Efficient Anytime Density-based Clustering

Son T. Mai * Xiao He * Jing Feng * Christian Böhm *

Abstract
Many clustering algorithms suffer from scalability problems on massive datasets and do not support any user interaction during runtime. To tackle these problems, anytime clustering algorithms are proposed. They produce a fast approximate result which is continuously refined during the further run. Also, they can be stopped or suspended anytime and provide an answer. In this paper, we propose a novel anytime clustering algorithm based on the density-based clustering paradigm. Our algorithm called A-DBSCAN is applicable to very high dimensional databases such as time series, trajectory, medical data, etc. The general idea of our algorithm is to use a sequence of lower-bounding functions (LBs) of the true similarity measure to produce multiple approximate results of the true density-based clusters. A-DBSCAN operates in multiple levels w.r.t. the LBs and is mainly based on two algorithmic schemes: (1) an efficient distance update scheme which restricts distance calculations to core-objects at each level of the LBs; (2) a local re-clustering scheme which restricts update operations to the relevant objects only. Extensive experiments demonstrate that A-DBSCAN acquires very good clustering results at very early stages of execution thus saves a large amount of computational time. Even if it runs to the end, A-DBSCAN is still orders of magnitude faster than DBSCAN.

Keywords. Anytime Clustering, Density-based Clustering, Lower-bounding distance

1 Introduction
Clustering is the task of assigning unlabeled objects into groups called clusters such that the similarity of objects within a group is maximized, and the similarity of objects between different groups is minimized. It plays a vital role for statistical data analysis in many fields including data mining, machine learning, pattern recognition, image analysis, information retrieval, etc. Although there are a vast amount of clustering algorithms proposed in literature, most of them work in a batch scheme. They only produce a single result, and there is no interaction with end users during their executions.

For large databases, the idea of exploring the results during execution time has been proved to be a very useful approach [14, 16, 17]. The algorithms quickly produce an approximate result which is continuously improved overtime and allow users interactions during runtime. Users can terminate the algorithms anytime whenever they satisfied with existing results to save computation time. Moreover, the final results of these algorithms should be close to that of the batch algorithms. Such algorithms are called anytime algorithms [14, 16, 17] and have recently attracted a lot of research efforts in many fields of data mining such as classification [14] and outlier detection [2]. However, anytime clustering has not been paid enough attention. There are only a few works on anytime clustering algorithms, e.g. I-kMeans [12, 13].

Among various kinds of clustering algorithms such as partitioning methods, hierarchical methods, etc., density-based clustering algorithms have attracted a lot of attention in the data mining community due to many advantages compared with the others [7, 8]. They can detect clusters of arbitrary shapes, do not require the number of clusters to be specified, and are robust to outliers. Besides many others, the density-based notion underlying the algorithm DBSCAN [8] is one of the most successful approaches to clustering with applications in many fields such as neuroscience, meteorology, etc. Many clustering algorithms are successfully proposed based on this notion [1, 7, 8]. However, all of them only work in the batch scheme.

In our paper, we propose for the first time a novel anytime clustering algorithm based on the clustering notion of DBSCAN [8]. Our algorithm called A-DBSCAN is applicable to many high dimensional databases such as time series, trajectory, medical data, etc. The core idea of A-DBSCAN is to use a sequence of lower-bounding functions (LBs) of the true similarity measure to produce multiple approximate results of DBSCAN. LBs are well-studied in the field of database indexing [5]. However, their applications in clustering have gained much less attention. By using LBs as distance measures, we are not only able to approximate the true clustering result but also speed up the algorithm significantly since the LBs often run very fast compared with the original function. A-DBSCAN operates in multiple levels w.r.t. the sequence of LBs. The result of each level is calculated by using the result of previous level. We propose an efficient distance and cluster update scheme from level to level based on theoretical study of the way clusters change under the effect of LBs. Also, some adaptations in the density-based notions of DBSCAN
are made to enhance the clustering quality. Further, an efficient heuristic to choose parameters is also presented. Extensive experiments on real datasets such as time series, trajectories, etc., demonstrate that A-DBSCAN acquires very good clustering results at very early stages of execution thus saving a large amount of runtime. Even if A-DBSCAN is run to the end, it is still 20 times faster than DBSCAN. Such advantages are impressive since A-DBSCAN, as an anytime algorithm, must perform clustering many times compared with only one time of the batch algorithm DBSCAN.

The rest of this paper is organized as follows. In Section 2, we present our anytime clustering algorithm A-DBSCAN. The similarity measure and lower-bounding functions are briefly described in Section 3. Section 4 reports experimental results. Section 5 is dedicated to related work and discussion. Section 6 concludes with a summary and some directions for future research.

