

# Towards Metric DBSCAN: Exact, Approximate, and Streaming Algorithms

GUANLIN MO\* and SHIHONG SONG\*, University of Science and Technology of China, China  
 HU DING†, University of Science and Technology of China, China

DBSCAN is a popular density-based clustering algorithm that has many different applications in practice. However, the running time of DBSCAN in high-dimensional space or general metric space (*e.g.*, clustering a set of texts by using edit distance) can be as large as quadratic in the input size. Moreover, most of existing accelerating techniques for DBSCAN are only available for low-dimensional Euclidean space. In this paper, we study the DBSCAN problem under the assumption that the inliers (the core points and border points) have a low intrinsic dimension (which is a realistic assumption for many high-dimensional applications), where the outliers can locate anywhere in the space without any assumption. First, we propose a  $k$ -center clustering based algorithm that can reduce the time-consuming labeling and merging tasks of DBSCAN to be linear. Further, we propose a linear time approximate DBSCAN algorithm, where the key idea is building a novel small-size summary for the core points. Also, our algorithm can be efficiently implemented for streaming data and the required memory is independent of the input size. Finally, we conduct our experiments and compare our algorithms with several popular DBSCAN algorithms. The experimental results suggest that our proposed approach can significantly reduce the computational complexity in practice. Our source code can be found at <https://github.com/MoGuanlin/Towards-Metric-DBSCAN>.

CCS Concepts: • **Theory of computation** → **Theory and algorithms for application domains**.

Additional Key Words and Phrases: Density-based Clustering, Outliers, Approximation, Doubling dimension, Streaming,  $k$ -center Clustering

## ACM Reference Format:

Guanlin Mo, Shihong Song, and Hu Ding. 2024. Towards Metric DBSCAN: Exact, Approximate, and Streaming Algorithms. *Proc. ACM Manag. Data* 2, 3 (SIGMOD), Article 178 (June 2024), 25 pages. <https://doi.org/10.1145/3654981>

## 1 INTRODUCTION

*Density-based clustering* is an important clustering model for many real-world applications [44]. Roughly speaking, the goal of density-based clustering is to identify the dense subsets from a given data set in some metric space. Unlike other clustering algorithms (*e.g.*, the center-based clustering algorithms like  $k$ -means clustering [63]), the density-based clustering algorithms are particularly useful to extract clusters of arbitrary shapes with outliers. So they also can be applied to solve outlier recognition problems [10]. In the past decades, a number of density-based clustering algorithms have been proposed [2, 5, 23, 51]. **DBSCAN** (Density Based Spatial Clustering of Applications

\*Both authors contributed equally to this research.

†Corresponding author.

---

Authors' addresses: Guanlin Mo, [moguanlin@mail.ustc.edu.cn](mailto:moguanlin@mail.ustc.edu.cn); Shihong Song, [shihongsong@mail.ustc.edu.cn](mailto:shihongsong@mail.ustc.edu.cn), University of Science and Technology of China, No.96, JinZhai Road Baohe District, Hefei, Anhui, China, 230026; Hu Ding, University of Science and Technology of China, No.96, JinZhai Road Baohe District, Hefei, Anhui, China, [huding@ustc.edu.cn](mailto:huding@ustc.edu.cn).

---

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 2836-6573/2024/6-ART178  
<https://doi.org/10.1145/3654981>

with Noise), which was proposed by Ester et al. [23], is one of the most popular density-based clustering algorithms and has been extensively applied to various practical areas [26, 33, 59]. Due to its substantial contributions in practice, the original DBSCAN paper was awarded *the test of time award* in ACM KDD 2014.

Though the DBSCAN method often achieves promising clustering performance in practice, the high computational complexity can be prohibitive for applications in large-scale data. Suppose the input data size is  $n$ . The worst-case time complexity of DBSCAN is  $\Theta(n^2)$ . Obviously this quadratic complexity is far from satisfying for large  $n$  in practice, and thus it motivated several faster implementations of DBSCAN in the past years. Gunawan and de Berg [28] proposed a grid-based DBSCAN algorithm, which can reduce the time complexity to  $O(n \log n)$  in  $2D$ . For general low-dimensional Euclidean space  $\mathbb{R}^d$  (assume  $d$  is small), Chen et al. [11] and Gan and Tao [26] independently proposed similar sub-quadratic complexity algorithms by using some computational geometry techniques. For example, the time complexity of [26] is  $O(n^{2 - \frac{2}{\lfloor d/2 \rfloor + 1} + \delta})$  with  $\delta$  being any arbitrarily small non-negative constant. However, as  $d$  increases, the time complexities of [11, 26] approach to  $O(n^2)$ . Furthermore, Gan and Tao [26] proved that the DBSCAN problem is at least as hard as the *unit-spherical emptiness checking (USEC)* problem [1], where it is widely believed that the time complexity for solving the USEC problem is  $\Omega(n^{\frac{4}{3}})$  for any  $d \geq 3$  [21, 22]. Therefore, most recent works mainly focus on the low-dimensional DBSCAN problem [15, 58, 61, 66].

Due to the hardness of the exact DBSCAN problem, another line of research on approximate DBSCAN attracts more attentions recently. For example, the aforementioned works [11] and [26] also proposed approximate DBSCAN algorithms with theoretical guarantees. The connectivity between points within each cluster, which is the most time-consuming step in exact DBSCAN, is relaxed in their algorithms. Their algorithms both achieve the  $O(n/\rho^{d-1})$  time complexity, where  $\rho > 0$  is the given parameter that measures the approximation quality (we provide the formal definition in Section 1.1). Other variants of approximate DBSCAN methods include sampling based DBSCAN++ [40], LSH based DBSCAN [70], and KNN based NG-DBSCAN [49].

However, the research on DBSCAN for abstract metric space and high-dimensional space is still quite limited, to the best of our knowledge. In this big data era, we often confront with high-dimensional data [24] and even non-Euclidean data [6]. For example, the input data can be texts, images, or biological sequences, which cannot be embedded into a low-dimensional Euclidean space; and moreover, the distance between different data items can be more complicated than Euclidean distance (e.g., edit distance [52]). We should emphasize that one can always run the original DBSCAN algorithm [23] for solving DBSCAN in abstract metric space or high-dimensional space, but the previously developed accelerating techniques cannot be directly applied to reduce the complexity for these cases. Ding et al. [17] proposed a fast exact DBSCAN algorithm for abstract metric space by using the  $k$ -center clustering heuristic [16]; but their result lacks strict theoretical analysis. Lulli et al. [49] and Yang et al. [72] considered distributed DBSCAN for metric space, but their methods mainly focus on the communication cost and load balance for distributed system rather than the time complexity (e.g., the approach of [72] just directly uses the original DBSCAN algorithm in each local machine).

**Our main results and the key ideas.** In this paper, we systematically study the DBSCAN problem in abstract metric space (we call it “**metric DBSCAN**” problem for short). Our results can be also extended to the case in high-dimensional Euclidean space, if the intrinsic dimension of the input data (except for the outliers) is low, which is a reasonable assumption for a range of applications in real world (e.g., image datasets [56]). We use the “doubling dimension”  $D$  to measure the intrinsic dimension (the formal definition is shown in Section 1.1). Also note that we do not require the value of  $D$  to be explicitly given in our algorithms. We do not add any constraint to

the outliers, because in practice they can locate anywhere in the space. For example, in the area of AI security, an attacker may add some carefully crafted outliers to mislead the machine learning model, where some recent works showed that the outliers often have significantly higher intrinsic dimension than normal data [35, 68]. We consider both the exact and approximate metric DBSCAN problems. Our main contributions are twofold (the results can be easily extended to the case that both inliers and outliers have low doubling dimensions).

(1) The exact DBSCAN problem involves two major tasks: identify the core points and merge the core points to form the clusters. Both of these two tasks take  $O(n^2)$  time in the worst case if using the original DBSCAN algorithm. First, we show that the first task can be completed with the runtime linear in  $n$ . The key idea is inspired by a novel insight that relates the task to a *radius-guided  $k$ -center clustering* method in metric space, which can reduce the search range for identifying the core points. After labeling the core points, we then build a set of cover trees [4] for the local regions, and solve the second task for merging the core points with the time linear in  $n$  as well.

(2) We also consider the  $\rho$ -approximate DBSCAN problem proposed in [26]. Our idea is to construct a “summary” for the set of core points. The summary should be much smaller than the core points set (which can be as large as  $n$ ); also the summary should be able to approximately represent the core points, so that we can efficiently recover the  $\rho$ -approximate DBSCAN clusters without dealing with all the core points. The approximate algorithm also has the time linear in  $n$ ; but comparing with our proposed exact DBSCAN algorithm, it compresses the data into a small-size summary, so the method can be easily extended for developing streaming algorithm where the required memory is independent of  $n$ . To the best of our knowledge, the current research for streaming DBSCAN (exact or approximate DBSCAN with theoretical quality guarantee) are still quite limited, though several other types of density-based streaming clustering algorithms were proposed before [8, 12, 25, 31].

We also take a comprehensive set of experiments to evaluate the performance of our proposed algorithms. Our algorithms can achieve significant speedups over several existing DBSCAN algorithms. Another advantage of our method is that it is friendly for parameter tuning. For example, one usually needs to try different values for the parameters (*e.g.*, the ball radius and density threshold) to achieve a satisfied DBSCAN solution; the radius-guided  $k$ -center clustering, which is a key part in our algorithms, do not need to be performed repeatedly during the parameter tuning, and therefore our overall running time can be further reduced in practice.

## 1.1 Preliminaries

**Notations.** We denote the metric space by  $(X, \text{dis})$ , where  $X$  is the set of  $n$  input data items and  $\text{dis}(\cdot, \cdot)$  is the distance function on  $X$ . For any  $p \in X$  and  $Y \subseteq X$ , we define  $\text{dis}(p, Y) = \min_{y \in Y} \text{dis}(p, y)$ . Let  $\mathbb{B}(p, r)$  be the ball centered at a point  $p \in X$  with radius  $r \geq 0$  in the space. For any  $Y \subseteq X$ , we use  $|Y|$  to denote the number of data items in  $Y$ . We assume that it takes  $t_{\text{dis}}$  time to compute the distance  $\text{dis}(p_1, p_2)$  between any two points  $p_1, p_2 \in X$ . For example,  $t_{\text{dis}} = O(d)$  for  $d$ -dimensional Euclidean space.

*1.1.1 Overview for DBSCAN.* For the sake of completeness, we briefly overview the DBSCAN algorithm [23] and the  $\rho$ -approximation [26]. DBSCAN has two input parameters:  $\epsilon \in \mathbb{R}^+$  and  $\text{MinPts} \in \mathbb{Z}^+$ . All the input data points of  $X$  are divided into three categories (see Figure 1 for an illustration): for any  $p \in X$ ,

- (1)  $p$  is a **core point** if it has at least  $\text{MinPts}$  neighbors within the distance  $\epsilon$ , *i.e.*,  $|\mathbb{B}(p, \epsilon) \cap X| \geq \text{MinPts}$ ;
- (2)  $p$  is a **border point** if  $p$  is not a core point, but there exists at least one core point  $q$  satisfying  $p \in \mathbb{B}(q, \epsilon)$ ;

(3)  $p$  is an **outlier**, if it does not satisfy neither (1) nor (2).



Fig. 1. An example for DBSCAN with  $MinPts = 4$ : the solid points are core points, the point  $p_1$  is a border point, and the point  $p_2$  is an outlier.

We also need to consider the “reachability” to define DBSCAN clusters. Let  $q$  be a core point. A point  $p \in X$  is **density-reachable** from  $q$  if we can find a sequence of points  $p_1, p_2, \dots, p_t \in X$  satisfying: all of them (with the possible exception of  $p_t$ ) are core points, and

- $p_{i+1} \in \mathbb{B}(p_i, \epsilon)$  for each  $i \in [1, t - 1]$ ;
- $p_1 = q$  and  $p_t = p$ .

**DEFINITION 1** (DBSCAN [23]). *We say a non-empty set of points  $C \subseteq X$  is a **cluster**, if  $C$  satisfies the following two conditions: (1) **maximality**:  $\forall p, q$ , if  $q \in C$  and  $p$  is density-reachable from  $q$ , then  $p \in C$ ; (2) **connectivity**: for any pair of points  $p, q \in C$ , there is a point  $o \in C$  such that both  $p$  and  $q$  are density-reachable from  $o$ . The **DBSCAN problem** is to find the set of DBSCAN clusters of  $X$ <sup>1</sup>*

Let  $\rho > 0$ . We say a point  $p \in X$  is  $\rho$ -**approximate density-reachable** from a core point  $q$  if the third condition of density-reachable “ $p_{i+1} \in \mathbb{B}(p_i, \epsilon)$ ” is relaxed to be “ $p_{i+1} \in \mathbb{B}(p_i, (1 + \rho)\epsilon)$ ” for each  $i = 1, 2, \dots, t - 1$ .

