W. K. Ng et al. (eds.): PAKDD 2006, LNAI 3918, pp. 179–188, 2006.

Parallel Density-Based Clustering of Complex Objects

Stefan Brecheisen, Hans-Peter Kriegel, and Martin Pfeifle

Institute for Informatics, University of Munich
{brecheis,kriegel,pfeifle}@dbs.ifi.lmu.de

Abstract. In many scientific, engineering or multimedia applications, complex distance functions are used to measure similarity accurately. Furthermore, there often exist simpler lower-bounding distance functions, which can be computed much more efficiently. In this paper, we will show how these simple distance functions can be used to parallelize the density-based clustering algorithm DBSCAN. First, the data is partitioned based on an enumeration calculated by the hierarchical clustering algorithm OPTICS, so that similar objects have adjacent enumeration values. We use the fact that clustering based on lower-bounding distance values conservatively approximates the exact clustering. By integrating the multi-step query processing paradigm directly into the clustering algorithms, the clustering on the slaves can be carried out very efficiently. Finally, we show that the different result sets computed by the various slaves can effectively and efficiently be merged to a global result by means of cluster connectivity graphs. In an experimental evaluation based on real-world test data sets, we demonstrate the benefits of our approach.

1 Introduction

Density-based clustering algorithms like DBSCAN [1] are based on ε-range queries for each database object. Thereby, each range query requires a lot of distance calculations. When working with complex objects, e.g. trees, point sets, and graphs, often complex time-consuming distance functions are used to measure similarity accurately. As these distance calculations are the time-limiting factor of the clustering algorithm, the ultimate goal is to save as many as possible of these complex distance calculations.

Recently an approach was presented for the efficient density-based clustering of complex objects [2]. The core idea of this approach is to integrate the multi-step query processing paradigm directly into the clustering algorithm rather than using it "only" for accelerating range queries. In this paper, we present a sophisticated parallelization of this approach. Similar to the area of join processing where there is an increasing interest in algorithms which do not assume the existence of any index structure, we propose an approach for parallel DBSCAN which does not rely on the pre-clustering of index structures.

First, the data is partitioned according to the clustering result carried out on cheaply computable distance functions. The resulting approximated clustering

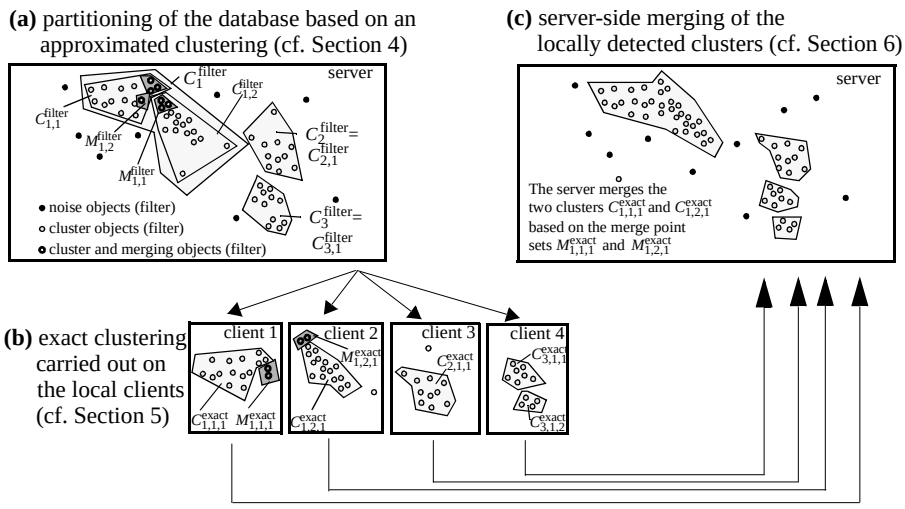


Fig. 1. Basic idea of parallel density-based clustering.

conservatively approximates the exact clustering. The objects of the conservative cluster approximations are then distributed onto the available slaves in such a way that each slave has to cluster the same amount of objects, and that the objects to be clustered are close to each other. Note that already at this early stage, we can detect some noise objects which do not have to be transmitted to the local clients. In addition to the objects to be clustered by a client, we send some filter merge points to this client. These filter merge points are also determined based on approximated distance functions. (cf. Figure 1a).

