# Efficient Computation and Visualization of Multiple Density-Based Clustering Hierarchies

Antonio Cavalcante Araujo Neto, Jörg Sander, Ricardo J. G. B. Campello, and Mario A. Nascimento

**Abstract**—HDBSCAN*, a state-of-the-art density-based hierarchical clustering method, produces a hierarchical organization of clusters in a dataset *w.r.t.* a parameter $mpts$. While a small change in $mpts$ typically leads to a small change in the clustering structure, choosing a "good" $mpts$ value can be challenging: depending on the data distribution, a high or low $mpts$ value may be more appropriate, and certain clusters may reveal themselves at different values. To explore results for a range of $mpts$ values, one has to run HDBSCAN* for each value independently, which can be computationally impractical. In this paper, we propose an approach to efficiently compute *all* HDBSCAN* hierarchies for a *range* of $mpts$ values by building upon results from computational geometry to replace HDBSCAN*'s complete graph with a smaller equivalent graph. An experimental evaluation shows that our approach can obtain over one hundred hierarchies for the computational cost equivalent to running HDBSCAN* about twice, which corresponds to a speedup of more than 60 times, compared to running HDBSCAN* independently that many times. We also propose a series of visualizations that allow users to analyze a collection of hierarchies for a range of $mpts$ values, along with case studies that illustrate how these analyses are performed.

**Index Terms**—Clustering, density-based clustering, hierarchical clustering, data mining, relative neighborhood graph

## 1 Introduction

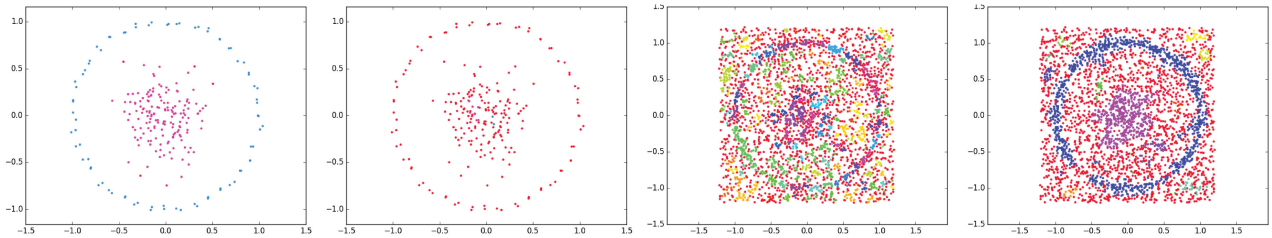
The discovery of groups within datasets plays an important role in the exploration and analysis of data. For scenarios where there is little to no prior knowledge about the data, clustering techniques are widely used. Density-based clustering, in particular, is a popular clustering paradigm that defines clusters as high-density regions in the data space, separated by low-density regions. Algorithms in this class, such as DBSCAN [16], DENCLUE [21], OPTICS [3] and HDBSCAN* [9], stand out for their ability to find clusters of arbitrary shapes and to differentiate between cluster points and noise. HDBSCAN*, the current state-of-the-art among those, computes a *hierarchy* of nested clusters, representing clusters at different density levels. It generalizes and improves several aspects of previous algorithms, and allows for a comprehensive framework for cluster analysis, visualization, and unsupervised outlier detection [9]. It requires a single parameter $mpts$, a smoothing factor that implicitly influences which clusters are detectable in the cluster hierarchy. Choosing a "correct" value for $mpts$ is typically not trivial. For instance, consider the examples in Fig. 1, which shows the results of HDBSCAN* (using automatic cluster extraction) for two datasets $A$ and $B$ and two sample $mpts$ values, $mpts = 5$ and 25. Dataset A is completely labeled as noise for $mpts > 24$, while the two structures in dataset B only start to be detected for $mpts > 24$. The main observation here is that (1) there is no single value of $mpts$ that would result in the extraction of the clusters in both cases, and (2) a user would not know which value for $mpts$ is suitable for a general dataset. It may even be the case that different values of $mpts$ are needed to reveal clusters in different areas of the data space of the same dataset.

To analyze clustering structures in practice, users typically run HDBSCAN* (like other algorithms with a parameter) multiple times with several different $mpts$ values, and explore the resulting hierarchies. Ideally, one would want to analyze cluster structures *w.r.t.* a large range of $mpts$ values, in order to fully explore a dataset in a given application. A larger range of HDBSCAN* solutions for multiple values of $mpts$ values offers greater insight into a dataset, also providing additional opportunities for exploratory data analysis. For instance, using internal cluster validation measures such as DBCV [27], one can identify promising density levels from different hierarchies, produced from different tunings of the algorithm's density estimates (based on $mpts$).

However, one is typically constrained by the non-negligible computational cost of running HDBSCAN* once for each desired value of $mpts$. The main component of the computational cost is due to the fact that HDBSCAN* is based on computing a Minimum Spanning Tree (MST) for a *complete* graph for a given value of $mpts$. Even though this complete graph does not need to be explicitly stored, it has $O(n^2)$ edges (for $n$ data points) whose weights depend on $mpts$. For each desired value of $mpts$, these weights have to be re-computed and an MST has to be constructed for the corresponding complete graph (we note that the computational cost for the MST

- *A. Cavalcante Araujo Neto, J. Sander, and M.A. Nascimento are with the Department of Computing Science, University of Alberta, Edmonton, AB T6G 2R3, Canada. E-mail: {antonio.cavalcante, jsander, mario.nascimento}@ualberta.ca.*
- *R.J.G.B. Campello is with the School of Mathematical and Physical Sciences, University of Newcastle, Callaghan, NSW 2308, Australia. E-mail: ricardo.campello@newcastle.edu.au.*

*Manuscript received 8 June 2018; revised 25 Sept. 2019; accepted 12 Dec. 2019. Date of publication 25 Dec. 2019; date of current version 7 July 2021. (Corresponding author: Antonio Cavalcante Araujo Neto.) Recommended for acceptance by X. Zhu. Digital Object Identifier no. 10.1109/TKDE.2019.2962412*



(a) $A$, $mpts = 5$ (b) $A$, $mpts = 25$ (c) $B$, $mpts = 5$ (d) $B$, $mpts = 25$

Fig. 1. Clusters from datasets $A$ and $B$ for $mpts = 5$ and 25. Noise points are colored in red in all plots.

construction depends on the number of edges in the input graph, $O(n^2)$ in this case).

As the main contribution we provide theoretical and practical results that lead us to a method for computing multiple hierarchies *w.r.t.* a *range* of $mpts$ values $(k_1, \ldots, k_{max})$, which is much more efficient than re-running HDBSCAN* for each individual value of $mpts$ in that range. This method gives access to a large range of HDBSCAN* solutions at a low computational cost, in fact equivalent to the cost of running the original HDBSCAN* for only 1 or 2 values of $mpts$. To achieve that, we show the following:

1) The smallest known neighborhood graph containing the Euclidean Minimum Spanning Tree (EMST) is the relative neighborhood graph (RNG)—as a first step towards finding a small, single spanning subgraph that can replace the complete graph in HDBSCAN*, while maintaining its correctness.
2) The proximity measure used in HDBSCAN*, which depends on $mpts$, can be used to define RNGs that can replace the complete graph in HDBSCAN* with one RNG for each value of $mpts$.
3) For a range of $mpts$ values, RNGs for smaller values are contained in RNGs for larger values of $mpts$, that is, a *single* RNG is sufficient to compute the hierarchies for the whole range of $mpts$ values.
4) Information related to "core-distances" that is needed in HDBSCAN* and that can be computed in a pre-processing step, allows us to formulate a highly efficient strategy for computing the single RNG, suitable for a whole range of $mpts$ values.

These results combined allow us to replace the (virtual) complete graph of the data in HDBSCAN* with a single, pre-computed RNG that contains all the edges needed to compute the hierarchies for every value of $mpts \in [1, k_{max}]$. Moreover, this RNG has typically much fewer edges than the complete graph so its construction cost is more than outweighed by the reduction in edge weight computations.

This paper is an extension of [29]. We add here all the proofs that guarantee the correctness of our proposed approach. More importantly, as a new contribution, we present a series of visualization methods that can be used to thoroughly analyze a set of hierarchies *w.r.t.* a range of $mpts$ values. We illustrate the use of those visualization methods by analyzing case studies based on real datasets.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 covers the concepts and techniques used in this paper. Section 4 presents our proposal and proves its correctness. Section 5 shows and discusses the results of our experimental evaluation. Section 6 introduces new means to visualize clusters that help with the analysis of a set of hierarchies. Finally, Section 7 offers conclusions and directions for future work.

## 2 Related Work

To the best of our knowledge, there is no previous proposal for computing *multiple* clustering hierarchies efficiently. There are works on automatic parameter selection strategies for density-based clustering, e.g., [13], [25], [30], which are loosely related to the issue illustrated in Fig. 1. However, those proposals are unsuitable to be used with HDBSCAN*, since they were developed for non-hierarchical clustering algorithms. In addition, they rely on assumptions that are often not satisfied in practice and there is not enough evidence to support their claims about parameter optimality.

If we denote the HDBSCAN*'s (virtual) complete graph for a given $mpts$ by $G_{mpts}$, a line of work related to our goal of reducing the cost for computing an MST for each $G_{mpts}$, are the works regarding (1) dynamically updating graphs, specifically MSTs, and (2) neighborhood graphs that could potentially replace HDBSCAN*'s (virtual) complete graph. We discuss some of those next.

The authors of [12], [20], [23], studied the problem of maintaining dynamic MSTs. However, these approaches are more suitable when the changes in the underlying graph take place sequentially, i.e., considering each operation (e.g., edge updates) individually. When it comes to major changes taking place globally and simultaneously across the entire graph, as opposed to a few localized changes, a sequence of applications of these techniques tends to be computationally very costly, possibly even more costly than the construction of the entire MST from scratch. This is the case for $G_{mpts}$, which is a complete graph whose majority of edges will likely change as a result of a change in the $mpts$ value.

The works on neighborhood graphs that are most related to our proposal aim at speeding up the special case of computing a *Euclidean* Minimum Spanning Tree, by first computing a spanning subgraph that is guaranteed to contain all the EMST edges. One of these strategies uses a Delaunay Triangulation [14] of the complete euclidean graph $G$, since it has been shown that the EMST is contained in the Delaunay Triangulation of $G$ [32]. Other spanning subgraphs of the complete graph $G$ that contain the EMST are the Gabriel Graph [17], [33] and the Relative Neighborhood Graph [32]. Unfortunately, these results are not simply applicable to our problem because $G_{mpts}$ lies in a transformed space of the data that depends on $mpts$ ($G_{mpts} \neq G$). It is one of the main contributions of this paper to formally show how to adapt an RNG so that it

can be used by HDBSCAN* as a suitable replacement for $G_{mpts}$ for different values of $mpts$.

## 3 Background

HDBSCAN* stands out for its ability to detect arbitrary-shaped clusters, and noise, as well as for building a hierarchical organization of cluster structures, rather than finding a single flat partitioning of the data. It can be considered a practical and theoretical generalization of its predecessors (DBSCAN and OPTICS). HDBSCAN*'s main output is a cluster hierarchy that describes the nested structure of density-based clusters in a dataset *w.r.t.* the parameter $mpts$. In order to determine this structure in a dataset $\mathbf{X}$, one needs to know (i) for each point $p \in \mathbf{X}$: the smallest radius $\varepsilon$ around $p$ that covers $mpts$ other points, called $p$'s "core distance" *w.r.t.* $mpts$; and (ii) for each value of $\varepsilon$: the clusters and the noise points *w.r.t.* $\varepsilon$ and $mpts$. The latter information can be derived conceptually from a complete, edge-weighted graph, called Mutual Reachability Graph, denoted by $G_{mpts}$, where nodes represent the points in $\mathbf{X}$, and the *edge weight* of an edge between two points $p$ and $q$ corresponds to the "mutual reachability distance" (*w.r.t.* $mpts$) between $p$ and $q$, defined as

$$mrd_{mpts}(p,q) = max\{c_{mpts}(p), c_{mpts}(q), d(p,q)\}, \quad (1)$$

where $d(\cdot,\cdot)$ represents the underlying distance function (typically Euclidean Distance), and $c_{mpts}(p)$ represents the core distance of $p$, which is formally the distance from $p$ to its $mpts$-th nearest neighbor, $mpts{-}\mathrm{NN}(p)$

$$c_{mpts}(p) = d(p, mpts{-}\mathrm{NN}(p)). \quad (2)$$

In this work, we assume that the underlying distance $d(\cdot,\cdot)$ is a proper metric, and, without loss of generality, we use Euclidean Distance in our examples.