2 Anytime Density-based Clustering

2.1 Anytime Clustering Algorithms. An anytime clustering algorithm works by trading execution time for quality of results [16, 17]. Anytime clustering produces a fast approximate result which is then refined during the further run. Users can examine the intermediate clustering results while the algorithm is continuing to produce the finer results at the next levels. According to [16, 17], an anytime clustering algorithm should satisfy some important properties: (1) it should produce good results which are close to the result of the batch algorithm at early stages; (2) the final result should be similar to or better than the batch algorithm; (3) the total cumulative runtime of the algorithm should be only slightly larger than the batch algorithm.

2.2 The Algorithm DBSCAN. In density-based clustering algorithms, clusters are considered as high density areas, separated by low density areas. The key idea of density-based clustering algorithm DBSCAN [8] is that the cardinality of the neighborhood of each object of a cluster has to exceed a predefined threshold.

Given a set of objects $S$, a distance function $d : S \times S \rightarrow R$ and two parameters $\epsilon \in R^+$ and $\mu \in N^+$.

**Definition 2.1.** ($\epsilon$-neighborhood) The $\epsilon$-neighborhood of $p \in S$, denoted as $N_\epsilon(p)$, is defined by $N_\epsilon(p) = \{q \in S : d(p, q) \leq \epsilon\}$.

Each object in $S$ is classified either as core object, border object or noise object.

**Definition 2.2.** (Core object property) An object $p \in S$ is a:

1. Core object, denoted as core$(p)$, iff $|N_\epsilon(p)| \geq \mu$.
2. Border object, denoted as border$(p)$, iff $|N_\epsilon(p)| < \mu$ and $\exists q \in N_\epsilon(p) : |N_\epsilon(q)| \geq \mu$.

Figure 1: The notions of DBSCAN: (a) $q$ is directly density-reachable from $p$; (b) $p$ and $q$ are density-connected; (c) object $a$ (red) is a core object, $b$ (green) is border object, $c$ (black) is noise object.

3. Noise object, denoted as noise$(p)$, iff it is not a core object or a border object.

**Definition 2.3.** (Directly density-reachable) An object $q \in S$ is directly density-reachable from object $p \in S$, denoted as $p \rightarrow q$, iff $|N_\epsilon(p)| \geq \mu$ and $q \in N_\epsilon(p)$.

**Definition 2.4.** (Density-connected) Two objects $p$ and $q \in S$ are density-connected, denoted as $p \bowtie q$, iff there exists a sequence $(x_1, \ldots, x_m)$ of objects such that $\forall x_i : |N_\epsilon(x_i)| \geq \mu$ and $p \bowtie x_1 \bowtie \cdots \bowtie x_m \bowtie q$.

A cluster is defined as a maximal set of density-connected objects and is composed of core objects and border objects. A border object could belong to several clusters depending on the order of objects.

**Definition 2.5.** (Cluster) A subset $C \subseteq S$ is called a cluster iff the two following conditions hold:

1. Maximality: $\forall p \in C, \forall q \in S \setminus C : \lnot p \bowtie q$
2. Connectivity: $\forall p, q \in C : p \bowtie q$

Figure 1 demonstrates some notions of DBSCAN. Interested readers please refer to [8] for more details.

2.3 Anytime Density-based Clustering. Assume that, there exists a set $D$ of $n$ lower-bounding functions $D = \{d_i : S \times S \rightarrow R \mid \forall p, q \in S : d_i(p, q) \leq d(p, q) \wedge d_n(p, q) = d(p, q)\}$. A-DBSCAN works in a sequence of levels from $L_1$ to $L_n$. At each level $L_i$, the clustering is performed by using the function $d_i$ as the distance measure.

A naive approach should calculate the new distances between all objects and perform the clustering on the whole dataset at each level. Though it is simple, it is inefficient since the properties of LBs and DBSCAN are totally not exploited. Using results at $L_i$ to restrict the calculation in $L_{i+1}$ is thus a reasonable approach to speed up the algorithm.

A-DBSCAN maintains a neighborhood graph $G = (S,E)$ which connects each object $p \in S$ with objects in its $\epsilon$-neighborhood. In the beginning, the graph $G$ is fully connected. At each level, the graph $G$ is updated to reflect the changes in the neighborhoods of objects w.r.t. the used distance function.

At level $L_{i+1}$, we define $\epsilon$-neighborhood of object $p$ w.r.t. the graph $G$ at $L_{i+1}$ ($G^{i+1}$) as follows:
Definition 2.6. ($\epsilon$-neighborhood w.r.t. $G^{i+1}$) The $\epsilon$-neighborhood of $p$ at $L_{i+1}$, denoted as $N^{i+1}_\epsilon(p)$, is defined by $N^{i+1}_\epsilon(p) = \{q \in S | (p,q) \in E^i \land d_{i+1}(p,q) \leq \epsilon\}$.