**DEFINITION 2** ( $\rho$ -APPROXIMATE DBSCAN [26]). *By using “ $\rho$ -approximate density-reachable” instead of “density-reachable” in the **connectivity** condition, one can define  $\rho$ -**approximate cluster** via the similar manner of Definition 1. The  $\rho$ -**approximate DBSCAN problem** is to find a set of  $\rho$ -approximate clusters such that every core point of  $X$  belongs to exactly one cluster.*

It is easy to see that  $\rho$ -approximate DBSCAN is equivalent to the exact DBSCAN problem if  $\rho = 0$ . Also, Gan and Tao [26] proved a “Sandwich Theorem” for it. Namely, the  $\rho$ -approximate DBSCAN solution  $S$  with  $(\epsilon, MinPts)$  is sandwiched by the exact DBSCAN solutions  $S_1$  and  $S_2$  with  $(\epsilon, MinPts)$  and  $((1 + \rho)\epsilon, MinPts)$  respectively: for any two points  $p, q \in X$ , if they fall into the same cluster of  $S_1$ , they must also fall into the same cluster of  $S$ ; if they fall into the same cluster of  $S$ , they must also fall into the same cluster of  $S_2$ .

**REMARK 1.** *The parameter  $MinPts$  is usually set to be a constant (say, less than 10) as recommended in previous articles like [15, 23, 28]. So we assume that  $MinPts$  is a small number throughout this paper.*

**1.1.2 Doubling Dimension.** We take “doubling dimension”, which is a natural and robust measure that has been widely used in machine learning and data analysis [29, 54], to indicate the intrinsic dimension of data. Intuitively, the doubling dimension measures how fast the data volume is growing in the space.

**DEFINITION 3** (DOUBLING DIMENSION [29]). *Given a metric space  $(X, dis)$ , let  $\Lambda$  be the smallest positive integer such that for any  $p \in X$  and any  $r \geq 0$ ,  $X \cap \mathbb{B}(p, 2r)$  can always be covered by the union of at most  $\Lambda$  balls with radius  $r$ . Then the doubling dimension of  $(X, dis)$  is  $D = \lceil \log_2 \Lambda \rceil$ .*

We also have the following proposition for doubling dimension that is used in our following analysis.

<sup>1</sup>In this definition, a border point may be assigned to multiple clusters.

**PROPOSITION 1** ([43, 62]). *Suppose the doubling dimension of a metric space  $(X, \text{dis})$  is  $D$ . For any point set  $Y \subseteq X$ , we have  $|Y| \leq 2^{D \lceil \log \alpha \rceil}$ , where  $\alpha$  is the aspect ratio of  $Y$ , i.e.,  $\alpha = \frac{\max_{y, y' \in Y} \text{dis}(y, y')}{\min_{y, y' \in Y} \text{dis}(y, y')}$ .*

We let  $X_{\text{out}}$  be the set of DBSCAN outliers and  $X_{\text{in}} = X \setminus X_{\text{out}}$ . For convenience, we let  $z = |X_{\text{out}}|$ . We have Assumption 1 following the previous articles on algorithms design in doubling metric [9, 17, 36].

**ASSUMPTION 1.** *We assume that the doubling dimension  $D$  of the sub-metric space  $(X_{\text{in}}, \text{dis})$  is constant.*

For the outliers  $X_{\text{out}}$ , we do not make any assumption for their doubling dimension. In other words, the outliers can scatter arbitrarily without any constraint in the space (e.g., the outliers may be manipulated by an attacker [35, 68]).

**1.1.3 Cover Tree.** We also introduce a spatial query structure “cover tree” which was proposed by Beygelzimer et al. [4] for data with low intrinsic dimension. To state the definition, we need to introduce “ $r$ -net” first [13].

**DEFINITION 4** ( $r$ -NET). *Given a number  $r > 0$  and a set  $P$  in some metric space, a subset  $Q \subseteq P$  is an  $r$ -net of  $P$  if it satisfies two requirements:*

- (1)  $r$ -packing:  $\text{dis}(q, Q \setminus \{q\}) \geq r$  for all  $q \in Q$ .
- (2)  $r$ -covering:  $\text{dis}(p, Q) \leq r$  for all  $p \in P$ .

A **Cover Tree** is a hierarchical tree structure  $T$  for storing a given data set  $P$ , where each node corresponds to a point of  $P$ . The set of nodes at the  $i$ -th level is denoted by  $T_i$  for  $l_{\text{bottom}} \leq i \leq l_{\text{top}}$ , where  $l_{\text{bottom}}$  and  $l_{\text{top}}$  indicate the lowest and highest levels of  $T$ , respectively;  $T_{l_{\text{top}}}$  is the root node, and  $T_i \subset T_{i-1}$  for each  $i > l_{\text{bottom}}$ . Also, we require that each  $T_i$  is a  $2^i$ -net of  $T_{i-1}$ .

For convenience, let  $\Delta_P$  and  $\delta_P$  denote the maximum and minimum pairwise distances of  $P$ , respectively. From the above definition of the cover tree, we can derive that  $l_{\text{top}} \leq \lceil \log \Delta_P \rceil$  and  $l_{\text{bottom}} \geq \lfloor \log \delta_P \rfloor$  (so  $l_{\text{bottom}}$  can be negative if  $\delta_P < 1$ ).

**CLAIM 1** (COMPLEXITIES OF COVER TREE). *Let the aspect ratio of  $P$  be  $\Phi_P = \Delta_P / \delta_P$ . Suppose the dataset  $P$  has a doubling dimension  $D$ . The construction time of the cover tree of  $P$  is  $O(2^{O(D)} |P| \log \Phi_P \cdot t_{\text{dis}})$ . For any query point  $p$ , finding its nearest neighbor in  $P$  takes  $O(2^{O(D)} \log \Phi_P \cdot t_{\text{dis}})$  time.*

**REMARK 2.** *In the original article [4], the authors assumed that the input data has a constant “expansion rate”, which is a similar but more stringent intrinsic dimension measure than the doubling dimension [29, Proposition 1.2]. For example, a data set has a constant expansion rate must also have a constant doubling dimension, but not vice versa; in other words, doubling dimension is a more general measure than the expansion rate. The complexities in terms of the doubling dimension of Claim 1 can be obtained by using the Navigating net idea [20, 43].*

In recent years, several new variants of the cover tree, such as the simplified cover tree [39] and the compressed cover tree [20], were also proposed. In this paper, we only use the vanilla cover tree for simplicity in our analysis, and one can also replace it by those variants in practice.

**The rest of this paper is organized as follows.** In Section 2, we propose the radius-guided  $k$ -center clustering algorithm that is the key technique for our algorithms. In Section 3, we show our algorithm for exact metric DBSCAN problem. In Section 4, we show our algorithm and its streaming version for  $\rho$ -approximate DBSCAN. Finally, we illustrate our experimental results in Section 5.

## 2 RADIUS-GUIDED GONZALEZ'S ALGORITHM

In this section, we revisit the Gonzalez's algorithm [27] which is a popular method for solving the  $k$ -center clustering problem. The goal of  $k$ -center clustering is to find  $k$  balls to cover the whole data with minimizing the maximum radius. The Gonzalez's algorithm is a standard greedy procedure. It iteratively selects  $k$  points, where each point has the largest distance to the set of the points that were selected in the previous rounds. The algorithm yields a 2-approximate solution, that is, the obtained radius is no larger than two times the optimal one. It was also shown that designing a  $(2 - \gamma)$ -approximate algorithm for any  $\gamma > 0$  is NP hard [34].

In this section, we propose a radius-guided version of the Gonzalez's algorithm (Algorithm 1). The key difference between our algorithm and the vanilla Gonzalez's algorithm is that the former one needs an upper bound  $\bar{r}$  of the radius instead of the number of centers  $k$ .

---

### Algorithm 1: RADIUS-GUIDED GONZALEZ

---

**Input:** The dataset  $X$  and an upper bound  $\bar{r} > 0$

- 1 Take an arbitrary point  $p_0 \in X$  and set  $E = \{p_0\}$ ; //  $E$  stores the center points.
- 2 Set  $d_{\max} = \max_{p \in X} \text{dis}(p, E)$ .
- 3 For any  $p \in X$ , define its closest center  $c_p = \arg \min_{e \in E} \text{dis}(p, e)$  (if there is a tie, we arbitrarily pick one as  $c_p$ ); // We call  $c_p$  as the center of  $p$ .
- 4 **while**  $d_{\max} > \bar{r}$  **do**
- 5     Let  $q = \arg \max_{p \in X} \text{dis}(p, E)$ , and add  $q$  to  $E$ .
- 6     Update  $d_{\max}$ ,  $c_p$  and  $\text{dis}(p, E)$  for each  $p \in X$ , and the set  $C_e = \{p \mid p \in X \ \& \ c_p = e\}$  for each  $e \in E$ ; // We call the set  $C_e$  as the cover set of  $e$ , and it is used in our following DBSCAN algorithms.
- 7 **end**
- 8 **return**  $E$

---

Below we analyze the complexity of Algorithm 1 under Assumption 1 introduced in Section 1.1.2.

**LEMMA 1.** *Suppose  $\Delta$  is the maximum pairwise distance of  $X_{\text{in}}$  and  $z = |X_{\text{out}}|$ . Algorithm 1 runs at most  $O((\frac{\Delta}{\bar{r}})^D) + z$  iterations, and each iteration takes  $O(nt_{\text{dis}})$  time.*

**PROOF.** The time complexity for each iteration is easy to obtain. When a new point  $q$  is added to  $E$ , we can update  $\text{dis}(p, E) = \min\{\text{dis}(p, E \setminus \{q\}), \text{dis}(p, q)\}$  in  $O(t_{\text{dis}})$  time for each  $p \in X$ ; the value  $d_{\max}$  then can be updated in  $O(n)$  time. Similarly, we can update all the centers  $c_p$ s and cover sets  $C_e$ s in  $O(n)$  time. So the overall time complexity for each iteration is  $O(nt_{\text{dis}})$ .

Now we focus on the maximum number of iterations. We first consider the inliers  $X_{\text{in}}$ . According to Algorithm 1, for any  $e_1, e_2 \in E$ , we claim that  $\text{dis}(e_1, e_2) \geq \bar{r}$ ; otherwise, Algorithm 1 should terminate earlier, and  $e_1$  or  $e_2$  cannot be added to  $E$ . Thus  $\min_{e_1, e_2 \in E} \text{dis}(e_1, e_2) \geq \bar{r}$ . The upper bound of the distance between two points in  $E \cap X_{\text{in}}$  is  $\Delta$ . So, according to Proposition 1,  $|E \cap X_{\text{in}}| \leq 2^{D \lceil \log \frac{\Delta}{\bar{r}} \rceil}$ . Also, we have  $|E \cap X_{\text{out}}| \leq z$ , where  $z = |X_{\text{out}}|$ . Therefore,  $|E| = |E \cap X_{\text{in}}| + |E \cap X_{\text{out}}| \leq 2^{D \lceil \log \frac{\Delta}{\bar{r}} \rceil} + z = O((\frac{\Delta}{\bar{r}})^D) + z$ , and Algorithm 1 runs for  $|E|$  iterations.  $\square$

**REMARK 3.** *We would like to emphasize that the number of iterations of Algorithm 1 usually is much lower than the theoretical bound in Lemma 1, especially when applying it to the DBSCAN problem. For example, a DBSCAN cluster can have any shape (e.g., a "banana" shape as shown in Figure 5) rather than a ball shape, i.e., does not fill an entire ball; so the size upper bound " $2^{D \lceil \log \alpha \rceil}$ " in Proposition 1 can be much higher than the real size, which also implies a lower number of iterations in Algorithm 1.*

We also introduce two other lemmas relevant to Algorithm 1, which are used in our following analysis. For any  $p \in X$ , its **neighbor ball center set** is:

$$\mathcal{A}_p = \{e \mid e \in E, \text{dis}(e, c_p) \leq 2\bar{r} + \epsilon\}, \quad (1)$$

where  $\epsilon$  is the parameter of DBSCAN described in Section 1.1. Note that for each  $p \in X$ , the set  $\mathcal{A}_p$  can be simultaneously obtained without increasing the complexity of Algorithm 1. For each  $q \in E$ , we just keep a set  $\{e \mid e \in E, \text{dis}(e, q) \leq 2\bar{r} + \epsilon\}$ . Note that the center set  $E$  is generated incrementally in Algorithm 1. When a new point  $e$  is added to  $E$ , we just update the set for each  $q \in E$ . Then the set  $\mathcal{A}_p$  is also obtained for each  $p \in X$  when the algorithm terminates. The approach of [17] also defines the neighbor region for each point  $p$  in a similar way as (1). But a major difference is that their approach needs an estimated upper bound “ $z$ ” for the number of outliers in their definition, which could be hard to obtain; moreover, it also needs to manually set the termination condition (please see Section 3.3 for the detailed discussion).