Intuitively, an edge weight in $G_{mpts}$ corresponds to the minimum radius $\varepsilon$ at which the corresponding endpoints are directly $\varepsilon$-reachable *w.r.t.* $mpts$, i.e., the smallest distance at which both points are in each other's $\varepsilon$-neighborhood, and both $\varepsilon$-neighborhoods contain at least $mpts$ points (i.e., both are dense). Moreover, $G_{mpts}$ has the following important characteristics related to $mrd_{mpts}$ and to how these edge weights change when changing the value of $mpts$: (1) Increasing the value of $mpts$ usually leads to higher values of $c_{mpts}$, never smaller; (2) When increasing $c_{mpts}$, more edges tend to have the same edge weight, since a point $p$ with a high $c_{mpts}$ value determines the weight of all edges between itself and its $mpts$-nearest neighbors with a smaller $c_{mpts}$, given the definition of $mrd_{mpts}$ as a max function; and (3) When decreasing the value of $mpts$, edge weights can either decrease or remain the same, but never increase. These characteristics allow us to devise and prove the correctness of the strategy proposed in this work.

Considering the concepts represented by $G_{mpts}$, the HDBSCAN* hierarchy *w.r.t.* $mpts$ for a dataset $\mathbf{X}$ is computed in the following way: First, the core distances of all points in $\mathbf{X}$ *w.r.t.* $mpts$ are computed. Then, an MST of $G_{mpts}$ is dynamically computed.[1] From this MST, the *complete* density-based cluster hierarchy *w.r.t.* $mpts$ is extracted, by removing edges from the MST in descending order of edge weight, and (re-)labeling the connected components and noise at the "next" resulting level. For a specific density level ($\varepsilon$ and $mpts$), removing all edges from $G_{mpts}$ with weights greater than $\varepsilon$ reveals the maximal, connected components, i.e., clusters, of that density level. The density-based clustering hierarchy can thus be compactly represented by (and more easily be extracted from) a Minimum Spanning Tree of $G_{mpts}$.

1. The authors of HDBSCAN* [7], [9] deem $G_{mpts}$ a conceptual graph as it does not need to be explicitly materialized or stored; edge weights can be computed on demand, when needed.

## 4 Approach

When HDBSCAN* has to be run for a range, $k_1, \ldots, k_{max}$, of $mpts$ values, one MST for each value of $mpts \in \{k_1, \ldots, k_{max}\}$ has to be computed by taking the complete, unweighted graph $G$ of the dataset, adding to it mutual reachability distances as edge weights, to obtain an $G_{mpts}$, and then computing the MST of this graph; the $O(n^2)$ edge weights of $G_{mpts}$ have to be re-computed for each $mpts \in \{k_1, \ldots, k_{max}\}$ by running a $k$-Nearest-Neigbor ($k$-NN) query for each point in the dataset with $k$ equal to the current $mpts$ value, in order to determine core distances.

One straightforward way to speed-up multiple runs of HDBSCAN* for all values of $mpts \in \{k_1, \ldots, k_{max}\}$ is to execute $k$-NN queries for each point only once, using the largest value $k_{max}$ in the range, and materialize the $k_{max}$-NN query results. Since the $k$-NNs for $k \leq k_{max}$ are part of the $k_{max}$-NNs, the core distances for all values in $\{k_1, \ldots, k_{max}\}$ can be easily obtained from the materialized $k_{max}$-NN query results. While this approach reduces the number of $k$-NN queries that have to be executed significantly, a main determining factor of the total runtime is still the large number of edges in the complete graphs that have to be processed to construct MSTs. In the following, we will formally prove that we can construct a single graph that is typically significantly smaller than a complete graph, yet still contains the edges of the MSTs of $G_{mpts}$ for all $mpts \in \{k_1, \ldots, k_{max}\}$. Thus, we can use this graph instead of the complete graph in HDBSCAN*, without changing the correctness of the results. How much speed-up can be achieved depends, of course, not only on the reduction in number of edges from the complete graph, but also on the added computational cost for constructing this graph.

### 4.1 Results From Computational Geometry

Consider first the special case of $mpts = 1$, where all core distances are equal to zero, and thus the mutual reachability distance $mrd_{mpts}$ reduces to the underlying distance function. With Euclidean Distance, what HDBSCAN* has to compute then is the Euclidean Minimum Spanning Tree of a dataset $\mathbf{X}$, i.e., the MST of a complete graph of $\mathbf{X}$ with Euclidean Distance between points as edge weights.

For the EMST, there are known results from computational geometry that relate the EMST to the Delaunay Triangulation (DT), the Gabriel Graph (GG) and the Relative Neighborhood Graph in the following way [32]:

$$EMST \subseteq RNG \subseteq GG \subseteq DT. \quad (3)$$

The RNG and GG are special cases of a family of graphs called $\beta$-skeletons [26], which can range from the complete

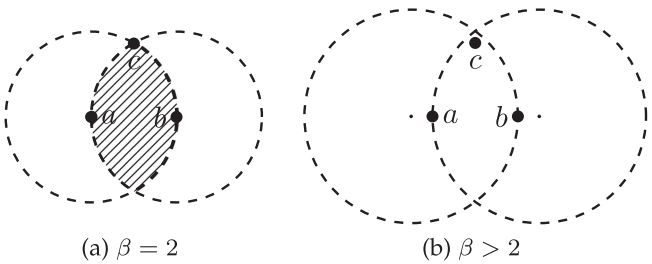



Fig. 2. $\beta$-skeletons.

graph to the empty graph, when $\beta$ goes from 0 to $\infty$. A value of $\beta = 1$ results in the GG and $\beta = 2$ in the RNG.

Given this result, the RNG, or possibly a $\beta$-skeleton with even fewer edges, may be a good replacement for a complete graph, if we can answer the following questions positively:

1) Can we determine the smallest $\beta$-skeleton, *w.r.t.* number of edges, that has the EMST as a subgraph?
2) Can the results for Euclidean Distance be generalized to other reachability distances *w.r.t.* $mpts > 1$?
3) Is there a single $\beta$-skeleton that contains all the edges needed to compute an MST of $G_{mpts}$ for each value of $mpts$ in a range of values $k_1, \ldots, k_{max}$?
4) Does the reduction in the number of edges justify the additional computational cost for constructing and materializing a $\beta$-skeleton for our task?

We will answer these questions in the following subsections.

### 4.2 The Smallest $\beta$-Skeleton Containing the EMST

The family of $\beta$-skeletons for a set of $d$-dimensional points is defined as follows. For a given $\beta$, an edge exists between two points $a$ and $b$ if and only if (*iff*) the intersection of the two balls centered at $(\beta/2)a + (1 - \beta/2)b$ and $(1 - \beta/2)a + (\beta/2)b$, both with radius $\beta d(a, b)/2$, is empty. For instance, when $\beta = 2$ (RNG), the centers of the balls coincide with $a$ and $b$, and the radius is equal to $d(a, b)$, as shown in Fig. 2a. The highlighted region, called $lune(a, b)$, must be empty for an edge to exist between $a$ and $b$. For $\beta = 2$, one can equivalently say that an edge exists between $a$ and $b$ *iff*

$$d(a, b) \leq max\{d(a, c), d(b, c)\}, \forall c \neq a, b. \quad (4)$$

The RNG is guaranteed to contain the EMST [32], which, in essence, can be demonstrated by considering a configuration of three points $a$, $b$, $c$, such that $lune(a, b)$ contains $c$, as shown in Fig. 3a. The edges $(a, b)$, $(a, c)$ and $(b, c)$ cannot all be part of an EMST, as they form a cycle. Since $(a, b)$ is the largest of these edges, $(a, b)$ cannot be part of the EMST. Thus, a necessary (but not sufficient) condition for an edge $(a, b)$ to be in an EMST is that all other points must lie outside $lune(a, b)$. Hence, for a complete graph $G = (V, E)$, $RNG = (V, E \setminus \{(a, b) : lune(a, b) \neq \emptyset\})$ contains the EMST.

Here, we prove by counter example that the RNG is also the smallest $\beta$-skeleton (i.e., there is no $\beta > 2$) with that property. Consider a dataset with three points $a$, $b$ and $c$, located at equal distance from each other, as illustrated in Fig. 2. When $\beta = 2$ (Fig. 2a), according to Inequality (4), there is an edge between every pair of points in the 2-skeleton of this dataset. For any $\beta > 2$ (Fig. 2b), however, the radius of the balls that define $lune(a, b)$ is increased by a factor of $(\beta - 2)/2$, and the centers of the balls are "pulled apart" accordingly, so that $c$ (equidistant from $a$ and $b$) must now be inside $lune(a, b)$. Thus, $a$ and $b$ are (by definition of $\beta$-skeleton) no longer connected by an edge. Analogously, there is no edge between the other pairs of points for $\beta > 2$, resulting in an empty $\beta$-skeleton that obviously cannot contain the EMST. Thus, it follows with the known result in Expression (3) that the RNG ($\beta = 2$) is the smallest $\beta$-skeleton that contains the EMST and, for this reason, we choose it as the basis for further analysis.

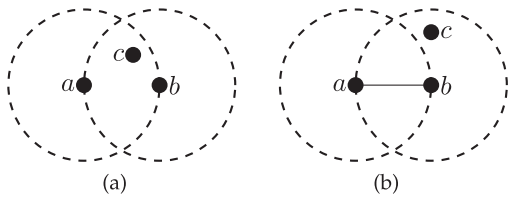



Fig. 3. Illustration for proofs of Theorems 1 and 2.

### 4.3 The RNG *w.r.t.* Mutual Reachability Distance

In this section, we prove that the results for RNGs in euclidean space can be extended to the space of mutual reachability distances.

*Notation:* (1) Let $G = (V, E)$ denote the undirected, unweighted complete graph corresponding to a dataset, i.e., the set of vertexes $V$ represents the data points, and the set of edges $E \subset V \times V$ represents all pairs of vertexes/points. (2) Let $G_i = (V, E, mrd_i)$ be the mutual reachability graph for $mpts = i$, i.e., the weighted, complete graph for the dataset with edge weights between points $p$ and $q$ equal to $mrd_i(p, q)$, the mutual reachability distance *w.r.t.* $mpts = i$.

We can define a relative neighborhood graph *w.r.t.* the mutual reachability distance $mrd_i$, $RNG^i$, as follows:

**Definition 1.** *$RNG^i = (V, E')$ where $E' \subseteq E$ and there is an edge $(a, b) \in E'$ if and only if*

$$mrd_i(a, b) \leq max\{mrd_i(a, c), mrd_i(b, c)\}, \forall c \neq a, b;$$

*and when there is an edge $(a, b) \in E'$, we say that $a$ and $b$ are relative neighbors w.r.t. $mrd_i$. The unweighted graph $RNG^i$ can be extended with edge weights defined by a distance function $mrd_j$, which results in the edge weighted graph $RNG^i_j$, where the weight of an edge connecting two points $p$ and $q$ is equal to $mrd_j(p, q)$.*

We can prove that the $RNG^i_i$ contains the MST of $G_i$, and thus we can replace $G_i$ with $RNG^i_i$ when running HDBSCAN* for $mpts = i$.