Second, each client carries out the clustering independently of all the other clients. No further communication is necessary throughout this second step. The presented local clustering approach also takes advantage of the approximating lower-bounding distance functions. The detected clusters and the detected exact merge point sets are then transmitted to the server (cf. Figure 1b).

Finally, the server determines the correct clustering result by merging the locally detected clusters. This final merging step is based on the exact merge points detected by the clients. Based on these merge points, cluster connectivity graphs are created. In these graphs, the nodes represent the locally detected clusters. Two local clusters are connected by an edge if a merge point of one cluster is a core object in the other cluster (cf. Figure 1c).

The remainder of this paper is organized as follows. In Section 2, we shortly sketch the work from the literature related to our approach. In Sections 3, 4 and 5, we explain the server-side partitioning algorithm, the client-side clustering algorithm, and the server-side merging of the results from the clients, respectively. In Section 6, we present a detailed experimental evaluation based on real world test data sets. We close the paper in Section 7 with a short summary and a note on future work.

2 Related Work

Complex Object Representations. Complex object representations, like high-dimensional feature vectors [3], vector sets [4], trees or graphs [5], are helpful to model real world objects accurately. The similarity between these complex object representations is often measured by means of expensive distance function, e.g. the edit distance. For a more detailed survey on this topic, we refer the interested reader to [6].

Clustering. Given a set of objects with a distance function on them, an interesting data mining question is, whether these objects naturally form groups (called clusters) and what these groups look like. Data mining algorithms that try to answer this question are called clustering algorithms. For a detailed overview on clustering, we refer the interested reader to [7].

Density-Based Clustering. Density based clustering algorithms apply a local cluster criterion to detect clusters. Clusters are regarded as regions in the data space in which the objects are dense, and which are separated by regions of low object density (noise). One of the most prominent representatives of this clustering paradigm is DBSCAN [1].

Density-Based Clustering of Complex Objects. In [2] a detailed overview can be found describing several approaches for the efficient density-based clustering of complex object. Furthermore, in [2] a new approach was introduced which performs expensive exact distance computations only when the information provided by simple distance computations is not enough to compute the exact clustering. In Section 4, we will use an adaption of this approach for the efficient clustering on the various slaves.

Parallel Density-Based Clustering of Complex Objects. To the best of our knowledge there does not exist any work in this area.

3 Server-Side Data Partitioning

The key idea of density-based clustering is that for each object of a cluster the neighborhood of a given radius ε has to contain at least a minimum number of *MinPts* objects, i.e. the cardinality of the neighborhood has to exceed a given threshold. An object p is called *directly density-reachable* from object q w.r.t. ε and *MinPts* in a set of objects D, if $p \in \mathcal{N}_\varepsilon(q))$ and $|\mathcal{N}_\varepsilon(q)| \geq$ *MinPts*, where $\mathcal{N}_\varepsilon(q)$ denotes the subset of D contained in the ε-neighborhood of q. The condition $|\mathcal{N}_\varepsilon(q)| \geq$ *MinPts* is called the *core object condition*. If this condition holds for an object q, then we call q a *core object*. Other objects can be directly density-reachable only from core objects. An object p is called *density-reachable* from an object q w.r.t. ε and *MinPts* in the set of objects D, if there is a chain of objects $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$, such that $p_i \in D$ and p_{i+1} is directly density-reachable from p_i w.r.t. ε and *MinPts*. Object p is *density-connected* to object q w.r.t. ε and *MinPts* in the set of objects D, if there is an object $o \in D$ such that both p and q are density-reachable from o. Density-reachability is the transitive closure of direct density-reachability and is not necessarily symmetric. On the other hand, density-connectivity is a symmetric relation.

DBSCAN. A flat density-based *cluster* is defined as a set of density-connected objects which is maximal w.r.t. density-reachability. Thus a cluster contains not only core objects but also border objects that do not satisfy the core object condition. The *noise* is the set of objects not contained in any cluster.

OPTICS. While the partitioning density-based clustering algorithm DBSCAN can only identify a flat clustering, the newer algorithm OPTICS [8] computes an ordering of the points augmented by the so-called *reachability-distance*. The reachability-distance basically denotes the smallest distance of the current object q to any core object which belongs to the current cluster and which has already been processed. The clusters detected by DBSCAN can also be found in the OPTICS ordering when using the same parametrization, i.e. the same ε and *MinPts* values. For an initial clustering with OPTICS based on the lower-bounding filter distances the following two lemmas hold.

