A Framework for Hierarchical Density-Based Clustering Exploration

by

Antonio Cavalcante Araujo Neto

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

University of Alberta

Abstract

HDBSCAN* is a hierarchical density-based clustering method that requires a single parameter *mpts*, a smoothing factor that implicitly influences which clusters are more detectable in the resulting clustering hierarchy. 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. This thesis aims at studying the problems related to the effects of *mpts* on the clustering hierarchies produced by HDBSCAN*. We present an analysis of HDBSCAN*'s density estimator and discuss how it could be improved to mitigate the issues with the choice of a *mpts* value. We also discuss how this modification affects the results obtained by HDBSCAN* and why one might still need to explore multiple parameter settings to better understand the cluster structures in the data. Hence, we are interested in the efficient computation and exploration of hierarchies constructed under different parameter settings, and in strategies that can ease the task of finding the appropriate amount of smoothing for density estimation in different regions of the data. More specifically, we propose a strategy that is able to efficiently compute over 100 clustering hierarchies with the computational cost of running HDBSCAN* twice. Our strategy is based on the replacement of the complete graph in HDBSCAN* with a much smaller graph, the RNG, that provably contains all the information needed to compute a set of clustering hierarchies for a range of *mpts* values. In order to help

with the analysis of the clustering hierarchies computed with our RNG-based strategy, we propose MustaCHE, a visualization tool that helps users explore a set of clustering hierarchies *and* focus their analyses on values of $mpts$ that produce "significantly" different results. Moreover, we observed that, for some datasets, a single value of $mpts$ is not enough to reveal all the cluster structures in the data simultaneously. Therefore, we discuss how HDBSCAN* can be used to compute hierarchies that contain cluster structures found with different values of $mpts$, and how users can select which values of $mpts$ are to be used in different parts of the data to construct clustering hierarchies in this fashion. While these contributions were made in the context of unsupervised clustering with HDBSCAN*, their relevance go beyond their original purpose. Thus, we discuss how each of the contributions presented in this thesis can be extended to a class of semi-supervised clustering and semi-supervised classification algorithms.

Preface

This thesis is composed of 8 chapters, parts of which consist of published material. The contributions in Chapter 4 have been published as a conference paper (ICDM'17) and as a journal paper (TKDE'19):

- Antonio Cavalcante Araújo Neto, J. Sander, R. Campello and M. Nascimento, "Efficient Computation and Visualization of Multiple Density-Based Clustering Hierarchies," in IEEE Transactions on Knowledge and Data Engineering.

- Antonio Cavalcante Araújo Neto, Jörg Sander, Ricardo J. G. B. Campello, Mario A. Nascimento: Efficient Computation of Multiple Density-Based Clustering Hierarchies. ICDM 2017: 991-996.

Parts of Chapter 5 have been published as a demo paper at a conference (VLDB'18):

- Antonio Cavalcante Araújo Neto, Mario A. Nascimento, Jörg Sander, Ricardo J. G. B. Campello: MustaCHE: A Multiple Clustering Hierarchies Explorer. Proc. VLDB Endow. 11(12): 2058-2061 (2018).

To my wife Camila

For all the love, support and companionship.

"O olhar para traz não deve ser uma forma nostálgica de querer voltar, mas um modo de melhor conhecer o que está sendo, para melhor construir o futuro"

– Paulo Freire, Pedagogia do Oprimido.

"Looking at the past must only be a means of understanding more clearly what and who they are so that they can more wisely build the future."

– Paulo Freire, Pedagogy of the Opressed.

Acknowledgements

In the last five years, I had the opportunity to cross paths with lots of people who have been an important part of my life in Edmonton. First, I would like to express my gratitude to my supervisor Prof. Jörg Sander for all the guidance and mentoring, for believing in me, and for teaching me so much over the years. Your wisdom and patience have inspired me to become a better researcher. I am also grateful to my co-supervisor Prof. Ricardo Campello, whose competence and strong commitment to high-quality science has pushed me to achieve great accomplishments. In addition, I am thankful to Prof. Mario Nascimento for the trust and support from the very beginning of my PhD program, which allowed me to get to where I am today. I feel lucky and privileged to have worked with all of you, and I owe you an enormous debt of gratitude. I would also like thank the other members of my examining committee, Prof. James Bailey and Prof. Osmar Zaiane, for the valuable and insightful comments on my work.

On a personal level, I would like to thank my wife Camila for being such an inspiration to me in so many ways. I feel very grateful and fortunate for having you by my side, encouraging me and sharing every bit of this journey with me. Thanks for all the love and all the laughs that make our life much lighter. I would like to thank my entire family for the love and support that I could always feel regardless the long distance separating us. I am thankful to my mother Ana Angélica for always supporting me and for encouraging me to follow my dreams and be happy. I am thankful to my sister Ana Cláudia for being my first role-model regarding a serious commitment to studies, which certainly came a long way. I am thankful to my late grandfather Cavalcante, who spared no effort to guarantee that I had access to a good education, and I

am thankful to my grandmother Sulamita for all the love and for never letting me forget that me and Camila are always in her prayers.

In Edmonton, I was lucky to have been able to count on many people that made Canada feel like home. I would like to thank Mario and Gabriela, for welcoming me and Camila to Edmonton and for helping us settle. Jadson, thanks for your friendship and for your patience to discuss clustering with me. Marjorie, thanks for being a reminder of home in Edmonton and for helping us become expert movers. Leandro, thanks for all the laughs and for the many lunches and coffees at SUB. Mariana, Priscila and Bandido, thanks for being part of this second family we had in our first year in Edmonton. Rodolfo, thank you for your friendship and for being an amazing curling partner/coach (go team Bravo). Ariane and Chris, thanks for sticking around and for always celebrating life with us. Cacau, Martin, Levi and Camila, thanks for all the awesome times we had making sushi, playing board games or simply sharing funny stories with each other. Beda and Regina, thank you for being such great friends for trips, barbecues, drinks and workouts. Augusto, thanks for always sharing your entertaining stories with us. Francesco, Antonio, Georg and Rubens, thanks for being part of our lives in the short time you stayed in Edmonton. Henrique, Victor and Megha, thanks for the many conversations we had, they were always a nice break from the day-to-day life.

I would also like to thank the people whom I was lucky to meet in my time in Australia. I am very thankful to Gabriel and Gabriela for welcoming me into their home and for being such good friends during my three months in Newcastle. JV, Vinicius and Emy, thank you for your friendship and for the many lunches and coffees together, my time at The University of Newcastle was much more enjoyable with you guys around.

I would like thank my best friends, Vituxo, Coutinho and Andrade, for all the love and support you have been sharing with us on a constant basis throughout the years. Thank you for always participating in our life and for the great friendship that only gets stronger with time.

Last, I would like to thank the Brazilian program Science without Borders/CNPq for funding the first four years of my PhD.

Contents

List of Tables

List of Figures

Chapter 1

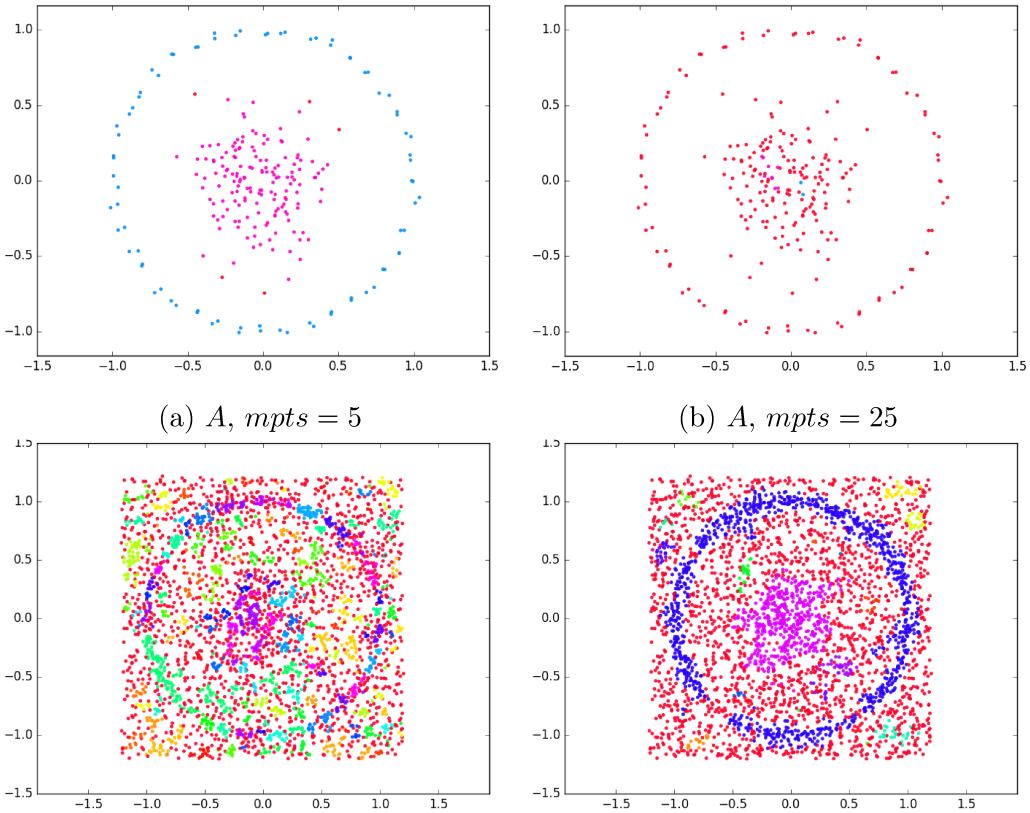
Introduction

The discovery of groups within datasets plays an important role in the exploration and analysis of data. For instance, information about customer groups purchasing particular products can guide providers in the planning of their business. Discovering groups in data can also aid the analysis of medical images [30] or the discovery of knowledge from text documents [48], to cite but a few examples. For scenarios where there is little to no prior knowledge about the data, clustering techniques are widely used. They aim at grouping the elements of a dataset in such a way that elements in the same group are more similar or interrelated to each other than they are to elements in other groups, according to a certain similarity or relatedness measure. 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 [24], OPTICS [2] 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 re-

quires 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 Figure 1.1, 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 (Figures 1.1a and 1.1b) is completely labeled as noise for $mpts > 24$, while the two structures in dataset B (Figures 1.1c and 1.1d) 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 [33], one can identify promising density levels from different hierarchies, produced from different parameter configurations 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



(a) A, $mpts = 5$ (b) A, $mpts = 25$

(c) B, $mpts = 5$ (d) B, $mpts = 25$

Figure 1.1: Clusters from datasets A and B for $mpts = 5$ and 25. Noise points are colored in red in all plots.

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 (Note that the computational cost for the MST construction depends on the number of edges in the input graph, $O(n^2)$ in this case).

This thesis aims at studying the problems related to the effects of $mpts$ on the clustering hierarchies produced by HDBSCAN*. We are interested in understanding how HDBSCAN*'s density estimator is dependent on its input parameter $mpts$, in the efficient computation and exploration of hierarchies constructed with different values of $mpts$, and in strategies that can ease the

task of finding the appropriate amount of smoothing for finding the density-based clusters in the data. Hence, this thesis is based on the investigation of the following research questions:

- *Can we improve HDBSCAN* and avoid the issues related to the choice of a mpts value?*
- *What can users do to select a good value for mpts?*
- *How can one efficiently explore the clustering results for a range of mpts values?*
- *Is a single value of mpts enough to reveal all the cluster structures in the data?*
- *Are these issues only observed in HDBSCAN*?*

In an attempt to answer these questions, we first studied the properties of HDBSCAN*'s density estimator and the influence of *mpts* on the clustering hierarchies computed by HDBSCAN*. We propose a new measure for density estimation in HDBSCAN* that provides advantages over its original formulation. We observed that the estimates obtained by this new measure are smoother and offer a better interpretation of the results, even though the conclusions drawn from the clustering results are similar to the ones drawn from using HDBSCAN*, and one cannot easily exploit the properties of such estimator for an efficient exploration of a range of parameter settings. Due to these observations, we have concentrated our research efforts on strategies that can help with the computation and exploration of clustering results based on the original formulation of HDBSCAN* for multiple parameter settings.

As one of the contributions of this thesis, we provide theoretical and practical results that lead us to a method for computing multiple clustering 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*. In order to achieve that, we replace the complete graph used by HDBSCAN* with the smallest known graph that can still guarantee the algorithm's correctness, and we prove that such graph only needs to be computed once and all the hierarchies for a range of *mpts* can be extracted from it. Moreover, this graph has typically much fewer edges than the complete graph so its construction cost is more than outweighed by the reduction in edge weight computations.

In this thesis, we also address the considerable challenge that users face when wanting to analyze a collection of hierarchies, which can be rather overwhelming without a proper tool. In order to help with the exploration of multiple clustering hierarchies computed for a range of values of *mpts*, we propose MustaCHE, a visualization tool that allows users to inspect clustering hierarchies in a visual and interactive manner. The use of MustaCHE allows users to get a deeper understanding of the cluster structures in a dataset with respect to different parameter values, and of how different clustering hierarchies compare to one another. MustaCHE can be used to narrow down the exploratory analysis to the values of *mpts* that yield the most significant hierarchies, preventing users from having to examine dozens of hierarchies unnecessarily.

Furthermore, to deal with cases where different cluster structures are unveiled at different values of *mpts*, we discuss how HDBSCAN* can be adapted to support the use of multiple values of *mpts* to construct a single hierarchy. By properly selecting the values of *mpts* to be used in different subsets of the data, users are then able to build hierarchies that contain cluster structures that would otherwise not be detectable with a global amount of smoothing.

This provides more flexibility and turns HDBSCAN* into a more adaptive method w.r.t. the local characteristics of different regions of the space or different subsets of the data.

Last, we discuss how other algorithms in the literature that are based on the same fundamental concepts as HDBSCAN* can suffer from the same issues related to the selection of a value of *mpts* and how our contributions for exploration of results with HDBSCAN* can be reused in other contexts other than unsupervised clustering.

