Connecting the Dots — Density-Connectivity Distance unifies DBSCAN, k-Center and Spectral Clustering

Anna Beer
beer@cs.au.dk
Aarhus University
Aarhus, Denmark

Andrew Draganov
draganovandrew@cs.au.dk
Aarhus University
Aarhus, Denmark

Ellen Hohma
ellen.hohma@tum.de
Technical University of Munich
Munich, Germany

Philipp Jahn
jahn@dbs.ifi.lmu.de
LMU Munich, DBS, MCML
Munich, Germany

Christian M.M. Frey
christianmaxmike@gmail.com
Fraunhofer IIS
Erlangen, Germany

Ira Assent
ira@cs.au.dk
Aarhus University
Aarhus, Denmark

ABSTRACT

Despite the popularity of density-based clustering, its procedural definition makes it difficult to analyze compared to clustering methods that minimize a loss function. In this paper, we reformulate DBSCAN through a clean objective function by introducing the *density-connectivity distance* (dc-dist), which captures the essence of density-based clusters by endowing the minimax distance with the concept of density. This novel ultrametric allows us to show that DBSCAN, k-center, and spectral clustering are equivalent in the space given by the dc-dist, despite these algorithms being perceived as fundamentally different in their respective literatures. We also verify that finding the pairwise dc-dists gives DBSCAN clusterings across all ε values, simplifying the problem of parameterizing density-based clustering. We conclude by thoroughly analyzing density-connectivity and its properties – a task that has been elusive thus far in the literature due to the lack of formal tools. Our code recreates every experiment below: github link.

CCS CONCEPTS

• **Information systems** → **Clustering**; **Data mining**; • **Theory of computation** → **Unsupervised learning and clustering**.

KEYWORDS

clustering, ultrametric space, density, minimax path

ACM Reference Format:
Anna Beer, Andrew Draganov, Ellen Hohma, Philipp Jahn, Christian M.M. Frey, and Ira Assent. 2023. Connecting the Dots — Density-Connectivity Distance unifies DBSCAN, k-Center and Spectral Clustering. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3580305.3599283

1 INTRODUCTION

Some of the most frequently used clustering algorithms approximate or find the optimum of a function. For example, k-center minimizes the maximum distance of any point to the closest of k centers [1] and spectral clustering optimizes a cut criterion on a graph obtained from the data [59]. However, for one of the most famous algorithms, DBSCAN [21], there exists only a procedural description of density-based clusters without a formal optimization criterion. As a result, the wealth of literature regarding density-based clustering has largely been heuristic due to the procedural nature of the algorithm [57].

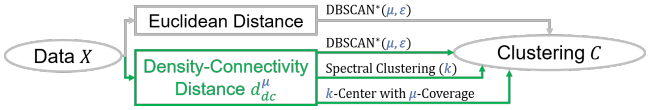

Figure 1: Equivalence among clustering algorithms under the dc-dist. k-center, spectral clustering, and DBSCAN* on the dc-dist are equivalent to DBSCAN* on the Euc. distance.

Our primary contribution is an objective function for density-based clustering. We achieve this by introducing the dc-dist, a novel distance measure that captures density-connectivity. To the best of our knowledge, such a measure has not been defined in the literature yet. We then show that it is an ultrametric and show that DBSCAN* [6] (DBSCAN without border points) optimally minimizes this function: Calculating the pairwise dc-dists provides DBSCAN* solutions across all ε values simultaneously. By a simple extension, we define an analogous distance measure and objective function for the full DBSCAN problem, with similar implications.

We then prove that any DBSCAN* clustering corresponds to the optimal solution for some k-center instance under the dc-dist. Put simply, if the k-center problem has an optimal solution with cost ε, then the density-based clustering problem for a given ε has the same optimal solution with k clusters. We extend this by incorporating the well-studied k-center with q-coverage problem [38] to ensure that each cluster has at least q points.

Due to the ultrametric nature of the dc-dist, we furthermore show that this optimal solution corresponds to cuts that can be obtained through the spectrum of the graph under our distance measure [37, 45]. Consequently, spectral clustering under the dc-dist returns the exact same clustering as DBSCAN* and k-center. Fig. 1 summarizes the connection of various clustering heuristics based on the proposed dc-dist.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599283

We conclude by performing a thorough experimental analysis of our distance measure. Since the dc-dist captures the spirit of density-connected clustering, studying its attributes gets at the fundamental properties underlying DBSCAN. For example, we show that the dc-dist is more robust to the single-link effect [34] than the minimax distance [42]. We also simplify the DBSCAN parameterization problem by noting that setting k roughly equal to the number of classes is a simple heuristic for high-quality DBSCAN clusterings. In summary, we:

- define the first distance measure that incorporates the concept of density-connectivity, called the dc-dist.
- show the equivalence of DBSCAN* (DBSCAN without border points), spectral clustering, k-center and k-center with q-coverage on the space defined by the dc-dist.
- give the first non-procedural definition of DBSCAN's clustering objective, including the concept of border points.
- perform extensive experiments showing that the dc-dist captures the structure of density-connected data better than other commonly used distance measures and how the dc-dist can be used to effectively select DBSCAN parameters.

2 AN ULTRAMETRIC CAPTURING DENSITY-CONNECTIVITY

Notation. Let $(\mathcal{X}, \delta)$ be a metric space of a finite set $\mathcal{X}$ and a metric δ. The input data $\mathcal{X} = \{x_1, \ldots, x_n\}$, where $x_i \in \mathbb{R}^d$, can be represented as an undirected, weighted graph $\mathcal{G}^\delta = (\mathcal{V}, \mathcal{E}, w)$ for any distance measure $\delta(\cdot, \cdot)$. Each data point $x_i \in \mathcal{X}$ implies a node $v_i \in \mathcal{V}$, where $|\mathcal{X}| = |\mathcal{V}| = n$, i.e., there exists a bijective mapping $\phi : \mathcal{X} \to \mathcal{V}$. Further, $\mathcal{E} = \{(v_i, v_j, w_{ij}^\delta) \mid \forall v_i, v_j \in \mathcal{V}\}$, where the weight/length of edges is defined by the distance δ between the points in high-dimensional space: $w^\delta(e_{ij}) = w_{ij}^\delta = \delta(x_i, x_j)$. As a result, each node has an edge of weight 0 to itself. A *path* P from a node v_p to another node v_q is a sequence of edges that connect a sequence of nodes, starting at v_p and ending at v_q: $P(v_p, v_q) = (e_{p,1}, e_{1,2}, \cdots, e_{l-1,q})$. $\mathcal{P}(v_p, v_q)$ is the set of all paths between v_p and v_q.

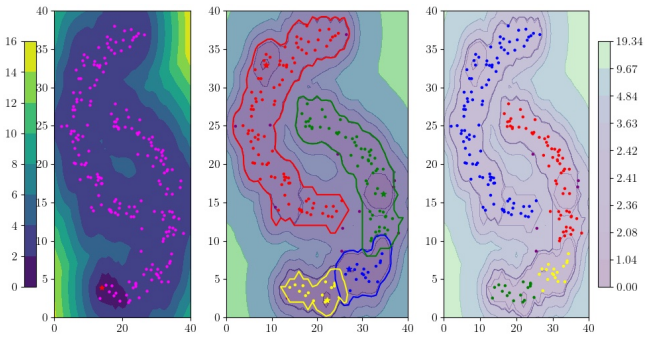

Figure 2: *Left*: dc-dist to the star, $\mu = 3$. *Middle*: k-center clusters for $k = 4$; centers are stars; heatmap gives dc-dist to the closest center; the cluster outlines are distance cost(C_i) to each center. *Right*: DBSCAN clusters with $\varepsilon = \max_i$ cost(C_i).

2.1 The Density-Connectivity Distance

With our measure, we want to combine *density* and *connectivity* in order to capture the essence of density-based clustering. The connectivity can be measured by the *minimax distance* [33, 42].