Lemma 1. *Let $C_1^{exact}, \ldots, C_n^{exact}$ be the clusters detected by OPTICS based on the exact distances, and let $C_1^{filter}, \ldots, C_m^{filter}$ be the clusters detected by OPTICS based on the lower-bounding filter distances. Then the following statement holds:*

$$\forall i \in \{1, \ldots, n\} \exists j \in \{1, \ldots, m\} : C_i^{exact} \subseteq C_j^{filter}.$$

Proof. Let $N_\varepsilon^{filter}(o)$ denote the ε-neighborhood of o according to the filter distances, and let $N_\varepsilon^{exact}(o)$ denote the ε-neighborhood according to the exact distances. Due to the lower-bounding filter property $N_\varepsilon^{exact}(o) \subseteq N_\varepsilon^{filter}(o)$ holds. Therefore, each object o which is a core object based on the exact distances is also a core object based on the lower-bounding filter distances. Furthermore, each object p which is directly density-reachable from o according to the exact distances is also directly density-reachable according to the filter functions. Induction on this property shows that if p is density-reachable from o based on the exact distances, it also holds for the filter distances. Therefore, all objects which are in one cluster according to the exact distances are also in one cluster according to the approximated distances.

Lemma 2. *Let $noise^{exact}$ denote the noise objects detected by OPTICS based on the exact distances and let $noise^{filter}$ denote the noise objects detected by OPTICS based on the lower-bounding filter distances. Then the following statement holds:*

$$noise^{filter} \subseteq noise^{exact}.$$

Proof. An object p is a noise object if it is not included in the ε-neighborhood of any core object. Again, let $N_\varepsilon^{filter}(o)$ and $N_\varepsilon^{exact}(o)$ denote the ε-neighborhood of o according to the filter distances and the exact distances, respectively. Due to the lower-bounding filter property $N_\varepsilon^{exact}(o) \subseteq N_\varepsilon^{filter}(o)$ holds. Therefore, if $p \notin N_\varepsilon^{filter}(o)$, it cannot be included in $N_\varepsilon^{exact}(o)$, proving the lemma.

Both Lemma 1 and Lemma 2 are helpful to partition the data onto the different slaves. Lemma 1 shows that exact clusters are conservatively approximated by the clusters resulting from a clustering on the lower-bounding distance

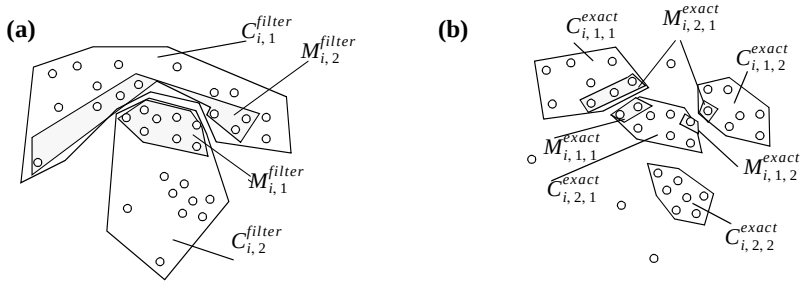
functions. On the other hand, Lemma 2 shows that exact noise is progressively approximated by the set of noise objects resulting from an approximated clustering. For this reason, noise objects according to the filter distances do not have to be transmitted to the slaves, as we already know that they are also noise objects according to the exact distances. All other N objects have to be refined by the P available slave processors. Let $C_1^{filter}, \ldots, C_m^{filter}$ be the approximated clusters resulting from an initial clustering with OPTICS. In this approach, we assign $P_{slave} = \sum_{i=1}^{m} |C_i^{filter}|/P$ objects to each of the P slaves. We do this partitioning online while carrying out the OPTICS algorithm. At each time during the clustering algorithm, OPTICS knows the slave j having received the smallest number L_j of objects up to now, i.e. the client j has the highest free capacity $C_j = P_{slave} - L_j$. OPTICS stops the current clustering at two different event points: In the first case, a cluster C_i^{filter} of cardinality $|C_i^{filter}| \leq C_j$ was completely determined. This cluster is sent to the slave j. In the second case, OPTICS determined C_j more points belonging to the current cluster C_i^{filter}. These points are grouped together to a filter cluster $C_{i,j}^{filter}$. Then, we transmit the cluster $C_{i,j}^{filter}$ along with the filter merge points $M_{i,j}^{filter}$ to the slave j. The set $M_{i,j}^{filter}$ can be determined throughout the clustering of the set $C_{i,j}^{filter}$ and can be defined as follows.