1.1 Contributions

The contributions of this thesis can be summarized as follows:

1. We analyse HDBSCAN*'s density estimator and assess whether the issues related to the choice of a value of *mpts* can be mitigated with an improved estimator.

2. We propose a highly efficient approach to compute all cluster hierarchies for a *range* of *mpts* values.

3. We propose MustaCHE, a visualization tool that allows users to analyze a collection of hierarchies for a range of *mpts* values, and we present case studies that illustrate how these analyses can be performed.

4. We discuss the use of multiple values of *mpts* in a single clustering hierarchy as an attempt to make HDBSCAN* more adaptive to the local density of points.

5. We discuss how our contributions made towards unsupervised clustering with HDBSCAN* can be directly applied or adapted to other algorithms in the literature.

1.2 Thesis Layout

This thesis contains eight chapters. Chapter 2 presents an overview of the background concepts used throughout this thesis. Chapter 3 presents an analysis of HDBSCAN*'s density estimator w.r.t. the parameter *mpts* and discusses the consequences of improving such estimator. Chapter 4 presents our efficient strategy to compute a set of HDBSCAN* hierarchies w.r.t. a range of values of *mpts*. Chapter 5 presents MustaCHE, a visualization tool for exploring a set of clustering hierarchies. Chapter 6 discusses the problem of using multiple values of *mpts* in a single hierarchy and presents the necessary adaptations in HDBSCAN*. Chapter 7 discusses how the contributions of this thesis can be adapted or directly applied to other methods in the literature. Chapter 8 recapitulates the contributions of this thesis and discusses future research directions.

Chapter 2

Background

In this chapter, we cover the fundamental concepts that at the foundation of this thesis. We formally introduce density-based clustering and present the classic algorithms that constitute the basis of this thesis. We discuss the intuition behind hierarchical density-based clustering and HDBSCAN*, the main object of study of this thesis. Last, we introduce the notion of proximity graphs and discuss the aspects that make them interesting for clustering applications.

2.1 Density-Based Clustering

Density-based clustering is a popular paradigm of clustering that aims at finding high-density regions in the data (*clusters*) separated by low-density regions. Contrary to the traditional notion of clustering, where objects are grouped together based solely on their pairwise similarity, density-based clustering captures contiguous regions in the space where the density does not drop below a certain threshold. Hartigan [21] formalized this intuition by introducing the concept of *density contour clusters*. Given a dataset X and a density level λ, Hartigan defines the *density-contour clusters* based on the unknown probability density function f from which X was sampled. Thus, the *density-contour clusters* of X at a level λ correspond to the *maximal* connected components of the upper level set $L(\lambda) = \{x : f(x) \geq \lambda\}$. Figure 2.1 illustrates a prob-

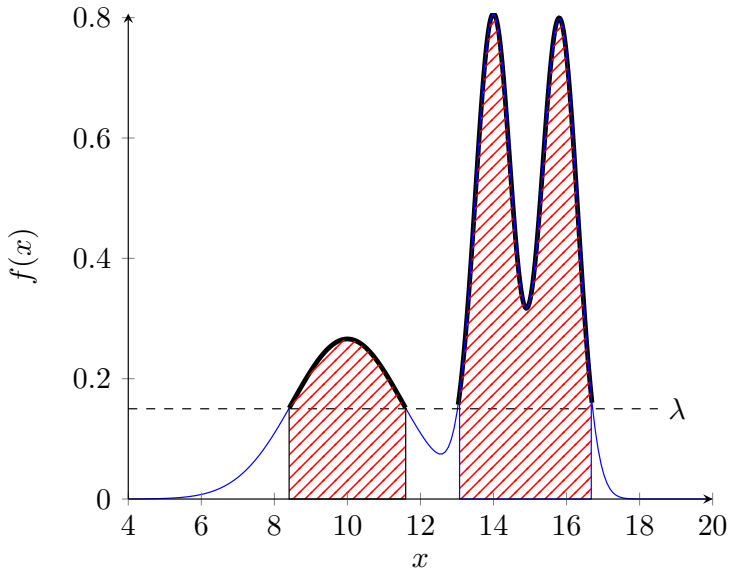


Figure 2.1: Upper-level Set

ability function f where the connected components w.r.t. the upper-level set $\lambda = 0.15$ correspond to the highlighted intervals. Note that any point x that is not in the highlighted intervals does not belong to the upper-level set and is considered noise at that level. It is easy to see that in order to go from one of the components to the other, one has to "go through" regions where the function f is smaller than λ.

Besides the ability to model arbitrary-shape clusters and noise in the data, another advantage of density-based clustering is that users typically do not need to select the number of clusters to be found by the algorithm. These characteristics have led to an abundance of works that followed this paradigm over the years (*e.g.* [2], [16], [24]). Among those works, DBSCAN [16] is considered one of the most popular density-based clustering algorithms in the literature. Given a dataset X and two input parameters ε and $mpts$, DBSCAN estimates the density at each point $x \in X$ by looking at their ε-neighborhood, denoted by N_ϵ and defined in Equation 2.1.

$$N_\varepsilon(x) = \{y : d(x, y) \leq \varepsilon\} \tag{2.1}$$

Points that have more than *mpts* many points in their ε-neighborhood are called *core-points*. In an analogy to Hartigan's model, the upper-level set L w.r.t. ε and *mpts* determined by DBSCAN can be defined as follows:

$$L(\varepsilon, mpts) = \{x : |N_\varepsilon(x)| \geq mpts\}$$

In fact, it is easy to see that $L(\varepsilon, mpts)$ is simply the set of *core-points* in X. Now, in order to find the density-based clusters of X, one has to construct the maximal connected components w.r.t. $L(\varepsilon, mpts)$. Connectivity in DBSCAN is determined according to the following definitions:

Definition 1 (Directly Density-Reachable) *A point x_i is directly density-reachable from a point x_j w.r.t. ε and mpts if x_j is a core-point and $x_i \in N_\varepsilon(x_j)$.*

Definition 2 (Density-reachable) *A point x_i is considered density-reachable from x_j w.r.t. ε and mpts if there is a chain of direct density-reachability from x_j to x_i.*

Definition 3 (Density-connected) *Two points x_i and x_j are density-connected w.r.t. ε and mpts if they are density-reachable from each other.*

Based on these definitions, a density-based cluster C w.r.t. ε and *mpts* is a non-empty subset of X that satisfies the following conditions:

- $\forall x_i, x_j \in C$: x_i and x_j are density-connected;
- $\forall x_i, x_j$: if $x_i \in C$ and x_j is density-reachable from x_i, then $x_j \in C$;

The first condition guarantees that there is a chain of connections between any two points in the same cluster where the density does not drop below the threshold determined by ε and *mpts*. The second condition guarantees that clusters are *maximal* w.r.t. the density threshold determined by ε and

mpts, *i.e.* no other points can be added to that cluster at that density level. Note that these properties conform with the model of *density-contour clusters* proposed by Hartigan.

2.2 Hierarchical Density-Based Clustering

Despite its advantages, DBSCAN is only able to find clusters w.r.t. to a single density-level determined by ε and *mpts*. As an attempt at mitigating this issue, OPTICS [2] was proposed as an evolution of DBSCAN with the purpose of detecting density-based clusters at different density levels. The main idea behind OPTICS consists in creating an ordering of the points in a dataset that reveals the structure of its density-based clusters. Roughly speaking, points that are in the same density-based cluster appear spatially close in the order.

The order imposed by OPTICS can be visualized in what the authors call a *reachability plot*. A reachability plot can be represented as a bar plot, where each bar corresponds to a point in the dataset, and the height of each bar is determined by the smallest distance at which its corresponding point is density-reachable from the points that precede it in the plot. Thus, the density-based clusters appear as *valleys* separated by *peaks*. Figure 2.2 illustrates the reachability plot constructed from the ordering induced by OPTICS in a toy dataset containing three clusters. Note that each valley in the plot corresponds to a cluster in the data. Along with OPTICS, the authors have proposed a post-processing procedure for extracting clusters from a reachability plot. However, this procedure requires setting an input parameter that is not intuitive to users. Thus, while OPTICS can be regarded as an important step towards finding the hierarchical organization of the density-based clusters at different density levels, its formulation does not include an explicit representation of a clustering hierarchy of the data.

One of the earliest formal definitions of hierarchical density-based clus-

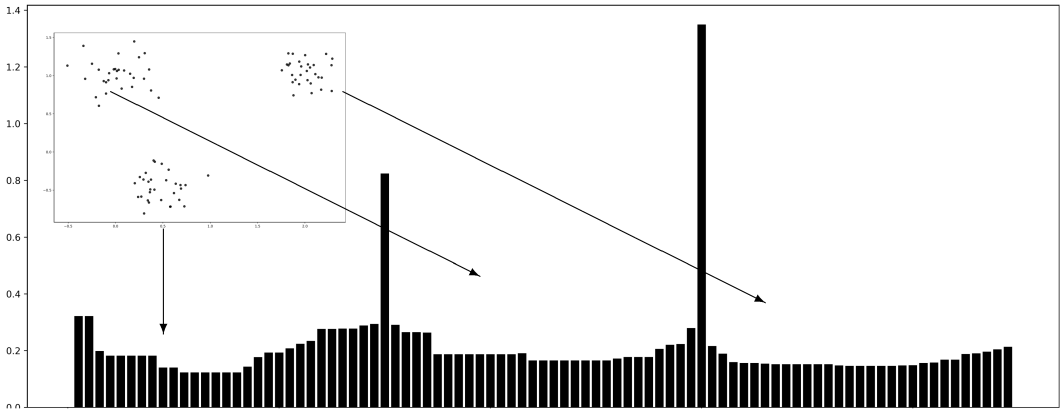


Figure 2.2: OPTICS reachability plot

tering comes from Hartigan's *density-contour tree* model [21]. Hartigan observed that, given two density levels λ_1 and λ_2, where $\lambda_1 < \lambda_2$, the *density-contour clusters* w.r.t. λ_1 contain the *density-contour clusters* w.r.t. λ_2. While a *density-contour cluster* is conceptually defined w.r.t. a density level λ, a *density-contour tree* corresponds to the structure generated w.r.t. several (ideally *all*) values of λ. For instance, Figure 2.3 illustrates the *density-contour clusters* C_1 and C_2 at $\lambda_1 = 0.10$, and C_3 and C_4 at $\lambda_2 = 0.40$. One can see that C_3 and C_4 are contained in C_2. Thus, one can construct the clustering hierarchy corresponding to a probability density function f by generating *all* possible density levels and organizing them as a tree.

Much like DBSCAN and OPTICS have tried estimating the density and connectivity levels of a dataset to produce a partitioning of the data and an ordering of the points in a dataset, respectively, the same fundamentals can be used to build an *explicit* clustering hierarchy that estimates the nested connected components of the underlying density function. Campello et al [8], [9] proposed HDBSCAN* as a practical and theoretical generalization of its predecessors (DBSCAN and OPTICS) with the purpose of overcoming their drawbacks. Similarly to other density-based clustering algorithms, HDBSCAN* is able to detect arbitrary-shaped clusters and noise. However, HDBSCAN*

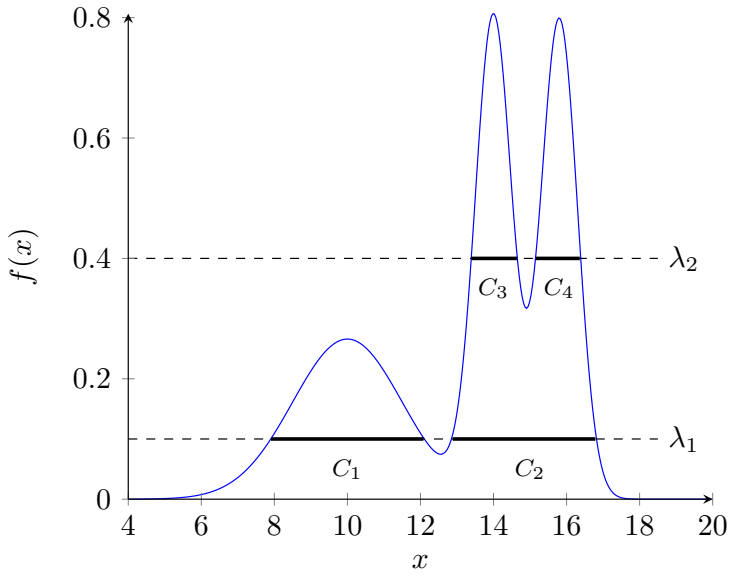


Figure 2.3: Upper-level Set

stands out for its ability to build a hierarchical organization of the cluster structures embedded in the data rather than just a single flat partitioning or an ordering of the points in the data. 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 ε around p that covers *mpts* other points, called p's "core distance" [1] w.r.t. *mpts*; and (ii) for each value of ε: the clusters and the noise points w.r.t. ε 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:

$$mrd_{mpts}(p,q) = max\{c_{mpts}(p), c_{mpts}(q), d(p,q)\} \tag{2.2}$$

where $d(\cdot,\cdot)$ represents the underlying distance function (typically Euclidean

[1] The core distance can be interpreted as inversely proportional to an unnormalized k-nearest neighbor (k-NN) density estimate.

distance), and $c_{mpts}(p)$ represents the core distance of p, which is formally the distance from p to its $mpts$-th nearest neighbor, $mpts$-NN(p):

$$c_{mpts}(p) = d(p, mpts\text{-NN}(p)) \tag{2.3}$$

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 ε at which the corresponding endpoints are directly ε-reachable w.r.t. $mpts$, *i.e.*, the smallest distance at which both points are in each other's ε-neighborhood, and both ε-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 $mpts$:

- Increasing the value of $mpts$ usually leads to higher values of c_{mpts}, never smaller;
- 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;
- When decreasing the value of $mpts$, edge weights can either decrease or remain the same, but never increase.

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.[2] From this MST, the *complete* density-based

[2]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.

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 (ε and *mpts*), removing all edges from G_{mpts} with weights greater than ε 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 (MST) of G_{mpts}.