**LEMMA 2.** *For any  $p \in X$ , we have  $X \cap \mathbb{B}(p, \epsilon) \subseteq \cup_{e \in \mathcal{A}_p} C_e$ .*

Lemma 2 can be proved by using the triangle inequality. Fix a point  $p \in X$ , and we suppose  $q \in X$  with  $\text{dis}(c_p, c_q) > 2\bar{r} + \epsilon$  (i.e.,  $q \in \cup_{e \notin \mathcal{A}_p} C_e$ ). Then we have  $\text{dis}(p, q) \geq \text{dis}(c_p, c_q) - \text{dis}(p, c_p) - \text{dis}(q, c_q) > 2\bar{r} + \epsilon - 2\bar{r} = \epsilon$ . Therefore, we know  $q \notin X \cap \mathbb{B}(p, \epsilon)$ , and consequently  $X \cap \mathbb{B}(p, \epsilon) \subseteq X \setminus \cup_{e \notin \mathcal{A}_p} C_e = \cup_{e \in \mathcal{A}_p} C_e$ . From Lemma 2, we know that it is sufficient to only consider the set  $\cup_{e \in \mathcal{A}_p} C_e$  for determining that whether  $p$  is a core point. This is particularly useful to reduce the neighborhood query complexity for DBSCAN.

**LEMMA 3.** *For each point  $p \in X$ , we have  $|\mathcal{A}_p| = O((\frac{\epsilon}{\bar{r}})^D + z)$ .*

**PROOF.** Note that  $|\mathcal{A}_p| = |\mathcal{A}_p \cap X_{\text{in}}| + |\mathcal{A}_p \cap X_{\text{out}}|$ . Since  $z = |X_{\text{out}}|$ , we have  $|\mathcal{A}_p \cap X_{\text{out}}| \leq z$ . So we only need to consider the size  $|\mathcal{A}_p \cap X_{\text{in}}|$ .

Similar to the proof of Lemma 1, we have  $\min_{e_1, e_2 \in E} \text{dis}(e_1, e_2) \geq \bar{r}$ . For any  $p_1, p_2 \in \mathcal{A}_p$ , we have  $\text{dis}(p_1, p_2) \leq \text{dis}(p_1, p) + \text{dis}(p, p_2) \leq 4\bar{r} + 2\epsilon$ , where the first inequality comes from the triangle inequality and the second inequality comes from the definition of  $\mathcal{A}_p$ . According to Proposition 1, we have  $|\mathcal{A}_p \cap X_{\text{in}}| \leq 2^{\lceil \log \alpha \rceil}$ , where

$$\begin{aligned} \alpha &= \frac{\max_{e_1, e_2 \in \mathcal{A}_p \cap X_{\text{in}}} \text{dis}(e_1, e_2)}{\min_{e_1, e_2 \in \mathcal{A}_p \cap X_{\text{in}}} \text{dis}(e_1, e_2)} \\ &\leq \frac{\max_{e_1, e_2 \in \mathcal{A}_p \cap X_{\text{in}}} \text{dis}(e_1, e_2)}{\min_{e_1, e_2 \in E} \text{dis}(e_1, e_2)} \leq \frac{4\bar{r} + 2\epsilon}{\bar{r}}. \end{aligned} \quad (2)$$

So we have  $|\mathcal{A}_p \cap X_{\text{in}}| \leq 2^{\lceil \log \frac{4\bar{r} + 2\epsilon}{\bar{r}} \rceil} = O((\frac{\epsilon}{\bar{r}})^D)$ . Therefore,  $|\mathcal{A}_p| = O((\frac{\epsilon}{\bar{r}})^D + z)$ .  $\square$

**REMARK 4.** *In the proof of Lemma 3, we directly count all the  $z$  points of  $X_{\text{out}}$  to  $\mathcal{A}_p$ . Actually, this is overly conservative in practice since the outliers usually are scattered and far away from the inliers. So  $|\mathcal{A}_p \cap X_{\text{out}}|$  usually is much less than  $z$ , and thus  $|\mathcal{A}_p|$  is also less than the theoretical bound  $O((\frac{\epsilon}{\bar{r}})^D + z)$ .*

### 3 A FASTER EXACT METRIC DBSCAN ALGORITHM

From Section 1.1.1, we know that the DBSCAN algorithm contains three key steps: (1) label all the core points; (2) merge the core points to form the clusters; (3) identify the border points and outliers. In this section, we analyze these three steps separately and show that our proposed radius-guided Gonzalez’s algorithm can help us to significantly reduce the overall complexity. First, we consider the problem under Assumption 1, and then show that the algorithm can be simplified

if Assumption 1 is replaced by a stronger assumption that the whole input data (including inliers and outliers) has a low doubling dimension.

### 3.1 Our Algorithm under Assumption 1

As the pre-processing, we run Algorithm 1 with  $\bar{r} = \epsilon/2$  on the given instance  $X$ . We also obtain the set  $\mathcal{A}_p$  for each  $p \in X$ . According to Lemma 3 and the assumption that  $D$  is a constant, we have

$$|\mathcal{A}_p| = O(z). \quad (3)$$

**Step (1): label the core points.** First, for any point  $p \in X$ , from the triangle inequality we know that

$$C_{c_p} \subseteq \mathbb{B}(p, \epsilon) \cap X, \quad (4)$$

where  $C_{c_p}$  is the cover set of  $c_p$  stored in Algorithm 1 and the size  $|C_{c_p}|$  can be obtained immediately in  $O(1)$  time. So if  $|C_{c_p}| \geq \text{MinPts}$ , we know  $|\mathbb{B}(p, \epsilon) \cap X| \geq \text{MinPts}$  as well; then we can safely label  $p$  as a core point. For the remaining points, we check their local regions based on  $\mathcal{A}_p$ . Let  $p$  be an unlabeled point. According to Lemma 2, we only need to count the size

$$\left| \mathbb{B}(p, \epsilon) \cap \left( \bigcup_{e \in \mathcal{A}_p} C_e \right) \right|. \quad (5)$$

The point  $p$  is a core point if and only if the size is  $\geq \text{MinPts}$ . Thus a natural idea for bounding the complexity of step (1) is to prove an upper bound for the size  $|\bigcup_{e \in \mathcal{A}_p} C_e|$ . Unfortunately, this size can be large in reality (e.g., one  $C_e$  can be very dense and contain a large number of points). To resolve this issue, a cute idea here is to consider the amortized query complexity over all the points of  $X$ ; we show that the time for labeling the core points is linear in  $n$ .

**LEMMA 4.** *The time complexity of Step (1) is  $O(nzt_{\text{dis}})$ .*

**PROOF.** Define  $E_1 = \{e \in E \mid |C_e| \geq \text{MinPts}\}$  and  $E_2 = E \setminus E_1$ , where  $E$  is the set returned by Algorithm 1. According to the previous analysis, the time complexity of step (1) is

$$\begin{aligned} & \sum_{e \in E_1} O(|C_e|) + \sum_{e \in E_2} \sum_{e' \in \mathcal{A}_e} O(|C_e||C_{e'}| \cdot t_{\text{dis}}) \\ &= O(n) + O(\text{MinPts}) \cdot \sum_{e \in E_2} \sum_{e' \in \mathcal{A}_e} O(|C_{e'}| \cdot t_{\text{dis}}), \end{aligned} \quad (6)$$

where the second equality comes from the facts  $\sum_{e \in E_1} O(|C_e|) \leq n$  and  $|C_e| < \text{MinPts}$  for any  $e \in E_2$ . We focus on the term “ $\sum_{e \in E_2} \sum_{e' \in \mathcal{A}_e} O(|C_{e'}| \cdot t_{\text{dis}})$ ” of (6). A key observation is that  $e' \in \mathcal{A}_e$  if and only if  $e \in \mathcal{A}_{e'}$ . Using this property, we can exchange the order of summation for  $e$  and  $e'$ , i.e.,

$$\sum_{e \in E_2} \sum_{e' \in \mathcal{A}_e} O(|C_{e'}| \cdot t_{\text{dis}}) = \sum_{e' \in E_1 \cup E_2} \sum_{e \in \mathcal{A}_{e'} \cap E_2} O(|C_{e'}| \cdot t_{\text{dis}}). \quad (7)$$

Note that  $O(|C_{e'}| \cdot t_{\text{dis}})$  is independent with  $e$ , so we have

$$\begin{aligned} (7) &= \sum_{e' \in E_1 \cup E_2} O(|C_{e'}| \cdot t_{\text{dis}}) \cdot O(|\mathcal{A}_{e'}|). \\ &= O(n \cdot t_{\text{dis}}) \cdot O(z), \end{aligned} \quad (8)$$

where the last equality comes from  $\sum_{e' \in E_1 \cup E_2} O(|C_{e'}|) = O(n)$  and  $|\mathcal{A}_{e'}| = O(z)$  in (3). Therefore, through combining (6) and (8), we have the total complexity of Step (1)

$$\begin{aligned} &= O(n) + O(\text{MinPts}) \cdot O(n \cdot t_{\text{dis}}) \cdot O(z) \\ &= O(\text{MinPts} \cdot n \cdot z \cdot t_{\text{dis}}) \\ &= O(nzt_{\text{dis}}), \end{aligned} \quad (9)$$



if assuming  $MinPts$  is a constant as discussed in Remark 1.  $\square$

**Step (2): merge the core points to form the DBSCAN clusters.** We consider the set  $E$  obtained in Algorithm 1. For each  $e \in E$ , denote by  $\tilde{C}_e$  the set of core points in  $C_e$  which are labeled in Step (1) (if  $\tilde{C}_e = \emptyset$ , we just simply ignore  $C_e$ ). First, all the core points of  $\tilde{C}_e$  should be merged into the same cluster; then through the triangle inequality, we only need to consider the core points of  $\tilde{C}_{e'}$  with  $e' \in \mathcal{A}_e$ . The core points of  $\tilde{C}_e$  and  $\tilde{C}_{e'}$  should be merged into the same cluster, if and only if their smallest pairwise distance  $\text{dis}(\tilde{C}_e, \tilde{C}_{e'}) = \min_{p \in \tilde{C}_e, q \in \tilde{C}_{e'}} \text{dis}(p, q) \leq \epsilon$ . Please see Figure 2 for an illustration.

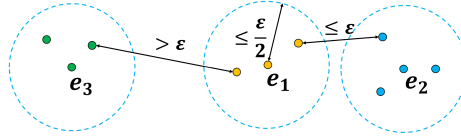


Fig. 2. The sets  $\tilde{C}_{e_1}$  (yellow points),  $\tilde{C}_{e_2}$  (blue points), and  $\tilde{C}_{e_3}$  (green points) are shown in the figure.  $\tilde{C}_{e_1}$  and  $\tilde{C}_{e_2}$  should be merged into the same cluster, because their closest pair distance is less than  $\epsilon$ ; on the other hand,  $\tilde{C}_{e_3}$  should be merged to a different cluster

Actually this is the classical *bichromatic closest pair (BCP)* problem in computational geometry [1]. Directly solving the BCP problem in a general metric space or a high-dimensional space can be quite challenging, *e.g.*, recently Rubinstein [57] showed that it requires at least nearly quadratic time to compute even only an approximate solution. A good news is that we have already distinguished the core points from other points in Step (1); that is, we can apply the indexing structure cover tree (Section 1.1.3) to solve the BCP problem efficiently under Assumption 1. Specifically, we build a cover tree for  $\tilde{C}_e$ , and traverse all the elements in  $\tilde{C}_{e'}$  and search for their nearest neighbors in  $\tilde{C}_e$ ; we return the smallest distance among all these nearest neighbor queries over  $\tilde{C}_{e'}$  as the solution of the BCP problem for the couple  $(\tilde{C}_e, \tilde{C}_{e'})$ .