Definition 2.1. Minimax path (distance)
The *minimax path* between two nodes v_p, v_q is the path between v_p and v_q that minimizes the maximum weight of any of its edges. The *minimax (path) distance* m^δ is the weight of the longest edge on the minimax path [33].

$$m^\delta(v_p, v_q) = min_{P \in \mathcal{P}(v_p, v_q)} max_{e \in P} w^\delta(e) \tag{1}$$

The minimax distance's lack of a density constraint leaves it prone to the single link effect, i.e., points that are far apart can have a small minimax distance when a single chain of noise points links them. In order to incorporate density into the minimax distance, we require a way to define the dense (core) distance between pairs of nodes. To this end, we translate the concept of dense (core) points (as known from clustering algorithms like DBSCAN [21] or Wishart's method [61]) to the graph setting:

Definition 2.2. Dense path, core nodes, and core distance
Given a graph $\mathcal{G}^\delta$, a *dense path* w.r.t. μ and ε is a path that contains only core nodes. *Core nodes* are nodes with at least μ edges (including the edge with weight zero to itself) that have a weight lower than ε. We call the length of the μ-th shortest edge connected to a node v_p its *core distance* $d_{core}^\mu(v_p)$. Equivalently, the *core distance* $d_{core}^\mu(p)$ is the distance to the μ-th nearest neighbor of p, where $\delta(\cdot, \cdot)$ is usually Euclidean.

We can now capture density-connectivity by measuring the connectivity with the minimax distance while ensuring density on these paths:

Definition 2.3. Density-Connectivity distance
The density-connectivity distance d_{dc}^μ (short: dc-dist) between two nodes p and q in a dataset $\mathcal{X}$ is the minimax distance on the graph $\mathcal{G}^{d_E}$ induced by the Euclidean distance on $\mathcal{X}$, where only dense paths are regarded.

We calculate this distance by working on a graph of only dense paths, i.e., all nodes of the graph are core nodes. For that, we propose the *mutual reachability distance* d_r^μ. It implies the distance for which two points can be "reached", i.e., the maximum of their Euclidean distance and the value for which both are core points. We provide a visualization of it in Fig. 2.

Definition 2.4. Mutual Reachability distance
The mutual reachability distance $d_r^\mu(p, q)$ for any $\mu \leq |\mathcal{X}|$ is 0 if $p = q$, and else

$$d_r^\mu(p, q) = max(d_E(p, q), d_{core}^\mu(p), d_{core}^\mu(q)) \tag{2}$$

Based on the mutual reachability distance, the density-connectivity distance can be formalized as follows:

Definition 2.5. The dc-dist is the minimax distance on the graph given by the mutual reachability distance:

$$d_{dc}^\mu(p, q) = m^{d_r^\mu}(v_p, v_q) = min_{P \in \mathcal{P}(v_p, v_q)} max_{e \in P} w^{d_r^\mu}(e) \tag{3}$$

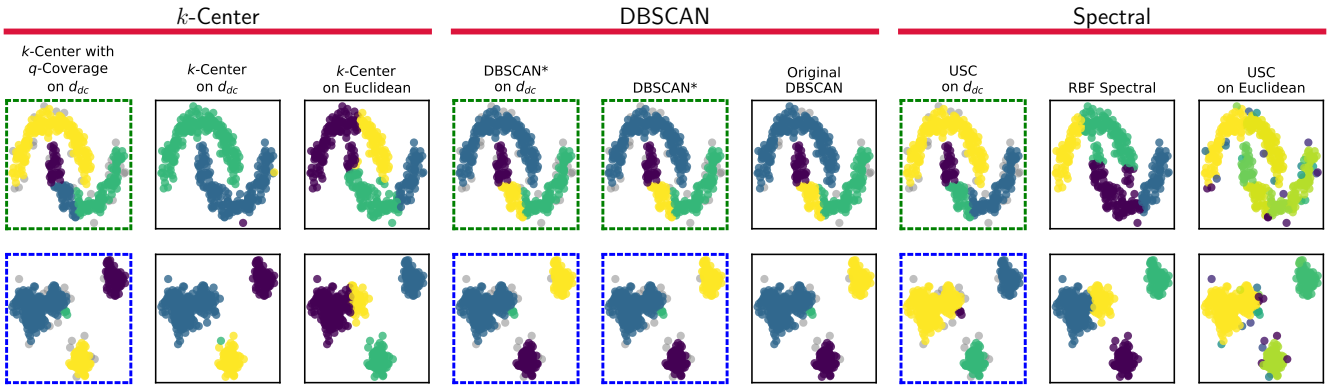

Figure 3: Illustration of the relationship between different clustering methods with dc-dist. DBSCAN without borderpoints, k-center with q-coverage, and ultrametric spectral clustering return the same results.

Working on the graph given by the mutual reachability distance ensures that all possible paths are dense. Thus, known algorithms for computing minimax distances on graphs, like Kruskal's or Prim's, can be used directly to compute the dc-dist efficiently.

2.2 Properties

We first observe that because the minimax paths lie on minimum spanning trees (MST) of the graph [14, 26], we can calculate the dc-dist by building the MST of $\mathcal{G}^{d_r^\mu}$ (see Sec. 2.3 for details). We also note that the dc-dist inherits the properties of an ultrametric (also referred to as *super metric* or *rooted tree metric*) from the minimax distance [13, 40]. As such, all pairwise distances between n objects can be represented in a tree with the objects as leaves and the dc-dists as nodes. Thus, the dc-dist only has up to $O(n)$ unique values while traditional distance metrics have up to $O(n^2)$. The dc-dist furthermore defines a metric space in which the *strong triangle inequality* holds, i.e., $\forall p, q, r \in \mathcal{X} : d_{dc}^\mu(p, r) \leq max\{d_{dc}^\mu(p, q), d_{dc}^\mu(q, r)\}$.

2.3 Calculating the dc-dist

We calculate the dc-dist through a modification to Kruskal's minimum spanning tree (MST) algorithm on the fully-connected graph induced by d_r^μ (for details on correctness and efficiency, see [14]). Kruskal's algorithm builds an MST by maintaining a set of forests and, at each step, connecting the two forests that have the shortest edge between them. The length of the edge that connected the forests of p and q corresponds to the dc-dist.

Algorithm 2 in the supplementary material describes the calculation of the dc-dist. We store the pairwise distances in a tree data structure that has one leaf for each point and one node for each edge that connected two forests during Kruskal's algorithm. We assign the length of this edge to its corresponding node. Obtaining the MST tree requires then $\tilde{O}(n^2)$ time [14]. Calculating the dc-dist between two points then requires $O(\log(n))$ time to find their lowest common ancestor and return the value in that node in the tree. Thus, the complexity to compute the entire distance matrix requires $\tilde{O}(n^2)$ time. We note that this binary tree is also used in Wishart's hierarchical classification method [61] and in HDBSCAN [6] (see Sec. 7.2).

3 DBSCAN$^{(*)}$ AND DC-DIST

DBSCAN [21] assigns core points (points that have at least *minPts* neighbors in their ε range) that are density-connected to a single cluster. Note, that two points can be density-connected without being core points; they are then called border points. Although DBSCAN is not usually presented as a hierarchical clustering algorithm, points that share a DBSCAN cluster for a ε_1 will also do so for $\varepsilon_2 \geq \varepsilon_1$. Furthermore, the DBSCAN clusters for $minPts = 1$ correspond – save for border points and noise – to the clusters found by Hierarchical Single Linkage clustering with cutoff at $threshold = \varepsilon$. Algorithms extending DBSCAN often ignore the border points and differentiate only between core and noise points (e.g., [6, 46]). In this adaptation of DBSCAN, called DBSCAN* [6], all border points are considered noise points[1]. We show example DBSCAN* clusterings in columns 4 and 5 of Fig. 3. Given this definition of DBSCAN*, we have the following theoretical results:

Lemma 3.1. *Let $C = \{C_1, ..., C_k\}$ be the clusters found by DBSCAN* with threshold ε. Then, for any $C_i \in C$, $d_{dc}(c_a, c_b) \leq \varepsilon \; \forall \; c_a, c_b \in C_i$. Furthermore, for any $C_i, C_j \in C$ with $C_i \neq C_j$, we have $d_{dc}(c_i, c_j) > \varepsilon \; \forall \; c_i \in C_i, c_j \in C_j$.*