**Theorem 1.** $MST(G_i) \subseteq RNG^i_i$

**Proof 1.** The argument for why $EMST \subseteq RNG$ [32] is in fact valid for any distance function as edge weight as long as it is symmetric and satisfies triangle inequality, which are all that is needed to guarantee that $(a, b)$ is in fact the largest edge in configurations like the one shown in Fig. 3a. Consequently, we only need to show that $mrd_i$ is symmetric and satisfies triangle inequality.

For symmetry, we can see from the definition of $mrd_{mpts}$ in Equation (1) that $mrd_i(a,b) = mrd_i(b,a)$ (given that the underlying distance $d$ is symmetric by assumption).

For the triangle inequality, we have to show that for all $a, b, c$ in a dataset $\mathbf{X}$

$$mrd_i(a,c) \leq mrd_i(a,b) + mrd_i(b,c). \tag{5}$$

By assumption (Section 3), the underling distance $d$ in the definition of $mrd_i$ satisfies the triangle inequality

$$d(a,c) \leq d(a,b) + d(b,c). \tag{6}$$

There are three cases according to the definition of $mrd_i(a,c)$, in all of which (5) must hold:

1) $mrd_i(a,c) = c_i(a)$. The $max$ function in the definition of $mrd_i$ implies $c_i(a) \leq mrd_i(a,b)$. Hence, $mrd_i(a,c) = c_i(a) \leq mrd_i(a,b) \leq mrd_i(a,b) + mrd_i(b,c)$.
2) $mrd_i(a,c) = c_i(c)$. (Analogous to case 1).
3) $mrd_i(a,c) = d(a,c)$. Since for any $x, y$ it holds that $x \leq max(x,y)$, we can replace the terms on the right side of Inequality (6) with $max$ functions to obtain, $d(a,c) \leq max\{d(a,b), c_i(a), c_i(b)\} + max\{d(b,c), c_i(b), c_i(c)\} = mrd_i(a,b) + mrd_i(b,c)$, and hence, also in this case: $mrd_i(a,c) = d(a,c) \leq mrd_i(a,b) + mrd_i(b,c)$.

Since $mrd_i$ satisfies symmetry and triangle inequality, it follows from [32] that $RNG_i^i$ contains the MST of $G_i$. □

### 4.4 One RNG To Rule Them All

We have established that we can use $RNG_i^i$ as a substitute for $G_i$ in HDBSCAN*. We will now show that all MSTs for HDBSCAN* *w.r.t.* $mpts \in \{k_1, \ldots, k_{max}\}$ can be obtained from the single graph $RNG^{k_{max}}$. For this we only need to show that $RNG^i \subseteq RNG^{k_{max}}$, for all $i < k_{max}$. Then, we can use the single graph $RNG^{k_{max}}$ to compute the MST of any $G_i$ by adding edge weights $mrd_i$ to $RNG^{k_{max}}$, and computing the MST of this edge-weighted graph $RNG_i^{k_{max}}$.

**Theorem 2.** $RNG^i \subseteq RNG^{k_{max}}, \forall i < k_{max}$.

**Proof 2.** To prove this by contradiction, assume that there is a $j < k_{max}$ for which $RNG^j \not\subseteq RNG^{k_{max}}$. Then, there must be at least one edge $(a,b)$ that belongs to $RNG^j$ but does not belong to $RNG^{k_{max}}$. According to the definition of relative neighborhood graphs, this means that there is a point $c$, such that for distance $mrd_{k_{max}}$, $c \in lune(a,b)$, and for distance $mrd_j$, $c \notin lune(a,b)$, as illustrated in Fig. 3. More formally:

For $RNG^{k_{max}}$ (Fig. 3a) *both* of the following inequalities must be satisfied so that $c \in lune(a,b)$.

$$mrd_{k_{max}}(a,b) > mrd_{k_{max}}(a,c) \tag{7}$$

$$mrd_{k_{max}}(a,b) > mrd_{k_{max}}(b,c). \tag{8}$$

For $RNG^j$ (Fig. 3b) *at least one* of the following inequalities must be satisfied so that $c \notin lune(a,b)$.

$$mrd_j(a,b) \leq mrd_j(a,c) \tag{9}$$

$$mrd_j(a,b) \leq mrd_j(b,c). \tag{10}$$

Using the definition of $mrd_{k_{max}}$, we can rewrite Inequalities (7) and (8) as follows:

$$\begin{matrix} max\{c_{k_{max}}(a), c_{k_{max}} & (b), d(a,b)\} > \\ max\{c_{k_{max}}(a), c_{k_{max}} & (c), d(a,c)\} \end{matrix} \tag{11}$$

$$\begin{matrix} max\{c_{k_{max}}(a), c_{k_{max}} & (b), d(a,b)\} > \\ max\{c_{k_{max}}(b), c_{k_{max}} & (c), d(b,c)\}. \end{matrix} \tag{12}$$

There are three cases, $c_{k_{max}}(a)$, $c_{k_{max}}(b)$, and $d(a,b)$, that the $max$ function on the left-hand side of the Inequalities (11) and (12) can evaluate to. If it evaluates to one of the core distances, we get an immediate contradiction with one of the Equations (11) and (12): In case $max\{c_{k_{max}}(a), c_{k_{max}}(b), d(a,b)\} = c_{k_{max}}(a)$, we get from Inequality (11) the following:

$$c_{k_{max}}(a) > max\{c_{k_{max}}(a), c_{k_{max}}(c), d(a,c)\}.$$

But since $max(c_{k_{max}}(a), \ldots) \geq c_{k_{max}}(a)$, it follows that $c_{k_{max}}(a) > c_{k_{max}}(a)$, a contradiction! In case $max\{c_{k_{max}}(a), c_{k_{max}}(b), d(a,b)\} = c_{k_{max}}(b)$, it follows, analogously to the previous case, from (12) that $c_{k_{max}}(b) > c_{k_{max}}(b)$, a contradiction again! If it does not evaluate to one of the core distances, i.e., $max\{c_{k_{max}}(a), c_{k_{max}}(b), d(a,b)\} = d(a,b)$, all the following inequalities must hold.

$$d(a,b) > c_{k_{max}}(a) \tag{13}$$

$$d(a,b) > c_{k_{max}}(b) \tag{14}$$

$$d(a,b) > c_{k_{max}}(c) \tag{15}$$

$$d(a,b) > d(a,c) \tag{16}$$

$$d(a,b) > d(b,c). \tag{17}$$

We also know that at least one of the Inequalities (9) and (10) must hold, under our assumption that $c \notin lune(a,b)$ for distance $mrd_j$. We can rewrite (9), using the definition of $mrd_j$ as follows:

$$\begin{matrix} max\{c_j(a), c_j & (c), d(a,c)\} \geq \\ max\{c_j(a), c_j & (b), d(a,b)\}. \end{matrix} \tag{18}$$

There are again the three possibilities, $c_j(a), c_j(c)$, $d(a,c)$, that the $max$ function on the left-hand side of Inequality (18) can evaluate to, and we show that each one contradicts what we already know about $a$, $b$, and $c$:

1) $max\{c_j(a), c_j(c), d(a,c)\} = c_j(a)$.

In this case, Inequality (18) yields the following.

$$c_j(a) \geq d(a,b). \tag{19}$$

Since core distances $c_{mpts}$ can only increase when $mpts$ increases, we have $c_{k_{max}}(a) \geq c_j(a)$ and, accordingly, we obtain the following from Inequality (19).

$$c_{k_{max}}(a) \geq d(a,b). \tag{20}$$

This contradicts Inequality (13)!

2) $max\{c_j(a), c_j(c), d(a,c)\} = c_j(c)$.

Analogously to the previous case, from (18) we get (21), and then from $c_{k_{max}}(c) \geq c_j(c)$ we get (22), which contradicts Inequality (15)!

$$c_j(c) \geq d(a,b) \tag{21}$$

$$c_{k_{max}}(c) \geq d(a,b). \tag{22}$$

3) $max\{c_j(a), c_j(c), d(a,c)\} = d(a,c)$.

In this case, we get from Inequality (18) that $d(a,c) \geq d(a,b)$, which is a contradiction to Inequality (16)!

This proves that Inequality (9) cannot hold under our assumption. We can prove analogously the same result for Inequality (10), which contradicts our assumption that there is a $j < k_{max}$ such that $RNG^j \not\subseteq RNG^{k_{max}}$. Hence $RNG^i \subseteq RNG^{k_{max}}, \forall i \leq k_{max}$. □

When we combine the results of Theorems 1 and 2, we obtain the following corollary, which states that the $MST(G_i)$ for all $i < k_{max}$ is contained in $RNG^{k_{max}}$, and can thus be computed from $RNG_i^{k_{max}}$, the graph obtained by extending $RNG^{k_{max}}$ with edge weights $mrd_i$.

**Corollary 1.** $MST(G_i) \subseteq RNG_i^{k_{max}}, \forall i \leq k_{max}$.

**Proof 3.** $MST(G_i) \subseteq RNG_i^i$ (Theorem 1), and $RNG^i \subseteq RNG^{k_{max}}$ (Theorem 2). By extending both graphs from Theorem 2 with edge weights $mrd_i$, we obtain $RNG_i^i \subseteq RNG_i^{k_{max}}$. Hence, $MST(G_i) \subseteq RNG_i^{k_{max}}$. □

### 4.5 RNG Computation

The performance gain when running HDBSCAN* *w.r.t.* all values of $mpts \in \{k_1, \ldots, k_{max}\}$ by using $RNG_i^{k_{max}}$ instead of the complete graph $G$ relies on a number of factors: the additional time to construct $RNG_i^{k_{max}}$ (recall that $G$ does not have to be explicitly constructed), the number of edges in $RNG_i^{k_{max}}$ compared to $G$, and the number of hierarchies $k_{max}$ to be computed.

The naive way to compute an RNG for a set of points $\mathbf{X}$ is to check for every pair of points $p, q \in \mathbf{X}$ and each third point $c \in \mathbf{X}$, whether $c$ is inside $lune(p,q)$. This algorithm runs in $O(n^3)$ time, which is inefficient for large datasets. More efficient strategies are surveyed in [24].

We adopt the approach in [2]—which has sub-quadratic expected time complexity under the assumption that points are in general position—with an adaptation of the definition of well-separated pairs proposed in [6]. In the first step, the entire dataset is decomposed recursively into smaller and smaller subsets, so that all pairs of obtained subsets $(A, B)$ are *well-separated* (see [6] for details).

Intuitively, two sets $A$ and $B$ are well-separated "if the diameter of each set is relatively small compared to the distance between the two sets" [5]. The distance is in our case the mutual reachability distance $mrd_{mpts}$, and the smallest possible $mrd_{mpts}$ between two point sets $A$ and $B$ is the shortest possible Euclidean Distance between a point $a \in A$ and a point $b \in B$, because of the max function in the definition of $mrd_{mpts}$. In order to avoid computing pairwise distances, one can use "safe" bounds instead of the exact distances to define well-separability (the only consequence of using bounds instead of exact distances is that more well-separated pairs may be generated than necessary). The distance between the sets $A$ and $B$ can be bounded, as in [6], by the distance $D(A,B)$ between the smallest enclosing balls $B_A$ and $B_B$ around the minimum bounding hyper-rectangles enclosing $A$ and $B$, respectively. The largest possible mutual reachability distance within the sets $A$ and $B$ can be bounded by $max\{diameter(B_A), diameter(B_B), \max_{p \in A \cup B}(c_{mpts}(p))\}$. Then, we can define that $A$ and $B$ are well-separated if

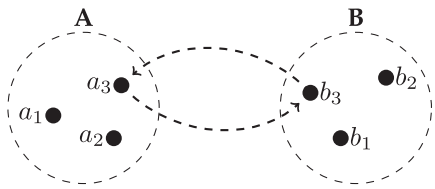


Fig. 4. Symmetric Bichromatic Closest Neighbor (SBCN).

$$D(A,B) \geq s \cdot \max\{diameter(B_A), diameter(B_B), \max_{p \in A \cup B}(c_{mpts}(p))\}.$$

The separation factor $s > 0$ determines how far both sets have to be from each other to be considered *well-separated*.