**LEMMA 5.** *The time complexity of Step (2) is  $O(nz \log \frac{\epsilon}{\delta} \cdot t_{\text{dis}})$ , where  $\delta$  is the smallest pairwise distance of  $X_{\text{in}}$ .*

**PROOF.** First, we need to build a cover tree for each  $\tilde{C}_e$ ,  $e \in E$ . According to Claim 1, we know that it takes  $O(|\tilde{C}_e| \log \Phi_e \cdot t_{\text{dis}})$  time to build a cover tree on  $\tilde{C}_e$ , where  $\Phi_e$  is the aspect ratio of  $\tilde{C}_e$ . Obviously,  $\Phi_e \leq \frac{\epsilon}{2\delta}$ . Therefore, the total time complexity to build the cover trees for all  $\tilde{C}_e$ s is:

$$\sum_{e \in E} O(|\tilde{C}_e| \log \Phi_e t_{\text{dis}}) = O(n \log \frac{\epsilon}{\delta} \cdot t_{\text{dis}}). \quad (10)$$

Next, for each  $\tilde{C}_e$  and its each neighbor  $e' \in \mathcal{A}_e$ , we solve the BCP problem of  $\tilde{C}_e$  and  $\tilde{C}_{e'}$ . According to Claim 1, the overall time complexity of this procedure is

$$\begin{aligned} & \sum_{e \in E} \sum_{e' \in \mathcal{A}_e} |\tilde{C}_e| O(\log \Phi_{e'} \cdot t_{\text{dis}}) \\ &= \sum_{e \in E} |\tilde{C}_e| |\mathcal{A}_e| O(\log \frac{\epsilon}{\delta} \cdot t_{\text{dis}}) \\ &= O(nz \log \frac{\epsilon}{\delta} \cdot t_{\text{dis}}). \end{aligned} \quad (11)$$

It is easy to see the complexity (11) dominates the complexity (10). So we obtain the complexity of Step (2) that is  $O(nz \log \frac{\epsilon}{\delta} \cdot t_{\text{dis}})$ .  $\square$

**Step (3): identify the border points and outliers.** We can apply the similar idea used in Step (1). For a non-core point  $p$ , just find its nearest core point  $c$  in  $\cup_{e \in \mathcal{A}_p} \tilde{C}_e$ . If  $\text{dis}(p, c) \leq \epsilon$ , we put  $p$  to the same cluster as  $c$  and label it as a border point; otherwise, label  $p$  as an outlier.

**LEMMA 6.** *The time complexity of Step (3) is  $O(nzt_{\text{dis}})$ .*

**PROOF.** Similar to the proof of lemma 4, we define  $E_1 = \{e \in E \mid |C_e| \geq \text{MinPts}\}$  and  $E_2 = E \setminus E_1$ . According to the definitions of border point and outlier, we know that a non-core point cannot locate inside some  $C_e$  with  $e \in E_1$ . Therefore, the time complexity of Step (3) is

$$\sum_{e \in E_2} \sum_{e' \in \mathcal{A}_e} O(|C_e||C_{e'}| \cdot t_{\text{dis}}), \quad (12)$$

which is as same as the second term of equation (6). So by using the similar idea of Lemma 4, we know the time complexity of step (3) is also  $O(nzt_{\text{dis}})$ .  $\square$

**The overall time complexity.** From the above Lemma 4, Lemma 5, and Lemma 6, together with the complexity of Algorithm 1, we know the overall time complexity for solving the DBSCAN problem under Assumption 1 is  $O\left(n\left(\left(\frac{\Delta}{\epsilon}\right)^D + z \log \frac{\epsilon}{\delta}\right)t_{\text{dis}}\right)$  which is linear in the input size  $n$ . Also note that usually  $z \ll n$ . For example, in most of the test instances in our experiments and the previous articles (e.g., [26][40][59]),  $z$  is less than 1%  $n$ .

**REMARK 5 (THE SETTING FOR  $\bar{r}$ ).** *In our proposed method, we set  $\bar{r} = \epsilon/2$  for simplicity when running Algorithm 1. Actually, we can prove that any  $\bar{r} \leq \epsilon/2$  works for our algorithm. We also would like to emphasize that the pre-processing step (i.e., Algorithm 1) only needs to be performed once, if we have an estimated lower bound  $\epsilon_0$  for  $\epsilon$  and set  $\bar{r} = \epsilon_0/2$ . When we magnify the parameter  $\epsilon$  or adjust the parameter  $\text{MinPts}$ , we only need to re-compute the  $\mathcal{A}_p$  sets and run the above Step (1)-(3) without running Algorithm 1 again. So our method is quite efficient for parameter tuning, which is a significant advantage for practical implementation. We also discuss this advantage in Section 5.5.*

### 3.2 When Outliers Also Have Low Doubling Dimension

In this section, we consider the scenario that the whole input data  $X$  has a low doubling dimension  $D$ , which is a more stringent version of Assumption 1. We show that the algorithm of the exact DBSCAN in Section 3.1 can be further simplified.

**Our main idea.** With a slightly abuse of notations, we still use  $\Delta$  and  $\delta$  to denote the maximum and minimum pairwise distances of  $X$ , respectively. So the aspect ratio of  $X$  is  $\Phi = \Delta/\delta$ . The main idea is to build a cover tree  $T$  for the whole input data  $X$  since we assume it has a low doubling dimension  $D$ . Then, we can directly obtain the ball center set  $E$  without running Algorithm 1. Let  $i_0 = \lfloor \log \epsilon/2 \rfloor$ . The set of nodes at the  $i_0$ -th level of  $T$ , which is denoted as  $T_{i_0}$ , is an  $\bar{r}$ -net with  $\bar{r} = 2^{\lfloor \log \epsilon/2 \rfloor}$  for  $X$ . So we can simply let  $E = T_{i_0}$ . For each  $p \in X$ , we also define the neighbor ball center set  $\mathcal{A}_p$  as (1). A key point here is that we can prove a much lower bound for  $|\mathcal{A}_p|$ .

**LEMMA 7.** *If  $(X, \text{dis})$  is a metric space with constant doubling dimension  $D$ , then for any  $p \in X$ , we have  $|\mathcal{A}_p| = O(1)$ .*

**PROOF.** Since  $E$  is an  $\bar{r}$ -net, then  $\min_{e_1, e_2 \in E} \text{dis}(e_1, e_2) \geq \bar{r}$ . For any  $p_1, p_2 \in \mathcal{A}_p$ , we have  $\text{dis}(p_1, p_2) \leq \text{dis}(p_1, p) + \text{dis}(p, p_2) \leq 4\bar{r} + 2\epsilon$ , where the first inequality comes from the triangle inequality and the second inequality comes from the definition of  $\mathcal{A}_p$ . According to Proposition 1 and using the same manner of the equality (2), we have  $|\mathcal{A}_p| \leq 2^{D \lceil \log \alpha \rceil}$ , where  $\alpha \leq \frac{4\bar{r} + 2\epsilon}{\bar{r}}$ . So we have  $|\mathcal{A}_p| \leq 2^{\lceil \log \frac{4\bar{r} + 2\epsilon}{\bar{r}} \rceil D} = O\left(\left(\frac{\epsilon}{\bar{r}}\right)^D\right)$ . By substituting  $\bar{r} = 2^{\lfloor \log \epsilon/2 \rfloor} \geq 2^{\log \epsilon/2 - 1} = \epsilon/4$ , we immediately have  $|\mathcal{A}_p| = O(4^D) = O(1)$ .  $\square$

Then we can prove the time complexity of the exact DBSCAN algorithm with the new  $O(1)$  bound of  $|\mathcal{A}_p|$ . In particular, we replace the previous  $O(z)$  bound by  $O(1)$  in Lemma 4, Lemma 5, and Lemma 6, respectively. Finally, we obtain the following theorem.

**THEOREM 1.** *If the metric space  $(X, \text{dis})$  has constant doubling dimension, our exact DBSCAN algorithm can be completed in  $O(n \log \Phi_{\text{dis}})$  time.*

### 3.3 Remark on The Comparison with [17]

As mentioned in Section 1, Ding et al. [17] proposed a randomized  $k$ -center clustering based DBSCAN algorithm. Here, we elaborate on the differences between our proposed method and their algorithm.

The first major difference comes from the way for pre-processing input data. In [17], they use a  $k$ -center with outliers algorithm to partition the data into  $k$  balls. But their algorithm needs to set the parameter “ $\tilde{z}$ ” as an estimated upper bound for the number of outliers, and the termination condition has to be manually set in practice. Thus a drawback of this method is that any improper parameter setting (e.g., underestimate the number of outliers) may lead to high computational complexity or even returning an incorrect DBSCAN result. In contrast, our pre-processing method, the Radius-guided Gonzalez algorithm proposed in Section 2, does not need to set those parameters and only constructs an  $\epsilon/2$ -net in the data space. Moreover, our method is a deterministic approach while the method of [17] is based on a “randomized”  $k$ -center with outliers algorithm that always has a failure probability.

Secondly, we have fundamentally different ways for constructing the DBSCAN clusters. Recall the three steps of DBSCAN as described in Section 3.1. Comparing with the original DBSCAN algorithm [23], the only improvement proposed by [17] is using the aforementioned randomized  $k$ -center with outliers algorithm to reduce the search range when performing brute force search for  $\epsilon$ -neighborhoods of each point. This is a heuristic improvement for Step (1), and its time complexity remains  $O(n^2)$  in the worst case. In contrast, our proposed algorithm attempts to establish an  $\epsilon/2$ -net data structure and convert the DBSCAN problem to a connectivity problem among the  $\epsilon/2$ -net spheres. Specifically, in Step (1), we divide the  $\epsilon/2$ -net spheres into two categories: dense and sparse spheres (corresponding to  $E_1$  and  $E_2$  in the proof of Lemma 4). Each point in dense spheres can be immediately marked as a core point, and then the complexity of Step (1) is reduced to be linear. In Step (2), instead of considering the connection between individual core points, we consider the connection between  $\epsilon/2$ -net spheres. Together with the cover tree technique, the time complexity of Step (2) is also improved to be linear. Consequently, our overall time complexity is linear in  $n$ .

Furthermore, another advantage of our techniques is that they can be extended to solve the approximate and streaming DBSCAN problems, as shown in the following Section 4.

## 4 $\rho$ -APPROXIMATE METRIC DBSCAN

In this section, we present a linear time  $\rho$ -approximate DBSCAN algorithm that can be applied to arbitrary metric space. The original  $\rho$ -approximate DBSCAN algorithm in [26] is based on the grid method in Euclidean space  $\mathbb{R}^d$ . However, their method cannot be directly extended to general metric space; moreover, it takes an  $O(n \cdot (1/\rho)^{d-1})$  running time that is unaffordable in practice for large  $d$ .

Comparing with our exact DBSCAN algorithm in Section 3.1, though both of the two algorithms have the time complexities linear in  $n$ , the approximate algorithm compresses the data into a small-size summary and thus it can be easily implemented for streaming data. Also in practice (as shown in our experiments), the approximate algorithm usually is faster than the exact algorithm.

The main reason is that for exact DBSCAN, we need to solve a number of BCP problems when merging the core points (Step (2) in Section 3.1). Although this step can be accelerated by using the cover tree, it is still relatively time-consuming in practice. In our approximate DBSCAN algorithm we can avoid this step by using the “summary” idea as shown below.

#### 4.1 Linear Time Algorithm via Core Points Summary

**Our main idea.** As shown in Section 3, we can apply our proposed radius-guided Gonzalez’s algorithm to efficiently reduce the complexity for labeling core points, border points, and outliers. To merge the core points to form the DBSCAN clusters, we consider to build a “summary”  $\mathcal{S}_*$  for the set of core points  $\cup_{e \in E} \tilde{C}_e$ , which is different to the cover tree idea. The summary set should satisfy two conditions: (1)  $|\mathcal{S}_*| \ll |\cup_{e \in E} \tilde{C}_e|$  and (2) we can correctly generate the  $\rho$ -approximate clusters (as Definition 2) through  $\mathcal{S}_*$ . Our idea for constructing  $\mathcal{S}_*$  is as follows. We check each point  $e \in E$  and its corresponding set  $C_e$  (obtained in Algorithm 1 with an appropriate  $\bar{r}$ ). If  $e$  is a core point, we add it to  $\mathcal{S}_*$  and ignore all the other points of  $C_e$ ; otherwise, we add all the core points  $\tilde{C}_e$  to  $\mathcal{S}_*$ . The details are shown in Algorithm 2.