Proof. Let $\{c_{i1}, c_{i2}, ..., c_{im}\}$ be a sequence of points in a DBSCAN* cluster. Since DBSCAN* operates by connecting those points that have a mutual reachability distance less than ε between one another (see [9]), it must be the case that each step in this path has mutual reachability distance less than or equal to ε. Thus, the longest step in any path within a DBSCAN* cluster is less than ε, proving the first part of the lemma. The second part is then trivially true, as distances across DBSCAN* clusters must have mutual reachability distance greater than ε. □

Corollary 3.2. *Let $[\mathbf{M}(\varepsilon)]_{ij} = \mathbb{1}_{d_{dc}(p_i,p_j)\leq\varepsilon}$ be a matrix that is 1 when the dc-dist of a pair of points is less than ε and 0 otherwise. Then $\mathbf{M}(\varepsilon)$ implies a valid DBSCAN* clustering for any value of ε: if two points have a non-zero entry in $\mathbf{M}(\varepsilon)$ then they are in the same DBSCAN* cluster for that ε.*

[1] We add the restriction that clusters must have at least μ points

Proof. Since DBSCAN* clusters must have intra-cluster dc-dist values less than ε, thresholding the matrix of pairwise distances by any value will leave only those pairs of points that are within an ε ball of each other. Furthermore, this thresholding forms an equivalence relation: it is inherently reflexive and symmetric due to acting on a distance matrix and is transitive by the ultrametric property $d(x,y) < \varepsilon \wedge d(y,z) < \varepsilon \Rightarrow d(x,z) < \varepsilon$. Thus, $\mathbf{M}(\varepsilon)$ corresponds to a DBSCAN* clustering. □

Importantly, Cor. 3.2 shows that the set of n^2 pairwise dc-dists defines DBSCAN* clusterings across all values of ε simultaneously – one only needs to threshold the matrix for an appropriate value of ε. Furthermore, since there are only $O(n)$ unique dc-dist values, searching for the optimal ε can be done quickly using a tree data-structure. We now present a result that states that DBSCAN* clusterings are equivalent under the Euclidean and dc-dist.

Lemma 3.3. *Let ε be a threshold value. Then $C = \{C_1, ..., C_k\}$ is a clustering found by DBSCAN* under the Euclidean distance if and only if C is a clustering found by DBSCAN* under the dc-dist .*

The proof is located in Sec. C in the appendix. This effect can be seen in columns 4 and 5 of Fig. 3, where the DBSCAN* result is identical under the Euclidean and density-connected distances.

3.1 DBSCAN*'s optimization criterion

We now utilize Lemma 3.1 to define an objective function for the DBSCAN* algorithm. Let $\mathcal{C} \subset \text{powerset}(\mathcal{X})$ be the space of solutions such that $C = \{C_1, \ldots, C_l\}$ partitions the dataset. Then we define the *ε-density-based-clustering* (εDBC) objective as:

$$\min_{\substack{C \subset \mathcal{C} \\ d_{dc}(p,q) \leq \varepsilon \forall p,q \in C_i \forall C_i \in C}} |C| \tag{4}$$

This can be read as "find the smallest set of clusters such that the maximum dc-dist within any cluster is ε." This leads to the natural loss function $\mathcal{L}_{\varepsilon DBC}(C) = |C|$, where we assume that C is a feasible solution to the εDBC problem. We note that the objective can be interpreted as a coverage problem, where we are given a set of points in the dc-dist space and must find the minimum number of ε-balls that cover them. The next theorem shows that DBSCAN* solves the εDBC problem optimally.

Theorem 3.4. *DBSCAN* with threshold ε obtains the optimal solution to the εDBC objective when applied on the ε core points of P.*

Proof. FSOC, assume there is another method that finds a solution with fewer clusters than DBSCAN*. Then there must exist a pair of core points p and q in the same optimal cluster but in separate DBSCAN* clusters. By the definition of the objective function, $d_{dc}(p,q) \leq \varepsilon$ since they reside in the same optimal cluster. However, since their dc-dist is less than ε, they must have been connected during the DBSCAN* procedure. Thus, we have a contradiction. □

Importantly, this objective function is consistent with the original introduction of DBSCAN [21]. Namely, the dc-dist corresponds to the connectivity criterion, while minimizing the number of clusters corresponds to the maximality criterion. We note that problems defined through an objective function have the advantage of allowing formal approximation and acceleration algorithms.

3.2 DBSCAN-Distance: Extension for Borderpoints

Thus far we have only discussed the dc-dist with respect to DBSCAN*, as it is cleaner than its original counterpart. We note, however, that just as the dc-dist gives the smallest ε such that two points are core points of the same DBSCAN* cluster, we can define the *DBSCAN-distance*, which gives the smallest ε, s.t. two points are in the same DBSCAN cluster (even if they are "only" border points of the same cluster). The difference to Def. 2.3 of the dc-dist is that, to handle the border points, we do not require the start and end node of a path to be dense. We note that this is no longer an ultrametric.

Definition 3.5. DBSCAN-distance
Let $\{v_p, ..., v_q\}$ be the minimax path in the mutual-reachability graph between the nodes of points p and q such that

(1) v_p and v_q are the only two nodes on the path that are allowed to not be core points, and
(2) the path contains at least one core point.

Then the DBSCAN-distance d^{μ}_{db} between two points p and q is the weight of the longest edge on this path.

We prove that this is a distance metric in Appendix E and give an algorithm for calculating it in Alg. 3. We note that, since the DBSCAN-distance corresponds to the smallest ε s.t. they are in the same DBSCAN cluster, we can extend the εDBC objective onto DBSCAN by simply substituting the new distance function. Thus, while other DBSCAN presentations only describe a *method* to find these clusters, the DBSCAN-distance allows the first non-procedural *definition* of the full DBSCAN optimization task. In Fig. 3, we show the difference between DBSCAN and DBSCAN* clusterings.

4 CONNECTING k-CENTER TO DBSCAN*

We now turn to the equivalence of DBSCAN* and k-center solutions under the dc-dist. The k-center problem is finding k centers s.t. the largest distance of any point to its closest center is minimized [27], formalized via the objective function

$$\mathcal{L}(\mathcal{X}, C) = \max_{x \in \mathcal{X}} \min_{c \in C} d(x,c).$$

We use the notation $\text{cost}(k, P)$ to refer to the cost of the optimal k-center solution on $\mathcal{X}$.

We can see a clear connection between this and the εDBC objective introduced in Eq. 4, we see that the εDBC objective aims to find the smallest number of ε-balls that cover a set of points while the k-center objective is searching for the smallest ε such that k ε-balls cover the points. We formalize this connection in the following theorems:

Theorem 4.1. *Let C be the unique optimal k-center solution on pointset $\mathcal{X}$ under the dc-dist for some μ and assume there does not exist an $l < k$ such that $cost(l, \mathcal{X}) = cost(k, \mathcal{X})$. Let ε_i^k be the cost of the i-th cluster and define $\varepsilon_{opt} = \max_i \varepsilon_i^k$. Then*

(1) C is the optimum of the DBC objective on $\mathcal{X}$ with ε_{opt}
(2) $|C| = k$
(3) DBSCAN finds this optimal solution.*

Proof. This is equivalent to saying that the k-center solution's intra- and inter-cluster dc-dists are respectively upper- and lower-bounded by ε_{opt} (see Sec. 3). The former occurs by definition. To see the latter, note first that the assumption that k is the smallest integer for which we can get $\text{cost}(k, \mathcal{X})$ implies that there is not an inter-cluster distance equal to ε_{opt}. If there were, we could join these together to obtain a $(k-1)$-center solution with equal cost. Now assume for the sake of contradiction that centers c_i and c_j have dc-dist $d_{dc}(c_i, c_j) < \varepsilon_{opt}$. We can then combine clusters c_i and c_j into a single cluster c_{ij} that has cost strictly less than ε_{opt}. This gives a solution with $k-1$ centers where there still exists an edge of length ε_{opt} within some cluster c_k. We can then split c_k into two clusters c_l and c_m along this longest edge. This new solution has cost less than or equal to that of C, which is a contradiction. Lastly, Theorem 3.4 states that DBSCAN* finds this optimum. □