Following Definition 2.6, the graph $G^{i+1}$ is created by removing every edge $(p,q) \in E^i$ which $d_{i+1}(p,q) > \epsilon$ from $G^i$. It is more efficient than the naive approach since only a part of the distances between objects must be updated at each level instead of all distances. However, while the neighborhoods of objects in naive algorithm at $L_{i+1}$ depend only on the distance function $d_{i+1}$, the notion of the neighborhood of A-DBSCAN considers not only the current distance $d_{i+1}$ but also the distances $d_i$ at previous levels represented by neighborhood graph $G$. As we shall see, this scheme is the heart of A-DBSCAN that allows it to be used with arbitrary sequences of LBs and distance functions thus enhancing the applicability of our algorithm.

How the clusters change. A-DBSCAN works by exploiting the way the clusters change at each level. Assume that we are currently at level $L_{i+1}$.

Lemma 2.1. For every object $p \in S$ the neighborhood of $p$ at $L_{i+1}$ is a subset of the neighborhood of $p$ at $L_i$.

Proof. Straightforward from Definition 2.6. □

According to Lemma 2.1, the neighborhood of each object $p$ decreases at each level. Thus, from $L_i$ to $L_{i+1}$, the core property of each object changes as following:

Lemma 2.2. From level $L_i$ to $L_{i+1}$.
1. If $p$ is a core object in $L_i$ then it is a core, a border or a noise object in $L_{i+1}$.
2. If $p$ is a border object in $L_i$ then it is a border or a noise object in $L_{i+1}$.
3. If $p$ is a noise object in $L_i$ then it remains a noise object in $L_{i+1}$.

Proof. According to Lemma 2.1 and Definition 2.2, a core object $p$ will become a border or a noise object if $|N^{i+1}_\epsilon(p)| < \mu$. A border object $p$ will never be a core object because its neighborhood size never increases. But it will become a noise object if it does not have any core object in its neighbors. Since the neighborhood of a noise object $p$ does not contain any core objects, it will remain a noise object in $L_{i+1}$. □

In DBSCAN, each cluster contains two kinds of objects: core and border objects and the core objects play a critical role to determine clusters. Because of their importance, we define a core cluster as follows:

Definition 2.7. (Core Cluster) A subset $C \subseteq S$ is called a core cluster iff $\forall p \in C : \text{core}(p) \land C \subseteq \xi : \xi$ is a cluster.

Lemma 2.3. For all objects $p, q \in S$, if $p$ and $q$ are not density-connected at $L_i$ ($\neg p \triangleright q$) then they are not density-connected at $L_{i+1}$.

Figure 2: From level $L_i$ to $L_{i+1}$, the core object $a$ changes to border object. Object $b$ changes from border to noise. The noise object $d$ remains unchanged. Cluster $C_1$ at $L_i$ is broken into two clusters $C_{11}$ and $C_{12}$ at $L_{i+1}$. Object $c$ becomes border of cluster $C_2$ at the new level.

Proof. According to Lemma 2.1 and 2.2, if $p \triangleright_{i+1} q \Rightarrow p \triangleright q$. Assume that $p \triangleright_{i+1} q$, there exists a sequence of objects $(x_1, \ldots, x_m)$ so that $p < x_1 < \cdots < x_m > q$ at $L_{i+1}$ according to Definition 2.4. Thus, $p < x_1 < \cdots < x_m > q$ at $L_i$. Therefore, we have $p \triangleright q$.

Lemma 2.4. For every core cluster $C_u$ at $L_{i+1}$ ($C_u^{i+1}$), there exists a core cluster $C_v$ at $L_i$ which $C_u \subseteq C_v$.

Proof. Assume that there exist $p, q \in C_u^{i+1}$ so that $p \in C_u^i$ and $q \in C_v^i$. From Definition 2.5 and Lemma 2.3, we have $p \triangleright q$ at $L_i \Rightarrow p \triangleright q$ at $L_{i+1}$. Thus, $p$ and $q$ belong to different clusters by Definition 2.5. □

Figure 2 illustrates the changes of clustering results and core properties of objects from level $L_i$ to level $L_{i+1}$. A cluster at $L_i$ may break into several smaller clusters at $L_{i+1}$ following the division of its core cluster. The border objects of a cluster may be possessed by other clusters at the next level due to the changes of core properties of its neighbors, even if this cluster does not split. Such property of A-DBSCAN reminds us the monotonicity property of subspace clustering algorithm SUBCLU [10]. Under Euclidean distance (ED), the distances in subspace projections of the data could be consider as a sequence of increasing lower-bounding distances ($\forall p, q \in S : d_i(p, q) \leq d_{i+1}(p, q)$), which is a special case of A-DBSCAN. By exploiting a graph structure, A-DBSCAN acquires the monotonicity property even for arbitrary sequences of LBs and is not restricted under ED like SUBCLU. Therefore, the applicability of the algorithm is significantly increased.