Though the above construction method for  $\mathcal{S}_*$  is simple, it is challenging to prove that it satisfies the two conditions. Below, we first show its correctness, that is, we can generate the  $\rho$ -approximate clusters through  $\mathcal{S}_*$ . Here we slightly modify the previous definition of the **neighbor ball center set**  $\mathcal{A}_p$  in (1). Recall that for any  $p \in X$ , we define  $\mathcal{A}_p = \{e | e \in E, \text{dis}(e, c_p) \leq 2\bar{r} + \epsilon\}$ . Now we change it to

$$\mathcal{A}_p = \{e | e \in E, \text{dis}(e, c_p) \leq 4\bar{r} + \epsilon\}. \quad (13)$$

Since we enlarge the set  $\mathcal{A}_p$  in this modification, the result of Lemma 2 is still true. Also, this modification does not change the asymptotic upper bound  $O((\frac{\epsilon}{\bar{r}})^D + z)$  of  $|\mathcal{A}_p|$  in Lemma 3. We set  $\bar{r} = \frac{\rho\epsilon}{2}$  in Algorithm 2, and then the equation (3) is replaced by:

$$|\mathcal{A}_p| = O\left(\left(\frac{1}{\rho}\right)^D + z\right). \quad (14)$$

**THEOREM 2 (CORRECTNESS).** *Algorithm 2 can correctly return a  $\rho$ -approximate DBSCAN solution.*

**PROOF.** We prove the correctness following Definition 2. The **connectivity** is easy to verify, since any two points are connected only when their distance is no larger than  $(1 + \rho)\epsilon$  in the algorithm. So we only need to prove the **maximality** of the obtained clusters. Namely,  $\forall p, q$ , if  $q$  is a core point and  $p$  is density-reachable from  $q$ , then  $p$  should have the same cluster ID with  $q$ .

To prove the **maximality**, we need to guarantee the uniqueness of the cluster ID for each core point  $q$ . If  $q \in \mathcal{S}_*$ , the uniqueness is satisfied when we merge inside  $\mathcal{S}_*$ ; else,  $q$  has the same cluster ID with  $c_q$ . So the uniqueness is guaranteed for all the core points, which implies that every core point belongs to exactly one cluster. Then we consider any two points  $p, q \in X$  with  $\text{dis}(p, q) \leq \epsilon$ . Let  $q$  be a core point and  $s_q$  be the “representative” of  $q$  in  $\mathcal{S}_*$  (according to the construction of  $\mathcal{S}_*$ , we let  $s_q = c_q$  if  $c_q$  is a core point; otherwise,  $s_q = q$ ).

**Case 1:** If  $p$  is also a core point (its representative in  $\mathcal{S}_*$  is denoted by  $s_p$ ), then from the triangle inequality we know

$$\text{dis}(s_p, s_q) \leq \text{dis}(s_p, p) + \text{dis}(p, q) + \text{dis}(q, s_q) \leq (1 + \rho)\epsilon. \quad (15)$$

It implies that  $s_p$  and  $s_q$  should have the same cluster ID; together with the ID uniqueness, we know  $p$  and  $q$  should have the same ID as well.

**Case 2:** If  $p$  is not a core point, actually  $p$  should be a border point for this case. If  $c_p \in \mathcal{S}_*$ , from the similar manner of (15) we know  $\text{dis}(c_p, s_q) \leq (1 + \rho)\epsilon$ . So  $c_p$  and  $s_q$  should be merged when we merge inside  $\mathcal{S}_*$ . Hence,  $p$  and  $q$  have the same cluster ID. If  $c_p \notin \mathcal{S}_*$ , we have  $\text{dis}(p, s_q) \leq$

**Algorithm 2:** METRIC  $\rho$ -APPROXIMATE DBSCAN VIA CORE POINTS SUMMARY

---

**Input:**  $\epsilon, MinPts, \rho, X$

- 1 Run Algorithm 1 with  $\bar{r} = \frac{\rho\epsilon}{2}$  and  $X$ , let  $E$  be the output set. We can also identify the core points of  $E$  after running Algorithm 1 (see our explanation in the proof of Lemma 10).  
Initialize  $\mathcal{S}_* \leftarrow \emptyset$ . ; // the summary for core points
- 2 **for each**  $e \in E$  **do**
- 3     **if**  $e$  is a core point **then**
- 4         | Add  $e$  to  $\mathcal{S}_*$ .
- 5     **else**
- 6         | Add all the core points in  $C_e$  to  $\mathcal{S}_*$ .
- 7     **end**
- 8 **end**
- 9 **Merge inside**  $\mathcal{S}_*$ : for any pair of  $c_1, c_2 \in \mathcal{S}_*$ , label them with the same cluster ID if  $\text{dis}(c_1, c_2) \leq (1 + \rho)\epsilon$ .
- 10 **Label other points:** for  $p \in X \setminus \mathcal{S}_*$  **do**
- 11     **if**  $c_p \in \mathcal{S}_*$  **then**
- 12         | Label  $p$  with the same cluster ID of  $c_p$ .
- 13     **else**
- 14         **if** there exists some  $s \in \mathcal{S}_*$  such that  $\text{dis}(p, s) \leq (\frac{\rho}{2} + 1)\epsilon$  **then**
- 15             | Label  $p$  with the same cluster ID of  $s$ .
- 16         **else**
- 17             | Label  $p$  as an outlier.
- 18         **end**
- 19     **end**
- 20 **end**

---

$\text{dis}(p, q) + \text{dis}(q, s_q) \leq (\frac{\rho}{2} + 1)\epsilon$ . So we label  $p$  with the same cluster ID with  $s_q$ . Also,  $s_q$  and  $q$  have same cluster ID, and thus  $p$  and  $q$  have the same cluster ID.

Overall, we know that the **maximality** is guaranteed from the above two cases. So algorithm 2 returns a qualified  $\rho$ -approximate DBSCAN solution.  $\square$

Now, we analyze the complexity of Algorithm 2, where the key is to prove the upper bounds for the complexity of constructing the summary  $\mathcal{S}_*$  and the induced query complexity in the labeling procedure. The idea is similar with the proof of Lemma 4, which considers the amortized complexity over the whole input instance.

**THEOREM 3 (COMPLEXITY).** *Let  $\rho \leq 2$ . Algorithm 2 runs in  $O\left(n\left(\left(\frac{\Delta}{\rho\epsilon}\right)^D + z\right) \cdot t_{\text{dis}}\right)$  time.*

Usually  $\rho$  is a small pre-specified number, since it measures the approximation degree and we do not want the result to be far away from the exact DBSCAN. So we suppose  $\rho \leq 2$  here (actually our analysis also works for the case  $\rho > 2$  with slight modification). To prove Theorem 3, we need to introduce some key lemmas first.

**LEMMA 8.** *For each  $e \in E$ ,  $|C_e \cap \mathcal{S}_*| \leq MinPts$ .*

**PROOF.** If  $e$  is a core point, then  $C_e \cap \mathcal{S}_* = \{e\}$  and obviously the size is no more than  $MinPts$ .

If  $e$  is not a core point, it implies  $|\mathbb{B}(e, \epsilon)| \leq \text{MinPts}$ . Since we let  $\rho \leq 2$ , then  $\frac{\rho\epsilon}{2} \leq \epsilon$  and  $C_e \subseteq \mathbb{B}(e, \epsilon)$ . Therefore,

$$|C_e \cap \mathcal{S}_*| \leq |C_e| \leq |\mathbb{B}(e, \epsilon)| \leq \text{MinPts}. \quad (16)$$

□

**LEMMA 9.** *The size of  $\mathcal{S}_*$  is  $O((\frac{\Delta}{\rho\epsilon})^D + z)$ .*

**PROOF.** The size of  $\mathcal{S}_*$  is

$$|\mathcal{S}_*| = \sum_{e \text{ is core point}} 1 + \sum_{e \text{ is not core point}} O(|C_e|). \quad (17)$$

According to the inequality (16), we have  $|C_e| \leq \text{MinPts}$  if  $e$  is not a core point, and thus

$$\begin{aligned} |\mathcal{S}_*| &= O(|E|) + \sum_{e \text{ is not core point}} O(\text{MinPts}) \\ &= O(\text{MinPts} \cdot |E|) \\ &= O(|E|) = O((\frac{\Delta}{\rho\epsilon})^D + z), \end{aligned} \quad (18)$$

where the last equality follows from Lemma 1. □

**LEMMA 10.** *The construction of  $\mathcal{S}_*$  can be completed in  $O((\frac{1}{\rho})^D + z)nt_{\text{dis}}$  time.*

**PROOF.** For each  $e \in E$ , we can determine whether  $e$  is core point or not when running Algorithm 1 without introducing additional time complexity, since we have the distance between any pair of  $e \in E$  and  $p \in X$ . Hence, in the step of constructing  $\mathcal{S}_*$ , if  $e$  is a core point, it takes  $O(1)$  time to add  $e$  to  $\mathcal{S}_*$ .

If  $e$  is not core point, we need to compute  $|\mathbb{B}(p, \epsilon)|$  for every  $p \in C_e$ . Based on Lemma 2, we only need to consider the set  $\cup_{e' \in \mathcal{A}_e} C_{e'}$ . In this case, determining whether  $p$  is a core point takes  $O(\sum_{e' \in \mathcal{A}_e} |C_{e'}|t_{\text{dis}})$  time.

So the time complexity of constructing  $\mathcal{S}_*$  (denoted by  $t_c$ ) is

$$t_c = \sum_{e \text{ is core point}} O(1) + \sum_{e \text{ is not core point}} \sum_{p \in C_e} \sum_{e' \in \mathcal{A}_e} |C_{e'}|t_{\text{dis}} \quad (19)$$

Similar with the idea for proving lemma 8, we know for a non-core point  $e \in E$ ,  $|C_e| \leq \text{MinPts}$ . As a consequence,

$$t_c = O(|E|) + O(\text{MinPts}) \cdot \sum_{e \text{ is not core point}} \sum_{e' \in \mathcal{A}_e} |C_{e'}|t_{\text{dis}}. \quad (20)$$

We then exchange the summation order in terms of  $e$  and  $e'$  (as the equation (7)) to have

$$\sum_{e \text{ is not core point}} \sum_{e' \in \mathcal{A}_e} |C_{e'}|t_{\text{dis}} = \sum_{e' \in E} \sum_{\substack{e \in \mathcal{A}_{e'} \\ e \text{ is not core point}}} |C_{e'}|t_{\text{dis}}. \quad (21)$$

Note that  $|C_{e'}|t_{\text{dis}}$  is independent with  $e$ , so we have

$$\begin{aligned} t_c &= O(|E|) + O(\text{MinPts}) \cdot \sum_{e'} O(|\mathcal{A}_{e'}|) |C_{e'}|t_{\text{dis}} \\ &= O(n) + O(\text{MinPts}) \cdot ((\frac{1}{\rho})^D + z) \sum_{e'} |C_{e'}|t_{\text{dis}} \\ &= O((\frac{1}{\rho})^D + z)nt_{\text{dis}}, \end{aligned} \quad (22)$$

from the facts  $\sum_{e'} |C_{e'}| = O(n)$  and  $|\mathcal{A}_{e'}| = O((\frac{1}{\rho})^D + z)$  in (14).  $\square$

**LEMMA 11.** *The step of merging inside  $\mathcal{S}_*$  needs  $O\left(\left(\frac{\Delta}{\rho\epsilon}\right)^D + z\right)\left(\left(\frac{1}{\rho}\right)^D + z\right)t_{\text{dis}}$  time.*

**PROOF.** In Algorithm 2, we set  $\bar{r} = \frac{\rho\epsilon}{2}$  for Algorithm 1. Hence, for any pair of  $s_1, s_2 \in \mathcal{S}_*$  whose pairwise distance  $\text{dis}(s_1, s_2) \leq (1 + \rho)\epsilon$ , we have  $\text{dis}(c_{s_1}, c_{s_2}) \leq \text{dis}(s_1, s_2) + \text{dis}(c_{s_1}, s_1) + \text{dis}(c_{s_2}, s_2) \leq (1 + 2\rho)\epsilon$ . According to (13), we have  $s_1 \in \cup_{e \in \mathcal{A}_{s_2}} C_e$  and vice versa. Therefore, when we merge inside  $\mathcal{S}_*$ , for every  $s \in \mathcal{S}_*$ , we only need to search inside  $(\cup_{e \in \mathcal{A}_s} C_e) \cap \mathcal{S}_*$  for the points who have the distance to  $s$  no larger than  $(1 + \rho)\epsilon$ . Hence, the time complexity for merging insides  $\mathcal{S}_*$  is

$$t_m = \sum_{s \in \mathcal{S}_*} \sum_{e \in \mathcal{A}_s} |C_e \cap \mathcal{S}_*| t_{\text{dis}}. \quad (23)$$

