Published in Proceedings of 1st International Conference on Knowledge Discovery and Data Mining (KDD-95)

A Database Interface for Clustering in Large Spatial Databases[1]

Martin Ester, Hans-Peter Kriegel, Xiaowei Xu

Institute for Computer Science, University of Munich
Leopoldstr. 11 B
D-80802 München, Germany
email: {ester | kriegel | xwxu}@informatik.uni-muenchen.de

Abstract

Both the number and the size of spatial databases are rapidly growing because of the large amount of data obtained from satellite images, X-ray crystallography or other scientific equipment. Therefore, automated knowledge discovery becomes more and more important in spatial databases. So far, most of the methods for knowledge discovery in databases (KDD) have been based on relational database systems. In this paper, we address the task of class identification in spatial databases using clustering techniques. We present an interface to the database management system (DBMS), which is crucial for the efficiency of KDD on large databases. This interface is based on a spatial access method, the R*-tree. It clusters the objects according to their spatial neighborhood and supports efficient processing of spatial queries. Furthermore, we propose a method for spatial data sampling as part of the focusing component, significantly reducing the number of objects to be clustered. Thus, we achieve a considerable speed-up for clustering in large databases. We have applied the proposed techniques to real data from a large protein database used for predicting protein-protein docking. A performance evaluation on this database indicates that clustering on large spatial databases can be performed both efficiently and effectively using our approach.

Keywords: discovery algorithms for large databases, database interfaces, spatial data sampling, clustering, application in molecular biology.

1. Introduction

Numerous applications require the management of geometric, geographic or *spatial* data, i.e. data related to space. The relevant space may be, e. g., a two-dimensional projection of the surface of the earth in the case of a geographic information system or a 3d-space containing a protein molecule in the case of an application in molecular biology. *Spatial Database Systems (SDBS)* (Gueting 1994) are database systems with additional support for the management of spatial data: they offer spatial data types in their data model and provide spatial access methods for efficient implementations of spatial operations (cf. Brinkhoff et al. 1993, Brinkhoff et al. 1994). Typical objects are points, lines, polygons and spheres with non-spatial attributes like landuse, color or electrostatic charge. While non-spatial attributes usually have atomar values, the values of spatial attributes often have an extension in space.

Both the number and the size of spatial databases are rapidly growing because of the large amount of data obtained from satellite images, X-ray crystallography or other scientific equipment. This growth by far exceeds human capacities to analyze the databases in order to find implicit regularities, rules or clusters hidden in the data. Therefore, automated knowledge discovery becomes more and more important in spatial databases. *Knowledge discovery in databases (KDD)* is the non-trivial extraction of implicit, previously unknown, and potentially useful information from databases (Frawley, Piatetsky-Shapiro & Matheus 1991). So far, most of the KDD methods have been based on relational database systems which are appropriate to handle non-spatial data, but not spatial data.

One of the well-known techniques for KDD is induction. Han, Cai, & Cercone (1993) assume the existence of concept hierarchies in the application domain and uses them to generalize the tuples of a relation into characteristic rules and classification rules. Liu, Han, & Ooi (1993) extend this method for SDBS by adding spatial concept hierarchies and performing spatial induction. However, these hierarchies may not be available in many applications and, if available, they will not be appropriate for all KDD tasks. Therefore, Ng & Han (1994) do not rely on any domain knowledge and explores the applicability of cluster analysis techniques for KDD in SDBS. An algorithm called CLARANS (Clustering Large Applications based on RANdomized Search) is presented, which is both, efficient and effective for databases of some thousand objects.

Ng & Han (1994) assume that all objects to be clustered can reside in main memory at the same time. However, this does not hold for large databases. Furthermore, the runtime of CLARANS is prohibitive on large databases. In general, the issue of interfacing KDD systems with a database management system (DBMS) has received little attention in the KDD literature and many systems are not yet integrated with a DBMS. Matheus, Chan, & Piatetsky-Shapiro (1993) propose an architecture of a KDD system including a DBMS interface and a focusing component. Well-known techniques are, e.g. focusing on a small subset of all tuples or focusing on a subset of all attributes. Agrawal, Imielinski, & Swami (1993) present a set of basic operations for solving different

1. This research was funded by the German Ministry for Research and Technology (BMFT) under grant no. 01 IB 307 B. The authors are responsible for the content of this paper.