Definition 1 (filter merge points). *Let C_i^{filter} be a cluster which is split during an OPTICS run into n clusters $C_{i,1}^{filter}, \ldots, C_{i,n}^{filter}$. Then, the* filter merge points *$M_{i,j}^{filter}$ for a partial filter cluster $C_{i,j}^{filter}$ are defined as follows: $M_{i,j}^{filter} = \{q \in C_i^{filter} - C_{i,j}^{filter} \mid \exists p \in C_{i,j}^{filter} : q$ is directly density-reachable from $p\}$.*

The filter merge points $M_{i,j}^{filter}$ are necessary in order to decide whether objects $o \in C_{i,j}^{filter}$ are core objects. Furthermore, a subset $M_{i,j}^{exact} \subseteq M_{i,j}^{filter}$ is used to merge exact clusters in the final merge step (cf. Section 5).

4 Client-Side Clustering

Each of the filter clusters $C_{i,j}^{filter}$ is clustered independently on the exact distances by the assigned slave j. For clustering these filter clusters, we adapt the approach presented in [2], so that it can also handle the additional merge points $M_{i,j}^{filter}$. The main idea of the client-side clustering approach is to carry out the range queries based on the lower-bounding filter distances instead of using the expensive exact distances. Thereto, we do not use the simple seedlist of the original DBSCAN algorithm, but we use a list of lists, called *Xseedlist*. The *Xseedlist* consists of an ordered *object list OL*. Each entry $(o, T, PL) \in OL$ contains a flag T indicating whether $o \in C_{i,j}^{filter}$ $(T = \mathrm{C})$ or $o \in M_{i,j}^{filter}$ $(T = \mathrm{M})$. Each entry of the *predecessor list PL* consists of the following information: a *predecessor* o_p of o, which is a core object already added to the current cluster, and the *predecessor distance*, which is equal to the filter distance $d_f(o, o_p)$ between the two objects.



During the server-side *partitioning step*, the cluster C_i^{filter} is split into two clusters $C_{i,1}^{filter}$ and $C_{i,2}^{filter}$ with their corresponding merge point sets.

During the server-side *merge step*, the cluster $C_{i,1,1}^{exact}$, $C_{i,1,2}^{exact}$, and $C_{i,2,1}^{exact}$ are merged based on their exact merge point sets to a cluster $C_{i,1}^{exact} \subseteq C_i^{filter}$. Furthermore, there exists a cluster $C_{i,2}^{exact} = C_{i,2,2}^{exact} \subseteq C_i^{filter}$.

Fig. 2. Server-side partitioning step (a) and merge step (b).

The result of the extended DBSCAN algorithm is a set of exact clusters $C_{i,j,l}^{exact} \subseteq C_{i,j}^{filter}$ along with their additional exact merge points $M_{i,j,l}^{exact} \subseteq M_{i,j}^{filter}$. To expand a cluster $C_{i,j,l}^{exact}$ we take the first element (o, T, PL) from OL and set o_p to the nearest predecessor object in PL.

Let us first assume that $T = \text{C}$ holds. If $PL = \text{NIL}$ holds, we add o to $C_{i,j,l}^{exact}$, delete o from OL, carry out a range query around o, and try to expand the cluster $C_{i,j,l}^{exact}$. If $PL \neq \text{NIL}$ holds, we compute $d_o(o, o_p)$. If $d_o(o, o_p) \leq \varepsilon$, we proceed as in the case where $PL = \text{NIL}$ holds. If $d_o(o, o_p) > \varepsilon$ and length of $PL > 1$ hold, we delete the first entry from PL. If $d_o(o, o_p) > \varepsilon$ and length of $PL = 1$ hold, we delete o from OL. Iteratively, we try to expand the current cluster $C_{i,j,l}^{exact}$ by examining the first entry of OL until OL is empty.

Let us now assume that $T = \text{M}$ holds. If $PL = \text{NIL}$ holds, we add o to $M_{i,j,l}^{exact}$, delete o from OL, and try to expand the exact merge point set $M_{i,j,l}^{exact}$. If $PL \neq \text{NIL}$ holds, we compute $d_o(o, o_p)$. If $d_o(o, o_p) \leq \varepsilon$, we proceed as in the case where $PL = \text{NIL}$ holds. If $d_o(o, o_p) > \varepsilon$ and length of $PL > 1$ hold, we delete the first entry from PL. If $d_o(o, o_p) > \varepsilon$ and length of $PL = 1$ hold, we delete o from OL. Iteratively, we try to expand the current exact merge point set $M_{i,j,l}^{exact}$ by examining the first entry of OL until OL is empty.