The larger the separation factor, the larger the number of generated pairs. For $0 < s < 1$, there is no guarantee that the resulting graph will contain the MST edges, and hence, to have as few edges as possible, we adopt $s = 1$.

In the second step, a supergraph of the RNG, which we will call RNG**, is constructed. For each well-separated pair $(A, B)$, the points $a_i \in A$ and $b_j \in B$ are connected with an edge if they are Symmetric Bichromatic Closest Neighbors (SBCN), i.e., if there is no other point in $B$ that is closer to $a_i$ than $b_j$ and vice versa. For example, in Fig. 4, $a_3$ and $b_3$ are SBCN and thus the edge $(a_3, b_3)$ is part of the RNG**.

The third step of the RNG computation consists of filtering RNG** to remove edges that are not in the RNG. Although RNG** has typically far fewer edges than the complete graph $G$, a naive filtering approach, which checks for each edge $(a,b)$ in RNG** whether each point $c$ is in $lune(a,b)$, can be very time consuming for large datasets. Therefore, we propose an alternative strategy based on information that is computed anyway for HDBSCAN*, which can make the overall filtering process more efficient. It is based on the intuition that points closer to $a$ or $b$ are more likely in $lune(a,b)$ than points that are farther away. For computing multiple HDBSCAN* hierarchies, we initially compute and store core distances (i.e., KNN-distances) $c_i$ for each value of $i \in k_1, \ldots, k_{max}$ by performing a $k_{max}$-nearest neighbor query for each point, which gives us access to the $k_{max}$ closest points to each point. To support our pruning strategy, we propose to also store the actual $k_{max}$-nearest neighbors, so that we can first check for each edge $(a,b)$ with weight $w$ if any of the $k_{max}$-nearest neighbors of $a$ and $b$ is inside $lune(a,b)$. As soon as we find one that is inside, we can safely remove the edge without further checking. If none of those neighbors is inside $lune(a,b)$, we check if $w$ is equal to the core-distance of $a$ or $b$. If that is the case (say for $a$), we know that no other point can be in $lune(a,b)$ (since $lune(a,b)$ is a subset of the ball

around $a$ with radius $w$ and we have checked all points inside this ball); hence we know without further checking that the edge is in the RNG. We can choose to perform only these $2 \times k_{max}$ checks per edge to obtain a graph, which we call RNG*, that is smaller than RNG** but may still contain edges that are not in the RNG. To obtain the exact RNG, we search the entire dataset whenever an edge cannot be excluded or included based on the described, $2 \times k_{max}$ tests, to determine whether or not there is a point in $lune(a, b)$.

Algorithm 1

**Input: X**: dataset; $n$: $|X|$; $[k_1, \ldots, k_{max}]$: $mpts$ range; $T$: graph to be computed (RNG, RNG*, RNG**);

```
 1: for i ∈ {1, ..., n} do
 2:   M[i] ← [1-NN(i), ..., k_max-NN(i)];
 3:
 4: wspd ← WSPD(X, M);
 5:
 6: for (A, B) ∈ wspd do
 7:   RNG^{k_max} ← RNG^{k_max} ∪ SBCN(A, B);
 8:
 9: if T ≠ RNG** then
10:   remove ← False;
11:   for (a, b) ∈ RNG^{k_max} do
12:     for x ∈ M[a] ∪ M[b] do
13:       if x ∈ lune(a, b) then
14:         remove ← True;
15:         break;
16:     if ¬ remove then
17:       if mrd_{k_max}(a, b) = max{c_{k_max}(a), c_{k_max}(b)} then
18:         remove ← False;
19:         continue;
20:     if ¬ remove and T = RNG then
21:       for x ∈ X do
22:         if x ∈ lune(a, b) then
23:           remove ← True;
24:           break;
25:     if remove then
26:       RNG^{k_max} ← RNG^{k_max} \ (a, b);
27:       remove ← False;
28:
29: for mpts ∈ {k_1, ..., k_max} do
30:   MST_mpts ← MST(RNG^{k_max}_mpts);
31:   compute−hierarchy(MST_mpts);
```

The pseudo-code for the overall strategy is shown in Algorithm 1. It takes as input a dataset $\mathbf{X}$ with $n$ points, a range of $mpts$ values, $[k_1, \ldots, k_{max}]$, and the type $T$ of the RNG to be computed, namely, the exact RNG, RNG*, or RNG**. The $k_{max}$-nearest neighbors for each point $x \in \mathbf{X}$ are computed and materialized in Line 2. It is important to emphasize that a single $k_{max}$-NN query is performed for each $x \in \mathbf{X}$. Next, the Well-Separated Pairs Decomposition (WSPD) is performed in Line 4. In Lines 6-7, the RNG** is constructed by adding one edge for each of the Symmetric Bichromatic Closest Neighbors between the pairs $(A, B) \in wspd$. The edge filtering occurs between Lines 11-27. In case the RNG** is chosen, the filtering process is completely skipped (Line 9). Otherwise, the filter steps based on the $k_{max}$-nearest neighbors are performed. The last filter, based on the sequential scan of the dataset (Lines 21-24), is only performed when the exact RNG is to be computed, and only for edges that cannot be excluded or included based on the previous tests. Finally, in Lines 29-31, the MSTs and hierarchies are computed for all the values of $mpts \leq k_{max}$, using the computed RNG.

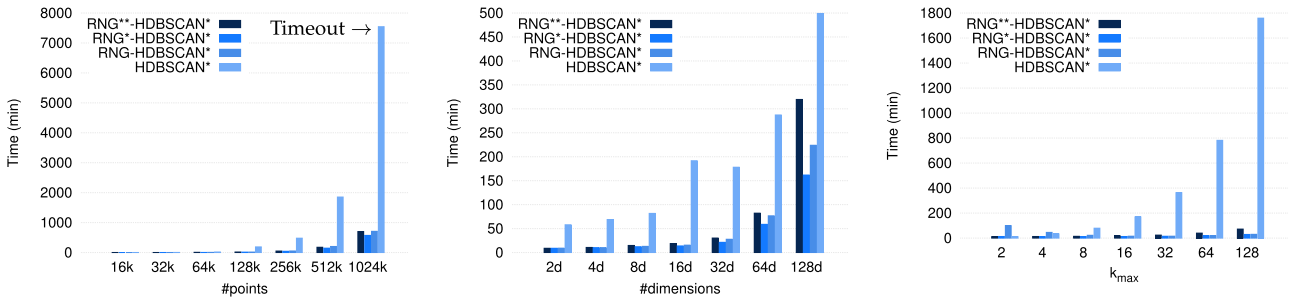
### 4.6 Computational Complexity

Our method can be decomposed into five main parts: (i) core-distance computations, (ii) Well-Separated Pair Decomposition (WSPD), (iii) RNG** construction, (iv) edge filtering, and (v) hierarchy constructions. In part (i), where the core-distances are computed, a $k_{max}$-NN query is executed for each point in the dataset, resulting in an $\mathcal{O}(n^2)$ time complexity. In part (ii), the WSPD is computed according to the method proposed in [5], which runs in $\mathcal{O}(n)$ time. In part (iii), the RNG** is constructed via the computation of the Symmetric Bichromatic Closest Neighbors for each of the pairs in the WSPD. Note that for every point $p$ in the dataset, the number of comparisons that involve $p$ is of order $n$, as the remaining $n-1$ points are placed in sets that are well separated from a set containing $p$. Therefore, one needs $\mathcal{O}(n^2)$ comparisons in order to find the SBCNs for all pairs of well-separated sets and, thus, building the RNG** takes $\mathcal{O}(n^2)$ time. In part (iv), the edges of the RNG** are filtered to produce either the RNG* or the RNG. This process relies on the information available from the core-distances to filter out the edges that do not belong in the final graph. Therefore, the computational complexity of this part depends directly on $k_{max}$ and on how the points are distributed in the space. In the best case scenario, checking whether an edge belongs to the RNG or not can be done in constant time, and the entire filtering process is done in $\mathcal{O}(|E'|)$ time, where $E'$ represents the set of edges in the RNG**. On the other hand, in the worst case scenario a linear scan of the points in the dataset has to be performed for each edge and the filtering is done in $\mathcal{O}(|E'| \cdot n)$ time. The number of edges in the RNG** can vary from $\mathcal{O}(n)$ to $\mathcal{O}(n^2)$, depending on the distribution and dimensionality of the points. Note that to produce the RNG* we only filter the edges that can be discarded in constant time. In part $(v)$, the Minimum Spanning Trees are computed in $\mathcal{O}(|E| + n \log n)$ time, where $E$ represents the set of edges in the graph after filtering (step $(iv)$). As the $RNG$ and its variants have in general much fewer edges than the complete graph, the computation of the hierarchies is much faster than applying the same algorithm in the complete graph.

## 5 Experiments

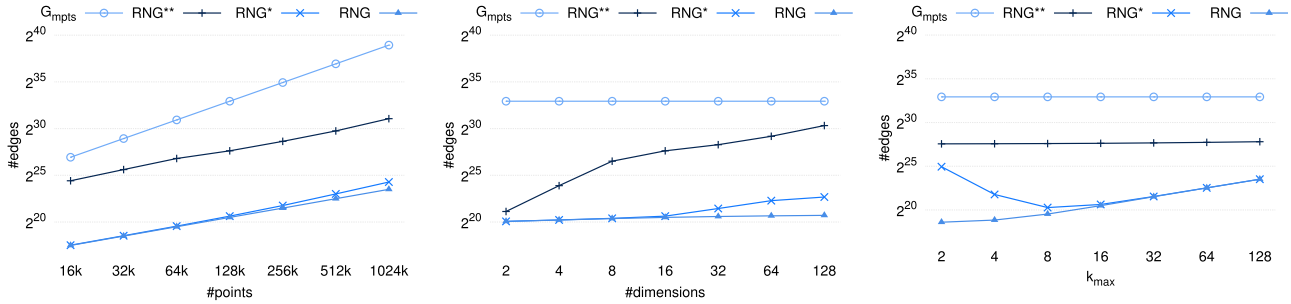
We conducted experiments to evaluate the efficiency of the proposed method with respect to changes in size and dimensionality of the dataset, and, most importantly, with respect to the number of hierarchies to be computed. We also show the sizes of the RNG, RNG*, and RNG** in comparison to the size of the $G_{mpts}$, since the reduction in the number of edges is the source of our performance gain.

To the best of our knowledge, no other strategy in the literature aims at computing multiple hierarchies efficiently. Thus, we compare our strategy to a straightforward baseline that runs HDBSCAN* multiple times, one for each $mpts$ value in the given range, but with the optimization of




(a) #points vs. Runtime (b) #dimensions vs. Runtime (c) $k_{max}$ vs. Runtime

Fig. 5. Runtime as a function of the dataset size, dataset dimensionality, and $k_{max}$. (Note that the $x$-axis is in log scale).




(a) #points vs. RNG size (b) #dimensions vs. RNG size (c) $k_{max}$ vs. RNG size

Fig. 6. RNG size as a function of the dataset size, dataset dimensionality, and $k_{max}$. (Note that both axes are in log scale).

pre-computing the core distances for all points, as we do in our approach, using a single $k_{max}$-NN query per point.