To conclude, the changes of clustering results of A-DBSCAN are monotonic w.r.t. the changes of graph $G$ at each level.

Anytime DBSCAN Algorithm. Our anytime algorithm is based on the notions of DBSCAN. Let us start with a new enhanced notion for a cluster:

Definition 2.8. (The border of a core cluster) A set $B$ of border objects is called the border of a core cluster $C$ iff $\forall p \in B : (1) \exists q \in C : q > p \land (2) \exists q \in S \setminus C : |N_\epsilon(q)| \geq \mu \land \forall r \in C : d(p, q) < d(p, r)$.
DEFINITION 2.9. (A-Cluster) An anytime cluster $C$ is an union of a core cluster $C_c$ and its border $C_b$.

Here, a border object is assigned to its nearest core object instead of being randomly assigned in DBSCAN. This brings up some benefits: (1) the compactness of clusters and thus the clustering quality are enhanced; (2) the clustering is independent with the order of objects.

Since A-DBSCAN operates in multiple levels, we have to efficiently solve 2 problems at each level: (1) How to upgrade the graph $G$? (2) How to perform the clustering?

Ordinarily, we have to update the whole graph $G$ from $L_i$ to $L_{i+1}$ according to Definition 2.6. It is more efficient than the calculation of the distances between all objects in the naive approach. However, Lemma 2.1 to 2.4 suggest a more efficient way as follows. In graph $G$, there exist five kinds of edges: core-core, core-border, border-border, border-noise and noise-noise (the core-noise edges do not exist according to Definition 2.2). Since the edges between border and noise objects do not involve in clustering process, they can be safely ignored to save computation cost. Therefore, we just need to update the parts which involve the core objects: the core-core and core-border edges. This update scheme significantly reduces the cost of graph construction at each level, especially when the number of core objects is small. In other words, we only consider the subgraph with core-core and core-border edges. Note that, the graph $G$ will no longer reflect the neighborhoods of all objects exactly. However, it will not affect the correctness of our algorithm as shown below.

For clustering algorithm, some clusters may be split but there is no merging of clusters from $L_i$ to $L_{i+1}$ according to Lemma 2.4.

COROLLARY 2.1. From level $L_i$ to $L_{i+1}$, an a-cluster $C_k$ may be split if:
1. $\exists u,v \in C_k : (u,v) \in E^i \setminus E^{i+1} \land \text{core}(u) \land \text{core}(v)$.
2. $\exists u \in C_k : \text{core}(u) \land E_i \land \neg \text{core}(u) \land L_{i+1}$.

Corollary 2.1 is directly inferred from Definition 2.4, 2.9. The deletion of an edge between two core objects or degradation of a core object in an a-cluster would break the density-connectivity of its core cluster thus causing the splitting of the a-cluster. Assume that this happened in cluster $C_k$ at $L_i$, all we need is to re-cluster all the core objects in $C_k$ instead of the whole dataset as in naive approach thus saving significant amount of time (roughly $O(|C|^2)$ time with $C$ is set of clusters). After that, all the border points are reassigned to the cluster labels of their nearest core objects following Definition 2.9. In case we use the original cluster notion of DBSCAN, only some border objects need to be reassigned, which is much faster but may be less effective in terms of clustering quality.

Figure 3 shows the pseudo codes for the algorithm A-DBSCAN. For every level, first the graph $G$ is updated (line 5-12). After that, we check all clusters to see if there are splitting possibilities, and re-cluster all of their core clusters to reflect the changes (line 13-19) using the clustering scheme of DBSCAN again. All the border objects will then be added to clusters following their nearest core objects (line 20-23).

Correctness of algorithm. We show that, the final clustering result of A-DBSCAN is identical to DBSCAN except for those objects which can be border objects of two or more different clusters.

Definition 2.6 guarantees that the final neighborhoods of objects of A-DBSCAN are similar to DBSCAN. The graph update scheme ignores only the edges between noise and border objects which do not play any role to determine core properties of objects and density-connectedness of clusters. Therefore, the core properties of objects and the core clusters of A-DBSCAN at the last level are identical to DBSCAN. Since the border objects are assigned to their nearest core objects as stated in Definition 2.9, they are the only difference between A-DBSCAN and DBSCAN. If we use original notion of cluster of DBSCAN then the final results of A-DBSCAN and DBSCAN are totally identical.