Moreover, one can extract a flat clustering partitioning from the hierarchy produced by HDBSCAN*. Trivially, a straight cut in the hierarchy at a level ε is equivalent to a partitioning found by DBSCAN* w.r.t. ε and *mpts* [3]. HDBSCAN*'s framework includes an automatic cluster extraction method, based on the FOSC framework [8], that allows the extraction of clusters at different ε levels according to suitable criteria.

2.3 Minimum Spanning Tree

The representation of structures formed by the points of a dataset in the space plays an important role in the detection of density-based clusters. Moreover, a suitable data structure for a density-based clustering algorithm must be able to model the distances between pairs of points of a dataset in a way that allows one to infer how groups of points are organized and separated. As previously mentioned, HDBSCAN* makes use of a *minimum spanning tree* in order to compute a clustering hierarchy of a dataset. We now introduce the formal definition of a *minimum spanning tree* and discuss why its properties are useful in clustering applications.

Let $G = (V, E)$ be an undirected connected graph with a set of vertices V and a set of edges E, and $w(\cdot)$ be a function that determines the weight

[3]HDBSCAN* is based on DBSCAN*, in which clusters contain only *core-points*; in DBSCAN clusters can contain *"border-points"*, which are not *core-points* but only in the ε-neighborhood of a *core-point*.

of the edges in E. The *minimum spanning tree* $T = (V, E')$ of G corresponds to a connected sub-graph of G such that $\sum_{e \in E'} w(e)$ is minimal among all connected sub-graphs (V, E').

By defining the function $w(\cdot)$ in a way such that edges represent the density levels at which groups of points are connected, one is able to estimate the connected components (*clusters*) at multiple levels by decomposing the *minimum spanning tree* in decreasing order of edge weight. Due to this property, the *minimum spanning tree* is a key component of some clustering algorithms in the literature (*e.g.* [9], [19], [43]).

Regarding practical aspects, the time complexity for computing an MST depends on the number of edges in the original graph G. When no index support is used, the construction of an MST takes $\mathcal{O}(|E|)$ time. In the case of the complete *mutual reachability graph* used in HDBSCAN*, this represents a complexity of $\mathcal{O}(|V|^2)$. This performance bottleneck is one of the issues explored in this thesis and will be addressed in more details in Chapter 4.

2.4 Proximity Graphs

In computational geometry, proximity graphs are often used to express geometric relationships between objects distributed in the space. In practice, the connectivity in these graphs is determined by how close objects are to each other, or by regions of the space that must satisfy certain constraints. While the focus of this thesis is mainly density-based clustering, proximity graphs can be rather useful as a way of representing how data points are organized/distributed in the space. Similarly to *minimum spanning trees*, proximity graphs can be used to represent the connections that correspond to the *density-based clusters* in the data. We discuss two proximity graphs, namely the *Relative Neighborhood Graph* [45] and the *Gabriel Graph* [17].

2.4.1 Relative Neighborhood Graph

The Relative Neighborhood Graph (RNG) [45] is characterized by the notion of *relative neighborhood* between pairs of points. In essence, two points are *relative neighbors* when no other point is closer to both of them than they are to each other, as expressed in Equation 2.4.

$$d(a,b) \leq \max\{d(a,c), d(b,c)\}, \forall c \neq a, b \tag{2.4}$$

Another intuitive way to express the connectivity between points in the RNG is through the geometric interpretation of relative neighborhood. Figure 2.4 shows two points a and b, and two circles (2-dimensional hyper-spheres) with radius $d(a,b)$ centered at a and b. The highlighted region corresponds to the intersection of the two circles and is called the *lune* of (a,b). Whenever the *lune* of a pair of points is empty, these points are considered relative neighbors and, consequently, are connected in the RNG.

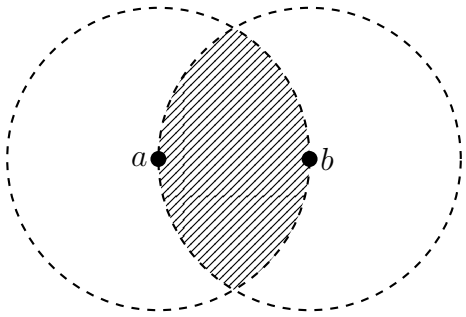


Figure 2.4: $lune(a,b)$.

One of the characteristics that make the RNG interesting for clustering applications, and especially to HDBSCAN*, is that it is a super-graph of the *Euclidean Minimum Spanning Tree*. Since the RNG of a set of points has typically much fewer edges than the complete graph, computing a *Euclidean Minimum Spanning Tree* from the RNG is typically much faster than doing so from the complete graph. In the context of our work, one must show that the same property is still valid when the edge weights are determined by a

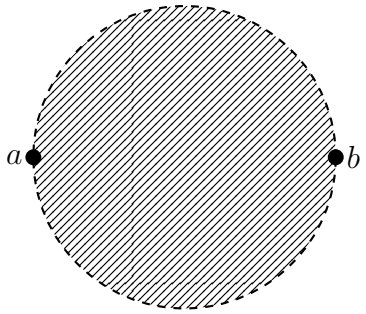


Figure 2.5: Gabriel Graph constraint region.

different measure. The use of the RNG as a replacement for the complete graph in HDBSCAN* is discussed in details in Chapter 4.

2.4.2 Gabriel Graph

Similarly to the RNG, the Gabriel Graph (GG) is also defined according to a spatial constraint. In this case, two points a and b are connected in the GG when the hyper-sphere of radius equals to $\frac{d(a,b)}{2}$ with center at the mid-point between a and b is empty. Figure 2.5 illustrates two points a and b and the highlighted circle (2-dimensional hyper-sphere) that has a and b on opposite sides of the circle.

It is easy to see that the GG is a super-graph of the RNG, as the region of the space that determines connectivity for the GG is completely contained in the region of the space that determines connectivity for the RNG, as illustrated in Figure 2.6. If two points a and b are connected in the RNG, there is no other point c in the *lune(a, b)*, and consequently, there is no other point in the region that determines connectivity for the GG. Furthermore, one can say that a and b are also connected in the GG. If a and b are *not* connected in the GG, there must exist a point c located in the hyper-sphere that has a and b on the opposite sides of its diameter. Thus, such point c must also be in the *lune(a, b)*, which prevents a and b from being connected in the RNG. Due to the relationship between the RNG and the GG (RNG $\subseteq$ GG), the

GG inherits all the properties w.r.t. the representation of the structures in the data discussed for the RNG. In this thesis, the GG is used in a pre-processing phase to filter out edges that do not belong to the RNG.

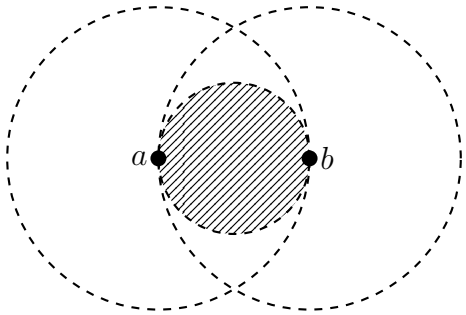


Figure 2.6: GG and RNG constraint regions.

Chapter 3

A Closer Analysis of HDBSCAN*

3.1 Introduction

In our motivation, we have argued that choosing a value of *mpts* is *not* trivial and that, for some datasets, multiple values of *mpts* might be necessary to reveal the cluster structures in the data. In this chapter we present an analysis of the density estimator used in HDBSCAN* and investigate whether changing its formulation can mitigate the issues studied related to the selection of a value for the parameter *mpts*.

The density estimator used in HDBSCAN* is based on the distances between data points and their *mpts*-nearest neighbors. Intuitively, when a point a has more than *mpts* many neighbors in its ε neighborhood, a is considered to be dense w.r.t. *mpts* at a density-level $\lambda = 1/\varepsilon$. This is then used to determine density-connectivity between points according to their *mutual reachability distance*. Moreover, the density-based clusters at a level $\lambda = 1/\varepsilon$ are determined by the connected components of the *minimum spanning tree* in the space of *mutual reachability distances* when all edges of weight equal to or larger than ε are removed from the tree. This formulation tries to capture the concepts established in Hartigan's *density-contour* model [21] that formalizes density-based clusters. In this model, when two points are part of the same cluster at

a density level λ, the density along every path connecting those two points, passing through other points in the same cluster, never drops below λ.

In the following, we analyze the paths connecting data points corresponding to the straight lines determined by the edges of the MST in the space of *mutual reachability distances*. When removing edges with weight larger than ε, the remaining edges that compose each of the resulting connected components have a weight necessarily smaller than ε. However, in a closer examination of HDBSCAN*'s formulation, one can observe that the density along the paths determined by these edges is not guaranteed to be above the density level $\lambda = 1/\varepsilon$.

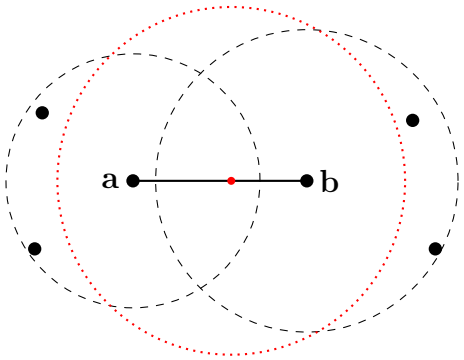


Figure 3.1: Overestimation of density

Figure 3.1 shows two points a and b, and their *core-distances* w.r.t. $mpts = 3$ represented by the black dashed circles. In this example, the *mutual reachability distance* between a and b is equal to the distance $d(a, b)$ between both points (see Chapter 2). Thus, in order for a and b to be part of the same cluster at a density level $\lambda = 1/d(a, b)$, one expects the density along the path between a and b w.r.t. $mpts$ to be above $\lambda = 1/d(a, b)$. This should mean that a circle (2-dimensional ball) of radius $d(a, b)$ placed anywhere along the path between a and b must enclose at least $mpts = 3$ points. However, one can see in Figure 3.1 that the red dotted circle of radius $d(a, b)$ placed along the path between a and b does not enclose $mpts = 3$ points. In this case, a circle with a

radius larger than $d(a, b)$ would be needed to guarantee that *all* points along the path between a and b are dense w.r.t. *mpts*. Thus, by determining that a and b are part of the same cluster at a level $\lambda = 1/d(a, b)$, HDBSCAN* is overestimating the density along the path between a and b.

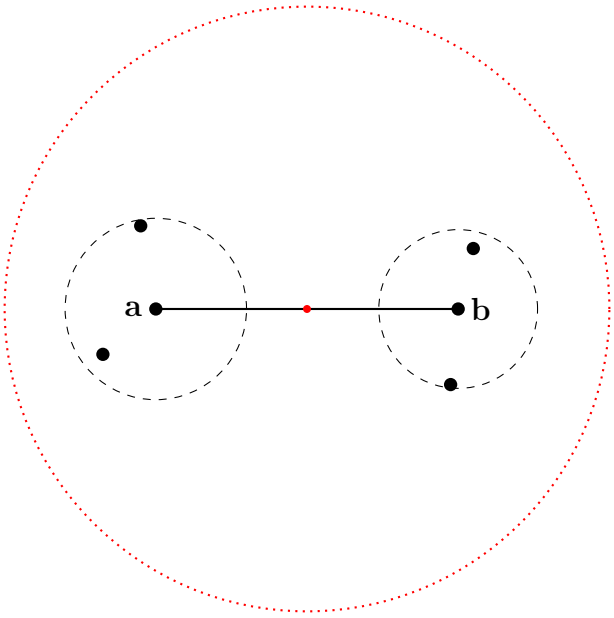


Figure 3.2: Underestimation of density.

On the other hand, Figure 3.2 shows a different configuration of points where a and b are further apart from each other than their *core-distances*. Similarly to the example in Figure 3.1, the *core-distances* of points a and b w.r.t. $mpts = 3$ are represented by the black dashed circles. As in the previous example, the *mutual reachability distance* between a and b also corresponds to the distance $d(a, b)$ between them. However, as seen in Figure 3.2, the red dotted circle with radius $d(a, b)$ is actually much larger than it needed to be in order to meet the density requirement determined by *mpts*. In this case, the formulation of HDBSCAN* underestimates the density along the path between a and b.

Both examples in Figures 3.1 and 3.2 show configurations of points where the density estimate as determined by the *mutual reachability distance* lead

to certain errors and the density along the paths between points cannot be interpreted the same way as the density at points in the data – number of points enclosed by a circle of radius ε. Motivated by these observations, we investigate an alternative measure for density estimation in HDBSCAN* that addresses the issues with underestimation and overestimation of density along the straight path between two points, and we discuss how this measure affects the results obtained by HDBSCAN* and the need for exploring results w.r.t. multiple values of *mpts*.

This chapter is organized as follows. In Section 3.2 we present a new measure for density estimation in HDBSCAN* that offers a more precise estimate of the density along straight paths between points and allows a better interpretation of the resulting clustering hierarchy. In Section 3.3 we present our conclusions and directions for future research. A discussion on why the formulation of HDBSCAN' cannot be easily exploited for the efficient computation of multiple clustering hierarchies w.r.t. to a range of values of *mpts* is provided in Appendix A.1.

3.2 HDBSCAN'

The current definition of *mutual reachability distance* (Equation 3.1) available in HDBSCAN* captures the levels at which data points are directly density-reachable, and these levels are ultimately used to build the clustering hierarchy of the dataset.

$$mrd_{mpts}(a, b) = max\{c_{mpts}(a), c_{mpts}(b), d(a, b)\} \tag{3.1}$$

However, that formulation imposes a strong requirement for connectivity between pairs of points – direct density-reachability – and can lead to underestimation or overestimation of density. As a consequence, the density along the paths (indicated by the edge weights of the MST) cannot be interpreted

the same way as the density measured at data points.