To study the computational trade-offs of the different edge filtering strategies described in Section 4.5, we show results for three variants: RNG**-HDBSCAN*, which just uses the RNG** without any additional filtering; RNG*-HDBSCAN*, which applies only the filtering based on $k_{max}$ nearest neighbors; and RNG-HDBSCAN*, which applies the complete filtering to obtain the exact RNG.

All methods have been implemented on top of the original HDBSCAN* code, provided by the authors of [9], in Java. The core-distances are computed with the aid of a $K$d-Tree index structure, adapted from [34]. The experiments were performed in a virtual machine with 64 GB RAM, running Ubuntu. For runtime experiments, we measure the total running time to compute core-distances and MSTs, and report the average runtime over 5 experiments.

The datasets were obtained using the generator proposed in [19], varying the number of dimensions from 2 to 128, and the number of points from 16k to 1M; the ranges of $mpts$ started with 1, varying the value of $k_{max}$ from 2 to 128. Table 1 shows these values and indicates in bold the default value for each variable when other variables are varied.

The datasets used to properly assess the efficiency of our method with regard to the effects of a specific variable were generated by varying only that variable, while the others were kept at their default values. For instance, in order to evaluate how our strategy behaves with regard to different dataset sizes, we take samples of different sizes from the largest dataset. Similarly, in order to evaluate the effects of dimensionality on the performance of our strategies, we vary the number of dimensions and keep the same number of clusters and points in the dataset. In the evaluation of the effects of $k_{max}$, the number of points and dimensions are kept fixed at their default values. Note that $k_{max}$ does not have any influence on the dataset generation.

TABLE 1
Experimental Setup

| Variables | Values |
|---|---|
| #points | 16k, 32k, 64k, **128k**, 256k, 512k, 1M |
| #dimensions | 2, 4, 8, **16**, 32, 64, 128 |
| $k_{max}$ | 2, 4, 8, **16**, 32, 64, 128 |

## 5.1 Effect of Dataset Size

Fig. 5a shows the total runtime as a function of the dataset size with default values for the remaining variables. As expected, the runtime tends to increase for all methods as the number of points increases. For datasets up to 64k points, all strategies have similar performances, but as the datasets become larger, the difference between our approaches and the baseline increases significantly. For 128k points, the baseline strategy already takes about twice as much time as our approaches, and for 1024k points, we actually interrupted each run of the baseline before it finished.[2]

Fig. 6a shows the number of edges in $G_{mpts}$, RNG**, RNG*, and RNG, as a function of the dataset size. As expected, the number of edges increases with the number of points. However, the RNGs are significantly smaller than the complete graph for all dataset sizes. In fact, even for the largest dataset, the sizes of the RNG* and RNG are smaller than the size of the $G_{mpts}$ for the smallest dataset.

2. The runs on this experiment were interrupted after 7,500 minutes ($\approx$ 5 days), as the observed performance was already enough for comparison purposes

TABLE 2
$k_{max}$ vs. Runtime (min.)

| $k_{max}$ | HDBSCAN* | RNG**-HDBSCAN* | RNG*-HDBSCAN* | RNG-HDBSCAN* |
|---|---|---|---|---|
| 2 | **12** | **12** | **12** | 99 |
| 4 | 33 | **12** | **12** | 45 |
| 8 | 79 | 14 | **12** | 22 |
| 16 | 169 | 17 | **13** | 15 |
| 32 | 363 | 23 | **14** | 15 |
| 64 | 781 | 40 | **18** | 19 |
| 128 | 1,759 | 72 | **29** | 30 |

The sizes of RNG and RNG** are quite different, yet their running times are quite similar (see Fig. 5a), indicating that the gain in MST computation due to a smaller graph size is canceled out by the time spent filtering to obtain the exact RNG. On the other hand, RNG*, which only uses the very fast filter based on materialized $k$-nearest neighbors, is very close in size to RNG, showing the effectiveness of our pruning heuristic, and leading to a much faster runtime.

### 5.2 Effect of Dimensionality

Fig. 5b shows the effect of dataset dimensionality on runtime. As expected, all approaches are affected by increasing dimensionality, due to a number of effects that are generally referred to as "curse of dimensionality." However, since our datasets do contain cluster structures, these effects are not critically severe even in 128 dimensions.

We can observe that all RNG-based strategies perform better than the baseline in all datasets, but as dimensionality increases, the difference between the unfiltered RNG (RNG**) and the filtered versions (RNG* and RNG) increases. This can be explained by looking at the number of graph edges shown in Fig. 6b. The size of the exact RNG is barely affected by an increase in dimensionality, while the size of the unfiltered RNG** grows significantly, approaching the complete graph $G_{mpts}$. This shows that the generation of well-separated pairs becomes less effective in implicitly excluding edges that cannot be in an RNG as the dimensionality increases. On the other hand, the exact RNG still has significantly fewer edges than a complete graph in these scenarios—although, theoretically, it also must eventually approach the complete graph [5], [24].

The number of edges of RNG* increases only slightly as the dimensionality increases, which shows that the pruning strategy using only the pre-computed $k$-nearest neighbors (16, as $k_{max} = 16$ in this experiment) stays quite effective, even in the 128-dimensional datasets, resulting in the best runtime performance overall.

### 5.3 Effect of Upper Limit $k_{max}$

Fig. 5c and Table 2 show the runtimes *w.r.t.* $k_{max}$. The runtime of all our methods is very low compared to the baseline, for which runtime increases linearly, as expected.

The runtime of HDBSCAN*-RNG** increases very slightly with $k_{max}$ as also the number of edges increases slightly, but it stays significantly below the number of edges in $G_{mpts}$, as shown in Fig. 6c.

RNG-HDBSCAN* shows a slightly higher runtime for $mpts = 2$, which then decreases for $mpts = 4$ and $mpts = 8$, after which it stays almost constant and becomes almost indistinguishable in performance to RNG*-HDBSCAN*.

RNG*-HDBSCAN*, which only uses the $k_{max}$-nearest neighbors for pruning RNG**, shows the most stable runtime behavior; its increase in runtime, as $k_{max}$ increases, is almost unnoticeable. For the largest value of $k_{max}$, the difference in runtime to the baseline corresponds to a speed-up of about two orders of magnitude. The runtime behavior of RNG and RNG* can be explained by their number of edges, shown in Fig. 6c. For $mpts = 2$, the number of edges in RNG* is much larger than in RNG (while still being smaller than in RNG**). The reason is that the filtering strategy is not yet very effective when only two nearest neighbors are considered. Thus, for many edges $(a, b)$ a sequential scan has to be performed to check $lune(a, b)$ in order to obtain RNG, outweighing the gain in performance runtime for computing the MST of RNG with fewer edges.

The results also show that (1) computing MSTs is very fast, compared to the rest of the computation, if the underlying graphs are already relatively small compared to the complete graph, and (2) that our pruning heuristic based on $k_{max}$-NNs becomes more effective as $k_{max}$ increases, leading to an almost indistinguishable performance between RNG and RNG* for $k_{max} \geq 16$.

The significance of our contribution and of the obtained speed-ups becomes even more clear, if we look at the runtime from a different perspective. Fig. 7 shows the ratio of the runtime to compute $k_{max}$ MSTs over the runtime to compute a single MST. RNG* exhibits a very stable ratio of about 2 for all values of $k_{max}$, *i.e*, we can use it to compute as many as 128 MSTs/hierarchies for the computational cost of naively computing about 2 MSTs/hierarchies.

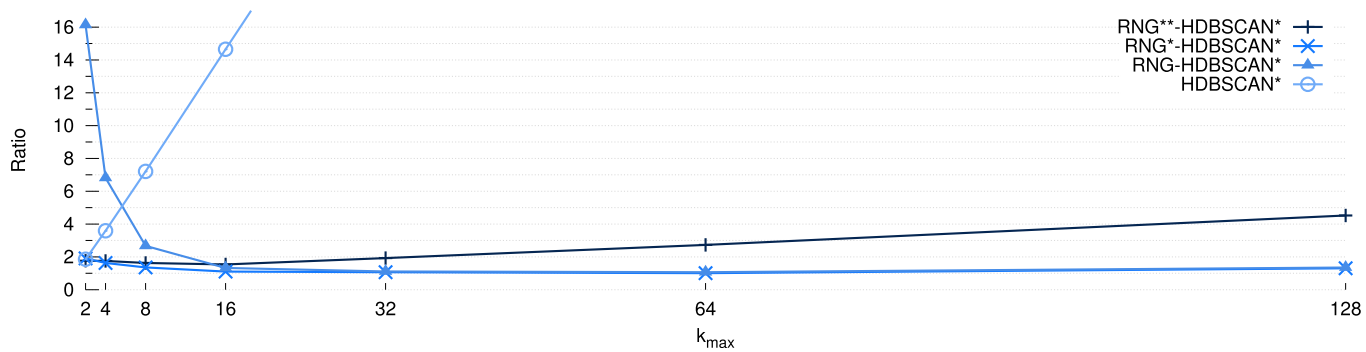



Fig. 7. Ratio: runtime to compute $k_{max}$ MSTs/hierarchies divided by the runtime to compute a single MST/hierarchy.

## 6 Application: Visualization

The ability to efficiently compute a set of HDBSCAN* hierarchies for a range of $mpts$ values opens the opportunity to interpret multiple hierarchies together in order to gain a better understanding of the data and its cluster structures. As argued earlier, different ranges of $mpts$ values produce different hierarchies and clusterings, and guessing the "right" range of $mpts$ for a specific dataset is not trivial. In what follows we propose a set of visualization methods that aim at helping a user to identify interesting ranges of $mpts$ values and what they mean in terms of clustering.[3] Note that this would not be practical without an efficient approach like RNG-HDBSCAN* which is capable of generating all the necessary metadata at an acceptable computational cost.

### 6.1 Hierarchy Similarity Plot

To identify different ranges of $mpts$ values that produce different relevant hierarchies, we first need to measure how similar (or dissimilar) two hierarchies are. In [1], the authors propose the Hierarchy Agreement Index (HAI) which captures how much two hierarchies agree with each other w.r.t. the distances between pairs of points in both hierarchies. The distance between a pair of points $x_i$ and $x_j$ in a hierarchy $H$, $d_H(x_i, x_j)$, is defined as the size of the smallest cluster where $x_i$ and $x_j$ appear together divided by the size of the whole dataset. The HAI similarity between two hierarchies $H_1$ and $H_2$ is then defined by the average, normalized difference of the distances $d_{H_1}$ and $d_{H_2}$, between all pairs of points, in the following way:

$$\text{HAI}(H_1, H_2) = 1 - \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} |d_{H_1}(x_i, x_j) - d_{H_2}(x_i, x_j)|.$$

After computing the HAI values for every pair of hierarchies, one is able to represent the similarities in a symmetric matrix where a row index $i$ and a column index $j$ represent $mpts_i$ and $mpts_j$, respectively, from the given range of $mpts$ values. A cell $(i, j)$ contains the HAI value for the pair of hierarchies with respect to $mpts_i$ and $mpts_j$, respectively. Plotting HAI values in a color scale allows one to visually identify $mpts$ values that result in similar hierarchies.

Fig. 8 shows the HAI similarity matrix for 50 hierarchies, one for each value of $mpts \in [1, 50]$, computed from a sample of the ALOI (Amsterdam Library of Object Images) dataset [18], which contains images of physical objects taken from different angles and under different light conditions. Intuitively, each cluster/category in this dataset correspond to a specific physical object. The sample contains 125 images in total from 5 different categories, and each object is represented with 144 dimensions [22]. In the plot, one is able to clearly identify two large ranges of $mpts$ values in which hierarchies have a high degree of similarity with each other, but are different from hierarchies in the other range. This means that across a range of $mpts$ values, there are mainly two different clustering structures.