A-DBSCAN and naive approach. At a middle level $L_i$, the neighborhoods of objects of A-DBSCAN are not identical to naive approach due to the new neighborhood notion in Definition 2.6. Therefore, the clustering result of A-DBSCAN at $L_i$ is actually an
approximation of the clustering result under distance function $d_i$. They are identical if and only if $D$ contains a sequence of increasingly LBs. Such condition, however, is hard to satisfy in reality.

**Complexity analysis.** In theory, time complexity of anytime algorithm is usually higher than the batch one since it has to run many times. In our setting, the complexity of naive approach and A-DBSCAN is $\sum_{i=1}^{|D|} \theta_i |S|^2$ (with $\theta_i$ is complexity of $d_i$). However, since A-DBSCAN has efficient graph update and re-clustering scheme as described above, A-DBSCAN is much faster than DBSCAN as we shall see in Section 4.

## 3 Similarity measure and Lower-bounding

Given a similarity measure $d$, providing a set of lower-bounding function $D = \{d_i|d_i: S \times S \to R \land \forall p, q \in S : d_i(p, q) \leq d(p, q) \land d_i(p, q) = d(p, q)\}$ is an essential problem in our approach. In order for A-DBSCAN to work properly, these conditions should be fulfilled: (1) $d_i$ should be significantly faster than $d$; (2) $d_{i+1}$ should be tighter than $d_i$ ($d_i(p, q) \leq d_{i+1}(p, q)$ in general).

### 3.1 Euclidean distance.

Recent research has introduced many similarity measures for high dimensional data such as Euclidean Distance (ED), Longest Common Subsequence (LCS), Dynamic Time Warping (DTW), etc. [6]. Though our algorithm can be used with all mentioned techniques above, we choose ED as similarity measure for our clustering algorithm due to its simplicity and ubiquitousness.

### 3.2 The Haar wavelet transform.

LBs for indexing of ED is a well-studied problem. There exist many proposed techniques such as Piecewise Aggregate Approximation (PAA) [13], Discrete Fourier Transform (DFT) [12] and Discrete Wavelet Transform (DWT) [5], etc. All of these techniques can be directly applied to A-DBSCAN. In this work, we simply choose DWT to build a sequence of LBs.

The Haar transform [5, 12] can be seen as a series of averaging and differencing operations between two adjacent values of a discrete time function $f(x)$ at a given resolution to form a smoothed, lower dimensional representation of signal. The wavelet decomposition is the combination of the coefficients at all resolutions: the first coefficient is the overall average of $f(x)$, while the other coefficients store the amount of information lost at each resolution. Due to space limitation, interested readers please refer to [5] for more details.

**Lemma 3.1.** The Haar transform preserves the Euclidean distance [5].

To build a sequence of LBs, we first transform all objects using DWT. Then, at each level $L_i$, we only use the first $k_i$ coefficients of each object to calculate the distance function $d_i$ with $k_i \leq k_{i+1}$. According to Lemma 3.1, the lower-bounding condition 2 holds. Since we only need a few coefficients to have a good approximation of original function, the runtime of the distance functions $d_i$ is significantly faster than the original ED function $d$, which satisfies the runtime condition 1. Due to linear complexity of DWT, the time needed to transform the whole dataset $S$ with dimensionality $n$ is $O(n|S|)$ which is negligible compared with $O(n|S|^2)$ for DBSCAN. For a large time series dataset with 9236 objects with the length of each object $n = 8192$, it costs only 2.1 seconds to transform the whole dataset and an hour for clustering with DBSCAN.

## 4 Experiments

We present our experiments on real datasets to prove the efficacy of our algorithm. All experiments are conducted on a Workstation with 2.0 Ghz CPU, 32GB RAM under Windows Server 2008 using Java.

### 4.1 Algorithms and Comparison Criteria.

The evaluations are conducted on 3 different algorithms: (1) DBSCAN [8]; (2) Multi-levels filter and refinement range query DBSCAN (M-DBSCAN) which uses LBs to speed up the range query process of DBSCAN thus enhancing the efficiency as stated in [3]; (3) Our algorithm A-DBSCAN. However, the batch algorithms DBSCAN and M-DBSCAN are only used as a rival technique to demonstrate the speed of our algorithm. It is important to note that, A-DBSCAN is not necessary to be faster than M-DBSCAN and DBSCAN in general since it has to perform clustering at each level. However, it should quickly produce good clustering results at early levels [16]. The comparison between naive anytime algorithm and A-DBSCAN is omitted for clarity since the naive algorithm clearly performs much worse than DBSCAN and M-DBSCAN.