With that in mind, we propose an adjusted definition of *mutual reachability distance* that provides an interpretation of the density along the straight paths (between data points) similar to the interpretation of the density at data points. Intuitively, we want to find the smallest value r such that, at any point p along the straight path between points a and b, the ball $B_r(p)$ with radius r encloses at least $mpts$ many points. At the points a and b, the smallest radius r such that the ball $B_r(p)$ encloses $mpts$ many points corresponds to the *core-distances* of each point. Thus, the value r must be at least equal to or larger than the *core-distances* of each point. Note that this is are already achieved by the current formulation expressed in Equation 3.1. However, the underestimation/overestimation of density can happen when the distance $d(a, b)$ is too small to enclose $mpts$ many points along the straight path between a and b or is larger than necessary to enclose $mpts$ many points along the same path. Therefore, we replace $d(a, b)$ in the formulation with a function $d^*(a, b)$, as shown in Equation 3.2, that is able to guarantee the density level along the path.

$$mrd'_{mpts}(a, b) = max\{c_{mpts}(a), c_{mpts}(b), d^*(a, b)\} \tag{3.2}$$

This formulation of *mutual reachability distance* mrd'_{mpts} defines what we call HDBSCAN', an attempt at mitigating the issues with HDBSCAN* discussed in Section 3.1.

A straightforward way to define the function $d^*(a, b)$ is to consider the radius r such that, for any point p along the path between a and b, the ball $B_r(p)$ encloses either a or b's $mpts$ nearest neighbors. Figure 3.3 shows two points a and b with their *core-distances* represented by the black dashed circles, and a red dotted circle B_r centered at the point p' along the path between a and b. Note that "moving" B_r towards a makes it enclose all of a's $mpts$ nearest neighbors, while and "moving" towards b makes it enclose all of b's $mpts$

nearest neighbors. Also, when centered exactly at p', B_r encloses both a and b's neighborhoods. Therefore, with B_r, we guarantee that, at any point along the straight path between a and b, either of a or b's $mpts$ nearest neighbors are enclosed. When looking at the exact configuration of points and position of B_r as illustrated in Figure 3.3, one can determine that the diameter of the ball B_r corresponds to the sum of the *core-distances* of a and b and the distance $d(a, b)$ between both points. Consequently, the radius r of B_r corresponds to half of this value, as expressed in Equation 3.3.

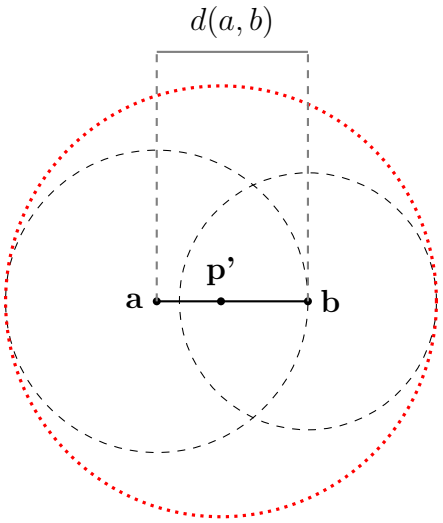


Figure 3.3: $d^*(a, b)$

$$d^*(a, b) = \frac{d_{core}(a) + d_{core}(b) + d(a, b)}{2} \tag{3.3}$$

The diagram in Figure 3.4 shows how mrd'_{mpts} compares to mrd_{mpts} w.r.t. several configuration of points and *core-distances*. The scale at the top of Figure 3.4 represents the values of mrd'_{mpts} in terms of mrd_{mpts}. In the two leftmost configurations, the circles with radii determined by the *core-distances* of each point do not intersect – in the first configuration both *core-distances* are zero. The density along the path between both points in these cases are usually underestimated by HDBSCAN* (higher values of *mutual reachability*

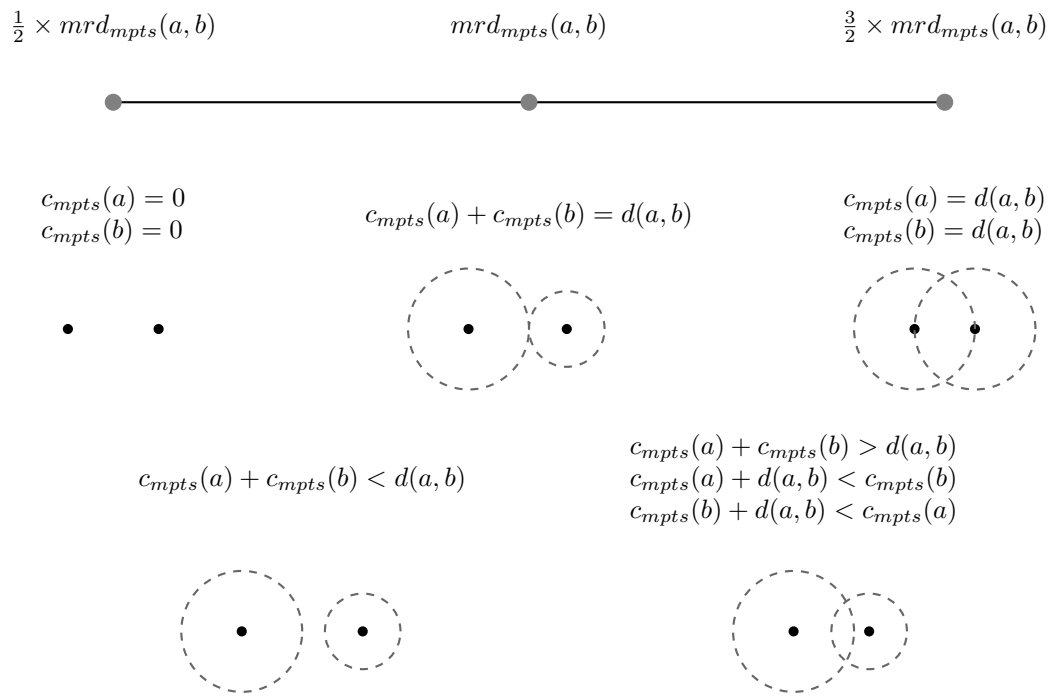


Figure 3.4: HDBSCAN* vs. HDBSCAN'

distance). On the other hand, the values estimated by mrd'_{mpts} in these cases are smaller than what is estimated by mrd_{mpts} up to a factor of 1/2. In the two rightmost configurations, where there is a non-empty intersection between the circles determined by the *core-distances* of each point, the density estimates found by HDBSCAN* are overestimated (lower values of *mutual reachability distance*). In this case, mrd'_{mpts} tries to compensate for the overestimation and evaluates to values up to 3/2 the values obtained by mrd_{mpts}. Last, in the particular configuration where the sum of the *core-distances* is equal to the distance between both points, both definitions of *mutual reachability distance* coincide.

Note that in HDBSCAN', as opposed to HDBSCAN*, when two points are connected at a certain density level, it is guaranteed that there is a path connecting these two points where the density does not drop below that density level, as stated in Hartigan's model. However, neither HDBSCAN' nor

HDBSCAN* can guarantee that if two points are on the same density level, they will be part of the same connected component (if the path that connects then is not along the straight lines connecting points). Also, note that the radius determined by Equation 3.3 does not correspond to the tightest estimate of the density along the path between a and b. In fact, when centered at point p', the ball B_r with radius determined by Equation 3.3 encloses $2 \times mpts$ many points, which is twice the number of points needed for a point to be dense. However, mrd'_{mpts} is a safe bound that only takes into account the *core-distances* of each point but not the actual position of each points' neighbors. For a tighter estimate, one can try and find smaller radii that enclose *only* $mpts$ many points. Figure 3.5 illustrates two points a and b and their *core-distances* w.r.t. $mpts$ from 1 to 4, with the highlighted dashed circles representing the *core-distance* of a w.r.t. $mpts = 3$ and the *core-distance* of b w.r.t. $mpts = 2$. When placing a circle B_r along the path between a and b such that it encloses exactly the two highlighted circles, one guarantees that at least 5 points are enclosed by B_r. This is represented in Figure 3.5 by the red dotted circle. The radius r in this case is determined by the *core-distances* of a and b w.r.t. $mpts = 3$ and $mpts = 2$, respectively, and the distance between a and b: $(c_3(a) + c_2(b) + d(a, b))/2$. In order to guarantee that *any* point along the straight path between a and b is dense w.r.t. $mpts$, one must look at all the pairs of *core-distances* of a and b that enclose a combined total of $mpts$ many points. This intuition is formally expressed in Equation 3.4.

$$d^{**}(a, b) = \max_{1 \leq i < mpts} \left\{ \frac{d(a, b) + d_{mpts-i}(a) + d_i(a)}{2} \right\} \tag{3.4}$$

We refer to the algorithm that uses the definition in Equation 3.4 as HDBSCAN". Note that the definitions in Equations 3.3 and 3.4 can be similarly interpreted and offer a minimum guarantee about the density along the paths between data points. Even though neither of these definitions corresponds to

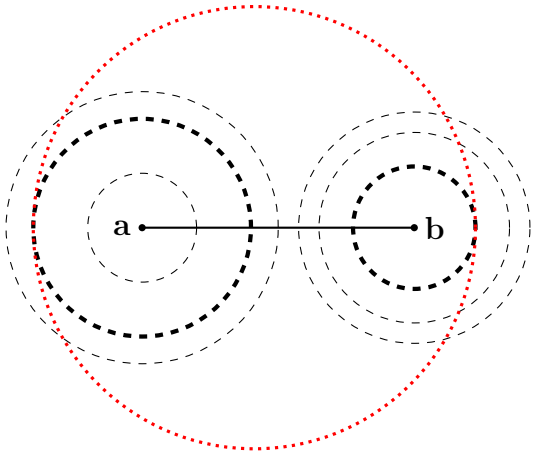
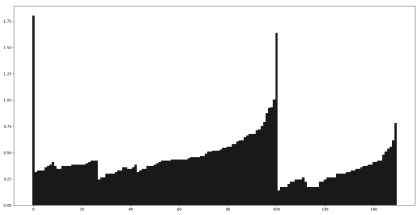


Figure 3.5: $d^{**}(a, b)$

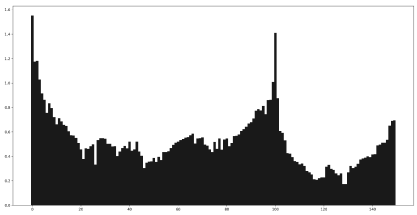
the most precise estimate one can possibly get from the data, they represent a trade-off between the accuracy of the estimates and the computational complexity to find these estimates: while Equation 3.3 is simple to compute, it corresponds to a less precise estimate; on the other hand, Equation 3.4 takes an extra effort in the order of $\mathcal{O}(mpts)$, but results in a tighter estimate.

Our preliminary tests with HDBSCAN' and HDBSCAN" have shown that the density estimates measured by mrd'_{mpts} are much smoother than the ones measured by the original formulation available in HDBSCAN*, as shown in Figures 3.6 and 3.7. Thus, when the clustering hierarchies computed with HDBSCAN' and HDBSCAN" are visualized as reachability plots, the separation between density-based clusters appear as gradual/smooth transitions instead of abrupt changes in the plot, as often observed in the hierarchies computed by HDBSCAN*. We also observed that the hierarchies constructed by HDBSCAN' and HDBSCAN" present the same level of dendrogram purity[1] [22] as the hierarchies constructed by HDBSCAN* and that all formulations lead to similar conclusions about the cluster structures in the data. In Table 3.1, we compared the best dendrogram purity scores that can be achieved with

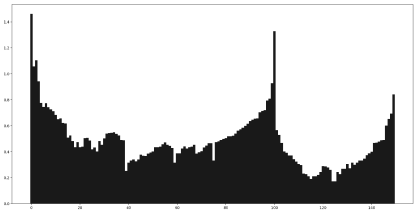
[1]The dendrogram purity measures the purity of the smallest clusters in the hierarchy that contain pairs of points with the same label. If a cluster only contains points with the same label, then the purity of that cluster is 1.



(a) HDBSCAN*



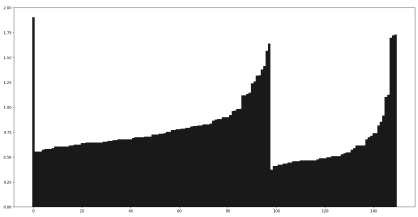
(b) HDBSCAN'



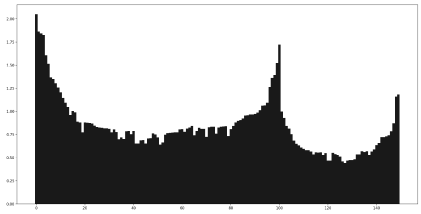
(c) HDBSCAN"

Figure 3.6: Iris - $mpts = 5$

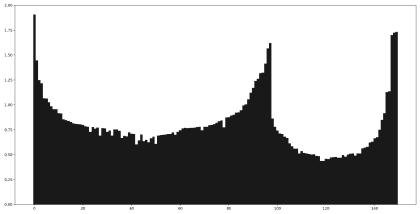
HDBSCAN*, HDBSCAN' and HDBSCAN" for a collection of real datasets for values of $mpts \in [1, 50]$. We observed that, for most datasets, all techniques are practically equivalent and can be applied to obtain equally good clustering hierarchies in terms of dendrogram purity. These results indicate that the new *mutual reachability distance* definitions do not eliminate the problem of selecting an appropriate value of $mpts$ to be used in the clustering process. Moreover, extracting clusters from the hierarchies constructed with HDBSCAN' and HDBSCAN" is not as effective as it is in HDBSCAN*. As HDBSCAN* makes certain "mistakes" that emphasize structures by underestimating the density between points that are far away from each other, the cluster structures in the data appear more pronounced in the clustering hierarchies and can more reliably be extracted with FOSC. These observations show that HDBSCAN*, despite the errors that are made in the density estimates, is effective in finding the nested density-based clusters in the data, which suggests that the errors are of a magnitude that does not affect the practical conclusions that can be drawn from a clustering result.