5 Server-Side Merging

Obviously, we only have to carry out the merge process for those clusters C_i^{filter} which were split in several clusters $C_{i,j}^{filter}$. The client detects that each of these clusters $C_{i,j}^{filter}$ contains t clusters $C_{i,j,1}^{exact}, \ldots, C_{i,j,t}^{exact}$. Note that t can also be equal to 0, i.e. no exact cluster is contained in the cluster $C_{i,j}^{filter}$. For each of the t exact clusters $C_{i,j,l}^{exact}$ there also exists a corresponding set of exact merge points $M_{i,j,l}^{exact} \subseteq M_{i,j}^{filter}$ (cf. Figure 2) defined as follows.

Definition 2 (exact merge points). *Let $C_{i,j}^{filter}$ be a cluster to be refined on the slave with the corresponding merge point set $M_{i,j}^{filter}$. Let $C_{i,j,l}^{exact} \subseteq C_{i,j}^{filter}$ be an exact cluster determined during the client-side refinement clustering. Then, we determine the set $M_{i,j,l}^{exact} \subseteq M_{i,j}^{filter}$ of* exact merge points *where $M_{i,j,l}^{exact} = \{q \in M_{i,j}^{filter} \mid \exists p \in C_{i,j,l}^{exact} : q$ is directly density-reachable from $p\}$.*

Based on these exact merge point sets and the exact clusters, we can define a "cluster connectivity graph".

Definition 3 (cluster connectivity graph). *Let C_i^{filter} be a cluster which was refined on one of the s different slaves. Let $C_{i,j,l}^{exact} \subseteq C_{i,j}^{filter} \subseteq C_i^{filter}$ be an exact cluster determined by slave j along with the corresponding merge point sets $M_{i,j,l}^{exact} \subseteq M_{i,j}^{filter}$. Then a graph $G_i = (V_i, E_i)$ is called a* cluster connectivity graph *for C_i^{filter} iff the following statements hold:*

- $V_i = \{C_{i,1,1}^{exact}, \ldots, C_{i,1,n_1}^{exact}, \ldots, C_{i,s,1}^{exact}, \ldots, C_{i,s,n_s}^{exact}\}$.
- $E_i = \{(C_{i,j,l}^{exact}, C_{i,j',l'}^{exact}) \mid \exists p \in M_{i,j,l}^{exact} : p \in C_{i,j',l'}^{exact} \wedge p$ *is a core point*$\}$.

Note that two clusters $C_{i,j,l}^{exact}$ and $C_{i,j',l'}^{exact}$ from the same slave $j = j'$ are never connected by an edge. Such a connection of the two clusters would already have taken place throughout the refinement clustering on the slave j. Based on the connectivity graphs G_i for the approximated clusterings C_i^{filter}, we can determine the *database connectivity graph.*

Definition 4 (database connectivity graph). *Let C_i^{filter} be one of n approximated clusters along with the corresponding cluster connectivity graph $G_i = (V_i, E_i)$. Then we call $G = (\bigcup_{i=1}^{n} V_i, \bigcup_{i=1}^{n} E_i)$ the* database connectivity graph.

The database connectivity graph is nothing else but the union of the connectivity graphs of the approximated clusters. Based on the above definition, we state the central lemma of this paper.

Lemma 3. *Let G be the database connectivity graph. Then the determination of all maximal connected subgraphs of G is equivalent to a DBSCAN clustering carried out on the exact distances.*

Proof. For each object o the client-side clustering determines correctly, whether it is a core object, a border object, or a noise object. Note, that we assign a border object which is directly density-reachable from core objects of different clusters redundantly to all of these clusters. Therefore, the only remaining issue is to show that two core objects which are directly density-reachable to each other are in the same maximal connected subgraph. By induction, according to the definition of density-reachability, two clusters then contain the same core objects. Obviously, two core objects o_1 and o_2 are directly density-reachable if they are either in the same exact cluster $C_{i,j,l}^{exact}$ or if $o_1 \in C_{i,j,l}^{exact}$ and $o_2 \in M_{i,j,l} exact$ resulting in an edge of the database connectivity graph. Therefore, depth-first traversals through all of the connectivity graphs G_i corresponding to a filter cluster C_i^{filter} create the correct clustering result where each subgraph corresponds to one cluster.