For A-DBSCAN and M-DBSCAN, we use a sequence of LBs with 10 different functions $D = \{5, 10, 15, 20, 25, 30, 35, 40, 45, 100\}$. Each function $d_i$ at $L_i$ uses first $D_i$ % Wavelet coefficients to calculate lower-bounding distance (100% means original ED distance). For the two parameters $\mu$ and $\epsilon$, we first run DBSCAN with $2 \leq \mu \leq 30$ and $100$ equally distributed values of $\epsilon$ from min to max distance between objects to find the optimal parameters for DBSCAN for every dataset. Then M-DBSCAN and A-DBSCAN are evaluated with these parameters. Therefore, we will have fair comparisons between all these algorithms. All the reported results are averaged over 100 runs. The selection of parameters will be further studied in Section 4.3. For A-DBSCAN, we report the cumulative runtime with all the costs (even the runtime for Wavelet transform).
To compare the results of different clustering algorithms with the ground truth, we use the Normalized Mutual Information (NMI) and Adjusted Mutual Information (AMI) [15]. However, we only show the NMI for clarity because the results of AMI are the same as NMI. The result of NMI is in [0, 1], with 0 means that the clustering result is independent of the ground truth and 1 means that the clustering result is the same as the ground truth.

4.2 Performance evaluation. We evaluate the performance of A-DBSCAN on various real datasets.

The UCR datasets. Figure 4 shows the clustering results for 5 datasets acquired from the UCR time series archives (http://www.cs.ucr.edu/~eamonn/time_series_data/) which contain small to large time series datasets from diverse fields. For each dataset, we report at each level the ratio between NMI score (the higher the values are, the better A-DBSCAN is) as well as the ratio between cumulative runtime of A-DBSCAN and runtimes of M-DBSCAN and DBSCAN (the lower the values are the better A-DBSCAN is). The NMI score of DBSCAN is also shown beside the name of each dataset. With dataset MALLAT (µ = 7, ϵ = 12.8), for example, the NMI score of DBSCAN (and also M-DBSCAN) is 0.824. The runtimes are 49.9 and 298.7 seconds for M-DBSCAN and DBSCAN respectively. At the first level L1, A-DBSCAN requires only 17.4 seconds to complete, which is 3 times faster than M-DBSCAN with a clustering score of 0.7475. The clustering scores come to 0.826 at level 2-3 and 0.824 at level 4-10. When it comes to the end, A-DBSCAN requires only 39.7 seconds, which is still faster than 49.9 seconds of M-DBSCAN and 8 times faster than DBSCAN.

For all datasets, the clustering scores become very close to the clustering scores of DBSCAN (more than 80% of NMI score of DBSCAN) from level 2. This means that A-DBSCAN acquires very good and stable clustering results at very early stages. For the dataset ECGFiveDays (µ = 5, ϵ = 8.18) and SonyAIBORobotSurface (µ = 24, ϵ = 9.2), users can terminate the algorithm at level 3 or later. For other datasets, the termination can be even earlier to acquire the speed up of 5 to 10 times compared with M-DBSCAN and more than 10 times compared with DBSCAN. It is a remarkable advantage, especially when using the cheap ED as distance measure. The difference would be significantly larger when we use expensive distance functions such as DTW or LCS [6], etc. For the dataset CinC_ECG_torso (µ = 22, ϵ = 63.9), the NMI scores at the early levels L1 to L3 (0.743, 0.739, 0.739 respectively) are even slightly better than the NMI scores of DBSCAN (0.738).

Figure 5 shows further comparisons for all 32 UCR datasets: a) The NMI ratios of A-DBSCAN and DBSCAN. A-DBSCAN improves the results on 13 datasets. b) The cumulative runtime ratios of A-DBSCAN and M-DBSCAN. A-DBSCAN is up to 2 times faster than M-DBSCAN on 31/32 datasets. c) The cumulative runtime ratio of A-DBSCAN and DBSCAN. A-DBSCAN is up to 20 times faster than DBSCAN.

The UCI datasets. Two datasets Character Trajectory (CT) and Australian Sign Language (ASL) from UCI archives (http://archive.ics.uci.edu/ml) are used to assess our algorithm. The CT dataset contains 2858 2D
The effects of parameter $\epsilon$ (with $\epsilon = 4$) and $\mu$ (with $\mu = 5$) and performance of A-DBSCAN for COIL20 dataset.