KDD tasks and shows how to apply them for efficient *classification*, i.e. finding rules that partition the database into a given set of groups. Good performance even on a large database is obtained by splitting the search space into independent parts, which is possible because different branches of a decision tree may be expanded independently from each other. Holsheimer & Kersten (1994) address the issue of classification in large relational databases. Splitting the given relation into a lot of relatively small binary relations, i.e. focusing on one attribute at a time, Holsheimer & Kersten (1994) always keep the relevant part of the database in main memory.

The task considered in this paper is *class identification*, i.e. the grouping of the objects of the database into meaningful subclasses (c.f. Matheus, Chan, & Piatesky-Shapiro 1993). We show how to integrate CLARANS with a SDBS in order to perform class identification on large spatial databases, which can only partially be loaded into main memory. The key to this integration is the use of a well-known spatial access method, the R*-tree (Beckmann et al. 1990). The R*-tree, designed for supporting spatial queries, provides an efficient interface to a SDBS. The rest of the paper is organized as follows. Chapter 2 gives a brief introduction into CLARANS and discusses its application to large databases. We present a DB interface (chapter 3) and a focusing component (chapter 4) for clustering in large SDBS. We evaluate the proposed method with respect to both efficiency and effectiveness on a protein database (chapter 5) and finish with the conclusion in chapter 6.

2. CLARANS on Large Databases

Clustering techniques are attractive for KDD, because they can find hidden structures in data without using any additional domain knowledge. We have choosen the *k*-medoid method as the basis of our clustering algorithm. In comparison with *k*-mean methods, e.g. ISODATA, *k*-medoid clustering algorithms have the following advantages. First, the method can be used not only on points or vectors for which the mean is defined but also on any objects for which a similarity measure between two objects is given. Second, the *k*-medoid methods are robust to the existence of outliers (i.e. objects that are very far away from the rest of the objects). Last but not least, *k*-medoid based algorithms can handle very large data sets quite efficiently. See (Kaufman & Rousseeuw 1990) for a more detailed comparison of *k*-medoid methods with other clustering methods.

In this paper, we consider the following clustering problem. Given the number of clusters *k*, a *set O of n objects*, and a distance function, *dist:* $O \times O \rightarrow R_0$, find a subset $M \subseteq O$ of *k* representative objects, i.e. the *set of medoids*, such that the average distance of all objects to their medoids is minimized. Each object $o \in O$ is assigned to the closest $m \in M$. In the following, we assume *dist* to be the euclidean distance, which is a natural choice for spatial clustering. We define a function medoid:

medoid: $O \rightarrow M$
medoid(o) $= m_i$, $m_i \in M$, $\forall\, m_j \in M$: $dist(o,m_i) \leq dist(o,m_j)$

Finally, we define the *cluster* of medoid m_i to be the subset of all objects from O with medoid(o) = m_i, i.e. $cluster(m_i) = \{o \mid o \in O \wedge medoid(o) = m_i\}$ and we define a *clustering* to be a set of clusters partitioning O. Let C_O be the set of all possible clusterings of O. The *average distance* of a clustering is used to measure its quality:

average_distance: $C_O \rightarrow R_0^+$

$$average_distance(c) = \sum_{m_i \in M} \sum_{o \in cluster(m_i)} dist(o, m_i) / n$$

The size of the search space C_O is of order $O(n^k)$, so that trying all possible solutions is prohibitive. Kaufman & Rousseeuw (1990) propose a method PAM (Partitioning Around Medoids) addressing this problem. PAM starts from an initial set of medoids and replaces one of these medoids by one of the non-medoids as long as the quality of the resulting clustering is improved. The complexity of PAM is $O(t \bullet k(n-k)^2)$, where *t* is the number of iterations. Thus, it is obvious that PAM is inefficient for large *n*.