According to Lemma 8,  $|C_e \cap \mathcal{S}_*| \leq \text{MinPts}$  and thus

$$t_m = \sum_{s \in \mathcal{S}_*} \sum_{e \in \mathcal{A}_s} O(\text{MinPts}) t_{\text{dis}} = \sum_{s \in \mathcal{S}_*} O(|\mathcal{A}_s| \cdot \text{MinPts}) t_{\text{dis}}. \quad (24)$$

Based on (14) we have

$$\begin{aligned} t_m &= \sum_{s \in \mathcal{S}_*} O\left(\left(\frac{1}{\rho}\right)^D + z\right) \cdot \text{MinPts} t_{\text{dis}} \\ &= O(|\mathcal{S}_*|) \cdot O\left(\left(\frac{1}{\rho}\right)^D + z\right) \cdot \text{MinPts} t_{\text{dis}} \\ &= O\left(\left(\frac{\Delta}{\rho\epsilon}\right)^D + z\right)\left(\left(\frac{1}{\rho}\right)^D + z\right) t_{\text{dis}}, \end{aligned} \quad (25)$$

where the last equality follows from Lemma 9.  $\square$

**LEMMA 12.** *The step of labeling other points (borders and outliers) needs  $n\left(\left(\frac{1}{\rho}\right)^D + z\right)t_{\text{dis}}$  time.*

Lemma 12 can be simply obtained from the proof of Lemma 6 by substituting  $|\mathcal{A}_p| = O(z)$  with  $|\mathcal{A}_p| = \left(\left(\frac{1}{\rho}\right)^D + z\right)$ .

Recall the time complexity of Algorithm 1 with  $\bar{r} = \frac{\rho\epsilon}{2}$  is  $O\left(n\left(\left(\frac{\Delta}{\rho\epsilon}\right)^D + z\right)t_{\text{dis}}\right)$ . Overall, through combining Lemma 10 - Lemma 12, we obtain the complexity of Theorem 3.

**REMARK 6.** *Similar with the discussion in Remark 5, we do not need to run Algorithm 1 repeatedly for  $\rho$ -approximate DBSCAN when tuning the parameters in practice.*

## 4.2 Implementation for Streaming Data

We further consider designing streaming algorithm for  $\rho$ -approximate DBSCAN with the space complexity being independent of the size of  $X$ . The major challenge for designing a streaming algorithm is that the number of core points can be as large as  $O(n)$ , which may result in large memory usage. Fortunately, we show that our proposed core point summary technique in Section 4.1 can help us to neatly circumvent this issue.

**Sketch of our algorithm.** We implement Algorithm 2 in a “streaming” fashion. Our algorithm contains three stages. **Stage 1:** we apply a streaming incremental technique instead of directly running the radius-guided Gonzalez’s algorithm. Our main idea is trying to assign every point in the data stream into existing balls within radius  $\bar{r}$ ; if fail, we just build a new ball centered at this point. In this stage, we aim to obtain the ball centers  $E$ . At the same time, we construct part of the summary  $\mathcal{S}_*$  by adding the core points in  $E$  into it. We also need to keep a set  $\mathcal{M} = \{p \in X \mid c_p \text{ is not core point}\}$ .

Intuitively, the purpose of keeping this set is to identify the potential core points in next stage. **Stage 2:** we identify the other necessary core points that should be added to  $S_*$ . Namely, we add  $p \in \mathcal{M}$  to  $S_*$  if  $p$  is a core point. This ensures the completeness of the constructed summary  $S_*$ . Then, we can perform the merge on  $S_*$  offline in our memory. **Stage 3:** we scan  $X$  in the final pass to assign other points (as line 10-20 in Algorithm 2). Overall, we have the following Theorem 4.

---

**Algorithm 3:** STREAMING  $\rho$ -APPROXIMATE DBSCAN
 

---

**Input:**  $\epsilon, MinPts, X, \rho$

```

1 Initialize  $E = \emptyset, \mathcal{M} = \emptyset$  and  $S_* = \emptyset$ .
2 The first pass: Scan  $X$ , for every  $p \in X$  do
3   if there is no  $e \in E$  such that  $\text{dis}(p, e) \leq \bar{r} = \frac{\rho\epsilon}{2}$  then
4     | Add  $p$  to  $E$ .
5   end
6   for every  $e \in E$  do
7     | if  $\mathbb{B}(e, \epsilon)$  has detected at least  $MinPts$  points then
8       |  $S_* \leftarrow S_* \cup \{e\}$ .
9     | else if  $\text{dis}(p, e) \leq \bar{r} = \frac{\rho\epsilon}{2}$  then
10    |  $\mathcal{M} \leftarrow \mathcal{M} \cup \{p\}$ .
11    | end
12  | end
13 end
14 The second pass: While scanning  $X$ , identify the core points from  $\mathcal{M}$  and add them to  $S_*$ .
15 Merge inside  $S_*$  offline as line 9 of Algorithm 2.
16 The third pass: Label the border points and outliers as line 10-20 of Algorithm 2.

```

---

**THEOREM 4.** Let  $\rho \leq 2$ , our streaming algorithm takes an  $O((\frac{\Delta}{\rho\epsilon})^D + z)$  memory usage (which is independent of  $n$ ) and an  $O(n((\frac{\Delta}{\rho\epsilon})^D + z) \cdot t_{\text{dis}})$  overall running time.

**Sketch of the proof.** By using the similar idea of Lemma 1, we can conclude that  $O(|E|) = O((\frac{\Delta}{\rho\epsilon})^D + z)$ . And from the process of building  $\mathcal{M}$ , we know that  $|\mathcal{M}| = O(MinPts \cdot |E|) = O(|E|)$ . The elements in the set  $S_*$  are selected from  $E$  and  $\mathcal{M}$ , so  $S_*$  does not occupy any extra space. Hence the overall memory usage is  $O(|E| + |\mathcal{M}|) = O((\frac{\Delta}{\rho\epsilon})^D + z)$ . Further, we can show that Algorithm 3 can be completed in  $O(n|E|t_{\text{dis}})$  time, i.e., its time complexity is  $O(n((\frac{\Delta}{\rho\epsilon})^D + z) \cdot t_{\text{dis}})$ .

## 5 EXPERIMENTS

### 5.1 Experimental environment and datasets

Our experiments were conducted on a server equipped with Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz CPU and 512GB memory. We implement our algorithms in C++.

The datasets are shown in Table 1 that includes both low dimensional and high dimensional datasets. In particular, to demonstrate the performance of our algorithm on general metric space, we also consider some non-Euclidean datasets, which include 4 real-world text datasets: **AG News** [74], **COLA** [67], **MNLI** [69], **MRPC**[18]. We use edit distance[48] to measure the distance between

<sup>2</sup>To enhance the density of high dimensional data in the space, we uniformly sampled 1000 data points from the original dataset and then duplicated them 10 times with adding random noise in the range of  $[-5, 5]$  for each dimension. We perform the same operation to process CIFAR 10, USPS HW, and Fashion MNIST.



Table 1. Datasets

Dataset	Dimension	n
Moons [60]	2	10,000
Cancer[19]	32	569
Arrhythmia[30]	262	452
Biodeg[50]	41	1,055
MNIST <sup>2</sup> [47]	784	10000
Fashion MNIST [71]	784	10,000
USPS HW[38]	256	10,000
CIFAR 10[45]	3072	10,000
DEEP1B [3]	96	9,990,000
GIST [42]	960	1,000,000
GloVe25 [53]	25	1,183,514
SIFT[42]	128	1,000,000
PCAM[64]	1024	2,493,440
Spotify_Session[7]	21	2,072,002,577
LSUN[73]	1024	2,943,300
COLA [67]	n/a	515
AG News [74]	n/a	7,600
MRPC [18]	n/a	1,725
MNLI [69]	n/a	9,815

texts. As shown in the table, we also take several million-scale datasets to evaluate the performance. In our experiments, each test instance is repeated 5 times and we report the average result.

## 5.2 Comparison on Running time

We compare the running time with several existing DBSCAN algorithms. In this section, we fix the parameters  $MinPts = 10$  and  $\rho = 0.5$ , and adjust  $\epsilon$  within a reasonable range for each dataset (more detailed discussion on  $\rho$  is shown in Section 5.3).

We compare the running time of our exact metric DBSCAN (denoted by Our\_Exact) and approximate DBSCAN (denoted by Our\_Approx) with the following algorithms: the original DBSCAN [23] (denoted by DBSCAN), DBSCAN++[40], the metric DBSCAN [17] (denoted by DYW\_DBSCAN), the exact and approximate DBSCAN [26] (denoted by GT\_Exact and GT\_Approx, respectively). Note that the clustering performance of the DBSCAN++ algorithm depends on the sampling ratio; we choose the ratio to be 30% as suggested in their paper for achieving promising clustering results.

We show the running time curves in Figure 3. The datasets given in Table 1 can be categorized into four classes corresponding to the four rows of Figure 3 from top to bottom: 4 low/medium-dimensional datasets, 4 high-dimensional datasets, 4 text datasets, and 4 million-scale datasets. Overall, our exact and approximate algorithms can achieve much lower running time comparing with the baselines, especially for large high-dimensional and non-Euclidean datasets. For example, for the largest two datasets GIST and DEEP1B, only our algorithms can complete within  $10^6$ s (about 12 days) on our workstation; for the dataset CIFAR 10, our algorithm runs in almost half the time of the original DBSCAN, and about 1/10 of GT\_Exact and GT\_Approx; for the text dataset AG News, our algorithm takes less than 1% of the time of the original DBSCAN. Also, our proposed approximate DBSCAN algorithm is comparable or faster than our exact DBSCAN algorithm in the experiments.

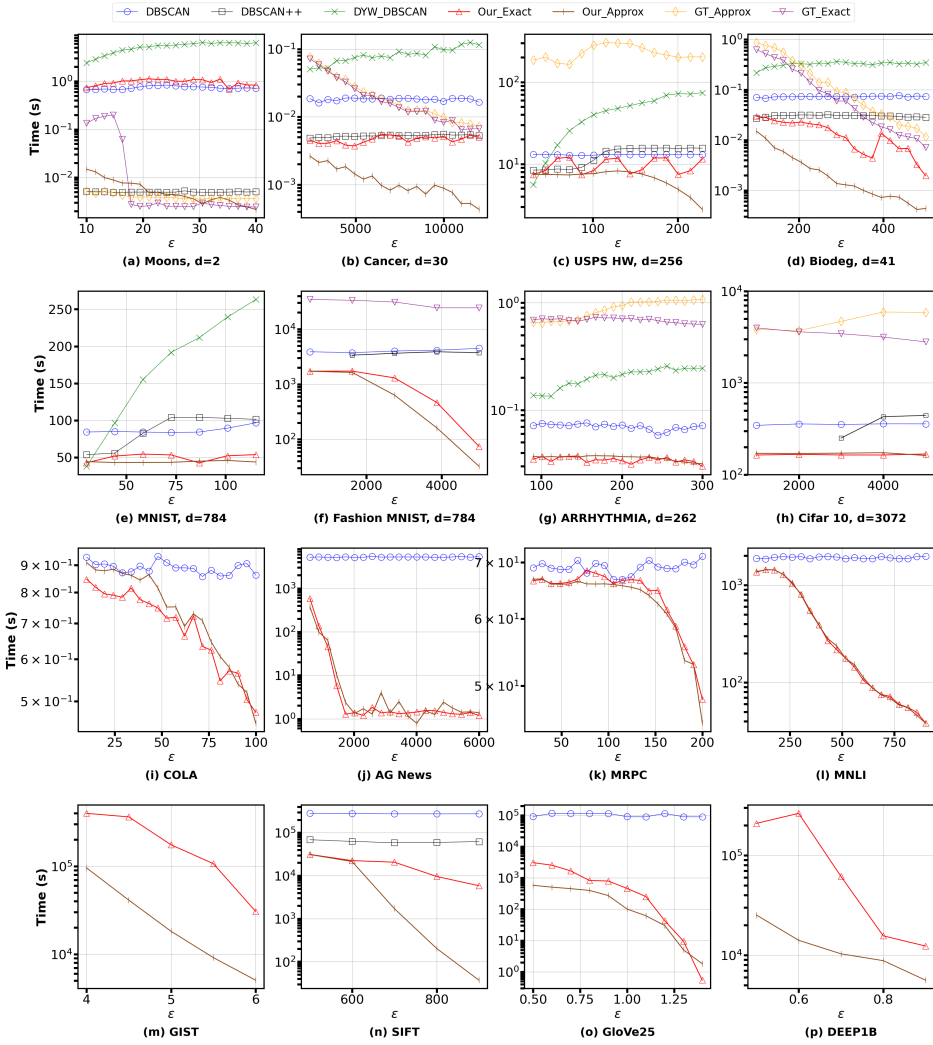


Fig. 3. Running time with varying  $\epsilon$ . Some baseline algorithms are not tested in some figures, because either they run too slowly ( $> 10^6$ s) on the high-dimensional data, or they cannot run on the non-Euclidean data.<sup>3</sup>

### 5.3 Discussion on clustering quality with $\rho$

In this section, we discuss the impact of  $\rho$  on the clustering quality. We run our approximation algorithm with a fixed  $\epsilon$  and different values of  $\rho$  and compare their clustering performance with the exact algorithm. We choose four datasets (MNIST, USPS HW, Fashion MNIST, and CIFAR 10), and we take the widely used measures *Adjusted Rand Index* (ARI) [37] and *Adjusted Mutual Information* (AMI) [65] to evaluate their performances (higher values for ARI and AMI indicate better clustering quality).