(a) HDBSCAN*



(b) HDBSCAN'



(c) HDBSCAN"

Figure 3.7: Iris - $mpts = 20$

	HDBSCAN*	HDBSCAN'	HDBSCAN"
banknote-authentication	0.85	0.93	**0.95**
cardiotocography	**0.94**	**0.94**	**0.94**
diggle-table	**0.95**	**0.95**	**0.95**
iris	**0.84**	**0.84**	**0.84**
mfeat-factors	**0.60**	**0.60**	**0.60**
mfeat-karhunen	**0.63**	**0.63**	**0.63**
seeds	0.69	**0.71**	**0.71**
segmentation-normcols	0.59	**0.60**	0.59
stock	**0.70**	**0.70**	**0.70**
wdbc	0.72	**0.73**	**0.73**
wine-187	**0.69**	**0.69**	**0.69**
yeast-galactose	**0.91**	**0.91**	**0.91**

Table 3.1: Best Dendrogram Purity - $mpts \in [1, 50]$

3.3 Conclusions

In this chapter we presented a thorough analysis of how density is estimated in HDBSCAN* and investigated HDBSCAN', an adaptation of HDBSCAN* based on a new definition of *mutual reachability distance* that offers a better interpretation of the density along the straight paths between points and mitigates the issues with overestimation and underestimation of density observed in HDBSCAN*. However, we have found that the results obtained by HDBSCAN' are not fundamentally different from the ones obtained by HDBSCAN*, and apart from the theoretical advantages, both strategies lead to essentially the same conclusions about the cluster structures in the data.

Overall, the important conclusion from the analysis in this chapter is that the replacement of the *mutual reachability distance* in HDBSCAN* with a theoretically improved density estimate does not necessarily imply "better" clustering results. Consequently, HDBSCAN* continues to be a state-of-the-art algorithm, even though it might require the computation of clustering hierarchies w.r.t. multiple values of *mpts*. In practice, the formulation of HDBSCAN' makes it harder for one to efficiently explore multiple parameter settings, since the new definition of *mutual reachability distance* cannot be exploited to achieve the same levels of performance as the original definition available in HDBSCAN*. In Chapter 4, we present an efficient strategy for the computation of a collection of clustering hierarchies based on the original density estimator available in HDBSCAN*. A more detailed discussion on why the same strategy cannot be applied to HDBSCAN' is available on Appendix A.1.

Chapter 4

RNG-HDBSCAN*

The first contribution in this thesis is 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. To achieve that, we show the following:

1. The smallest known proximity 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 the correctness of the results.

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. Our 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.

4.1 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], [27], [39], which are loosely related to the issue illustrated in Figure 1.1. 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 [10], [23], [25], 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 (EMST), 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 [45]. Other spanning subgraphs of the complete graph G that contain the EMST are the Gabriel Graph [17], [46] and the Relative Neighborhood Graph (RNG) [45]. 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 thesis 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 $mpts$.

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.

4.2 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 (EMST) 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 (RNG) in the following way[45]:

$$EMST \subseteq RNG \subseteq GG \subseteq DT \tag{4.1}$$

The RNG and GG are special cases of a family of graphs called β-skeletons [29], which can range from the complete graph to the empty graph, when β goes from 0 to ∞. A value of $\beta = 1$ results in the GG and $\beta = 2$ in the RNG.

Given this result, the RNG, or possibly a β-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 β-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 β-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 β-skeleton for our task?

We will answer these questions in the following sections.

4.3 The Smallest β-Skeleton Containing the EMST

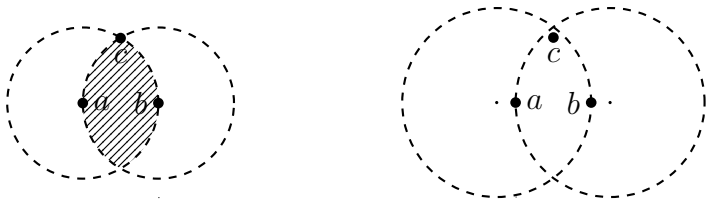
The family of β-skeletons for a set of d-dimensional points is defined as follows. For a given β, an edge exists between two points a and b if and only if (*iff*) the intersection of the two balls centered at $(\beta/2)a + (1 - \beta/2)b$ and $(1 - \beta/2)a + (\beta/2)b$, both with radius $\beta d(a, b)/2$, is empty. Larger values of β result in

fewer edges in the β-skeleton. 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 Figure 4.1a. 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*

$$d(a, b) \leq max\{d(a, c), d(b, c)\}, \forall c \neq a, b \tag{4.2}$$

The RNG is guaranteed to contain the EMST [45], 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 Figure 4.2a. 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 β-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 Figure 4.1: When $\beta = 2$ (Figure 4.1a), according to Inequality (4.2), there is an edge between every pair of points in the 2-skeleton of this dataset. For any $\beta > 2$ (Figure 4.1b), 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 β-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 β-skeleton that obviously cannot contain the EMST . Thus, it follows with the known result in Expression (4.1) that the RNG ($\beta = 2$) is the smallest β-skeleton that contains the EMST and, for this



(a) $\beta = 2$ (b) $\beta > 2$

Figure 4.1: β-skeletons

reason, we choose it as the basis for further analysis.

4.4 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 4 *$RNG^i = (V, E')$ 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 weighed 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* $EMST \subseteq RNG$ *[45] 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 Figure 4.2a. 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 (2.2) that* $mrd_i(a, b) = mrd_i(b, a)$ *(given that the underlying distance* d *is symmetric by assumption - (Chapter 2)). For the triangle inequality, we have to show that for all* a, b, c *in a dataset* $\boldsymbol{X}$*:*

$$mrd_i(a, c) \leq mrd_i(a, b) + mrd_i(b, c) \tag{4.3}$$

By assumption, the underling distance d *in the definition of* mrd_i *satisfies the triangle inequality:*

$$d(a, c) \leq d(a, b) + d(b, c) \tag{4.4}$$

Consider the definition of $mrd_i(a, c)$*, as expressed in Equation 4.5:*

$$mrd_i(a, c) = max\{c_i(a), c_i(b), d(a, b) \tag{4.5}$$

Thus, there are three cases in which (4.3) 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 (4.4) with max functions to obtain the following inequality:*

$$d(a,c) \leq max\{d(a,b), c_i(a), c_i(b)\} + max\{d(b,c), c_i(b), c_i(c)\} \tag{4.6}$$

Note that the terms on the right-hand side of Inequality 4.6 are equivalent to $mrd_i(a,b)$ *and* $mrd_i(b,c)$, *which allows us to rewrite Inequality 4.6 as follows:*

$$d(a,c) \leq mrd_i(a,b) + mrd_i(b,c)$$

Hence, according to the assumption in this case that $mrd_i(a,c) = d(a,c)$, *we can show that the triangle inequality holds:*

$$mrd_i(a,c) \leq mrd_i(a,b) + mrd_i(b,c)$$

Since mrd_i *satisfies symmetry and triangle inequality, it follows from [45] that* RNG_i^i *contains the MST of* G_i.

4.5 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*

Figure 4.2: Illustration for proofs of Theorems 1 and 2

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 Figure 4.2. More formally:

For $RNG^{k_{max}}$ (Figure 4.2a) both *of the following inequalities must be satisfied so that $c \in lune(a,b)$.*

$$mrd_{k_{max}}(a,b) > mrd_{k_{max}}(a,c) \tag{4.7}$$

$$mrd_{k_{max}}(a,b) > mrd_{k_{max}}(b,c) \tag{4.8}$$

For RNG^j (Figure 4.2b) at least one *of the following inequalities must be satisfied so that $c \notin lune(a,b)$.*

$$mrd_j(a,b) \leq mrd_j(a,c) \tag{4.9}$$

$$mrd_j(a,b) \leq mrd_j(b,c) \tag{4.10}$$

Using the definition of $mrd_{k_{max}}$, we can rewrite Inequalities (4.7) *and* (4.8) *as follows:*

$$\begin{aligned} &max\{c_{k_{max}}(a), c_{k_{max}}(b), d(a,b)\} > \\ &max\{c_{k_{max}}(a), c_{k_{max}}(c), d(a,c)\} \end{aligned} \tag{4.11}$$

$$\begin{aligned} &max\{c_{k_{max}}(a), c_{k_{max}}(b), d(a,b)\} > \\ &max\{c_{k_{max}}(b), c_{k_{max}}(c), d(b,c)\} \end{aligned} \tag{4.12}$$

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 (4.11) *and* (4.12) *can evaluate to. If*

it evaluates to one of the core distances, we get an immediate contradiction with one of the equations (4.11) *and* (4.12)*: In case* $max\{c_{k_{max}}(a), c_{k_{max}}(b), d(a,b)\} = c_{k_{max}}(a)$*, we get from Inequality* (4.11) *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* (4.12) *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.*

$$d(a,b) > c_{k_{max}}(a) \tag{4.13}$$

$$d(a,b) > c_{k_{max}}(b) \tag{4.14}$$

$$d(a,b) > c_{k_{max}}(c) \tag{4.15}$$

$$d(a,b) > d(a,c) \tag{4.16}$$

$$d(a,b) > d(b,c) \tag{4.17}$$

We also know that at least one of the Inequalities (4.9) *and* (4.10) *must hold, under our assumption that* $c \notin lune(a,b)$ *for distance* mrd_j*. We can rewrite* (4.9)*, using the definition of* mrd_j *as follows:*

$$\begin{aligned} &max\{c_j(a), c_j(c), d(a,c)\} \geq \\ &max\{c_j(a), c_j(b), d(a,b)\} \end{aligned} \tag{4.18}$$

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* (4.18) *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 (4.18) *yields the following.*

$$c_j(a) \geq d(a,b) \tag{4.19}$$

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 (4.19).

$$c_{k_{max}}(a) \geq d(a,b) \tag{4.20}$$

This contradicts Inequality (4.13)*!*

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

Analogously to the previous case, from (4.18) *we get* (4.21)*, and then from $c_{k_{max}}(c) \geq c_j(c)$ we get* (4.22)*, which contradicts Inequality* (4.15)*!*

$$c_j(c) \geq d(a,b) \tag{4.21}$$

$$c_{k_{max}}(c) \geq d(a,b) \tag{4.22}$$

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

In this case, we get from Inequality (4.18) *that $d(a,c) \geq d(a,b)$, which is a contradiction to Inequality* (4.16)*!*

This proves that Inequality (4.9) *cannot hold under our assumption. We can prove analogously the same result for Inequality* (4.10)*, 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, $MST(G_i) \subseteq RNG_i^{k_{max}}$.*

4.6 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 [26].

We adopt the approach in [1] — 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. Based on that decomposition, we find the *smallest* collection of pairs of *well-separated* subsets (A, B) such that, for any two points $p, q \in \mathbf{X}$, there is exactly one pair (A, B) in the *well-separated pair decomposition* where $p \in A$ and $q \in B$ (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:

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

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 *well-separated pairs*. This happens due to a stronger requirement on separation between sets of points. For instance, if two sets A and B are *not* well-separated w.r.t. s, one must look at *all* the pairs involving subsets of A and subsets of B to find pairs that are *well-separated* w.r.t. s. Therefore, for large values of s, the pairs of sets that are *well-separated* have to be either very far away from each other or have very small diameters. Thus, most of the pairs that are *well-separated* for large values of s are the ones containing singleton sets (with *diameter* zero), which could be as many as $\mathcal{O}(n^2)$. For $0 < s < 1$, there is no guarantee that the resulting graph will contain all the RNG edges and, consequently, the MST edges. Figure 4.3 shows two sets A and B, whose distance between their enclosing balls is smaller than the diameters of both balls. In this example, the closest point to a_3 in B is b_2, and the closest point to b_2 in A is a_2. According to the graph construction from the *well-separated pairs* (discussed next in the second step), the edge between a_3 and b_2 is not included in the resulting graph, even though a_3 and b_2 are *relative neighbors*. On the other hand, when a pair of sets (A, B) is *well-separated* w.r.t. $s \geq 1$, the points in one set are *necessarily* closer to each other than to the points in the other set. Thus, if two points $a \in A$ and

$b \in B$ are the closest pair of points from different sets, no other point in B can be closer to a than b, and vice versa. Consequently, no point in A or B can be closer to both a and b than they are to each other, which means that a and b are potentially *relative neighbors*. Therefore, any value of $s \geq 1$ is enough to guarantee that pairs of points that are potentially *relative neighbors* are connected with an edge in the graph constructed from the *well-separated pair decomposition*. Hence, to have as few edges as possible and guarantee the correctness of the resulting graph, we adopt $s = 1$.

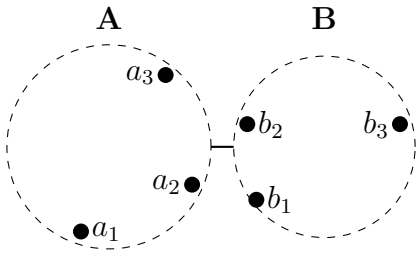


Figure 4.3: Well-Separated Pair Decomposition ($0 < s < 1$)

In the second step, a supergraph of the RNG, which we will call RNG**, is constructed. This graph is built with edges between pairs of *well-separated sets* and connects points that are potentially *relative neighbors*. 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 Figure 4.4, a_3 and b_3 are SBCN and thus the edge (a_3, b_3) is part of the RNG**. This step is based on the fact that when two points are *relative neighbors*, there is no other point in the data that is closer to both of them than they are to each other. Thus, connecting the SBCN between sets that are *well-separated* w.r.t. $s = 1$ results in a graph that contains all the edges in the RNG, but also contains many more edges between points that are not *relative neighbors*.

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 filtering strategy based on information that must be computed for HDBSCAN* to determine *core-distances*, 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.*, k-NN 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. This 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,

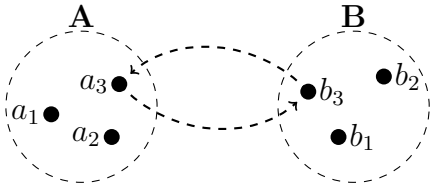


Figure 4.4: Symmetric Bichromatic Closest Neighbor (SBCN)

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

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

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

The pseudo-code for the overall strategy is shown in Algorithm 1. It takes as input a dataset **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 (SBCN) 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.