Ng & Han (1994) propose a clustering method based on PAM with a new heuristic search strategy. This strategy does not try all possible clusterings, but only a small number of them, which are selected in a random way. In figure 1, we briefly sketch the algorithm CLARANS. O, *k* and *dist* are given as an input. Furthermore, CLARANS requires the parameters *numlocal* (number of locally optimal clusterings to be considered) and *maxneighbor* (number of exchanges of one medoid and one non-medoid to be tried) to control the heuristic search strategy. Ng & Han (1994) suggest to set *numlocal* to 2 and to set *maxneighbor* to the maximum of *1.25%*k(n-k) and* 250. The output consists of a set of *k* clusters.

```
FOR i FROM 1 TO numlocal DO
   ❶ create randomly an initial set of k
   medoids;
   WHILE NOT maxneighbor exchanges tried DO
      ❷ select randomly one of the k medoids
      and one of the n-k non-medoids;
      ❸ calculate the difference of average
      distance implied by exchanging the two
      selected objects;
      IF the difference is less than 0 THEN
         ❹ exchange the selected medoid and
         non-medoid;
   END WHILE;// INNER LOOP
   ❺ calculate the average distance of the
      current clustering;
   IF this distance is less than the
      distance of the best clustering THEN
      remember current clustering as the
      best clustering;
END FOR;// OUTER LOOP
```

figure 1: Algorithm CLARANS

Now, we want to analyse the cost of CLARANS, when applied to a database. Our analysis is based on the following as-

sumptions. Let c be the *average number of objects stored on one page*. The small set of medoids is resident in main memory, while the large set of non medoids has to reside on disk. The I/O-cost heavily dominates the CPU cost. Therefore, we take the number of disk pages to be read as the cost measure, which is a common approach for database systems.

We obtain the following cost for the different operations. The procedure ❷ (see figure 1) has to read just one of the non medoid objects, i.e. one page. For similar reasons procedure ❶ will have to read k pages in the worst case. Procedure ❸ accesses all objects, which means it reads n/c pages. Procedure ❹ only changes the set of medoids and thus does not access to the disk. Procedure ❺ again has to access all objects, i.e. has cost n/c.

Our analysis indicates that the most expensive operation of CLARANS is the calculation of the difference of distance (procedure ❸). Its cost is $O(n)$, i.e. it needs to read all data pages each time. Furthermore, it is called very frequently, depending on n, k and on the distribution of the database objects in space. In chapter 4, we will present focusing techniques addressing the improvement of efficiency for CLARANS.

3. The DBMS Interface

The *DB interface* allows the KDD system to access the database using queries. A simple query for a relational DBMS, e.g., could request some of the attributes of all tuples in a relation with a given attribute value. Typical queries for SDBS are *region queries* (return all objects from the database intersecting a query polygon) and *nearest neighbor queries* (return the object closest to a query object) (Gueting 1994).

Spatial access methods (SAM) have been developed to support efficient processing of such queries. They organize both the relevant space and the objects of the database such that only a minimal subset of the database has to be considered to find the answers of a query. An *approximation* is a simple object with limited complexity preserving the main properties of a complex spatial object. The use of approximations leads to a two step strategy of query processing. First, the filtering step excludes as many objects as possible from the set of answers based on the approximations and returns a set of candidates. Second, the refinement step checks the exact geometry of the candidates in order to decide whether they fulfill the query condition or not.

The most common approximation is the *bounding box* (BB), i.e. the minimal rectilinear rectangle containing a given spatial object. Therefore, most SAM's are designed to manage rectangles (c.f. Gueting 1994 for an overview of SAM's). The R*-tree (Beckmann et al. 1990) is a SAM which is very efficient both for points and rectangles. It is a balanced search tree, similar to the B^+-tree, storing in each node a set of rectangles. Each node of the tree represents a *page*, the unit of secondary storage. Therefore, the number of rectangles per node is constrained by a *lower limit (min)* and an *upper limit (max)* with the goal to obtain a high storage utilization and to keep the number of disk pages to be read for query processing as small as possible. The nodes storing the data objects are called *data pages*, the nodes organizing the data pages are called directory pages. Figure 2 illustrates the data structure of the R*-tree giving an example of 2-dimensional rectangles.