<sup>3</sup>In the version published at the SIGMOD '24 conference, we identified an error in the coding of the program, which resulted in incorrect displays of subfigures (i), (j), (k), and (l) in Figure 3. We have corrected this error in the current version of Figure 3. We apologize for this oversight.

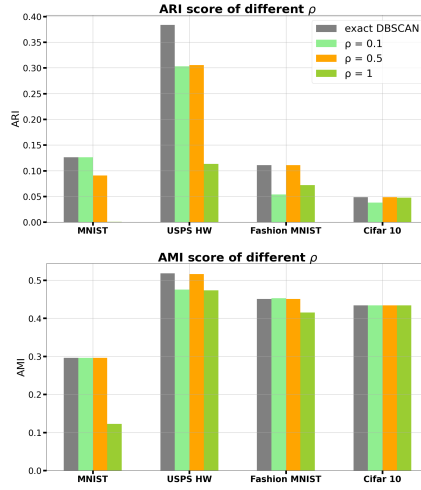


Fig. 4. The ARI and AMI with fixed  $\epsilon$  and different  $\rho$ .

From Figure 4 we can observe that when  $\rho$  is set to 0.5, our approximate algorithm can achieve similar clustering qualities with the exact algorithm for most instances. As an illustration, Figure 5 (a)-(d) show two examples of the clustering results of exact DBSCAN and our approximate algorithm with  $\rho = 0.5$ ; we can see that the results are very close.

**REMARK 7.** *In a few cases of Figure 4, we observe that the ARI or AMI score is slightly improved from  $\rho = 0.1$  to  $\rho = 0.5$ . One possible explanation is that the goal of DBSCAN may not be exactly consistent with the best ARI and AMI values. The change of the clustering performance with the varying of  $\rho$  can be complex. So when  $\rho$  becomes smaller, though the clustering result is closer to the exact DBSCAN, it does not always have increased ARI or AMI. The experiments in [40] also show that DBSCAN may not yield perfect ARI and AMI.*

#### 5.4 Comparison with non-DBSCAN Algorithms

For completeness, we also consider the comparison with some non-DBSCAN clustering algorithms. Specifically, we consider the following baselines: “Dirichlet process (DP)-means” [46] (a non-parametric  $k$ -means clustering algorithm that does not require to input the number of clusters  $k$ ), the streaming  $k$ -means algorithm “BICO” [25], and two density clustering algorithms “Density-peak” [55] and “Meanshift” [14]<sup>4</sup>. For fairness in the comparison, we try our best to tune the parameters for each baseline. It is also worth noting that BICO requires to manually specify the number of clusters  $k$ . In other words, the performance of BICO is likely to have downgrade in practical applications, if the number of clusters is not given. As for DP-means, the cluster penalty parameter  $\lambda$  is set by taking the maximum distance of the  $k$ -center initialization, following the suggested setting in their original paper.

To evaluate their performances on high-dimensional dense datasets, we particularly construct the “MNIST\_noisy” and “Fashion\_noisy” datasets: we duplicate each point in the MNIST and Fashion MNIST datasets 10 times and add a random perturbation in the range of  $[-5, 5]$  to each dimension of each duplicated point, and then randomly generate 1% noisy points within  $[0, 255]^d$  where  $d$  is

<sup>4</sup>We are aware of the recent improved algorithm Meanshift++ [41], but Meanshift++ is only applicable to low-dimensional data, and thus is not suitable for most of our experimental datasets.

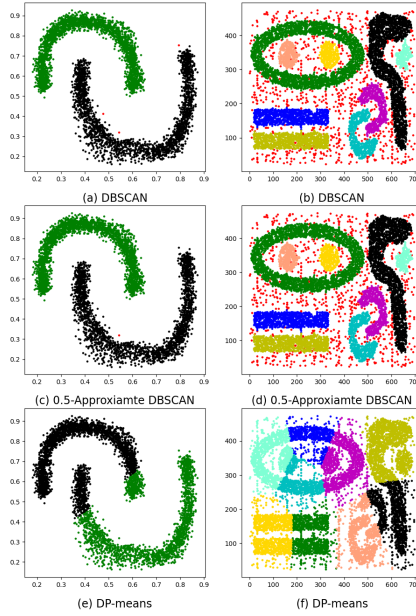


Fig. 5. Clustering results of exact DBSCAN, our approximate algorithm with  $\rho = 0.5$  and DP-means. The points with same color belong to the same cluster, and the red points are outliers.

the dimensionality of the image space. From Table 3, we can see that our exact and approximate DBSCAN outperform the baselines in terms of the ARI and AMI scores on most of the datasets.

As for the runtime, we observe that BICO has similar speed with our algorithms (for example, the average difference on dataset MNIST is less than 20%), and the two baseline density clustering algorithms (Density-peak and Meanshift) are much slower (at least 5 times on average, and Density-peak even encounters memory overflow issue on some large datasets.). The baseline DP-means, though usually has worse clustering performance than our algorithms as shown in Table 3 (two examples are also given in Figure 5 (e)-(f)), is always the fastest one, which is about 12 times faster on average than our algorithms and BICO. We attribute this to the inherent algorithmic simplicity of DP-means than other testing algorithms (especially the density-based algorithms often need to conduct more complicated operations towards spatial neighborhood).

### 5.5 Time Taken by the Radius-guided Gonzalez

As discussed in Remark 5 and Remark 6, we do not need to repeatedly run Algorithm 1 when tuning the parameters  $\epsilon$  or  $MinPts$ . This property is quite useful in practical implementation because the time of Algorithm 1 often takes a large part of the whole DBSCAN procedure. To verify this, we illustrate the runtime proportions taken by Algorithm 1 in our exact DBSCAN algorithm in Table 2.

From Table 2 below, we can see that the runtime of Algorithm 1 takes more than 60% of the total DBSCAN procedure; it indicates that when we increase  $\epsilon$  or adjust  $MinPts$  during parameter tuning, a large amount of runtime can be saved without repeatedly running Algorithm 1. In our approximate algorithm, this ratio is even higher, e.g., in our approximate algorithm on the MNIST dataset, the Radius-Gonzalez procedure takes almost 98% of the total time. We leave the detailed results to our full version.

Table 2. The runtime proportions of Algorithm 1 in our exact DBSCAN algorithm

Dataset	Radius-guided Gonzalez (ms)	Total time (ms)	Proportion
Moons	4.16	4.44	94%
Cancer	5.10	5.75	89%
USPS HW	6090	7994	76%
Biodeg	35.1	38.8	90%
MNIST	40474	43597	93%
Fashion MNIST	5113	5136	99%
Arrhythmia	35.1	38.8	90%
CIFAR 10	155966	156611	99%
COLA	702	779	90%
AG News	73131	114830	64%
MRPC	67148	67158	99%

Table 3. Comparison of ARI and AMI scores with the non-DBSCAN algorithms. The symbol "-" denotes values less than 0.01 and "\*" denotes memory overflow (&gt;500GB)

Dataset	DBSCAN		0.5-approx DBSCAN		DP-means		BICO		Density-peak		Meanshift	
	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI
Moons	<b>1.0</b>	<b>1.0</b>	0.99	0.96	0.26	0.30	0.19	0.19	0.51	0.67	0.39	0.30
Cluto	0.94	0.91	<b>0.95</b>	<b>0.92</b>	0.28	0.40	0.59	0.50	0.41	0.62	0.19	0.33
MNIST	0.25	0.48	0.28	<b>0.51</b>	-	-	<b>0.31</b>	0.41	0.01	0.06	0.02	0.43
MNIST_Noisy	<b>0.28</b>	<b>0.57</b>	<b>0.28</b>	0.51	-	0.05	0.17	0.22	*	*	0.02	0.49
Fashion	0.14	0.46	<b>0.25</b>	<b>0.50</b>	-	-	0.23	0.39	0.02	0.06	0.02	0.43
Fashion_Noisy	0.15	0.52	<b>0.24</b>	<b>0.52</b>	-	0.05	0.14	0.38	*	*	0.02	0.49
PCAM	0.02	<b>0.13</b>	0.02	<b>0.13</b>	<b>0.05</b>	0.04	0.01	0.02	*	*	-	-
LSUN	0.03	<b>0.13</b>	0.02	0.12	<b>0.04</b>	0.02	0.01	0.02	*	*	-	-

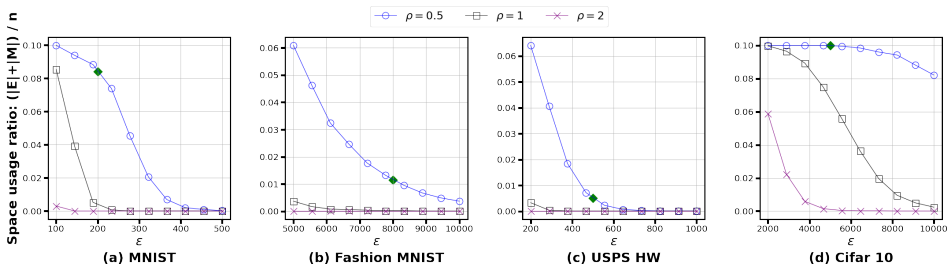


Fig. 6. Memory usage of our streaming algorithm; the green diamond in each figure corresponds to the parameter that we use in Table 4

## 5.6 Streaming DBSCAN Algorithm

We compare the clustering result of our streaming algorithm (*i.e.*, Algorithm 3) with several popular streaming density clustering algorithms such as DBStream [31], D-Stream [12], evoStream [8], and BICO [25]. We set  $\rho = 0.5$  for our algorithm. For each baseline algorithm, we show its performance under the properly tuned parameters. We also take some real stream data with large scales: the billion-scale dataset Spotify\_Session, and the two million-scale datasets PCAM and LSUN (please

Table 4. Results of the streaming algorithms, the symbol "-" denotes values less than 0.01

Dataset	Our algorithm		DBStream		D-Stream		evoStream		BICO	
	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI	ARI	AMI
Moons	<b>0.97</b>	<b>0.90</b>	0.42	0.38	0.19	0.19	0.23	0.30	0.21	0.24
Cancer	<b>0.70</b>	<b>0.58</b>	0.13	0.13	0.59	0.50	0.44	0.40	0.53	0.40
Arrhythmia	<b>0.23</b>	<b>0.13</b>	0.16	0.13	0.01	0.01	0.18	0.12	0.05	0.10
Biodeg	<b>0.09</b>	<b>0.06</b>	-	0.03	0.01	0.02	0.01	0.05	0.01	0.04
MNIST	0.28	<b>0.51</b>	0.04	0.12	-	-	0.01	0.07	<b>0.31</b>	0.41
CIFAR 10	0.02	<b>0.43</b>	0.04	0.07	0.01	0.02	-	0.01	<b>0.03</b>	0.13
Fashion MNIST	<b>0.26</b>	<b>0.50</b>	-	-	-	-	-	-	0.23	0.39
USPS HW	0.19	<b>0.53</b>	-	-	0.01	0.10	0.22	0.40	<b>0.37</b>	0.49
PCAM	<b>0.02</b>	<b>0.03</b>	-	-	-	-	-	-	0.01	0.02
LSUN	-	<b>0.04</b>	-	-	-	-	-	-	<b>0.01</b>	-
Spotify_Session 1%	<b>0.02</b>	<b>0.14</b>	0.02	-	-	-	-	-	0.01	-
Spotify_Session 10%	<b>0.02</b>	<b>0.15</b>	0.02	-	-	-	-	-	0.01	-
Spotify_Session 50%	<b>0.02</b>	<b>0.15</b>	0.02	-	-	-	-	-	0.01	-
Spotify_Session 100%	<b>0.02</b>	<b>0.09</b>	0.02	-	-	-	-	-	0.01	-

refer to Table 1 for details). Additionally, we divide the Spotify\_session dataset into four datasets by date, namely the earliest 1%, 10%, 50%, and the full set; since the recorded data in Spotify\_Session may have changing trend over time, we can view them as four different datasets. We also use ARI and AMI to evaluate their performances. The results are shown in Table 4. We can see that for most of the test instances, our streaming algorithm achieves better quality over other baselines.