4.7 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 (SBCN) 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 on the complete graph.

4.8 Experimental Evaluation

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 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 4.6, 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 Kd-Tree index structure, adapted from [47]. The experiments were performed in a virtual machine with 64GB 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 [20], 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 4.1 shows these values and indicates in bold the default value for each variable when other variables are varied.

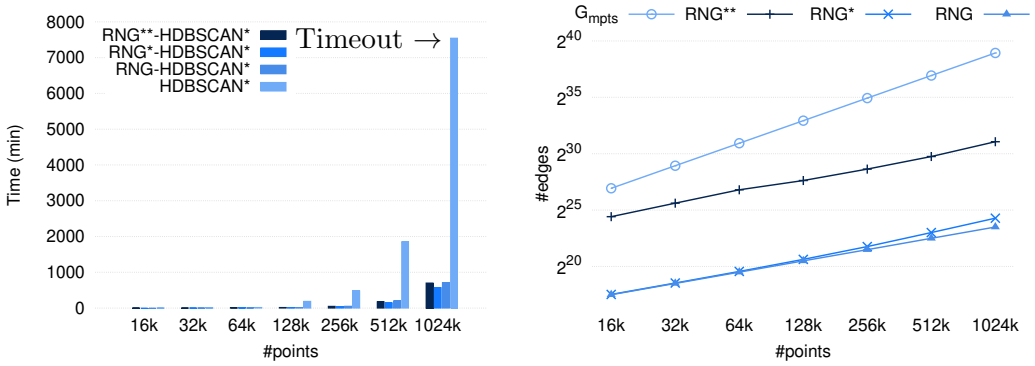
The datasets used to properly assess the efficiency of our method with regard to the effects of a specific variable were generated by varying only that variable, while the others were kept at their default values. For instance, in order to evaluate how our strategy behaves with regard to different dataset 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

Table 4.1: 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

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.

4.8.1 Effect of Dataset Size



(a) #points vs. Runtime (b) #points vs. RNG size

Figure 4.5: Runtime and RNG size as a function of dataset size.

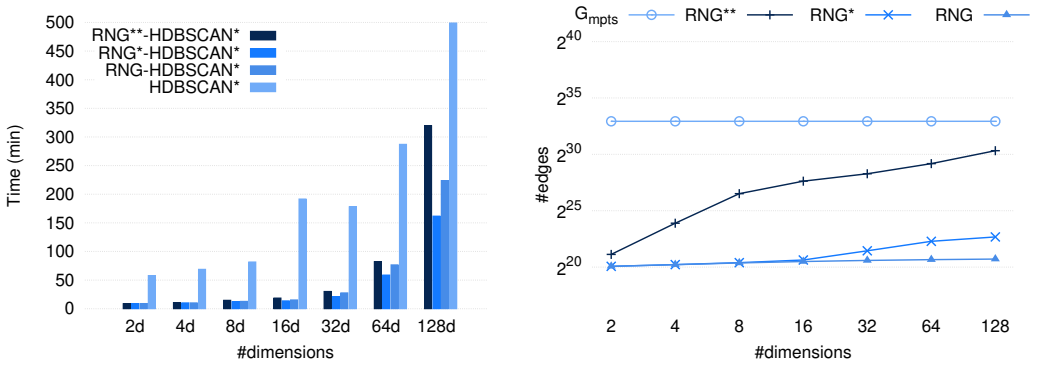
Figure 4.5a shows the total runtime as a function of the dataset size with default values for the remaining variables. As expected, the runtime tends to increase for all methods as the number of points increases. For datasets up to 64k points, all strategies have similar performances, but as the datasets 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 [1].

[1]The runs on this experiment were interrupted after 7500 minutes ($\approx$ 5 days), as the

Figure 4.5b 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.

The sizes of RNG and RNG** are quite different, yet their running times are quite similar (see Figure 4.5a), 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.

4.8.2 Effect of Dimensionality



(a) #dimensions vs. Runtime (b) #dimensions vs. RNG size

Figure 4.6: Runtime and RNG size as a function of dataset dimensionality.

Figure 4.6a 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."

observed performance was already enough for comparison purposes

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 Figure 4.6b. 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], [26].

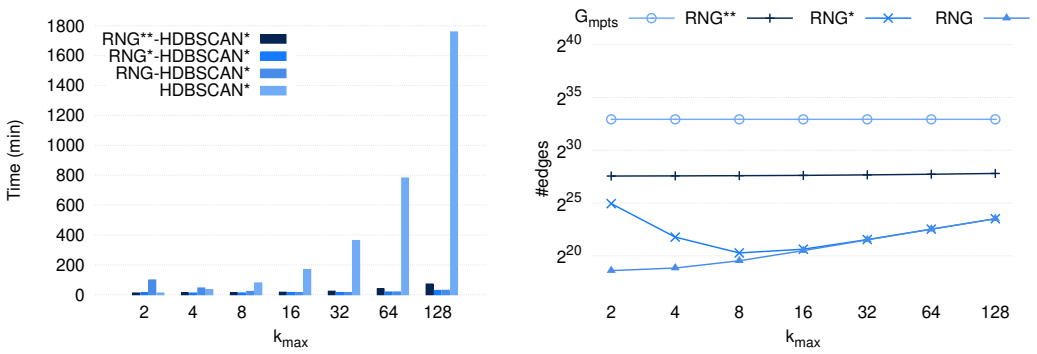
The number of edges of RNG* increases only slightly as the dimensionality increases, which shows that the pruning strategy using only the pre-computed k-nearest neighbors (16, as $k_{max} = 16$ in this experiment) stays quite effective, even in the 128-dimensional datasets, resulting in the best runtime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

4.8.3 Effect of Upper Limit k_{max}

Figure 4.7a and Table 4.2 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 RNG**-HDBSCAN* 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 Figure 4.7b.

RNG-HDBSCAN* shows a slightly higher runtime for $mpts = 2$, which



(a) k_{max} vs. Runtime

(b) k_{max} vs. RNG size

Figure 4.7: Runtime and RNG size as a function of k_{max}.

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 Figure 4.7b. 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 $k_{max} \geq 16$.

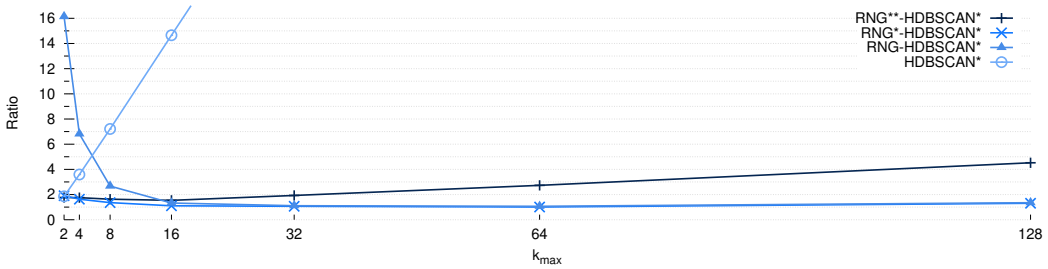


Figure 4.8: Ratio: runtime to compute k_{max} MSTs/hierarchies divided by the runtime to compute a single MST/hierarchy.

Table 4.2: 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	1759	72	**29**	30

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. Figure 4.8 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.

4.9 Conclusion

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

SCAN* 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.

Chapter 5

MustaCHE: Multiple Clustering Hierarchies Explorer

5.1 Introduction

RNG-HDBSCAN* [11], [36], presented in Chapter 4, is an efficient strategy for computing multiple HDBSCAN* clustering hierarchies w.r.t. a range of parameter values, and is able to compute over a hundred clustering hierarchies at a very low computational cost – equivalent to running HDBSCAN* twice – while producing results identical to the ones produced by the original HDBSCAN* formulation. These developments have made possible for users to have access to a potentially large collection of clustering hierarchies for analysis and exploration. However, finding the "best" value of *mpts* remains an open problem, and analyzing a very large number of clustering hierarchies, and *learning* from them, is still practically challenging. While close values of *mpts* are likely to result in similar hierarchies, different *ranges* of *mpts* values may produce *significantly* different hierarchies. Therefore, we address in this chapter the following non-trivial questions: for a given dataset, (1) *how many of these ranges exist?*, (2) *how does one identify these ranges?* and (3) *how do the hierarchies in each of these ranges look like?*.

To address these questions, we propose MustaCHE[1], a tool that leverages

[1] `https://github.com/antoniocavalcante/mustache`

the main results of RNG-HDBSCAN* and allows a visual and interactive analysis and exploration of multiple clustering hierarchies, thus helping users to better understand their data and its cluster structures. Users can then explore hierarchies individually and, at the same time, see how they compare to the other hierarchies. The simultaneous visualization of multiple clustering hierarchies provided by MustaCHE makes it feasible (and easy) for a user to gain a deeper understanding of the data and how its cluster structure reveals itself under different parameter settings.

Next we present the different visualizations available in MustaCHE and discuss what a user can learn from each of them, followed by a description of a demonstration scenario that illustrates MustaCHE's usability. We also present a couple of small, real-world case studies to demonstrate the effectiveness of our visualization tools for exploratory cluster analysis with multiple clustering hierarchies.

5.2 Visualizations in MustaCHE

Given a set of HDBSCAN* hierarchies of a given dataset for different *mpts* values, efficiently pre-computed using RNG-HDBSCAN*, MustaCHE offers a set of visualizations that simplify and aid in the analysis of those hierarchies. Its main overall goals are to assist the user to (1) (visually) find "good" values for *mpts* and (2) to understand which cluster structures are detectable in the data for different parameter values. In the following, we discuss each of the visualizations available in MustaCHE and the motivations behind them.

5.2.1 Similarity Matrix

A common representation for a cluster hierarchy, which depends on a given value of *mpts*, is a dendrogram. However, in order to gain a broad understanding of how the parameter *mpts* affects the hierarchical organization of the data

it is not necessary to look at actual dendrograms; but rather to identify the ranges of *mpts* values that produce similar hierarchies.

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 [31], 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 number of points in the 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 these values in a color scale, where lighter colors indicate a higher similarity, makes it possible to visually identify the *mpts* values that result in similar hierarchies. For instance, Figure 5.1 shows the pairwise HAI values for 50 hierarchies from a sample dataset w.r.t. $mpts \in [1, 50]$. Note that any two hierarchies w.r.t. $mpts \in [1, 15]$ are highly similar.

Likewise, any two hierarchies w.r.t. $mpts \in [16, 50]$ are also similar among themselves, but to a lesser degree than in the previous case. Furthermore one can observe that within both of these ranges of *mpts* values, there are smaller sub-ranges for which the similarity is higher than in the larger one. Finally,

any two hierarchies with *mpts* outside those those two ranges are dissimilar.

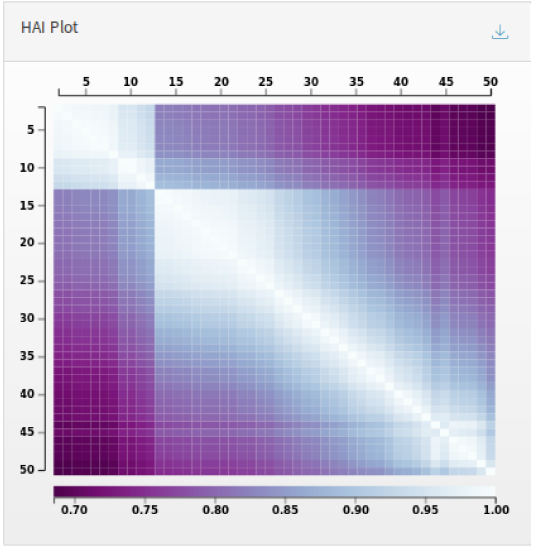


Figure 5.1: Pairwise HAI Similarity

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 [3]. 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.

5.2.2 Meta-Clustering Dendrogram

While the similarity matrix plot presents an overview of the similarity among hierarchies for a range of *mpts* values, extracting the exact ranges that produce similar hierarchies only from this plot can still be difficult. 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 plot. Also, there might be cases where two hierarchies for non-consecutive values of *mpts* have a higher similarity than for consecutive values. In order to deal with such cases, we propose to cluster these hierarchies while at the same time allowing the user to set the similarity threshold needed to consider two hierarchies as part of the same group.

To combine these two requirements, we do a meta-clustering process using the HAI values to construct a *clustering hierarchy of clustering hierarchies* with HDBSCAN*.[2] This meta-hierarchy can be visualized as a dendrogram, where the user can see similar hierarchies next to each other and is also able to distinguish similarity levels more clearly.

Figure 5.2 shows the dendrogram of the 50 hierarchies computed from the HAI values presented in the example in Figure 5.1. The meta-hierarchy 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 as 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. Note that it highlights four main (meta) clusters selected by the automatic extraction method (FOSC) [8] provided by HDBSCAN*, which is based on the stability of each

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

cluster. At this point, instead of having to examine 50 *"different"* hierarchies, the user knows that there are 4 main different hierarchical organizations of the data. Notice that there are also outliers (plotted in light gray) that might represent interesting results as well, since they are unique in the sense that they haven't been included into any of the found clusters. In Section 5.3 we discuss how these outliers can be interactively explored and how the user can select different partitionings from meta-hierarchies.