The computation of the similarity matrix involves the comparison of each pair of hierarchies according to the HAI measure. Each comparison takes $\mathcal{O}(n^2)$-time with the use of auxiliary techniques and data structures, such as Euler Path, Range Minimum Queries, and Sparse Tables, to compute the Lowest Common Ancestor (LCA) of every pair of points in each hierarchy—given that LCA queries can be computed in $\mathcal{O}(1)$-time [4]. It is important to note that the HAI measure is only one possible measure; alternative similarity measures that would be computationally less expensive could be explored. For instance, one could investigate how to extract a similarity measure from the reachability plots that represent the hierarchies or investigate strategies that would approximate a similarity measure based on the observation that adjacent values of $mpts$ are likely to produce similar hierarchies. We have chosen to use an existing measure in the literature as we consider that the evaluation and comparison of hierarchies is a relevant research topic on its own, which would deserve attention as a separate work with a proper discussion around theoretical and practical aspects.

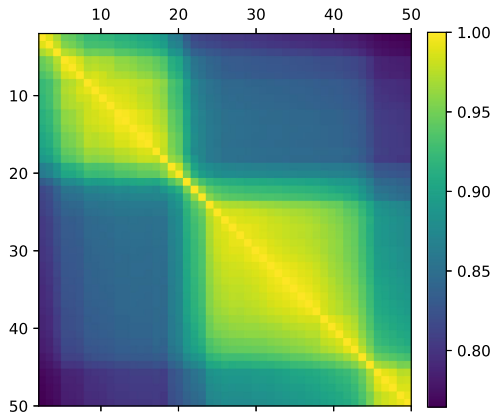



Fig. 8. ALOI - HAI similarity matrix plot; lighter colors indicate a higher similarity.

### 6.2 Meta-Hierarchy Plot

The HAI similarity matrix plot presents an overview of the similarities between hierarchies *w.r.t.* a range of $mpts$ values. Identifying exact sets of similar hierarchies from this plot can, however, be challenging. For example, in cases where changing the value of $mpts$ leads to a smooth decrease or increase of similarity, it is hard to draw boundaries that separate two ranges of values just by looking at the HAI similarity plot. Also, there might be cases where two hierarchies for non-consecutive values of $mpts$ have a higher similarity than for consecutive values.

To find classes of similar clustering hierarchies, we can perform a meta-clustering using the HAI values to construct a *clustering hierarchy of clustering hierarchies*.[4] This meta-hierarchy can then be visualized as a dendrogram, where the user can see groups of similar hierarchies at different similarity levels more clearly.

Previous works in the literature have used meta-clustering techniques to analyze multiple clustering results [10], [11]. Unlike ours, however, these works produce a single flat clustering solution (a partition), rather than a clustering hierarchy, which carries more information.

3. A web-based demonstration of these visualizations can be found at http://webdocs.cs.ualberta.ca/~joerg/mustache.

4. Note that (1) we use HDBSCAN* with $mpts = 1$ to cluster the clustering hierarchies, which is equivalent to using Single Linkage; and (2) as the HAI values express similarity, one has to convert them into dissimilarity before using them with HDBSCAN*.

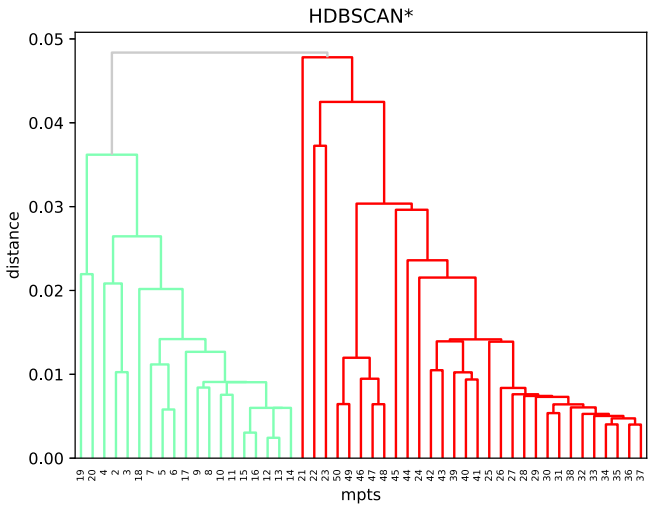



Fig. 9. ALOI - Meta-hierarchy plot.

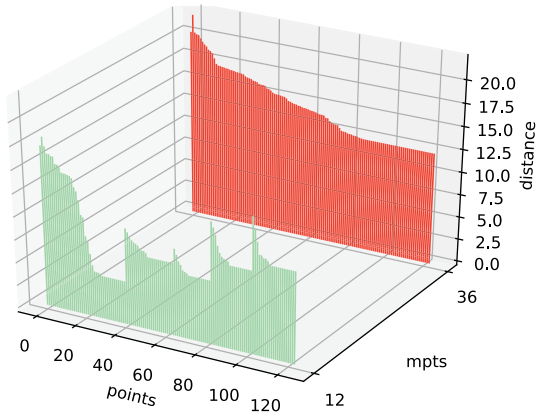



Fig. 10. ALOI - Reachability plots of the medoids of the two meta-clusters in Fig. 9. Five valleys in the reachability plot of the green meta-cluster ($mpts = 12$) correspond to clusters.

Fig. 9 shows the dendrogram resulting from a meta-clustering of the HAI similarity matrix in Fig. 8. Note that there is a clear separation of two main meta-clusters.

A meta-hierarchy plot makes it easier to identify groups of hierarchies whose elements can be considered as representing essentially the same clustering structure of the data, as well identify groups of hierarchies that represent *"significantly"* distinct clustering structures of the data. Furthermore, deciding what is significant becomes more intuitive and more practical with the use of dendrograms.

### 6.3 Reachability Plots

In the next step, we examine the hierarchies in each meta-cluster of hierarchies. Since the hierarchies in each meta-cluster are, according to the meta-clustering, similar to each other, we can select and visualize only the *medoid* hierarchy from each meta-cluster as its representative. The medoid of a meta-cluster is the hierarchy for which the average HAI similarity to other hierarchies in the same meta-cluster is the highest.

We could use again a dendrogram to visualize individual clustering hierarchies of the data, but dendrograms are only a good visualization when the number of leaf nodes (i.e., elements being clustered) is "relatively small", as is typically the case for our meta-hierarchy. The number of leaf nodes in a meta-hierarchy corresponds to the number of obtained clustering hierarchies *w.r.t.* the different values of $mpts$ used, which in practice is rarely larger than a few tens. However, when inspecting individual clustering hierarchies of a data set, such as the meta-cluster medoids or outliers in the meta-hierarchy, the number of leaf nodes corresponds to the number of data points in the dataset, which may be too large to be properly visualized as a dendrogram. Therefore, we choose in these cases *reachability plots* [3] to visualize density-based clustering hierarchies. Reachability plots are bar plots where each bar corresponds to an object in the dataset, and they are sorted in such a way that objects that belong to the same cluster at every density level are next to each other. The height of each bar is defined by the lowest density level that makes its corresponding object join the preceding objects in the plot, so that density-based clusters appear as "valleys" or "dents" in the plot.

Fig. 10 shows the reachability plots that correspond to the medoids of the two meta-clusters highlighted in Fig. 9. The medoids are the hierarchies *w.r.t.* $mpts = 12$ and $mpts = 36$, respectively, and they are colored according to the color of the meta-cluster they represent. For $mpts = 12$, one can identify 5 clusters in the data, while the plot for $mpts = 36$ does not reveal any prominent cluster structure. This type of visualization of multiple, representative reachability plots can be very powerful in practice as it allows model selection to be performed (in this case, the choice of $mpts$ around 12) in a completely unsupervised way.

This example shows how the combined visualization of different reachability plots for meta-cluster medoids can be used to easily compare the main different hierarchical organizations of the data across multiple $mpts$ values. One can also investigate a specific reachability plot in more detail, possibly coloring the plot according to clusters selected from the corresponding hierarchy. In particular, HDBSCAN* is equipped with an optional post-processing routine, called FOSC [8], to automatically extract clusters from optimal local cuts through the hierarchy using the notions of cluster lifetime and stability. Once a flat clustering solution is extracted using this method, the reachability plot can be colored according to the extracted clusters.

This type of visualization is illustrated in Fig. 11, which shows the reachability plot for $mpts = 12$ with the extracted clusters shown in different colors (black represents data objects left unclustered as noise). Fig. 11 also shows the physical objects corresponding to the images in each of the clusters. (Recall that this dataset contains 125 different images of 5 unique objects [22].)

The previous discussion illustrates how the proposed visual tools can be used to learn more from a collection of clustering hierarchies about a data set. Next, we present a few small, real-world case studies to demonstrate the effectiveness of our visualization tools for exploratory cluster analysis with multiple clustering hierarchies.

### 6.4 Case Study #1: Text Data

In this case study, we analyze hierarchies constructed for the dataset Articles-1442-5 [28] containing 253 documents extracted from journals of different fields (DNA Research,

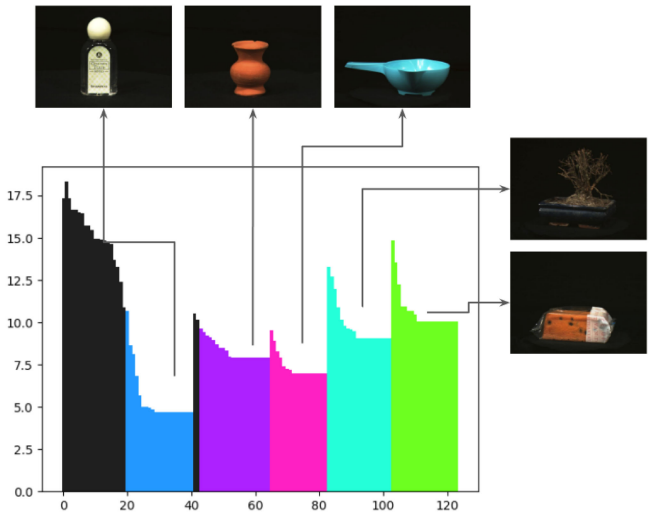
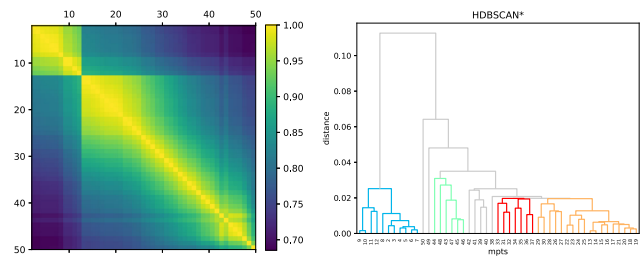


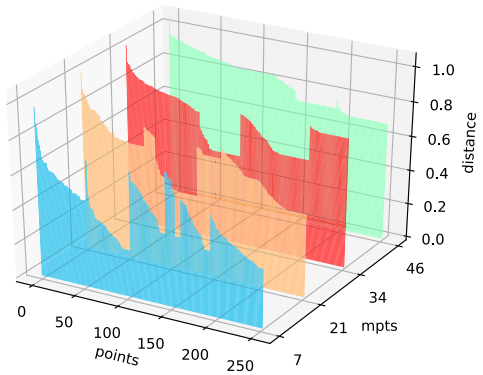
Fig. 11. Reachability plot ($mpts = 12$) colored according to clusters selected by HDBSCAN*'s automatic cluster extraction method (FOSC).

British Food Journal, Transactions on Mobile Computing, American Political Science Review and Monthly Weather Review). Each document is a research paper described by 4636 dimensions following a "bag-of-words" representation, and documents from the same journal/field are expected to correspond to a cluster in the feature space. For this dataset, we used the angular distance in order to generate the set of HDBSCAN* hierarchies for a range of $mpts$ values. The angular distance is equivalent to the normalized angle between two vectors in the feature space and is also a metric.