We note that if there is some $l < k$ such that $\text{cost}(l, \mathcal{X}) = \text{cost}(k, \mathcal{X})$, then $|C| = l$. We refer to Algorithm 1 for a method that optimally solves the k-center problem under the dc-dist[2]. It operates over a binary tree where each node in the tree corresponds to one edge in the mutual reachability MST. Each node's value is then equal to the length of its corresponding edge in the MST and the distance between two nodes in the tree is the value of their lowest common ancestor. Importantly, it is the ultrametric nature of the dc-dist that allows the k-center problem to be solved optimally:

Theorem 4.2. *Algorithm 1 returns the optimal k-center solution under the dc-dist.*

Proof sketch. Algorithm 1 begins by recursively finding the nodes of the ultrametric's density-connectivity tree that have at most k elements and assigning a center to each one. Thus, each node in the recursive stack corresponds to a k-center solution on its respective subtree. Since the cost of a cluster is the value of its lowest-common-ancestor (lca), merging children by the smallest lca guarantees that we retain optimality after each merge. □

We provide a full proof in the appendix. These theorems can be interpreted as equivalence statements between k-center and DBSCAN* under the dc-dist. Namely, if the k-center problem has an optimal solution with cost ε, then the εDBC problem has an optimal solution with k clusters[3].

4.1 k-center with q-Coverage

Note that the k-center problem is strongly influenced by outliers. We see this effect in column 2 of Fig. 3, where the k-center solution on the two-moons dataset has three clusters consisting of one point each. Thus, although k-center solutions provide us with valid DBSCAN* clusterings, they may not be the most practical ones.

This issue stems from low density at each outlier and has been addressed by modifying the k-center problem to incorporate a minimum number of points. Specifically, the popular k-center with q-coverage problem [38] is an extension of the k-center objective such that each center accounts for at least q points. Formally, the objective function is given by

$$\mathcal{L}(P, C) = \max_{p \in P} \min_{c \in C} \max \left(d(p, c), d_q(P, c) \right),$$

where $d_q(P, c)$ is the distance of the q-th nearest neighbor to the center c. Thus, any optimal solution to the k-center with q-coverage problem has at least q points belonging to each cluster. We can transfer this to the density setting – a core point is defined as having at least μ nearest neighbors within its ε-ball. Thus, under the dc-dist, the k-center with q-coverage problem constrains our clusters to be density-connected with $\mu = q$.

It is easy to amend Algorithm 1 to solve the k-center with q-coverage problem by pruning the ultrametric tree. Specifically, we make a copy of the tree that only includes nodes with at least μ children. We then apply Algorithm 1 on this pruned tree. Lastly, we take the peak N_i of each cluster in the pruned tree and set $c_i = \{l \mid l \text{ is a leaf of } N_i \text{ in the original tree}\}$. We describe this formally in Algorithm 4 in the appendix. We now show that it optimally solves the k-center with q-coverage problem.

Theorem 4.3. *Algorithm 4 optimally solves the k-center with q-coverage problem under the dc-dist.*

Proof. The principal step in Algorithm 4 is Algorithm 1 on the pruned tree. By definition, the pruned tree consists only of those nodes that have at least q children. Thus, any k-center solution on the pruned tree corresponds to a q-coverage solution on the original dataset. To show optimality, consider that Algorithm 1 finds the optimal k-center solution on the pruned tree. Since solutions on the pruned tree have equal cost to those on the original tree, Algorithm 4 returns the optimal k-center with q-coverage solution. □

The first column of Fig. 3 shows that this provides a significantly more practical solution, where clusters are evenly distributed in size. Furthermore, k-center with q-coverage solutions retain the property that they must correspond to a DBSCAN* solution. Indeed, we see that the DBSCAN* solutions in columns 4 and 5 of Fig. 3 are identical to those in column 1. Thus, solving the k-center with q-coverage problem on the dc-dist with k roughly equal to the number of classes offers a heuristic for finding a reasonable ε DBSCAN parameter. We show this experimentally in Sec. 6.

5 ULTRAMETRIC SPECTRAL CLUSTERING

We now regard spectral clustering on the dc-dist. Spectral graph analysis converts a clustering problem in arbitrary space into a graph partitioning problem. For a given similarity matrix S, where $S_{i,j} \geq 0$ is a similarity score between x_i and x_j, the steps of SC reduce to: (1) normalization, where often some Laplacian matrix is used, (2) computation of the eigendecomposition, and (3) partitioning based on the eigenvectors. Depending on the affinity matrix and type of partitioning, a minimum ratio cut or the closely related minimum normalized cut gives the final clustering [59]. In this paper, we regard the minimum cut $mincut = \sum_{k=1}^{K} \sum_{i \in C_k, j \notin C_k} S_{i,j}$, where the similarity matrix S is based on the dc-dist and has row vectors $S_{[i,:]} \coloneqq 1 - \frac{d_{dc_{[i,:]}}}{\|d_{dc_{[i,:]}}\|}$. In the following, we provide an intuition for how spectral partitioning yields the same clustering as thresholding the dc-dist-tree (Sec. 3). We refer to Theorem 5 of [3]

[2]The $C[i].merge(C[i+1])$ function in Algorithm 1 appends the points in $C[i+1]$ to $C[i]$ but does not change $C[i]$'s center.

[3]If these clusters have size less than k, then they are noise points

Algorithm 1: Tree-k-Center

Input: Root node R; k
Output: k clusters

$C \leftarrow [\,]$;
if $|R| <= k$ **then**
  C.append($child$) for $child \in R$;
  return C ;
end if
C.append(Tree-k-Center(r.left, k));
C.append(Tree-k-Center(r.right, k));
while $|C| > k$ **do**
  $min\_dist, min\_index \leftarrow \infty, -1$;
  for $i < |C| - 1$ **do**
    $left, right \leftarrow C[i].center, C[i+1].center$;
    $peak \leftarrow$ LeastCommonAncestor($left, right$);
    $dist \leftarrow peak$.value ;
    if $dist < min\_dist$ **then**
      $min\_dist, min\_index \leftarrow dist, i$;
    end if
  end for
  $C[min\_index]$.merge($C[min\_index + 1]$);
end while
return C

for a full proof of the statement and [40] for a deeper discussion of spectral analysis using ultrametrics.

It is known that spectral clustering identifies eigenvectors that describe minimal cuts between components [59]. Now consider that the ultrametric distance matrix D_{dc} can be rearranged in a block matrix C representing the clusters. Following the proof scheme of [3, 62], two rows r_k, r_l of C are similar, i.e., $r_k \approx r_l$, iff nodes k and l belong to the same cluster and the distance $d_{dc}(k,l) < d_{dc}(k,m)$ for m in a different cluster. This holds for all sub-clusters as well, implying a hierarchical partition structure. We refer to SC applied on an ultrametric matrix D_{dc} as *Ultrametric Spectral Clustering* (USC) and provide the following proposition (proven in [3]):

Proposition 5.1. *Let the $n \times n$ similarity matrix S induce Laplacian $L = D - S$, where D is the degree matrix given by the affinity matrix S. Then the first non-constant eigenvector induces a partition on $\mathcal{G}^{d_r^\mu}$. Applying this partitioning recursively until all dc-dist values are less than ε gives the DBSCAN* clustering.*

We start with the ultrametric distance matrix D_{dc} and an ascending ordering on the values stored therein, i.e., $d_{dc_1} \leq \ldots d_{dc_n}$ for n distinct distance scores. To transform the values of D_{dc} into a similarity matrix S, we first normalize the values of D_{dc} to retrieve $\overline{D_{dc_{[i,:]}}} = \frac{d_{dc_{[i,:]}}}{\|d_{dc_{[i,:]}}\|}$ such that values are in the range $[0,1]$. Thus, our affinity matrix $S := 1 - \overline{D_{dc}}$ gives similarity scores that are in reverse order compared to the distances. We use S for the Spectral Clustering approach described in [50], where eigenpairs correlate to edges in the graph having lowest values. By the minimization problem of Spectral Clustering $min_{f \in \mathbb{R}^n} f'Sf$ (cf. [59]), the method favors cuts having low similarity scores. Consequently, larger distances are favored by the proposed *Ultrametric Spectral Clustering*. Applying the cutting procedure in an iterative manner, we get a hierarchical partition clustering up to a specific ε that the user can set as threshold.