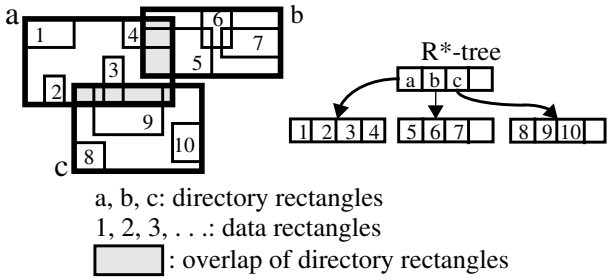

figure 2: R*-tree for two-dimensional rectangles

Data rectangles may overlap, which does not hold for point data. If overlapping data rectangles do not fit into the same page, an overlap of directory pages will occur. This overlap of directory rectangles has to be minimized for efficient query processing. Therefore, the R*-tree uses the following *splitting strategy* when the capacity of a page is exceeded after the insertion of a new object. The page (set of rectangles) is split into two pages (two sets of rectangles) such that the bounding boxes of the resulting pages have minimal area, margin and overlap. This heuristics is both efficient and effective as experiments on realistic data point out (Beckmann et al. 1990).

4. The Focusing Component

The *focusing component* determines which parts of the database are relevant for pattern extraction. In a relational DBS, e.g. one could focus on some attributes of the tuples yielding most information or focus on a randomly drawn sample of all tuples. Finally, the focusing component asks queries to the DB interface obtaining the input for pattern extraction.

According to chapter 2, there are two approaches to improve the efficiency of CLARANS on large SBDS. First, a reduction of the number n of the considered objects will result in a significant speed up, because the calculation of the distance difference is linear. Second, a careful analysis shows that actually not all n objects contribute to the result of the distance difference, such that efficiency can be improved by restricting the access to the relevant objects. We have designed several focusing techniques following these approaches. In this paper, due to the strict page limit, we only present *focusing on representatives* in section 4.1 and a method for determining the number of representatives in section 4.2.

4.1 Focusing on Representatives

We propose to apply CLARANS not to the whole database, but to select a relatively small number of representatives from the database and to apply CLARANS only to these representatives. This is a kind of sampling, a technique common in KDD systems.

Kaufman & Rousseeuw (1990) propose the clustering algorithm CLARA (Clustering LARge Applications) based on sampling. CLARA draws a set of $40 + 2k$ representatives

from the n objects of the database and applies PAM to the representatives only. The method for sampling is as follows:

Assign each object to a unique index in the range [1..*n*]. Draw *40+2k* random numbers from the interval [1..*n*] and return the objects with the respective indices as representatives.

Step (1) is the crucial one. In order to obtain representatives with a good distribution over the geometric space, the indices have to be assigned such that objects with similar indices are close to each other in the geometric space. In other words, we need a method for defining a linear order for objects reflecting their geometric neighborhood as much as possible. Kaufman & Rousseeuw (1990) give no hint how to obtain such an order. The R*-tree, however, offers a solution to this problem. In fact, the R*-tree has been designed to store a SDB such that neighboring objects, which are often requested by the same query, are stored in neighboring pages on the disk.

In the following, we define a method for spatial sampling based on the R*-tree. Let N denote the set of the *natural numbers*, $P \subseteq N$ denote the set of all *page numbers* and $E \subseteq N$ the set of all *entry numbers*, i.e. the relative indices of objects in their page. Let *page* be the function returning the set of objects of the page with a given page number and *object* be the function returning the object of a page with a given entry number:

page: $P \rightarrow 2^O$

object: $2^O \text{ x } E \rightarrow O$

Let a function *random* be defined as *random:* $2^N \times N \rightarrow 2^N$ drawing a set of m random numbers from a given set of numbers, either page numbers or entry numbers.

Kaufman & Rousseeuw (1990) report experiments indicating that *40 + 2k* is a reasonable choice for the number of representatives. However, these experiments have been performed with values for n (≤ 1000) and k (≤ 30) which are not realistic for large databases. Our experimental evaluation in chapter 5 reveals that a reasonable definition for the heuristic parameter *maxneighbor* of CLARANS differs considerably between midsize and large databases. Therefore, we introduce two parameters allowing to scale the number of representatives drawn from a database. Let *np* be the *number of pages* to be considered for sampling, and *ne* be the *number of entries* to be drawn from the same page. A value of *n/c* for *np* and a value of 1 for *ne*, e.g., yields one representative for each page, i.e. *n/c* representatives. In general, the *number r of representatives* can be calculated as $r = np * ne$.

