spatial database systems and data mining. He received his MS in 1984 from the University of Dortmund and his Ph.D. in 1990 from ETH Zurich. From 1990 to 1993 he worked for Swissair, Zurich, specializing in expert systems.

Hans-Peter Kriegel is a full professor for database systems in the Institute for Computer Science at the University of Munich. His research interests are in spatial database systems, particularly in query processing, performance issues, similarity search, high-dimensional indexing and in parallel systems. Data exploration using visualization led him to the area of knowledge discovery and data mining. Kriegel received his MS and Ph.D. 1973 and 1976, respectively, from the University of Karlsruhe, Germany.

Xiaowei Xu is a research and teaching assistant in the Institute for Computer Science at the University of Munich. His research interests include data mining and high-dimensional indexing. He received his MS 1987 from Shen Yang Institute for Computer Technology, Chinese Academy of Sciences.