Fig. 12a shows that there are two main ranges of $mpts$ values for which the hierarchies are more similar among themselves. Overall, the hierarchies in the larger range ($mpts \in [13, 50]$) have a weaker degree of similarity than the ones in

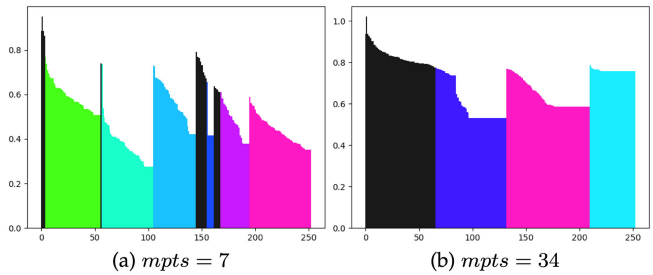


(a) HAI Matrix (b) Meta-Clustering



(c) Reachability Plots - Medoids

Fig. 12. Journal documents ("Articles-1442-5") results.



(a) $mpts = 7$ (b) $mpts = 34$

Fig. 13. Selected plots for the "Articles-1442-5" dataset.

the smaller range ($mpts \in [2, 12]$), although there are sub-ranges of larger values within which similarity is more prominent, e.g., around [13, 25] and around [42, 49]. Notice that in a dataset with these many dimensions, selecting a meaningful value of $mpts$ would be particularly challenging.

Fig. 12b shows the meta-hierarchy for this dataset, with meta-clusters (as detected by FOSC) highlighted in different colors. The reachability plot of each meta-cluster's medoid is shown in Fig. 12c. Fig. 13 shows the reachability plots for $mpts = 7$ and $mpts = 34$ colored according to the partitioning automatically extracted from the corresponding hierarchies. Observe that $mpts = 7$ is able to detect most clusters successfully, just splitting cluster 3 into two sub-clusters. We can also observe that $mpts = 34$ results in a larger amount of noise objects (colored in black), because the density of objects as estimated by the inverse of their core distance decreases as $mpts$ increases, turning more objects into noise for any density cutting threshold $\varepsilon$. A relevant aspect that can be learned from this is that clusters $c_1$, $c_3$ and $c_4$ can still be detected for $mpts = 34$, whereas clusters $c_2$ and $c_5$ become noise at this level. This shows that clusters $c_1$, $c_3$ and $c_4$ are more stable than the others, as they persist across varied parameter ranges and hierarchies.

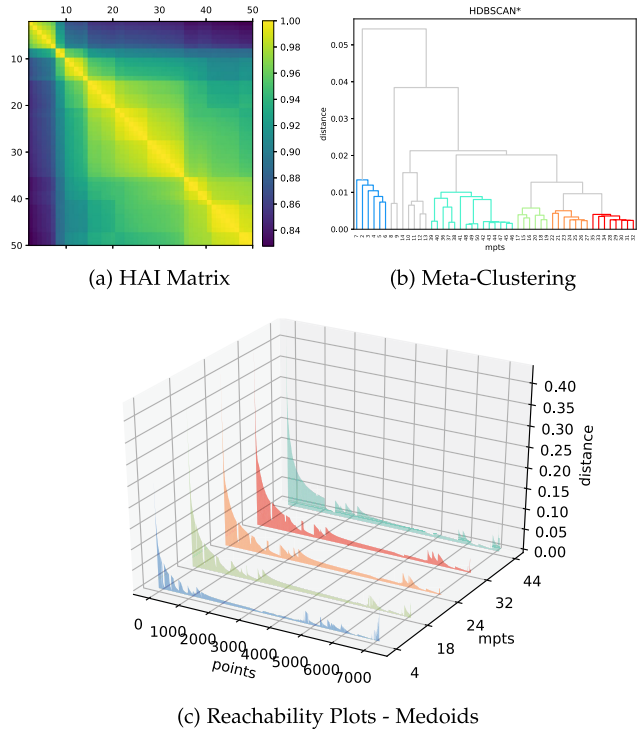
The reachability plots for the other values of $mpts$ may be similarly analyzed in order to get insights about the data. Also, one should keep in mind that the reachability plots showed in Fig. 12c are only the representatives of their respective meta-clusters, which can serve as a first guide to what ranges of values the user should focus on primarily.

### 6.5 Study Case #2: Sound Data

In this second case study, we analyze a dataset of sound features extracted from frog (Anuran order) calls [15]. The dataset contains 7,000 entries with 22 dimensions each and has been used in tasks related to species recognition. We want to investigate if there are clusters that match the taxonomic classification of the frogs. After running HDBSCAN* in this dataset, one may expect to get a clustering hierarchy that represents, as close as possible, the taxonomy that describes which specimen belongs to the each species, genus and families. This depends on the $mpts$ value though.

Choosing $mpts$ for this dataset is far from trivial. Fig. 14a shows that there are several intervals of $mpts$ values for which hierarchies are very similar to each other. After performing the meta-clustering on the HAI values, five more prominent meta-clusters are identified (Fig. 14b).

Fig. 15 shows the reachability plots for each medoid meta-cluster colored according to the partitioning found by FOSC. For $mpts = 4$, FOSC extracted 78 clusters. When compared to the true labels, we observed that clusters

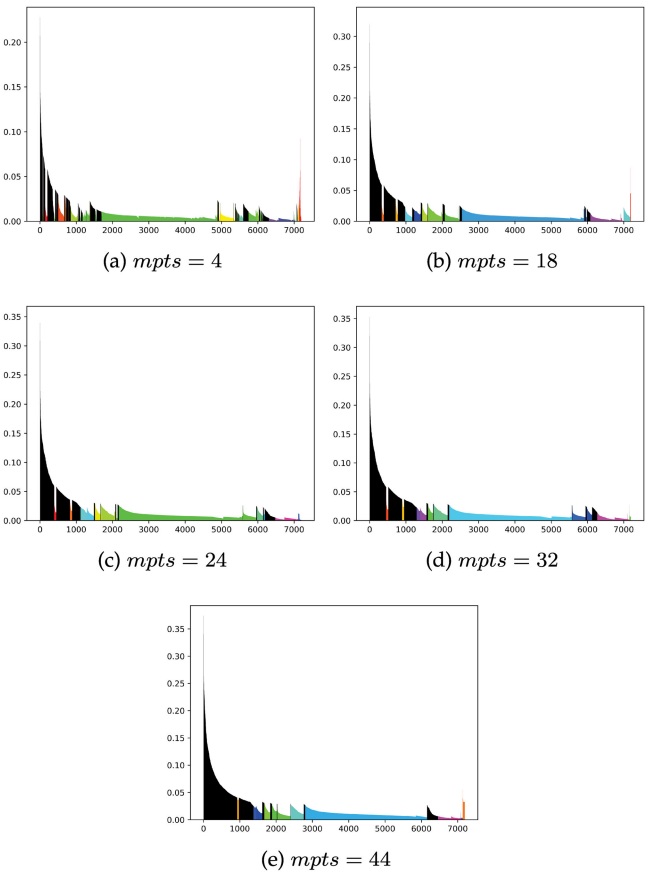



(a) HAI Matrix (b) Meta-Clustering

(c) Reachability Plots - Medoids

Fig. 14. Anuran sounds results.

corresponding to the same family/genus/species were split into several micro sub-clusters. On the other hand, for the higher values of $mpts$, we observed that some entries that belong to the same family, but not same genus or specie,

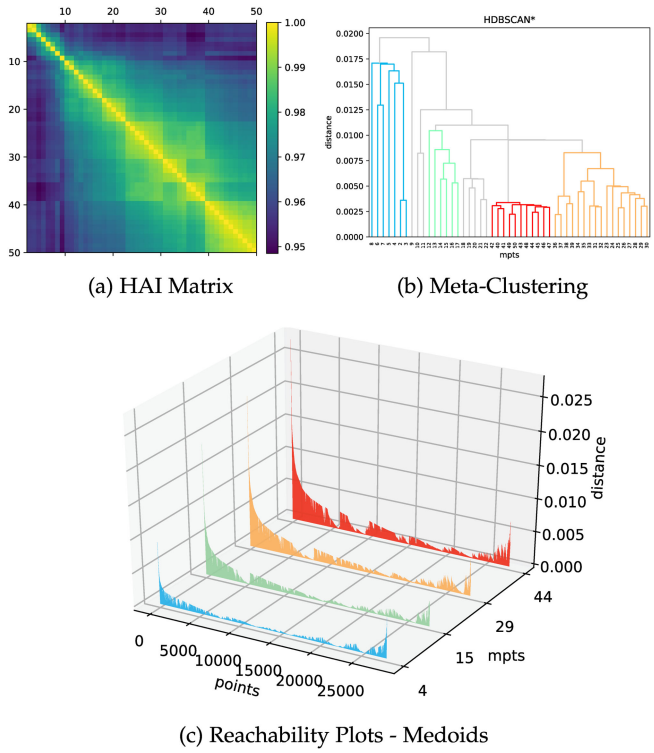


(a) $mpts = 4$ (b) $mpts = 18$

(c) $mpts = 24$ (d) $mpts = 32$

(e) $mpts = 44$

Fig. 15. Anuran - Reachability plots.




(a) HAI Matrix (b) Meta-Clustering

(c) Reachability Plots - Medoids

Fig. 16. Flickr Paris results.

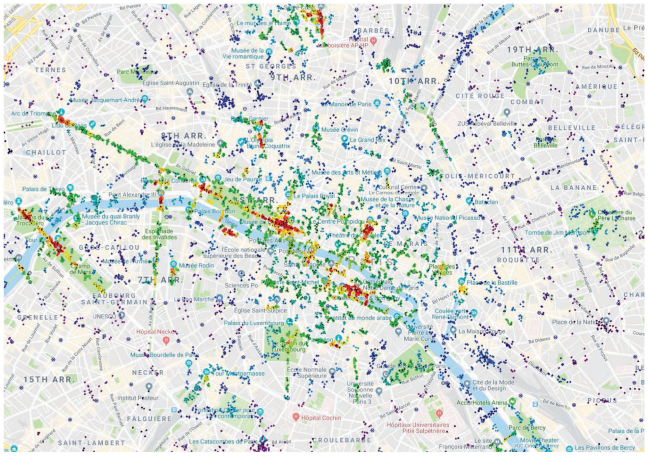
were put into the same cluster. In rare cases, a small number of objects (e.g., 2 or 3) were put into the wrong cluster, but the overall results are aligned with the taxonomic classification at some level. Note that, even in cases where the ground-truth is unknown and this type of comparison is not an option, narrowing the $mpts$ search space to 5 values is already a good starting point to a more detailed analysis, i.e., clustering as a primary exploratory data analysis tool.

The plots in Fig. 15 also reveal that the number of objects labeled as noise increases as the value of $mpts$ increases, for the reasons already explained in the previous case study. Also, there is a large cluster that can be detected in all of the reachability plots. This cluster is not only the largest one, but is also very dense and stable, as it can be observed for these very different density estimate settings. This observation is only possible when multiple and "distant enough" values of $mpts$ are inspected. The medoids represent exactly these distinct values across a range of $mpts$ values for which one had no knowledge before.