Now, we define the main function *spatial_sampling* with a set of objects, *np* and *ne* as input:

spatial_sampling: $2^O \text{ x } P \text{ x } E \rightarrow 2^O$

spatial_sampling(set_of_objects, np, ne) =

$\{o \in set_of_objects \mid p \in random(P, np) \wedge e \in random(E, ne) \wedge o = object(page(p), e)\}$

An obvious question is, whether it is reasonable to let the R*-tree perform the whole clustering in one step without using CLARANS in a second step. The answer is "no" because of the following reasons: (i) The R*-tree does not allow the user to specify the number k of clusters, it derives k indirectly from n and from the capacity of a page. This k may be inappropriate for a given application and may yield clusterings with a high average distance. (ii) All clusters (i.e. the directory rectangles) have a rectangular shape and, furthermore, these rectangles have to be parallel to the axes of the coordinate system.

One of the goals of the R*-tree is to guarantee high storage utilization, whereas the purpose of CLARANS is to generate clusterings with maximum quality according to an application specific distance function. The R*-tree is efficient for large databases, CLARANS, on the other hand, is inefficient for large values of n and k. Therefore, we propose to combine the good properties of the R*-tree and of CLARANS in the following two step approach: (1) Extract representatives according to *np* and *ne* from the R*-tree. (2) Cluster the representatives using CLARANS and return k medoids.

4.2 Determining the Number of Representatives

We have introduced two parameters allowing to scale the number of representatives drawn from a database: the number of pages and the number of entries. In this section, we will determine values for these parameters yielding a good trade-off between effectiveness and efficiency for the resulting clustering. We use experiments to determine suitable parameters.

In the following experiments, we cluster some 50.000 3-D points from a protein database (c.f. chapter 5 for a description of the application). These points have been clustered with the number of representatives varying from 256 (*np* = 256, *ne* = 1) to 4108 (*np* = 1027, *ne* = 4). Figure 3 and figure 4 depict the resulting effectiveness and efficiency for the varying number of representatives.

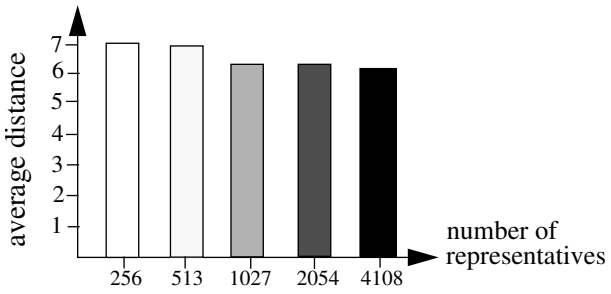

figure 3: Average Distance

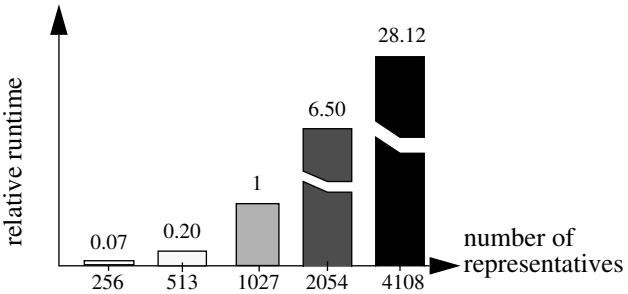

figure 4: Relative Runtime

We observe no significant reduction of the average distance for more than 1027 representatives, whereas the runtime increases more or less linearly with the number of representatives. We conclude that one representative from each data page is an appropriate choice for the number of representatives. Therefore, the experiments reported in chapter 5 will be based on a value of *n/c* for *np* and on a value of 1 for *ne*.

5. Application and Performance Evaluation

We apply the proposed clustering techniques to a large protein database and evaluate their performance in this context. We introduce the protein database (section 5.1) and evaluate focusing on representatives with respect to effectiveness and efficiency (section 5.2).

5.1 BIOWEPRO - a SDBS for Protein-Protein Docking

Proteins are biomolecules consisting of some hundreds to some thousands of atoms. Their mode of operation lies in the interaction with other biomolecules, for example proteins, DNA or smaller partner molecules. These interactions are performed by connecting the partner molecules, and are therefore called *docking*.