The parameters $\mu$ and $\epsilon$. Figure 8 shows the relationships between the two parameters $\mu$ (with $\epsilon = 4$), $\epsilon$ (with $\mu = 5$) and the performance of A-DBSCAN on COIL20 dataset. The runtime of the algorithm increases with $\epsilon$ and decreases with $\mu$ due to the reduction of graph size and number of core objects. The clustering quality is strongly affected by the choices of $\epsilon$. For $\epsilon = 3.5$, A-DBSCAN quickly reaches good NMI scores at level 2. The clustering quality slightly increases at each level until level 6 and decreases until the end. For $\epsilon = 5.0$, the clustering quality of A-DBSCAN increases and reaches the maximum value at the last level. In contrast, the clustering quality of A-DBSCAN at each level seems less affected by the choices of $\mu$. All the NMI score curves are generally the same with different values of $\mu$. Moreover, by using LBs, A-DBSCAN may reach results which are hard to acquire with DBSCAN at some middle levels. For example, the best found score for COIL20 is 0.908 for A-DBSCAN ($\mu = 3$, $\epsilon = 4$) at level 7, while the best found score for DBSCAN is only 0.857 ($\mu = 5$, $\epsilon = 4.75$). These phenomena are caused by the lower-bounding property of the distance function w.r.t parameter $\mu$ and $\epsilon$. For example, if $\epsilon$ is smaller than the optimal value, A-DBSCAN may acquire near optimal results at some mid-levels since all the distances between objects are smaller than the true distance.

Though A-DBSCAN seems more robust to the choices of parameters than DBSCAN due to its anytime scheme, the choices of $\epsilon$ and $\mu$ are still data-dependent and thus hard to select in general. Therefore, some existing heuristics such as the k-dist graph [8] and entropy [11] are still helpful.

The lower-bounding functions. Choosing a sequence of LBs is an important aspects of A-DBSCAN. Depending on their characteristics, different datasets require different numbers of coefficients to closely approximate the true ED distance. For example, for dataset ECGFiveDays, the use of the first 5% coefficients at level 1 is too small to have a good approximation of ED distance. Thus, the clustering quality is too low. In contrast, the first 5% coefficients are too many for

4.3 Parameters analysis. We study the impact of parameters $\mu$, $\epsilon$ and $D$ of A-DBSCAN.

Other datasets. Figure 7 shows the clustering results for COIL20 dataset in computer vision field acquired from Columbia Object Image Library (http://www1.cs.columbia.edu/CAVE/software/softlib/coil-20.php). This dataset contains 1440 pictures of 20 different objects with 72 pictures per object. The same results as above are also observed.

Summary. A-DBSCAN acquires close clustering scores with DBSCAN at early stages of its execution. Thus, it helps to accelerate the runtime up to orders of magnitude compared with M-DBSCAN and DBSCAN. Though they both use LBs, A-DBSCAN is faster than M-DBSCAN because it can exploit the monotonicity property to reduce redundant distance upgrades and re-clustering as described in Section 2. Note that, the UCR and UCI datasets are re-interpolated to the length of $2^\lceil \log(n) \rceil + 3$ to use with DWT.

4.3 Parameters analysis. We study the impact of parameters $\mu$, $\epsilon$ and $D$ of A-DBSCAN.
dataset Symbols. The lower-bounding distance is close to the ED distance. Thus, the clustering qualities are the same at all levels. Figure 9 shows the relationship between performance of A-DBSCAN and tightness of LBs, which is the average of ratios between lower-bounding functions and true functions, w.r.t. the numbers of used Wavelet coefficients for dataset Symbol \((\mu = 30, \epsilon = 16.1)\). A-DBSCAN is run with 3 different sequences of LBs \(D_1\) to \(D_3\) (10 levels with different numbers of coefficients). As we see, the higher the numbers of the used coefficients at each level are, the tighter the lower-bounding distances and the better the clustering qualities are. However, the runtime of the algorithm is increased.

Therefore, by randomly drawing a subset of data and calculating the averaged tightness of LBs w.r.t. the numbers of the used coefficients, users can easily choose the number of levels and number of used coefficients for each level based on their purposes.

5 Related Works and Discussion

Anytime Clustering Algorithms. Anytime algorithms are algorithms that trade execution time for quality of result [17]. The quality of result of an anytime algorithm typically improves as the time increases to reach the result of the batch algorithm in the end. An anytime algorithm should satisfy some important properties described in [16, 17]. Anytime algorithms are currently an area of active research in many fields of data mining such as classification [14] and outlier detection [2]. However, there is only little work on anytime clustering algorithms such as [9, 12, 13, 16]. Zhu et al. [16] proposed an approximation technique for Dynamic Time Warping which allows it to be used with anytime clustering algorithms. Kraken et al. [9] proposed an anytime clustering algorithm for streaming data.

Lin et al. [12, 13] exploited the multi-resolution property of DWT and PAA to casting k-Means into an anytime algorithm called I-kMeans. I-kMeans works by using the final cluster centers of level \(i\) as initial centers for level \(i+1\). Though it is simple and efficient, I-kMeans is limited only for spherical shape clusters while A-DBSCAN is able to detect clusters with arbitrary shapes and robust to outliers. The lower-bounding property of DWT and PAA are also not exploited to construct clusters as in A-DBSCAN.