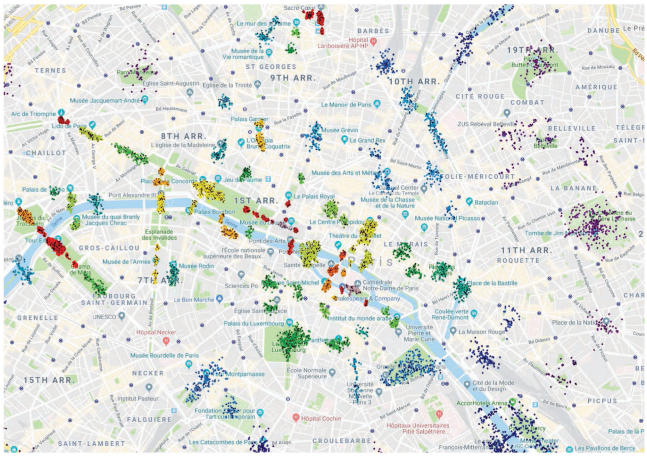
### 6.6 Case Study #3: Geolocated Data

In this dataset, we analyze geolocations from photos posted on Flickr [31]. We selected a sample of the dataset containing 28,000 geo-tagged photos in Paris, France. In a city like Paris, one expects most photos to be concentrated on tourist hotspots such as the Eiffel Tower and the Louvre Museum. However, the amount of outliers in some regions of the city makes it difficult to identify the boundaries of clusters that correspond to a specific touristic place from another. For instance, photos related to the Arc de Triomphe and the Champs-Élysées Avenue are hard to separate, because the former is located on the latter.

Fig. 16a indicates that there are a few ranges and sub-ranges of $mpts$ values for which the hierarchies have a



(a) $mpts = 4$



(b) $mpts = 29$

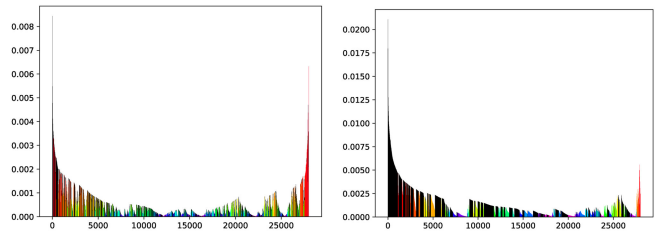
Fig. 17. Paris Flickr: Geolocations of extracted clusters.

higher degree of similarity. When clustered, the hierarchies for this dataset form 4 main meta-clusters, as shown in Fig. 16b. Fig. 16c shows the reachability plots for the meta-cluster medoids ($mpts = 4$, 15, 29, and 44).

Fig. 17 shows on the map the actual geolocations corresponding to the clusters extracted from the hierarchies for $mpts = 4$ and $mpts = 29$. Note that for $mpts = 4$, it is hard to identify the boundaries of each cluster. When it comes to the reachability plot, one can also identify the clusters more easily with $mpts = 29$ than with $mpts = 4$, as shown in Fig. 18. Even though it is possible to see some prominent valleys in the reachability plot for $mpts = 4$, these valleys are composed of several micro-valleys that end up being selected by FOSC. On the other hand, for $mpts = 29$ the clusters are clearly separated on the map, and at the same time it is possible to identify the major valleys in the reachability plot that cover larger areas in the map, merging together smaller nearby clusters, typically around the same hotspot. In this case, the use of larger values of $mpts$ not only helps clearing outliers that end up unclustered as noise, but it also reduces the large amount of micro-clusters that might be found.

## 7 Conclusion

In this paper we presented RNG-HDBSCAN*, an efficient strategy for computing multiple density-based clustering




(a) $mpts = 4$ (b) $mpts = 29$

Fig. 18. Selected reachability plots - Paris Flickr dataset.

hierarchies. The key for its efficiency is the replacement of the Mutual Reachability Graph by a suitable Relative Neighborhood Graph which allows to incrementally explore HDBSCAN* solutions *w.r.t.* a range of values of $mpts$. Our experiments showed that RNG-HDBSCAN* can be more than 60 times faster than running the original HDBSCAN* algorithm for the same ranges of $mpts$. In particular, it scales significantly better when running on large datasets and more prominently for broader ranges of $mpts$ values. Moreover, we proposed a series of visualizations that allow the analysis of multiple hierarchies *w.r.t.* several $mpts$ values, and we used real datasets from different domains to show how one can use these visualizations to explore and better understand the different hierarchical organizations of the data under different parameter settings.

We also found that in some cases there is no single value of $mpts$ that is able to detect all the cluster structures in the data in a single hierarchy; that is, different relevant clusters are revealed for different values of $mpts$ as part of different hierarchies. As future work, we intend to investigate how one can analyze the multiple HDBSCAN* hierarchies as a single structure that considers different density scenarios. This will help one understand how the cluster structures change with $mpts$ as we progress towards a parameter-free hierarchical density-based clustering framework.

During this research we realized that the literature still lacks in works that explore similarity measures between clustering hierarchies. As future work, we would also like to investigate alternative methods to estimate how similar two hierarchies are, as well as possible validity indexes able to evaluate the quality of a clustering hierarchy in an unsupervised way.

## Acknowledgments

Research partially supported by NSERC, Canada, and by CNPq, under the program Science without Borders, Brazil.

## References

[1] D. M. Johnson, C. Xiong, J. Gao, and J. J. Corso, "Comprehensive cross-hierarchy cluster agreement evaluation," in *Late-Breaking Developments in the Field of Artificial Intelligence*, vol. WS-13–17, 2013.

[2] P. K. Agarwal and J. Matoušek, "Relative neighborhood graphs in three dimensions," *Comput. Geom.*, vol. 2, pp. 1–14, 1992.

[3] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1999, pp. 49–60.

[4] M. A. Bender and M. Farach-Colton, "The LCA problem revisited," in *Proc. 4th Latin Amer. Symp.Theor. Informat.*, 2000, pp. 88–94.

[5] P. B. Callahan, "Dealing with higher dimensions: The well-separated pair decomposition and its applications," PhD thesis, Dept. Comput. Sci., Johns Hopkins Univ., Baltimore, MD, 1995.
[6] P. B. Callahan and S. R. Kosaraju, "A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields," *J. ACM*, vol. 42, no. 1, pp. 67–90, 1995.
[7] R. J. G. B. Campello, D. Moulavi, and J. Sander, "Density-based clustering based on hierarchical density estimates," in *Proc. Pacific-Asia Conf. Knowl. Discovery Data Mining*, 2013, pp. 160–172.
[8] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "A framework for semi-supervised and unsupervised optimal extraction of clusters from hierarchies," *Data Mining Knowl. Discovery*, vol. 27, no. 3, pp. 344–371, 2013.
[9] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discovery Data*, vol. 10, no. 1, pp. 5:1–5:51, 2015.
[10] C. Carpineto and G. Romano, "Optimal meta search results clustering," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2010, pp. 170–177.
[11] R. Caruana, M. Elhawary, N. Nguyen, and C. Smith, "Meta clustering," in *Proc. 6th Int. Conf. Data Mining*, 2006, pp. 107–118.
[12] G. Cattaneo, P. Faruolo, U. F. Petrillo, and G. F. Italiano, "Maintaining dynamic minimum spanning trees: An experimental study," *Discrete Appl. Math.*, vol. 158, no. 5, pp. 404–425, 2010.
[13] X. Chen, Y. Min, Y. Zhao, and P. Wang, "GMDBSCAN: Multi-density DBSCAN cluster based on grid," in *Proc. IEEE Int. Conf. e-Bus. Eng.*, 2008, pp. 780–783.
[14] B. Delaunay, "Sur la sphère vide. A la mémoire de Georges Voronoï," *Bulletin de l'Académie des Sciences de l'URSS*, no. 6, pp. 793–800, 1934.
[15] D. Dheeru and G. Casey, "UCI machine learning repository," University of California, Irvine, School of Information and Computer Sciences, 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[16] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 226–231.
[17] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," *Syst. Biol.*, vol. 18, pp. 259–278, 1969.
[18] J. Geusebroek, G. J. Burghouts, and A. W. M. Smeulders, "The amsterdam library of object images," *Int. J. Comput. Vis.*, vol. 61, no. 1, pp. 103–112, 2005.
[19] J. Handl and J. D. Knowles, "Improvements to the scalability of multiobjective clustering," in *Proc. IEEE Congress Evol. Comput.*, 2005, pp. 2372–2379.
[20] M. R. Henzinger and V. King, "Maintaining minimum spanning trees in dynamic graphs," in *Proc. Int. Colloquium Automata Lang. Program.*, 1997, pp. 594–604.
[21] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proc. 4th Int. Conf. Knowl. Discovery Data Mining*, 1998, pp. 58–65.
[22] D. Horta and R. J. G. B. Campello, "Automatic aspect discrimination in data clustering," *Pattern Recognit.*, vol. 45, no. 12, pp. 4370–4388, 2012.
[23] G. A. II, G. Cattaneo, and G. F. Italiano, "Experimental analysis of dynamic minimum spanning tree algorithms (extended abstract)," in *Proc. 8th Annu. ACM-SIAM Symp. Discrete Algorithms*, 1997, pp. 314–323.
[24] J. W. Jaromczyk and G. T. Toussaint, "Relative neighborhood graphs and their relatives," *Proc. IEEE*, vol. 80, no. 9, pp. 1502–1517, Sep. 1992.
[25] A. Karami and R. Johansson, "Choosing DBSCAN parameters automatically using differential evolution," *Int. J. Comput. Appl.*, vol. 91, pp. 1–11, 2014.
[26] D. G. Kirkpatrick and J. D. Radke, "A framework for computational morphology," *Mach. Intell. Pattern Recognit.*, vol. 2, pp. 217–248, 1985.
[27] D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proc. 14th SIAM Int. Conf. Data Mining*, 2014, pp. 839–847.
[28] M. C. Naldi, R. J. G. B. Campello, E. R. Hruschka, and A. C. P. L. F. Carvalho, "Efficiency issues of evolutionary k-means," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1938–1952, 2011.
[29] A. C. A. Neto, J. Sander, R. J. G. B. Campello, and M. A. Nascimento, "Efficient computation of multiple density-based clustering hierarchies," in *Proc. IEEE Int. Conf. Data Mining*, 2017, pp. 991–996.
[30] A. Smiti and Z. Elouedi, "DBSCAN-GM: An improved clustering method based on Gaussian means and DBSCAN techniques," in *Proc. IEEE 16th Int. Conf. Intell. Eng. Syst.*, 2012, pp. 573–578.
[31] B. Thomee *et al.*, "YFCC100M: The new data in multimedia research," *Commun. ACM*, vol. 59, no. 2, pp. 64–73, 2016.
[32] G. T. Toussaint, "The relative neighbourhood graph of a finite planar set," *Pattern Recognit.*, vol. 12, no. 4, pp. 261–268, 1980.
[33] D. W. Matula and R. R. Sokal, "Properties of gabriel graphs relevant to geographic variation research and the clustering of points in the plane," *Geogr. Anal.*, vol. 12, no. 3, pp. 205–222, 1980.
[34] J. Wetherell, "Java: Algorithms and data structure," 2017. [Online]. Available: https://github.com/phishman3579/java-algorithms-implementation

**Antonio Cavalcante Araujo Neto** received the bachelor's and master's degree from the Federal University of Ceará, Brazil, in 2012 and 2015, respectively. He is currently working toward the PhD degree in the Department of Computing Science, University of Alberta's, Edmonton, Canada. His main research interests are clustering and data mining.

**Joerg Sander** received the MS and PhD degrees in computer science from the University of Munich, Munich, Germany. He is currently a full professor in computing science at the University of Alberta, Edmonton, Canada. His research interests include knowledge discovery in databases, clustering, spatial data mining, and similarity search.

**Ricardo J. G. B. Campello** received the BSc degree in electronics engineering from the State University of São Paulo, Brazil, in 1994, and the MSc and PhD degrees in electrical engineering from the State University of Campinas, Brazil, in 1997 and 2002, respectively. He is currently a full professor with the School of Mathematical and Physical Sciences, University of Newcastle, Callaghan, Australia. His current research interests include data mining, data science, machine learning, and computational intelligence.

**Mario A. Nascimento** is a full professor with the Department of Computing Science, University of Alberta's, Edmonton, Canada. He has also been a visiting professor with the National University of Singapore, Singapore, Aalborg University in Denmark, LMU Munich in Germany and with the Federal University of Ceara in Brazil. His current research interests lie in the domain of spatio-temporal data management.