6 EXPERIMENTS

We now turn to the experimental study of the dc-dist, demonstrating that it indeed captures the essence of density-based clustering. We describe our setup in Sec. 6.1. We study the separability of intra- and inter-cluster distances in Sec. 6.2 and robustness w.r.t. the single link effect in Sec. 6.3. We perform extensive experiments on synthetic and real-world benchmark datasets to illustrate how the dc-dist captures the structure of the data in Sec. 6.4. Sec. 6.5 shows the connection between the number of clusters k for k-center clustering and ε for DBSCAN*. Our code recreates every experiment below: https://github.com/Andrew-Draganov/dc_dist.

6.1 Experimental Setup and datasets

We start with experiments on synthetic datasets containing density-based clusters (d1, d2, d3) or Gaussian-distributed clusters (b3). Datasets d1 and d2 consist of 10k datapoints in density-based clusters with same and varying density respectively. Datasets b3 and d3 consist of 9k datapoints in unbalanced clusters (resp. clusters with varying density) and 1k uniformly distributed noise points. We used the data generator functions from the scikit-learn library[4] and implemented a data generator (online accessible via github[5]) adapting the Seed Spreader described in [24]: Each density-based cluster is produced by a random walk through the d-dimensional space. For each step, we uniformly generate points in the local hypersphere around the current position. We adapt the generator s.t. it produces a given number of clusters that are well-separated. Noise is added uniformly and assigned to a cluster if it is within a minimum distance. A more detailed description of the data generator is included in Appendix H. We examine the suitability of dc-dist for real-world use cases on state-of-the-art benchmark datasets (driver faces, coil5, coil10, pendigits, coil20, olivetti, coil). Appendix G gives an overview of all datasets.

6.2 Capturing density-connectivity

We begin by demonstrating that the dc-dist indeed captures density-connectivity by showing that the gap between intra-cluster and inter-cluster distances is much more pronounced for the dc-dist than for commonly used distance measures.

Fig. 4 shows this effect for the Euclidean, Manhattan, Cosine, and dc-dist with $\mu \in \{3, 5, 10\}$ on synthetic density-based clusters. Intra- and inter-cluster distances are plotted in blue and yellow, respectively. The x-axis corresponds to 5% distance buckets from 0 to 100% of the maximal distance in the dataset. The y-axis then shows the percentage of distances that fall into each bucket. We use the dotted lines to show the effect for each dimensionality and average over 10 runs.

We start by examining the low-dimensionality, lighter-colored lines. There, we see that the dc-dist imposes a large valley between

[4] https://scikit-learn.org/stable/, last accessed: Jan 28, 2023
[5] https://anonymous.4open.science/r/DensityDimReduction-077A

the intra-cluster distances that skew left and the inter-cluster distances that skew right. Comparing this to the other distances, the dc-dist obtains a larger differentiation between intra- and inter-cluster distances. This implies that other distance measures blur the distinction between clusters in low-dimensionality more than the dc-dist. Then, as dimensionality increases, the expected density decreases and thus the contrast decreases as well.

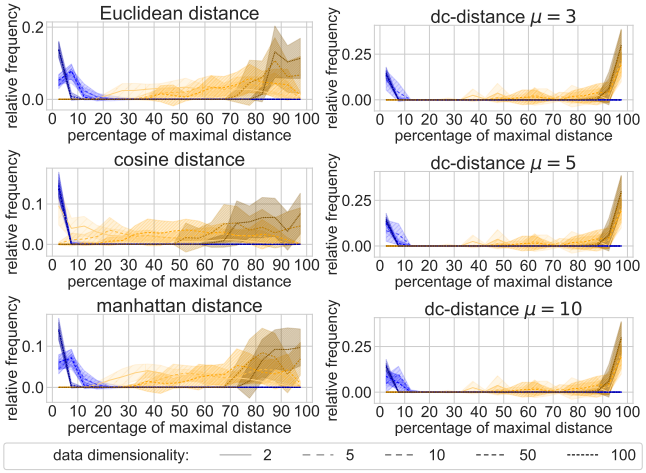

Figure 4: Distribution of distance values on data with density-based clusters. We can see a "valley" separating intra- (blue) from inter- (yellow) cluster distances for the dc-dist across dimensionalities, but not for the standard distance measures.

6.3 Robustness w.r.t. single link effect

We now show that while the minimax distance is prone to the single-link (SL) effect, the dc-dist is more robust to noise with increasing μ, thanks to incorporating density. The SL effect means clusters cannot be separated by density-based clustering, i.e., a path of density-connected noise connects them. Quantitatively, this means that the inter-cluster distance is less than or equal to the intra-cluster distances. We illustrate the robustness of dc-dist to this effect in Fig. 5 by plotting the difference between the minimum inter-cluster and mean intra-cluster distances for two multivariate Gaussians with centers at $(-3, 0)$ and $(3, 0)$. The x-axis shows their variances and the average over 10 runs. Values in the plot below 0 imply that clusters cannot be separated due to the SL effect, as the length of the smallest inter-cluster distance is smaller than the average intra-cluster distance. Fig. 5 shows that with increasing values of μ, this happens to a much lesser degree. Thus, incorporating density into the dc-dist increases robustness w.r.t. the SL effect.

6.4 Clustering Algorithms and the dc-dist

We now turn to the relationship between distance measures and clustering algorithms in Fig. 6. In order to perform a comprehensive set of experiments, we find the pairwise distances across a dataset and embed them into Euclidean space using multidimensional scaling (MDS) [8]. This allows us to apply clustering algorithms that may not accept pairwise distance matrices. We then measure the

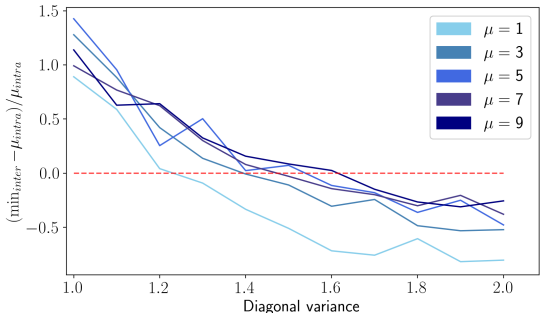

Figure 5: Robustness of the dc-dist to SL effect. The higher the values for μ, the noisier data can be before the single link effect occurs (i.e., before values fall below 0).

Adjusted Rand Index (ARI) [30] of the resulting clustering with its ground truth. We use some of the most fundamental clustering algorithms (DBSCAN [21], k-Means[4], and spectral clustering[4]) as well as a recent density-based clustering algorithm suitable for large data of varying densities (DCF [58]) across synthetic and real datasets, where real datasets are ordered by number of classes. To account for the effect of the embedding, we vary over the MDS embedding dimension in the large rows of the figure. For comparability, parameters were chosen equally among all experiments. As suggested in [55], $minPts$ for DBSCAN was set to $2 * d$, ε was set to the maximum distance of the $minPts$-th nearest neighbor. For spectral clustering, an RBF-Kernel with $\gamma = 2/\sqrt{d}$ was used. DCF parameters were chosen according to the suggested default settings, with $k = \sqrt{n}$ and $\beta = 0.4$.

We first see that on synthetic datasets, the clusterings for low-dimensional embeddings maintain a higher ARI with the ground truth when using the dc-dist. This holds true for data of different structures (Gaussian or density-based clusters, with/ without noise, balanced/ imbalanced densities) and for different clustering methods. This echoes the results shown in Fig. 4. Furthermore, we see that the dc-dist is robust to decreasing the embedding dimensionality. This is intuitively clear when considering the dc-dist's ultrametric nature: since many of the distances are exactly equivalent, there is less information that is lost when reducing dimensionality.