We also consider the memory usage of our streaming algorithm. Recall that in Algorithm 3, we need to store the set  $E$  and  $\mathcal{M}$  ( $\mathcal{S}_* \subseteq E \cup \mathcal{M}$ , so that we do not need extra memory for  $\mathcal{S}_*$ ) in memory. So we evaluate the memory usage by the ratio  $\frac{|E|+|\mathcal{M}|}{n}$ . For each dataset, we conducted the experiments with different values of  $\rho \in \{0.5, 1, 2\}$ , and the results are shown in Figure 6. We can see that our algorithm can greatly save the memory usage, e.g., for the Fashion MNIST dataset, our algorithm only needs to store about 1% of the data points (see the green diamond in the curve).

## 6 CONCLUSION AND FUTURE WORK

In this paper, we study the metric DBSCAN problem and present its exact, approximate, and streaming algorithms. We first study their quality guarantees in theory, and then conduct a set of experiments to compare with other DBSCAN algorithms. Following this work, there are several opportunities to further improve our methods from both theoretical and practical perspectives. For example, is it possible to design a faster Gonzalez's algorithm under Assumption 1? We are aware that a fast Gonzalez's algorithm was proposed by Har-Peled and Mendel [32] in doubling metric, but they assume that the whole input data (including both inliers and outliers) has a low doubling dimension. As discussed in Section 5.4, we will also consider to reduce the overall time complexity by designing new algorithmic techniques, such as new  $r$ -net method with lower complexity. In particular, the improvement on efficiency may lead to real-world applications for database systems research (e.g., preprocessing large-scale NLP database to support training machine learning model more efficiently). As for streaming DBSCAN, we may consider to reduce the number of passes for our streaming algorithm via developing some more succinct data structure in memory; it also has certain practical significance to deal with other operations and issues, like data deletion and drift.

## 7 ACKNOWLEDGEMENT

The research of this work was supported in part by the National Natural Science Foundation of China 62272432, the National Key Research and Development Program of China 2021YFA1000900, and the Natural Science Foundation of Anhui Province 2208085MF163. We want to thank the anonymous reviewers for their helpful comments. We are grateful to Qingyuan Yang for his help on refining our code and for inspiring discussions.

## REFERENCES

- [1] Pankaj K Agarwal, Herbert Edelsbrunner, Otfried Schwarzkopf, and Emo Welzl. Euclidean minimum spanning trees and bichromatic closest pairs. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 203–210, 1990.
- [2] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.
- [3] Artem Babenko and Victor Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2055–2063, 2016.
- [4] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104, 2006.
- [5] Panthadeep Bhattacharjee and Pinaki Mitra. A survey of density based clustering algorithms. *Frontiers of Computer Science*, 15:1–27, 2021.
- [6] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.
- [7] Brian Brost, Rishabh Mehrotra, and Tristan Jehan. The music streaming sessions dataset. In *Proceedings of the 2019 Web Conference*. ACM, 2019.
- [8] Matthias Carnein and Heike Trautmann. evostream—evolutionary stream clustering utilizing idle times. *Big data research*, 14:101–111, 2018.
- [9] Matteo Ceccareello, Andrea Pietracaprina, and Geppino Pucci. Solving k-center clustering (with outliers) in mapreduce and streaming, almost as accurately as sequentially. *Proceedings of the VLDB Endowment*, 12, 2019.
- [10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.
- [11] Danny Z Chen, Michiel Smid, and Bin Xu. Geometric algorithms for density-based data clustering. *International Journal of Computational Geometry & Applications*, 15(03):239–260, 2005.
- [12] Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2007.
- [13] Kenneth L Clarkson. Building triangulations using  $\epsilon$ -nets. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 326–335, 2006.
- [14] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.
- [15] Mark de Berg, Ade Gunawan, and Marcel Roeloffzen. Faster dbscan and hdbscan in low-dimensional euclidean spaces. *International Journal of Computational Geometry & Applications*, 29(01):21–47, 2019.
- [16] Hu Ding, Haikuo Yu, and Zixiu Wang. Greedy strategy works for k-center clustering with outliers and coresets construction. In *27th Annual European Symposium on Algorithms (ESA 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.
- [17] Hu Ding, Fan Yang, and Mingyue Wang. On metric dbscan with low doubling dimension. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 3080–3086, 2021.
- [18] William B Dolan, Chris Quirk, and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.
- [19] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- [20] Yury Elkin and Vitaliy Kurlin. A new near-linear time algorithm for k-nearest neighbor search using a compressed cover tree. In *International Conference on Machine Learning*, pages 9267–9311. PMLR, 2023.
- [21] Jeff Erickson. New lower bounds for hopcroft’s problem. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 127–137, 1995.
- [22] Jeff Erickson. On the relative complexities of some geometric problems. In *CCCG*, volume 95, pages 85–90, 1995.
- [23] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

- [24] Jianqing Fan, Fang Han, and Han Liu. Challenges of big data analysis. *National science review*, 1(2):293–314, 2014.
- [25] Hendrik Fichtenberger, Marc Gillé, Melanie Schmidt, Chris Schwiegelshohn, and Christian Sohler. Bico: Birch meets coresets for k-means clustering. In *Algorithms-ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings 21*, pages 481–492. Springer, 2013.
- [26] Junhao Gan and Yufei Tao. Dbscan revisited: Mis-claim, un-fixability, and approximation. In *Proceedings of the 2015 ACM SIGMOD international conference on management of data*, pages 519–530, 2015.
- [27] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38: 293–306, 1985.
- [28] A Gunawan and M de Berg. A faster algorithm for dbscan, master’s thesis. *Technical University of Eindhoven*, 2013.
- [29] Anupam Gupta, Robert Krauthgamer, and James R Lee. Bounded geometries, fractals, and low-distortion embeddings. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 534–543. IEEE, 2003.
- [30] H Altay Guvenir, Burak Acar, Gulsen Demiroz, and Ayhan Cekin. A supervised machine learning algorithm for arrhythmia analysis. In *Computers in Cardiology 1997*, pages 433–436. IEEE, 1997.
- [31] Michael Hahsler and Matthew Bolaños. Clustering data streams based on shared density between micro-clusters. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1449–1461, 2016.
- [32] Sarel Har-Peled and Manor Mendel. Fast construction of nets in low-dimensional metrics and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006.
- [33] Yaobin He, Haoyu Tan, Wuman Luo, Shengzhong Feng, and Jianping Fan. Mr-dbscan: a scalable mapreduce-based dbscan algorithm for heavily skewed data. *Frontiers of Computer Science*, 8:83–99, 2014.
- [34] Dorit S Hochbaum and David B Shmoys. A unified approach to approximation algorithms for bottleneck problems. *Journal of the ACM (JACM)*, 33(3):533–550, 1986.
- [35] Michael E Houle. Characterizing adversarial subspaces using local intrinsic dimensionality. In *6th International Conference on Learning Representations (ICLR 2018), CoRR abs/1801.02613*, volume 6, pages 1–15, 2018.
- [36] Lingxiao Huang, Shaofeng H.-C. Jiang, Jian Li, and Xuan Wu. Epsilon-coresets for clustering (with outliers) in doubling metrics. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 814–825. IEEE Computer Society, 2018.
- [37] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2:193–218, 1985.
- [38] Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994.
- [39] Mike Izbicki and Christian Shelton. Faster cover trees. In *International Conference on Machine Learning*, pages 1162–1170. PMLR, 2015.
- [40] Jennifer Jang and Heinrich Jiang. Dbscan++: Towards fast and scalable density clustering. In *International conference on machine learning*, pages 3019–3029. PMLR, 2019.
- [41] Jennifer Jang and Heinrich Jiang. Meanshift++: Extremely fast mode-seeking with applications to segmentation and object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4102–4113, 2021.
- [42] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.
- [43] Robert Krauthgamer and James R Lee. Navigating nets: Simple algorithms for proximity search. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 798–807. Citeseer, 2004.
- [44] Hans-Peter Kriegel, Peer Kröger, Jörg Sander, and Arthur Zimek. Density-based clustering. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(3):231–240, 2011.
- [45] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.
- [46] Brian Kulis and Michael I Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. *arXiv preprint arXiv:1111.0352*, 2011.
- [47] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [48] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.
- [49] Alessandro Lulli, Matteo Dell’Amico, Pietro Michiardi, and Laura Ricci. Ng-dbscan: scalable density-based clustering for arbitrary data. *Proceedings of the VLDB Endowment*, 10(3):157–168, 2016.
- [50] Kamel Mansouri, Tine Ringsted, Davide Ballabio, Roberto Todeschini, and Viviana Consonni. Quantitative structure–activity relationship models for ready biodegradability of chemicals. *Journal of chemical information and modeling*, 53(4):867–878, 2013.
- [51] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205, 2017.
- [52] Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.



- [53] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
- [54] James C Robinson. *Dimensions, embeddings, and attractors*, volume 186. Cambridge University Press, 2010.
- [55] Alex Rodriguez and Alessandro Laio. Clustering by fast search and find of density peaks. *science*, 344(6191):1492–1496, 2014.
- [56] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.
- [57] Aviad Rubinfeld. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th annual ACM SIGACT symposium on theory of computing*, pages 1260–1268, 2018.
- [58] Aditya Sarma, Poonam Goyal, Sonal Kumari, Anand Wani, Jagat Sesh Challa, Saiyedul Islam, and Navneet Goyal.  $\mu$ dbscan: an exact scalable dbscan algorithm for big data exploiting spatial locality. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–11. IEEE, 2019.
- [59] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21, 2017.
- [60] scikit-learn developers. scikit learn. [https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\\_moons.html#sklearn.datasets.make\\_moons](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html#sklearn.datasets.make_moons), 2007–2023.
- [61] Hwanjun Song and Jae-Gil Lee. Rp-dbscan: A superfast parallel dbscan algorithm based on random partitioning. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1173–1187, 2018.
- [62] Kunal Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 281–290, 2004.
- [63] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. Pearson Education India, 2016.
- [64] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16–20, 2018, Proceedings, Part II 11*, pages 210–218. Springer, 2018.
- [65] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning*, pages 1073–1080, 2009.
- [66] Yiqiu Wang, Yan Gu, and Julian Shun. Theoretically-efficient and practical parallel dbscan. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 2555–2571, 2020.
- [67] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.
- [68] Sandamal Weerasinghe, Tansu Alpcan, Sarah M Erfani, and Christopher Leckie. Defending support vector machines against data poisoning attacks. *IEEE Transactions on Information Forensics and Security*, 16:2566–2578, 2021.
- [69] Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics, 2018. URL <http://aclweb.org/anthology/N18-1101>.
- [70] Yi-Pu Wu, Jin-Jiang Guo, and Xue-Jie Zhang. A linear dbscan algorithm based on lsh. In *2007 International Conference on Machine Learning and Cybernetics*, volume 5, pages 2608–2614. IEEE, 2007.
- [71] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
- [72] Keyu Yang, Yunjun Gao, Rui Ma, Lu Chen, Sai Wu, and Gang Chen. Dbscan-ms: distributed density-based clustering in metric spaces. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 1346–1357. IEEE, 2019.
- [73] Fisher Yu, Yinda Zhang, Shuran Song, Ari Seff, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.
- [74] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

Received October 2023; revised January 2024; accepted February 2024