Density-based clustering. In density-based clustering algorithms, clusters are considered as high density areas, separated by low density areas. Among various kinds of density-based clustering algorithm, DBSCAN is one of the most successful algorithms with many extensions and applications in many fields such as medicine, etc. [1, 3, 4, 7, 10].

Ester et al. [7] proposed an incremental version of DBSCAN in a data warehousing environment. Based on the fact that insertion or deletion of an object affects the current clustering only in the neighborhood of this object, their algorithm called I-DBSCAN significantly speeds up DBSCAN even for large numbers of updates in a data warehouse. I-DBSCAN also exploits the nature of DBSCAN to do the clustering like our algorithm. However, I-DBSCAN is an incremental clustering algorithm, not an anytime algorithm. The changes of clusters in I-DBSCAN are caused by inserted or deleted objects, while the changes in A-DBSCAN are directed by the changes of distance function.

Density-based subspace clustering algorithm SUBCLU [10] is based on the monotonicity property of DBSCAN w.r.t. distances in subspace projections of the data. The changes of clusters in A-DBSCAN are also monotonic. However, the monotonicity of A-DBSCAN is caused by the reduction of a special neighborhood graph related to a sequence of LBs, which is more general than SUBCLU. Thus, A-DBSCAN can be used with many different kinds of distance measures such as DTW, LCS, etc. [6] and arbitrary sequences of LBs while SUBCLU is limited under ED. Moreover, SUBCLU is also not an anytime algorithm.

In [4], a client-server parallel version of DBSCAN is proposed. The monotonicity property of DBSCAN [8] and OPTICs [1] is used to split objects to different clients and to merge the results returned from clients into final clusters in server. This monotonicity property is also similar to SUBCLU and is a special case of the monotonicity property used with A-DBSCAN.

LBs could be used in density-based clustering to accelerate the range query process. In [3] the authors integrate a lower-bounding function into DBSCAN [8] and OPTICs [1] to speed up these algorithms based on a data structure called XSeedlist to reduce the number of true distance calculations. Although it is fast, it produces only one result and thus is not an anytime algorithm. In our algorithm, we use many LBs to produce multiple approximate clustering results during the runtime. A-DBSCAN relies on the monotonicity property and restricted neighborhood graph to perform
the clustering, which is fundamentally different from [3]. The ideas of A-DBSCAN can also be combined with the ideas of [3]. To update the clusters at each level, as described in Section 2.2, we can replace the original Seedlist of DBSCAN by Xseedlist. As our algorithm exploits the monotonicity property, it yields better pruning power than [3]. Unlike the original seedlist, the concept of Xseedlist requires a time-consuming sorting procedure. So it pays off only for expensive distance functions. In this paper, we use the original seed list of DBSCAN.

**Anytime DBSCAN.** In general, A-DBSCAN is unique in the ways that it: (1) exploits multi-levels lower-bounding functions to produce multiple approximate results of the final clustering result; (2) maintains the graph structure to acquire the monotonicity property even for arbitrary sequences of LBs. By this way, A-DBSCAN can be used with any kind of distance measures and arbitrary sequences of lower-bounding functions. Thus it would have great applicability in reality. Due to its update scheme, A-DBSCAN works very well on noisy datasets which contain large amount of noise and border objects. Also, A-DBSCAN is extremely useful when using with very expensive distance functions such as DTW or LCS [6]. Moreover, A-DBSCAN can easily be parallelized.

**6 Conclusion and Future Remark**

We propose an anytime density-based clustering algorithm called A-DBSCAN which is applicable for many high dimensional databases such as time series, trajectory, etc. Our algorithm works by exploiting a sequence of LBs to produce multiple approximate results of the true density-based clusters. To enhance performance, we propose an efficient distance update scheme which partially updates the distances of objects, and a local re-clustering scheme to save computational time at each level. Some changes in the notions of DBSCAN are made to improve the clustering results. An efficient heuristic for parameter setting is also proposed. Experiments on real data sets have shown that A-DBSCAN produces very good clustering results at very early stages of execution thus saving a large amount of computational time. Even if it runs to the end, A-DBSCAN is still much faster than DBSCAN and M-DBSCAN, despite the fact that it has to produce clustering results at every level. Our future work aims at applications of A-DBSCAN on emerging research fields such as biomedicine, etc.

**Acknowledgements.** We thank to Claudia Plant, Sebastian Goebel, Nina Hubig, Diep M. T. Phan, Ha H. T. Mai, Hahn M. T. Vo, Anh X. Nghien, and Franz Krojer for their helps and supports. We special thank for anonymous reviewers for their helpful comments.

**References**