Turning to the benchmark dataset, we see that both of these effects become less pronounced but that there are still clear patterns that one can observe. First, the dc-dist loses some of its robustness to the embedding dimensionality but this change seems smaller for higher μ. Second, we see datasets with fewer classes are better clustered on average across all distance measures, but the standard distance measures seem more resilient to larger numbers of classes. We hypothesize that this is due to the single-link effect being more likely to occur across classes as the number of classes grows. Third, for density-connected data with few classes (e.g., c5 and c20) the dc-dist permits better clusterings compared to the other distance measures. We note that the reachability distance, which also captures density, outperforms the standard distance measures on these

datasets as well. Fourth, the quality of clusterings naturally decreases if we choose the wrong μ: e.g., $\mu = 10$ is too high for the Olivetti dataset, which only has clusters of size 10.

6.5 Choosing k leads to good values for ε

We lastly consider Fig. 7, which shows how the number of classes affects the ε value in DBSCAN. Specifically, we increase k and observe that the resulting k-center with q-coverage leads to the ε value consistently dropping. Furthermore, the figure shows that, as expected, setting k equal to the number of classes is a reasonable heuristic for parameterizing DBSCAN. We argue that practitioners often have a rough idea for the number of classes in their dataset and suggest this as a simple method for obtaining DBSCAN clusterings without hyperparameter tuning.

7 RELATED WORK

Sec. 7.1 gives some background on distance measures in general, as well as the minimax distance and the reachability distance in particular, as the dc-dist is based on them. Subsequently, we give an overview of the research areas that are brought together by our paper: density-connectivity (Sec. 7.2), k-center (Sec. 7.3), and Spectral clustering (Sec. 7.4).

7.1 Distance Measures

In the experiments in Sec. 6 we compare the dc-dist to the most commonly used distances measures: Euclidean, Manhattan, and cosine distance, all working in $\mathbb{R}^d$. Link-based or path-based distances (e.g., shortest path distance [25] or minimax distance [33]), as well as metrics based on MSTs (e.g., the critical distance [54]) and ultrametrics in general, are more closely related to the dc-dist. The minimax distance is known to be suitable for clustering and is used as basis for several cluster algorithms (e.g., [11, 36, 40, 41, 44]). For an extensive overview of minimax distances and related concepts, we refer to [26]. Note, that it is also referred to as the widest path [60], bottleneck shortest path [23, 32], longest leg path [40], or as transitive distance [62] or connectivity kernel [22] and it is closely related to the maximum capacity problem [51]. It is implied by the Single Link Distance, which is defined on sets of points and typically used for hierarchical clustering [47]. Additionally, it offers the desirable properties of ultrametrics [26]. In general, ultrametrics can be useful for, e.g., minimum spanning tree (MST) based clustering, efficient graph-based image segmentation (EGS), and normalized tree partition (NTP) [62]. In [22], a "pairwise clustering cost function" on the "connectivity kernel" is optimized, however, the authors omit that it optimizes a graph cut on the minimax distance. Nevertheless, they perform the cut with kernel PCA imitating a spectral approach. The potential of representing distances in the form of trees is known for a long time [17]. Our novel dc-dist combines the minimax distance with an adaption of the original reachability distance, which was first defined with the introduction of OPTICS [2], and later reused by various algorithms, e.g., LOF [5] or LDBSCAN [20]. A visualization of the reachability distance in comparison to the dc-dist is given in Fig. 8, where the distances used by OPTICS are framed in black. The reachability distance builds on top of the definitions for (direct) density-reachability introduced with DBSCAN [21] and is defined for $|N_\varepsilon(p)| \geq \mu$ as $d_{or}(p,q) = max(core - distance(p), distance(p,q))$. Note that d_{or} (as used in, e.g., [2, 5]) is not a distance measure as it is, e.g., not symmetric. Also the "mutual reachability distance" defined by HDBSCAN [6] is not a metric as the distance of a point to itself there is defined as its core-distance, not as 0. Power-weighted shortest path distances (PWSPD) [39] account for the density along paths but assume a continuous density-function. We leave the connection to non-discrete settings for future work.

7.2 Density-Connectivity

Wishart's density-based hierarchical clustering approach from 1969 [61] and Hartigan [28] laid the foundation for many later published density-based algorithms. Wishart defined density via the distance to the k-th nearest neighbor, equivalent to the *minPts* parameter DBSCAN [21] paper published over 20 years later. Its results differ from DBSCAN's mainly in that "points which are non-dense [are] allocated to the cluster containing its nearest dense point" [61], instead of getting labeled as noise. Hartigan [28] described, among others, the fundamental concepts of density-based clustering, *connectivity* and *maximality*, which were later often readdressed (e.g., [34] or [4]). DBSCAN introduced the term "density-connected" and builds a basis for a wealth of clustering algorithms, e.g. AnyDBC [43], HDBSCAN [6], and Density Peaks [52]. Here, points are density-connected, if there is a "chain" of core points between them. Based on DBSCAN, HDBSCAN [6, 7] builds a clustering hierarchy that also corresponds to the one described by Wishart [61]. They offer a method on how to cut the tree for a flat clustering by optimizing cluster stability. The DBSCAN-based hierarchical clustering algorithm DBHC [35] merges DBSCAN clusterings resulting from different parameter settings until the "number of clusters matches the user's desired number" [35]. This allows users to set the number of clusters instead of ε and *minPts*, however, it requires to run DBSCAN $O(\sqrt{n})$ times and comes, thus, with a high runtime. We point out that calculating the pairwise dc-dists accomplishes the same task in a simpler manner. $SL(\alpha)$ [44] is another clustering method that combines density with hierarchical clustering, but their concept of density defined by simplicial complexes differs from the concept used by DBSCAN-like approaches. Cluster trees [12] come close to the idea of dc-dists, but do so by hierarchical density *estimates* which are used to build a hierarchy. Some heuristics on how to combine density with hierarchical clustering, e.g., by requiring a minimum cluster size, are given in [53]. Finding the "best"parameters for *minPts* and ε is a much-discussed challenge [57].

7.3 k-Center

In [41], the authors employ the minimax distance and show that cluster centers have a minimum density within them. They propose an *Improved Path-based Clustering* (IPC) algorithm using an elimination process to find cluster centers. DBSCAN++ [31] speeds up DBSCAN by only recovering a fraction of the core points. It requires expensive density estimation queries on only m of the n samples. The subset of m samples is chosen according to the k-center problem minimizing the maximum distance of any point to its closest one in that subset. The connected k-center problems presented in [19] use a connectivity graph such that each cluster has a well-defined connected subgraph. The authors prove that if

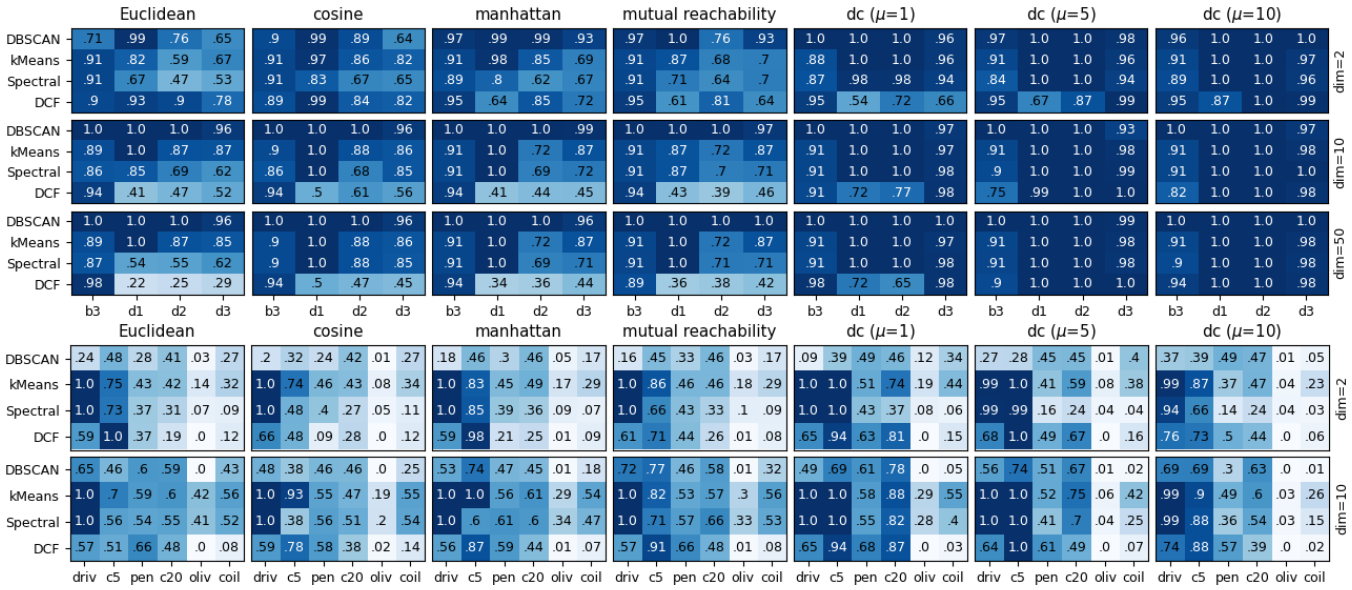

Figure 6: ARI of clusterings for different dimensionalities $d \in \{2, 10, (50)\}$ (top to bottom) and distance measures (left to right). Within each table, rows show results of different clustering algorithms, columns are different datasets. The top figure uses synthetic data, the bottom is on benchmark datasets.