6 Experimental Evaluation

In this section, we present a detailed experimental evaluation based on real-world data sets. We used CAD data represented by 81-dimensional feature vectors [3] and vector sets where each element consists of 7 6D vectors [4]. Furthermore, we used graphs [5] to represent image data. The used distance functions can be characterized as follows: (i) The exact distance computations on the graphs are very expensive. On the other hand, the filter is rather selective and can efficiently be computed. (ii) The exact distance computations on the feature vectors and vector sets are also very expensive as normalization aspects for the CAD objects are taken into account [4, 3]. As a filter for the feature vectors we use their Euclidean norms [9] which is not very selective, but can be computed very efficiently. The filter used for the vector sets is more selective than the filter for the feature vectors, but also computationally more expensive. If not otherwise stated, we used 3,000 complex objects from each data set.

The original OPTICS and DBSCAN algorithms, their extensions introduced in this paper, and the used filter and exact distances functions were implemented in Java 1.4. The experiments were run on a workstation with a Xeon 2.4 GHz processor and 2 GB main memory. All experiments were run sequentially on one computer. Thereby, the overall time for the client-side clustering is determined by the slowest slave. If not otherwise stated, we chose an ε-parameter yielding as many flat clusters as possible, and the *MinPts*-parameter was set to 5.

Characteristics of the partitioning step. Figure 3 compares the number of merge points for different split techniques applied to filter clusters. As explained in Section 3, we split a filter cluster during the partitioning step along the ordering produced by OPTICS. Note that OPTICS always walks through a cluster by visiting the densest areas first. Figure 3 shows that this kind of split strategy yields considerably less merge points than a split strategy which arbitrarily groups objects from a filter cluster together. Thus, the figure proves the good clustering properties of our metric space filling curve OPTICS.

Dependency on the Number of Slaves. Figure 4 shows the absolute runtimes of our parallel DBSCAN approach dependent on the number of available slaves for the vector sets and for the graph dataset. The figure shows the accumulated times after the partitioning, client-side clustering, and the merge step. The partitioning times also include simulated communication times for the transfer of the objects

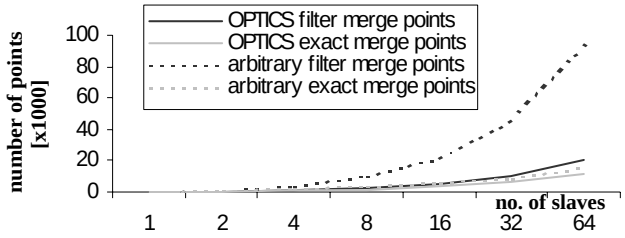
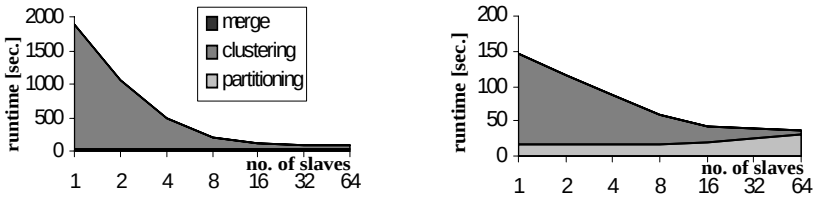


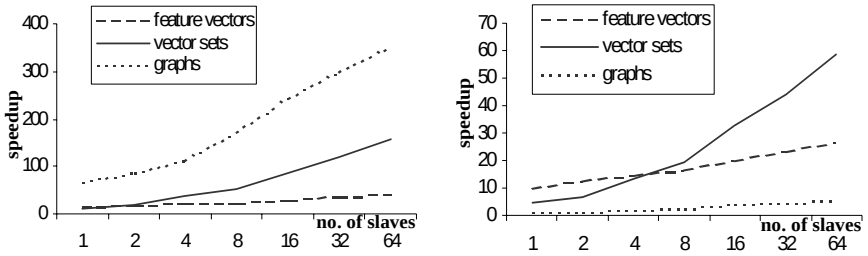
Fig. 3. Number of merge points w.r.t. a varying number of slaves for the graph dataset.



(a) Vector sets.