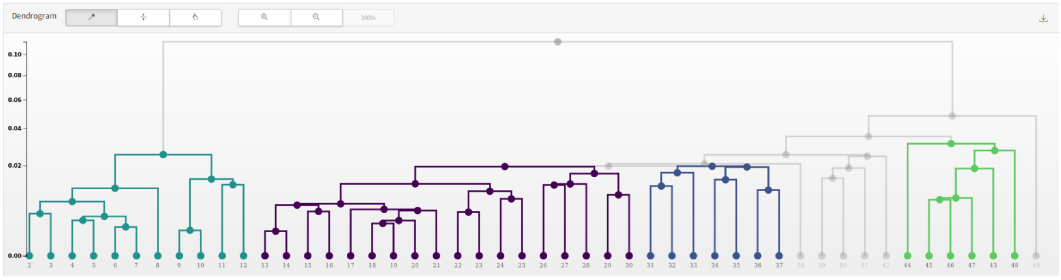


Figure 5.2: Hierarchy of hierarchies (meta-clustering) - FOSC

In addition to FOSC, MustaCHE provides two other methods for extracting clusters from the meta-hierarchy. In the first, the meta-clusters are selected through a horizontal cut in the dendrogram. Users will be able to move the threshold bar up and down along the y axis in order to choose the cutting point, or input the value where the bar should be placed. The clusters selected with this method will be the ones that will appear below the threshold bar. Note that in Figure 5.3, the threshold bar method is selected, and the bar is setting the threshold as 0.03, which results in three meta-clusters. The last method

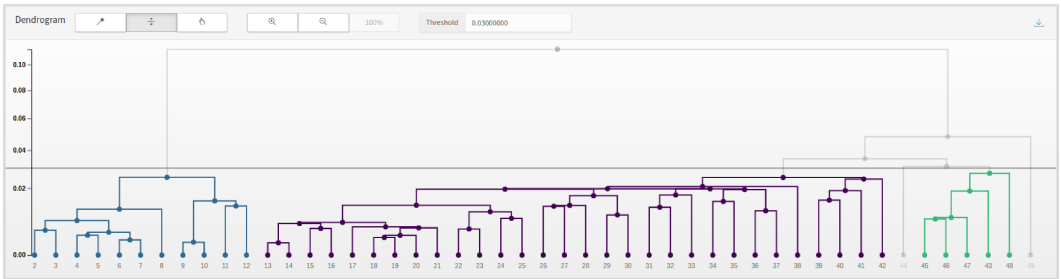


Figure 5.3: Hierarchy of hierarchy (meta-clustering) - Threshold Bar

allows users to manually select the meta-clusters in the dendrogram, *i.e.*, they will be able to click the meta-clusters they want to investigate, as shown in Figure 5.4. This feature gives users flexibility to try different arbitrary selections.

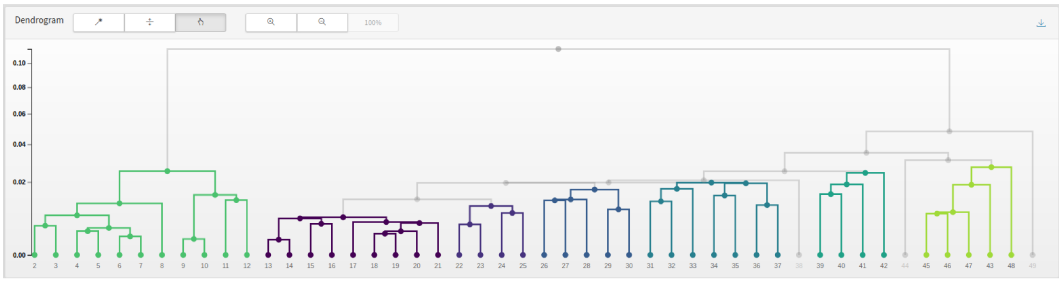


Figure 5.4: Hierarchy of hierarchy (meta-clustering) - Manual Selection

5.2.3 Reachability Plots

At this point, after perusing the visualizations previously described, the user has a better understanding of how the parameter *mpts* affects the similarity of the hierarchies and should be able to identify the ranges of values that produce *"significantly"* different hierarchies. The next step now is to examine how the hierarchies from each meta-cluster look like. Since the hierarchies in each meta-cluster are similar to each other, there is no need to examine all of them. In this kind of visualization we select 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.

Even though we choose to visualize our meta-hierarchy as a dendrogram, dendrograms are more suitable for visualization when the number of leaf nodes (*i.e.* elements being clustered) is "relatively small". Fortunately, the number of leaf nodes in the meta-hierarchy corresponds to the number of clustering hierarchies (or, alternatively, the number of *mpts* values) under analysis, which in practice is rarely larger than a few tens in common real datasets. However,

when we want to inspect individual clustering hierarchies, such as the meta-cluster medoids or outliers in the meta-hierarchy, the number of leaf nodes corresponds to the number of data points (*i.e.*, the size of the dataset), which may be too large to be properly visualized as a dendrogram.

When the number of points is large, the use of *reachability plots* [2] is more appropriate to visualize density-based clustering hierarchies. In a nutshell, reachability plots are bar plots where each bar corresponds to a point in the dataset, and they are sorted in such a way that points 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 point join the preceding points in the plot, so density-based clusters appear as "valleys" (or "dents") in the plot. Figure 5.5 displays four reachability plots, each of which corresponds to the medoid of one of the four meta-clusters found in Figure 5.2. For instance, the reachability plot for $mpts = 7$ indicates that the dataset might be decomposed into six main clusters, whereas the hierarchy for $mpts = 34$ shows only two main clusters.

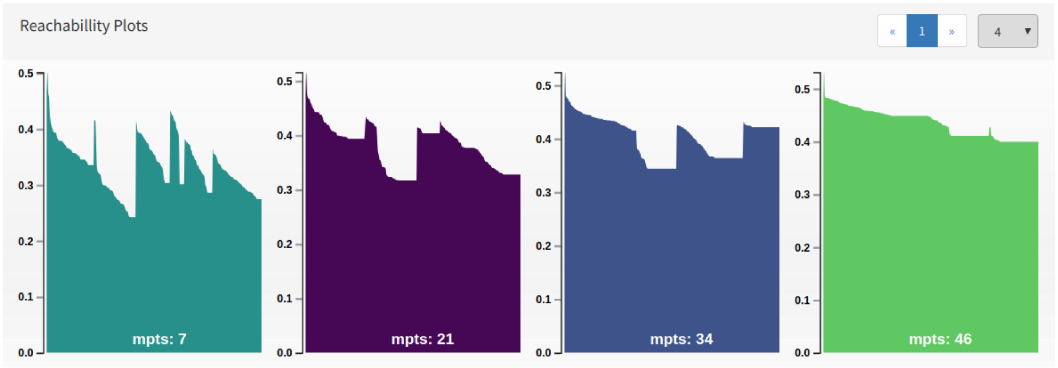


Figure 5.5: Reachability Plots

The combined visualization of the reachability plots for the meta-cluster medoids allows users to easily compare the different hierarchical organizations of the data across multiple *mpts* values. MustaCHE also allows the investigation of reachability plots in more details, by coloring the selected plot according

to the FOSC partitioning for that corresponding hierarchy. This type of visualization is illustrated in Figure 5.6, which shows the reachability plot for $mpts = 7$ with the clustering represented by different colors, and black representing noise. This visualization shows which points belong to which clusters in the partitioning performed by FOSC/HDBSCAN*, and which points are noise. We note, this visualization can be produced for any value of $mpts$, *i.e.*, not necessarily only the ones corresponding to meta-clusters medoids (Figure 5.5), but also other elements of the meta-hierarchy (Figure 5.2), including outliers.

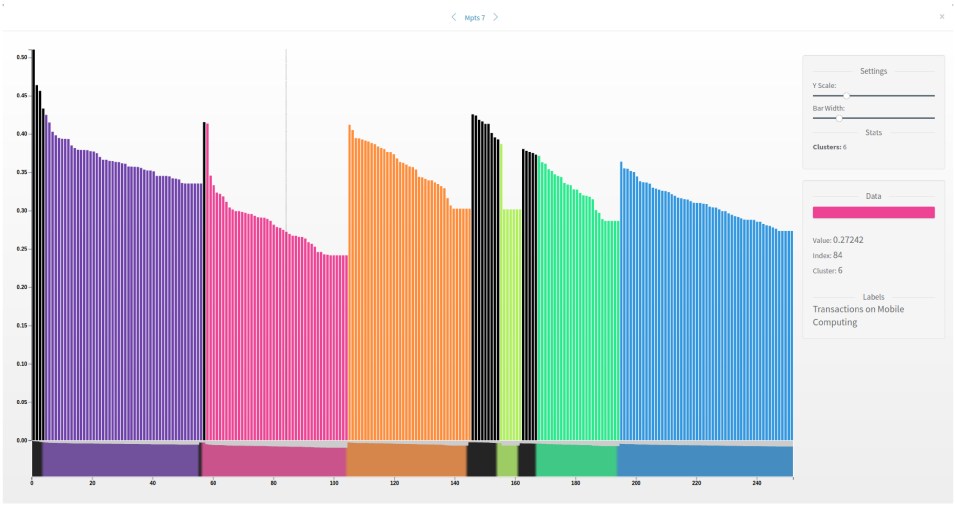


Figure 5.6: Reachability Plot for $mpts = 7$

At this stage, the user can see which points compose each of the clusters to understand the cluster structures in greater detail. By hovering the mouse over the bars, MustaCHE shows on the bottom right panel the index in the dataset correspondent to that point, the cluster to which it belongs, and any labels associated to that point.

5.2.4 Circle Packing

In addition to reachability plots, users can visualize clustering hierarchies in different ways according to their goals and according to the aspect of the hi-

erarchies they want to explore. While reachabilty plots are a great alternative for visualizing the overall hierarchical structure of the nested density-based clusters in the data, these plots do *not* offer an explicit representation of the clusters in the hierarchy. As the elements in the plot (the bars) correspond to data points, it might be challenging for users to identify with more precision where clusters begin and end. Moreover, when performing an analysis, users might want inspect the stability of clusters at different levels or how many points there are in each cluster, which is unpractical with just reachability plots.

In a more cluster-oriented visualization, each element in the plot corresponds to a cluster, and nested clusters appear nested in the plot, implying a hierarchical structure. Figure 5.7 shows a *circle packing* plot that represents the hierarchical organization of the clusters in the data, where each circle corresponds to a cluster, their sizes are proportional to the number of points in the cluster, and their colors represent their stability values – the *warmer* the color of a cluster, the higher its stability.

When clustering hierarchies are represented in this fashion, one is able to have access to other information about the clusters that are not practical to access or even feasible with reachability plots. When a user hovers the mouse over a circle representing a cluster in the plot, MustaCHE shows a *tooltip* containing (1) the cluster's identifier; (2) the number of points in that cluster; (3) its birth and death levels; and (4) its stability value, as shown in Figure 5.7. In order to allow the exploration of clusters at deeper levels in the hierarchy, MustaCHE provides a zooming feature that makes it easy for users to navigate the hierarchical structure by just clicking the clusters they want to inspect closely.

Last, users are also able to highlight the clusters selected by FOSC according to their stability/relative excess of mass. By clicking the button labeled

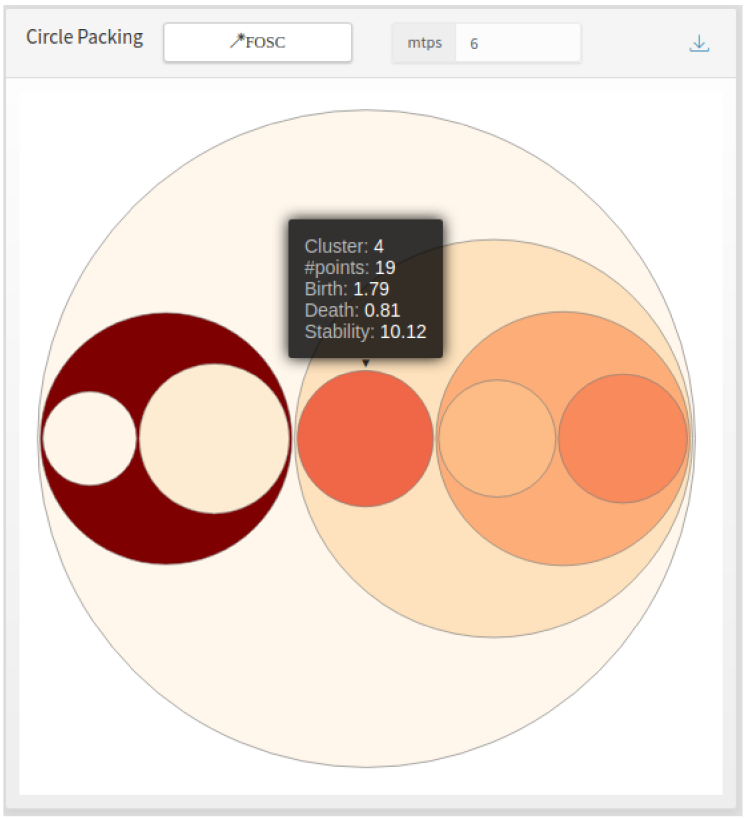


Figure 5.7: Circle Packing

FOSC at the top of the plot, MustaCHE shows which clusters have a higher combined stability as computed by FOSC.

5.2.5 Condensed Cluster Trees

As another option for a cluster-oriented visualization, users can choose to visualize their clustering hierarchies as *condensed cluster trees*. In this type of visualization, clusters are plotted according to the levels at which they exist in the clustering hierarchy. Therefore, if a cluster C appears at a level λ_1 in a clustering hierarchy and disappears (splits/dies) at a level λ_2, the representation of C in the plot reflects, at each intermediate level between λ_1 and λ_2, the configuration of C w.r.t. number of points. Figure 5.8 shows a *condensed cluster tree*, where the clusters correspond to the colored polygons, the colors represent the stability of that cluster – the lighter the color, the

higher the stability –, and the width of the cluster corresponds to the number of points in the cluster.

Figure 5.8: Condensed Cluster Tree

Note that, due to the representation of the clusters in the plot, it is possible to visualize how a cluster shrinks along the density levels in the hierarchy. Thus, users can have a comprehensive overview of the cluster structures w.r.t. hierarchical organization, size and stability.

Similarly to the *circle packing* plot, users can hover the mouse over clusters and access more detailed information about the cluster, as well as highlight the clusters with higher combined stability as found by FOSC. Additionally, both the *circle packing* and the *condensed cluster tree* visualizations can be considered more scalable w.r.t. the number of points in the data, as the elements in the plot correspond to the clusters rather than data points.