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. Therefore, a database system for protein-protein docking has to process queries for proteins with similar or complementary surfaces.

In the BIOWEPRO (Biomolecular Interactions of Proteins) project (Ester et al. 1995), we are developing a SDBS to support protein-protein docking. We use the crystallographically determined atom coordinates of proteins and protein complexes from the Brookhaven Protein Data Bank (PDB 1994), presently containing some 3,000 proteins. Each protein has a triangulated surface consisting of some 10,000 3D points. For each point on the protein surface, several geometric and physicochemical features are computed. The *solid angle (SA),* e.g., (Connolly 1986) is a geometric feature describing the degree of convexity or concavity of the surface in the neighborhood of the considered point.

The search for similar protein surfaces is not performed at the level of surface points, but at the level of surface segments, resulting in a significant reduction of the number of both, the objects in the database and the answers to a given query. A *segment* is defined as a set of neighboring surface points with similar non-spatial 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. Thus, the KDD task is to find a segmentation of protein surfaces supporting the processing of docking queries.

5.2 Evaluation of Focusing on Representatives

In this section, we present experimental results from the BIOWEPRO database evaluating the technique of focusing on representatives with respect to efficiency and effectiveness. Our measure of effectiveness is the average distance of the resulting clusterings. Efficiency is measured by the CPU runtime of the whole processing. All experiments have been run on an HP 9000/735 workstation.

We use the protein hemoglobin for our experiments, because it is one of the largest objects in the database. The surface of hemoglobin consists of 50,559 points, and for each of these points we store the 3D-coordinates along with the value of SA. The number of clusters is set to 10 and *numlocal* set to 2. Ng & Han (1994) has proposed a heuristic to determine *maxneighbor*, i.e. *maxneighbor = max(250, k(n-k)*1.25%).* This proposal is based on experiments with n varying from 100 to 3000 and k varying from 5 to 20. Our experiments indicate that this heuristic is not appropriate for values of *n* varying between 10,000 and 100,000. The resulting values for *maxneighbor* are so large, that the gain in effectiveness is very small compared to the increase of runtime. *maxneighbor = 0.1*k(n-k)*1.25%* seems to be a good trade-off between effectiveness and efficiency in our case. In the first set of experiments, we directly apply CLARANS on hemoglobin with *maxneighbor* varying from 250 over 500 to 1,000. In the second set of experiments, we use "focusing on representatives". We obtain 1,027 representatives out of the 50,559 points for hemoglobin, and CLARANS is then applied to this set of representatives.

The results on effectiveness in terms of the average distance are presented in figure 5. Using the focusing technique, we observe a decrease of effectiveness ranging from 1.5% to 3.2% compared to clustering without focus. Figure 6 depicts the results of the comparison of efficiency.

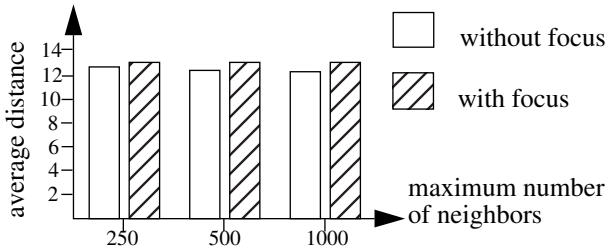

figure 5: Comparison of Effectiveness

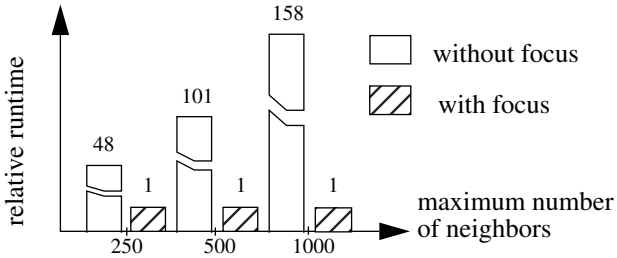

figure 6: Comparison of Efficiency

Focusing improves the efficiency by a factor ranging from 48 to 158 in comparison to clustering without focus.

To conclude, focusing improves efficiency by a factor of 48 to 158, whereas the loss of effectiveness is only 1.5% to 3.2%.