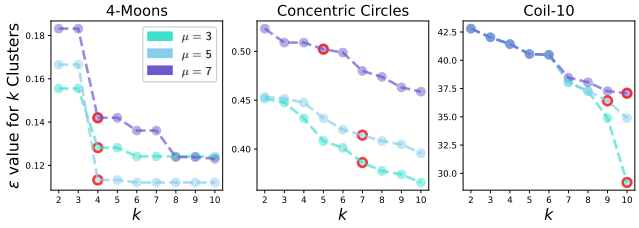

Figure 7: The effect of the number of clusters k on ε for varying μ on different datasets. The red circle indicates that this value of k obtained the highest NMI with the ground truth.

the connectivity graph G is a tree, the connected k-center problem can be solved optimally in $O(n^2 logn)$ even if the distance measure is not metric. In [16], the authors lay down the theoretical foundations for *admissible* objective functions in the field of *Hierarchical Clustering* where the ground-truth clustering has an optimal value.

7.4 Spectral Clustering

The relationship between spectral (SC) and density-based clustering has been studied as they are both able to find non-convex clusters. E.g., it has been shown that DBSCAN* returns connected components of the ε- neighborhood graph of the core points in a dataset [15] and they can be found by inspecting the multiplicity of 0 eigenvalues without applying an additional clustering step [56]. The authors of [56] also show the equivalence of ratio cut to kernel k-means [29] and determine the best parameter settings for *minPts* in DBSCAN to find clusters with a large average density by analyzing the spectrum. However, these works are based on a ε-graph for a *given* ε. In contrast, in this paper, we neither create an ε-graph nor is ε required for calculating the dc-dist. In [46], SC is applied to the results of DBSCAN* to examine substructures and it is accelerated by employing sparse representations of MSTs in [10]. SC is combined with the minimax distance in various works to identify the number of clusters [40] or achieve robustness [11, 22].

8 CONCLUSION

In this paper, we introduced the dc-dist, which is an ultrametric combining the concepts of density and minimax distance. It allowed us to formalize density-based clustering by a loss function instead of a procedural description. With an extension to the dc-dist, the DBSCAN-distance, we can also capture the exact definition of DBSCAN including border points. Most importantly, we showed that DBSCAN*, k-center, and spectral clustering are equivalent in the space given by the dc-dist, even though they are perceived as fundamentally different in current literature. Our extensive experiments showed, that the dc-dist captures the structure of data and improves clustering data that contains density-based clusters compared to other distance measures. By establishing the connection between three fundamental clustering methods, we offer plenty of advantages and build a stable foundation for future work: E.g., it allows choosing the number of clusters k for DBSCAN instead of the sensitive parameter ε. Furthermore, analyses can be performed more formally and extensions as well as acceleration methods can be transferred between the established algorithms.

ACKNOWLEDGMENTS

This project is partially funded by the Villum Foundation (project number 34326).

A PROOF OF CORRECTNESS OF ALGORITHM 2

Proof. Let G_X be a fully connected graph where edge lengths are the reachability distance between the corresponding points. Then the connectivity distance is defined as

$$d_c^{\mu}(x_i, x_j) = \min_{\text{path}(x_i,x_j)\in G_X} \left(\max_{e\in \text{ path}} |e| \right),$$

where μ is the min points parameter. Consider that Algorithm 2 builds up forests across the dataset in increasing reachability distance. Before the forests of x_i and x_j are connected, there does not exist a path between x_i and x_j. Upon connecting x_i and x_j, all paths must go through the recently added edge. Since Algorithm 2 adds the smallest such edge, this edge's length must be the connectivity distance. □

B DC-DIST ALGORITHM

Algorithm 2: Density-Connectivity Distance

Input: Dataset $\mathcal{X} = \{x_0, \cdots, x_n\}, x_i \in \mathbb{R}^d$
Output: Distance matrix **D**

```
D ← zeros((n, n));
MST ← MstClass();
for x ∈ X do
    MakeTree (MST, {x});
for edge(x_i, x_j) ∈ PairwiseDistances(X) increasing do
    if Tree(x_i) ≠ Tree(x_j) then
        Union(MST, Tree(x_i), Tree(x_j));
        for u ∈ Leaves(Tree((x_i))) do
            for v ∈ Leaves(Tree(x_j)) do
                D[u, v] ← distance(x_i, x_j);
return D
```

C PROOF OF LEMMA 3.3

Proof. Part (A): if points are core points of the same cluster for d_{dc}^{μ}, then the same holds for d_E and μ. Assume two points p, q are core points of the same DBSCAN*(ε, μ) cluster for d_{dc}^{μ}. Thus, there is a chain of core points (resp. d_{dc}^{μ}) between them where all $d_{dc}^{\mu}(v_i, v_j) \leq \varepsilon$ and p, q are core points themselves. I.e., $d_{dc}^{\mu}(v_i, v_j) = max(d_E(v_i, v_j), d_{E,core}^{\mu}(v_i), d_{E,core}^{\mu}(v_j)) \leq \varepsilon$. Thus, $d_E(v_i, v_j) \leq \varepsilon$ and all points on the chain are core points for that ε. Thus, p and q are also in a DBSCAN*(ε, μ) cluster for d_E.
Part (B): if points are core points of the same cluster for d_E and μ, then the same holds for d_{dc}^{μ}. Assume two points p, q are core points of the same DBSCAN*(ε, μ) cluster for d_E. Thus, there is a chain of core points (resp. d_E) between them where all $d_E(v_i, v_j) \leq \varepsilon$ and p, q are core points themselves: $d_{E,core}(v_i) \leq \varepsilon$ for all v_i on the path. Thus, as $d_{dc}(v_i, v_j) \leq d_r(v_i, v_j) = max(d_E(v_i, v_j), d_{E,core}(v_i), d_{E,core}(v_j)) \leq \varepsilon$ for all points on this path. Furthermore, all points on this path are core points w.r.t. d_{dc}^{μ}: Let us regard an arbitrary point p on the path and the distance to its μ-th nearest neighbor n_{μ}. The set of points on the path is a subset of the full dataset, thus, the distance to p's μ-th nearest neighbor *on the path*, n_{μ}^{path}, is larger than or equal to the distance to the μ-th nearest neighbor in the full dataset. As $d_{dc}(p, n_{\mu}^{path}) \leq max(d_{dc}(v_i, v_j))$ it holds that $d_{dc}(p, n_{\mu}^{path}) \leq \varepsilon$, thus $d_{dc}(p, n_{\mu}) \leq \varepsilon$, meaning p is a core point. □

D PROOF OF THEOREM 4.2

Proof. We proceed inductively by showing that, given two optimal solutions in the left and right subtrees, our merge criteria returns an optimal solution on the node in question. We first note that the base case is clearly optimal, as there are up to k centers where each only has itself as a point.

Now assume that we have an optimal $2k$ clustering – k centers each in the left and right subtrees. By the nature of our tree, it is always the case that the parent node has a higher value than either of its children. Thus, the maximum distance in any cluster is the value in the cluster's lowest common ancestor (or peak) – the node that is the parent to every element of the cluster. Furthermore, it must be the case that each cluster is closest to its immediate left or right neighbor and that the distance between two child nodes is larger than any distance within each child. Thus, our merge criteria must *always* merge two clusters whose peaks are the left and right child of a node.