(b) Graphs.

Fig. 4. Absolute runtimes w.r.t. a varying number of slaves.



(a) Speedup w.r.t. DBSCAN based on a full table scan.

(b) Speedup w.r.t. DBSCAN based on the traditional multi-step query processing paradigm.

Fig. 5. Overall speedup w.r.t. a varying number of slaves.

to the slaves in a 100 Mbit LAN. No communication costs arise from the client-side clustering step, as each client already received all needed filter merge points. A growing number of slaves leads to a significant speedup of the client-side clustering. A lower bound of the achievable total runtime is given by the time needed for the initial partitioning step. It is worth to note the time needed for the final merging step is negligible even for a high number of slaves. Although the number of exact merge points grows with an increasing number of slaves (cf. Figure 3), the merge step remains cheap.

Speedup. Finally, Figure 5 depicts the speedup achieved by our new parallel DBSCAN approach based on a server-side partitioning with OPTICS. We compared this approach to a DBSCAN approach based on a full table scan and compared to a DBSCAN approach based on the traditional multi-step query processing paradigm. The figure shows that for the feature vectors we achieve a speedup of one order of magnitude already when only one slave is available. In the case of the graph dataset we have a speedup of 67 compared to DBSCAN based on a full table scan. These results demonstrate the suitability of the client-side clustering approach. For the vector sets the benefits of using several slaves can clearly be seen. For instance, our approach achieves a speedup of 4 for one slave and a speedup of 20 for eight slaves compared to DBSCAN based on traditional multi-step range queries.

7 Conclusions

In this paper, we applied the novel concept of using efficiently computable lower-bounding distance functions for the parallelization of data mining algorithms to the density-based clustering algorithm DBSCAN. For partitioning the data, we used the hierarchical clustering algorithm OPTICS as a kind of space filling curve for general metric objects, which provides the foundation for a fair and suitable partitioning strategy. We showed how the local clients can carry out their clustering efficiently by integrating the multi-step query processing paradigm directly into the clustering algorithm. Based on the concept of merge points, we constructed a global cluster connectivity graph from which the final clustering result can easily be derived. In the experimental evaluation, we demonstrated that our new approach is able to efficiently cluster metric objects. We showed that if several slaves are available, the benefits achieved by the full computational power of the slaves easily outweigh the additional costs of partitioning and merging by the master. In our future work, we will demonstrate that also other data mining algorithms can beneficially be parallelized based on lower-bounding distance functions.

References

1. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: Proc. 2nd Int. Conf. on Knowledge Discovery and Data Mining (KDD'96), Portland, OR. (1996) 291–316
2. Brecheisen, S., Kriegel, H.P., Pfeifle, M.: "Efficient Density-Based Clustering of Complex Objects". In: Proc. 4th IEEE Int. Conf. on Data Mining (ICDM'04), Brighton, UK. (2004) 43–50
3. Kriegel, H.P., Kröger, P., Mashael, Z., Pfeifle, M., Pötke, M., Seidl, T.: "Effective Similarity Search on Voxelized CAD Objects". In: Proc. 8th Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan. (2003) 27–36
4. Kriegel, H.P., Brecheisen, S., Kröger, P., Pfeifle, M., Schubert, M.: "Using Sets of Feature Vectors for Similarity Search on Voxelized CAD Objects". In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'03), San Diego, CA. (2003) 587–598
5. Kriegel, H.P., Schönauer, S.: "Similarity Search in Structured Data". In: Proc. 5th Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'03), Prague, Czech Republic. (2003) 309–319
6. Kailing, K.: New Techniques for Clustering Complex Objects. PhD thesis, Institute for Computer Science, University of Munich (2004)
7. Jain, A.K., Murty, M.N., Flynn, P.J.: "Data Clustering: A Review". ACM Computing Surveys **31(3)** (1999) 265–323
8. Ankerst, M., Breunig, M.M., Kriegel, H.P., Sander, J.: "OPTICS: Ordering Points to Identify the Clustering Structure". In: Proc. ACM SIGMOD Int. Conf. on Management of Data (SIGMOD'99), Philadelphia, PA. (1999) 49–60
9. Fonseca, M.J., Jorge, J.A.: "Indexing High-Dimensional Data for Content-Based Retrieval in Large Databases". In: Proc. 8th Int. Conf. on Database Systems for Advanced Applications (DASFAA'03), Kyoto, Japan. (2003) 267–274