6. Conclusion

We use the clustering algorithm CLARANS (Ng & Han 1994) for class identification in SDBS. We have presented a DB interface and a focusing component supporting clustering in large SDBS. The DB interface of our KDD system is based on a spatial access method, the R*-tree. It clusters the objects according to their spatial neighborhood and supports efficient processing of spatial queries. Focusing on representative objects significantly reduces the number of objects to be clustered. Thus, we achieve a considerable speed-up for clustering in large databases. Our DB interface provides an effective way of determining these representatives.

We have applied the proposed clustering techniques to real data from a large protein database. In general, our approach can be used in any database where spatial access methods are provided. In (Ng & Han 1994) an application of CLARANS to a geographic database consisting of point data is described. We are going to apply the proposed methods to a database managing CAD parts with non-spatial features. Our goal is to identify classes of these parts based on their features.

We have performed an evaluation of focusing on representatives on the protein database. Focusing reduces the runtime for clustering by factors, whereas the decrease of effectiveness is very small. Thus, focusing on representatives offers a very good trade-off between efficiency and effectivity.

Future research will have to consider the following issues. CLARANS randomly selects two objects to be exchanged and does not consider any alternatives if the exchange results in a reduction of the average distance of the clustering. Heuristic strategies for selection should reduce the huge size of the search space and thus improve the efficiency of pattern extraction. So far, we have created crisp clusterings, i.e. each object has been assigned to a unique cluster. However, due to the spatial nature of the objects, it is possible that an object intersects the area of two clusters at the same time. Therefore, for SDBS fuzzy clustering techniques seem to be more appropriate than crisp clustering methods and will be explored. Besides clustering techniques, there are other methods for class identification in SDBS, e.g. region growing, which should also be supported by a KDD system. It is an open question, how these methods compare with clustering methods.

Acknowledgments

We thank Thomas Seidl for engaging and fruitful discussions on the subject of this paper and for his support in the performance evaluation on the BIOWEPRO data.

References

Agrawal R.; Imielinski T.; and Swami A. 1993. Database Mining: A Performance Perspective. *IEEE Transactions on Knowledge and Data Engineering* 5(6):914-925.

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, 1990, pp. 322-331.

Brinkhoff T.; Horn H.; Kriegel H.-P.; and Schneider R. 1993. A Storage and Access Architecture for Efficient Query Processing in Spatial Database Systems. Proc. 3rd Int. Symp. on Large Spatial Databases (SSD '93), Singapore, Lecture Notes in Computer Science, Springer, Vol. 692, pp. 357-376.

Brinkhoff T.; Kriegel H.-P.; Schneider R.; and Seeger B. 1994 Efficient Multi-Step Processing of Spatial Joins, Proc. ACM SIGMOD Int. Conf. on Management of Data, Minneapolis, MN, 1994, 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.; Seidl T.; and Xu X. 1995 Shape-based Retrieval of Complementary 3D Surfaces in Protein Databases, (in German), Proc. GI Conf. on Database Systems for Office Automation, Engineering, and Scientific Applications, Springer.

Frawley W.J.; Piatetsky-Shapiro G.; and Matheus J. 1991 Knowledge Discovery in Databases: An Overview. Knowledge Discovery in Databases, 1-27. AAAI Press, Menlo Park.

Gueting R.H. 1994. An Introduction to Spatial Database Systems. *The VLDB Journal* 3(4): 357-399.

Han J.; Cai Y.; and Cercone N. 1993. Data-driven Discovery of Quantitative Rules in Relational Databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(1): 29-40.

Holsheimer M., and Kersten M.L. 1994. Architectural Support for Data Mining, Proc. AAAI Workshop on Knowledge Discovery in Databases, 217-228. Seattle, Washington.

Kaufman L., and Rousseeuw P.J. 1990. *Finding Groups in Data: an Introduction to Cluster Analysis*. John Wiley & Sons.

Lu W.; Han J.; and Ooi B.C. 1993 Discovery of General Knowledge in Large Spatial Databases, Proc. Far East Workshop on Geographic Information Systems, 275-289. Singapore.

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.

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, 144-155. Santiago, Chile.

Protein Data Bank, 1994. *Quarterly Newsletter No. 70,* Brookhaven National Laboratory, Upton, NY.