We prove the optimality of this as follows: Assume FSOC that Algorithm 1 returns C and let the optimal clustering be C_{opt}. Then C_{opt} must have a cluster with lowest common ancestor N that could not be obtained from our merging process in the original $2k$ clusters. Since $\text{cost}(C_{opt}) < \text{cost}(C)$, it must be the case that N is contained in one of the clusters in C, i.e. c', which was obtained by merging c'_l and c'_r. It must also be the case that N's peak is greater than one of the peaks of c'_l or c'_r. Thus, N', the peak of our 'optimal' cluster, is between the peaks of c' and one of its children. Since there is no additional node between c' and its children, we have reached a contradiction. □

E ADDITIONS TO THE DBSCAN-DISTANCE

Proof. We show that d_{db}^{μ} is a metric. Per definition, it is symmetric and $d_{db}^{\mu}(p, p) = 0$. As all values for d_{db} are given by an underlying distance measure (either some $w^{d_E}(e)$ or d_{core}^{μ}), it holds that $d_{db}^{\mu}(p, q) \geq 0$ for all p, q.
For $\mu > 1$, for all $p : coredist(p) > 0$ (as we are on a data*set* containing only distinct objects), thus also for all $q \neq p$: $d_{db}^{\mu}(p, q) > 0$. As $d_{db}^{\mu}(p, p) = 0$ per definition, it holds that iff $d_{db}^{\mu}(p, q) = 0$ then $p = q$.
For proving the triangle inequality, let $d_{db}^{\mu}(p, q) = \varepsilon_{pq}$ and $d_{db}^{\mu}(q, r) = \varepsilon_{qr}$ and $d_{db}^{\mu}(p, r) = \varepsilon_{pr}$. Note that points in a cluster might not be core points themselves, but they are connected by a path of core points. Thus, there is a core point $q_p \in N_{\varepsilon_{pq}}(q)$ and a core point $q_r \in N_{\varepsilon_{qr}}(q)$. The distance between q_p and q_r is at most $\varepsilon_{pq}+\varepsilon_{qr}$ because of the triangle inequality of the underlying distance measure. Thus, also p and r are in a cluster for $\varepsilon_{pr} \leq \varepsilon_{pq} + \varepsilon_{qr}$. □

Note that the DBSCAN-distance is not an ultrametric and that it does not only use weights occurring in the mutual reachability graph.

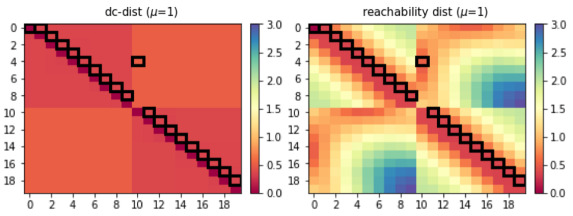

Figure 8: dc-dist (left) and mutual reachability distance (right) on the two-moons dataset. Black frames imply equal values.

Algorithm 3: DBSCAN distance

Input: Dataset $X = \{x_0, \cdots, x_n\}, x_i \in \mathbb{R}^d$
Output: Distance matrix **D**

$D_E \leftarrow$ euclidean_distance_matrix(X); $D_{dc} \leftarrow [\infty]$;
$sorted\_dists \leftarrow sort\_values(D_E, ascending)$;
$neighbors(v_i) \leftarrow \{\emptyset\}$ for all $v_i \in nodes$;
for $d_{ij} \in sorted\_dists$ **do**
  $\varepsilon \leftarrow d_{ij}$; $neighbors(v_i) += v_j$; $neighbors(v_j) += v_i$;
  if $m > 3$ **then**
    if d_{ij} *is m-th neighbor of* $v_x \in \{v_i, v_j\}$ **then**
      for $n_{x1}, n_{x2} \in neighbors(v_x) \times neighbors(v_x)$ **do**
        if $comp[n_{x1}] \neq comp[n_{x2}]$ **then**
          $merge\_comps(comp[n_{x1}], comp[n_{x2}], \varepsilon)$
  if $len(neighbors(v_i)) \geq m$ *and* $len(neighbors(v_j)) \geq m$ **then**
    for $v_k \in comp[v_i], v_l \in comp[v_j]$ **do**
      $merge\_comps(comp[v_k], comp[v_l], \varepsilon)$
  else if $len(neighbors(v_i)) \geq m$ **then**
    for $v_k \in comp[v_i]$ **do** $D_{dc}[v_k, v_j] \leftarrow \varepsilon$
  else if $len(neighbors(v_j)) \geq m$ **then**
    for $v_l \in comp[v_j]$ **do** $D_{dc}[v_l, v_i] \leftarrow \varepsilon$

Algorithm 4: Tree-k-Center-with-q-coverage

Input: Root node R; k, q
Output: k clusters

$C \leftarrow [\,]$ $S \leftarrow Prune(R)$;
/* Prune(R, q) returns a copy of those nodes in R that have at least q children */
$C^S \leftarrow$ Tree-k-Center(S, k) ;
$\varepsilon \leftarrow$ MaxEpsilon(C) ;
/* MaxEpsilon(C) gives the cost of the k-center solution C */
for $C_i^S \in C^S$ **do**
  $peak \leftarrow$ LeastCommonAncestor(C_i^S.points) ;
  while $peak.parent.value < \varepsilon$ **do**
    $peak \leftarrow peak$.parent ;
  C_i.points $\leftarrow peak$.orig_node.children ;
  /* .orig_node refers to the node in R that the peak of C_S is a copy of */
return C

F RELATIONSHIP TO OPTICS

Fig. 8 gives a visualization of d_r^μ for the two-moons dataset. Adjacent points have low mutual reachability distances resulting in low values around the diagonal. However, points that are equally well connected, but at opposite ends of a "moon", have high values for d_r^μ because of their high Euclidean distance. In contrast, the dc-dists on the left are uniformly low for *all* points of a cluster, which facilitates subsequent clustering tasks. Black frames imply the reachability distances d_{or} as used by OPTICS [2], for more details see Sec. 7.1.

G DETAILS ON THE EXPERIMENTS

Tab. 1 gives an overview of the properties of the datasets used in the experiments. For Fig. 4, the data was generated with the density-based data generator for dimensionalities $[2, 5, 10, 50, 100]$ for 10 seeds each. The number of data points was limited to 1000 from 5 disjoint clusters. For Fig. 7, two shifted copies of the 2-moons dataset (left) generated with sklearn, 6 concentric circles of radii $[1, 2, ..., 6]$ with noise $\sim \mathcal{N}(0, 0.1)$ (middle) and 10 classes from the coil-100 dataset (right) were used. Experiments were performed on an AMD Ryzen 7 2700X Eight-Core Processor @ 3.70GHz using 64GB RAM as well as an Intel(R) Core(TM) i7-7700K Eight-Core Processor @ 4.20GHz.

Table 1: Dataset properties used in the analysis.

	name	n	d	k	source
syn.	b3	10k	50	10	sklearn
	d1, d2, d3	10k	50	10	own
	Fig. 4	1k	2-100	5	own
benchmark	driver faces	606	921,600	4	[18]
	coil-5	360	186,368	5	[49]
	pendigits	7,494	16	10	[18]
	coil-20	1,440	16,384	20	[49]
	olivetti	400	4,096	40	sklearn [6]
	coil-100	7,200	49,152	100	[48]

H DATA GENERATOR FOR DENSITY-BASED CLUSTERS

The data generator used in this paper is based on the Seed Spreader described by Junhao Gan and Yufei Tao in [24]. The basic procedure uses a random walk with restarts where the data points themselves are generated uniformly at random in a local hypersphere. After a set amount of points, the hypersphere position is adjusted in a random direction, and the process is repeated.
We made adaptions s.t. the number of clusters can be given, clusters have a meaningful size, and density can differ between different clusters. We furthermore ensure that clusters are not overlapping. Noise points are generated uniformly across the data space spanned by the cluster points. They are assigned to a cluster if they are within a minimum distance of the initial data points of a cluster.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_olivetti_faces.html