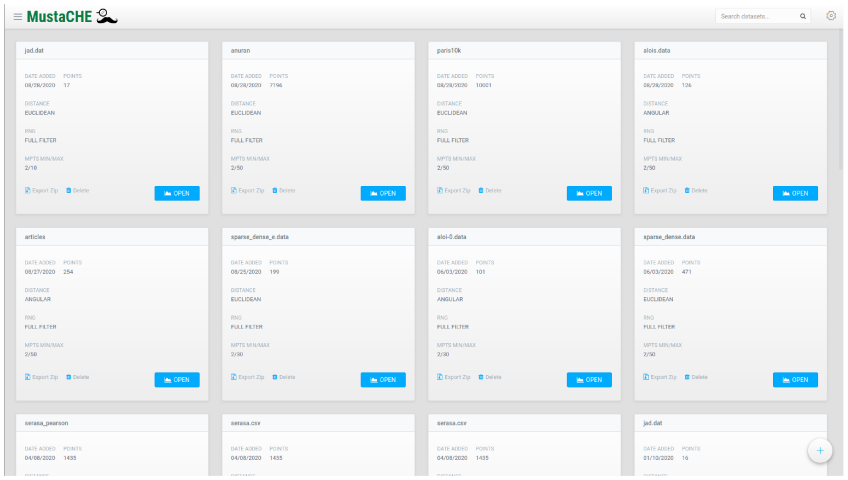
5.3 MustaCHE Overview

In this section, we provided an overview of how users can interact with MustaCHE. The first interaction is the selection of a dataset to be analyzed, or the upload of a new dataset along with the parameter values for the computation of clustering hierarchies with RNG-HDBSCAN*. Thus, when opening MustaCHE, users are directly taken to the *Datasets* screen, where all datasets available in MustaCHE are listed (Figure 5.9a). In order to submit a new dataset, users must click the + icon at the bottom right of the *Datasets* screen. Then, MustaCHE launches a pop-up window (Figure 5.9b) that allows users to submit their data and labels (if available). In this context, labels are merely references to the data points so users can easily identify them in the visualizations provided by MustaCHE. After uploading their data, users then move onto the next screen to enter the parameters settings for running RNG-HDBSCAN* (Figure 5.9c) – dataset name, distance function, RNG filter, range of *mpts* values, and the minimum cluster size [3].

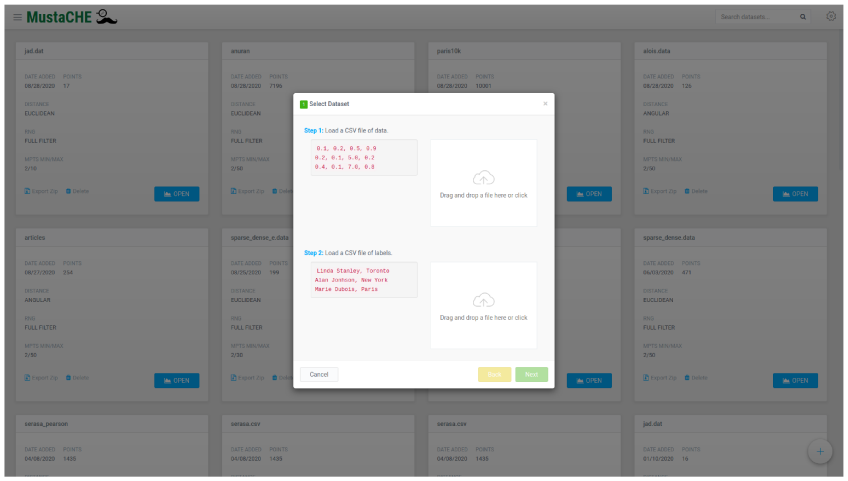
Then, when users click the *Run* button, MustaCHE launches the RNG-HDBSCAN* processing and the newly added dataset entry in the *Datasets* screen is updated according to the progress of the computation of the clustering hierarchies and the meta-clustering information. Once the processing is finished, users can then click the *Open* button and access the visualizations of the clustering hierarchies corresponding to that dataset.

In the main visualization screen (Figure 5.10), users have access to the HAI similarity matrix (Figure 5.10, part **b**), the meta-hierarchy of clustering hierarchies (Figure 5.10, part **c**), and the medoids of the meta-clusters identified in the meta-hierarchy (Figure 5.10, part **d**). When positioning the cursor

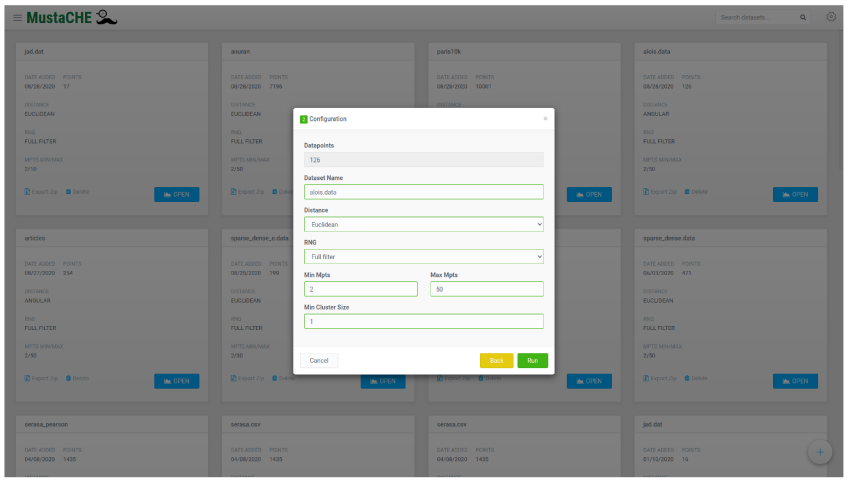
[3]HDBSCAN* receives a *minimum cluster size* parameter to decide whether a cluster is splitting into multiple clusters or simply shrinking. As a result, all clusters in the hierarchy have at least that many points.



(a) MustaCHE - Datasets



(b) MustaCHE - Input (1)



(c) MustaCHE - Input (2)

Figure 5.9: MustaCHE - datasets and input screens

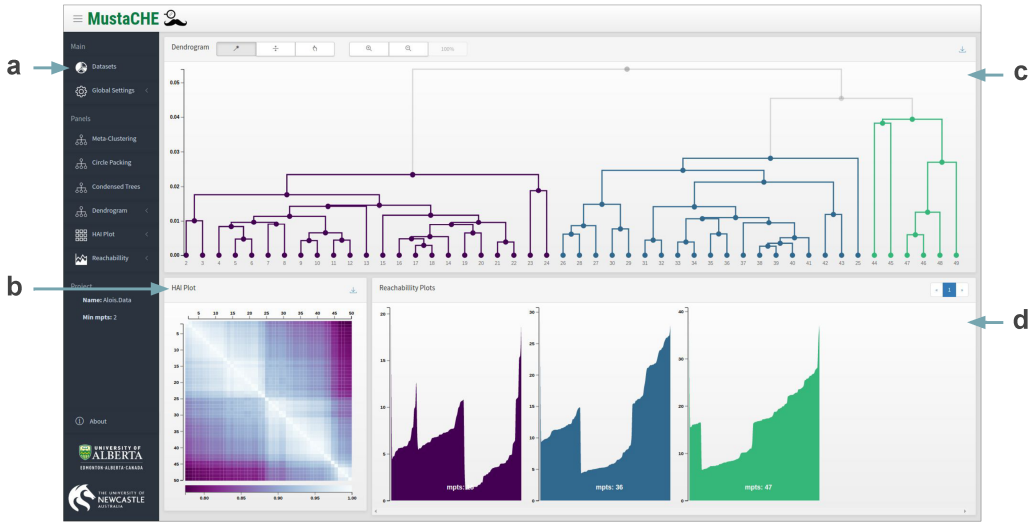


Figure 5.10: MustaCHE

over a cell of the HAI matrix plot, MustaCHE shows which values of *mpts* correspond to that cell and their similarity as computed by the HAI measure. Moreover, users are also able to change the method for selecting clusters from the meta-hierarchy dendrogram (FOSC, threshold bar, manual selection), and MustaCHE automatically reacts to the changes in the meta-clusters selection and updates the reachability plots accordingly (Figure 5.10, part **d**).

In order to inspect individual hierarchies, including outliers, in more details, users can select them either from the medoids (Figure 5.10, part **d**) or from the dendrogram (Figure 5.10, part **c**), and MustaCHE will show the detailed reachability plot colored according to its cluster partitioning (extracted by FOSC) in a pop-up window as shown in Figure 5.6.

In the sidebar menu, users can choose to change their visualization mode to *circle packing* or *condensed trees*. In the *circle packing* mode (Figure 5.11), users have access to a large version of the *circle packing* visualization (Figure 5.11, part **a**) that can be used to inspect any hierarchy for any value of *mpts* in more detail, as well as to a set of *circle packing* plots that represent the clustering hierarchies for all values of *mpts* in the range given as input (Figure

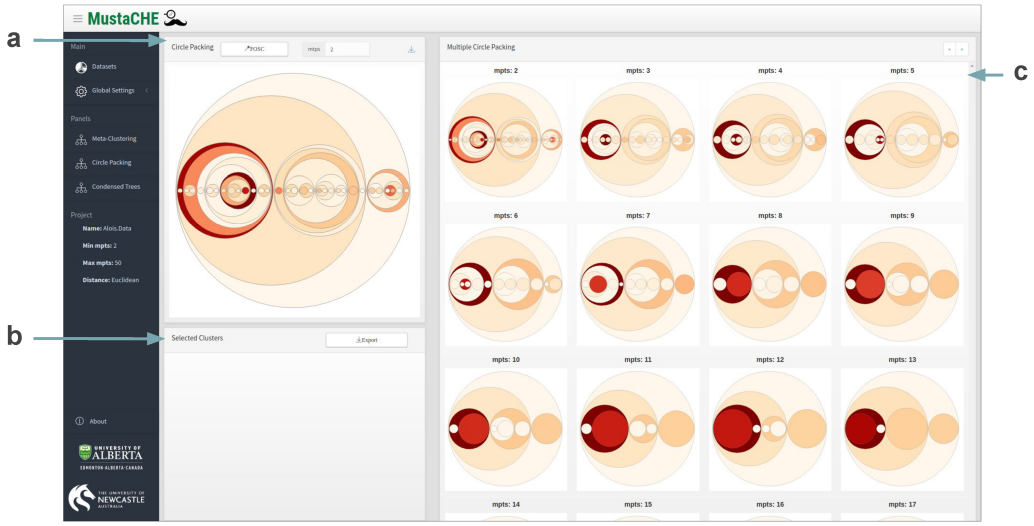


Figure 5.11: Circle Packing Screen

5.11, part **c**). Moreover, when users click the *FOSC* button (Figure 5.11, part **a**), MustaCHE highlights the clusters selected by FOSC in each of the plots, offering a broad view of the number of clusters selected, their size and stability in each hierarchy.

Similarly to the *circle packing* mode, the *condensed trees* mode also provides a larger version of the *condensed tree* plot (Figure 5.11, part **a**) for a detailed analysis, and a set of smaller plots (Figure 5.11, part **c**) corresponding to the clustering hierarchies computed w.r.t. the range of *mpts* values given as input.

In both the *circle packing* and *condensed tree* modes, users are able to select clusters they find interesting for further exploration. When users double click a cluster, MustaCHE adds that cluster to the list of selected clusters (Figure 5.11, part **b** and Figure 5.12, part **b**). As clustering is commonly used in early stages of exploratory data analysis, the selection of clusters performed with MustaCHE can be rather useful in later stages of the analysis. Thus, after making their selection, users are able to download the information about the selected clusters – partitioning of the points and reference to the clusters

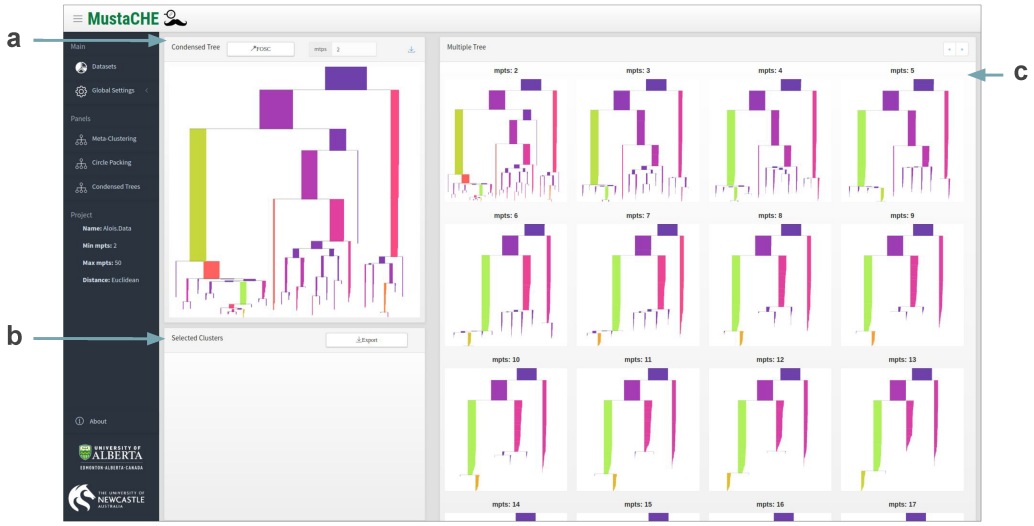


Figure 5.12: Condensed Trees Screen

selected by the user – by clicking the *Export* button.

5.4 Case Studies

Next, we present a couple of case studies to showcase how exploratory cluster analysis can be performed with the visualizations available in MustaCHE.

5.4.1 Case Study #1: Text Data

In this study case, we analyze hierarchies constructed for the dataset Articles-1442-5 [34] containing 253 documents extracted from journals of different fields (DNA Research, Transactions on Mobile Computing, Monthly Weather Review, British Food Journal and American Political Science 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.

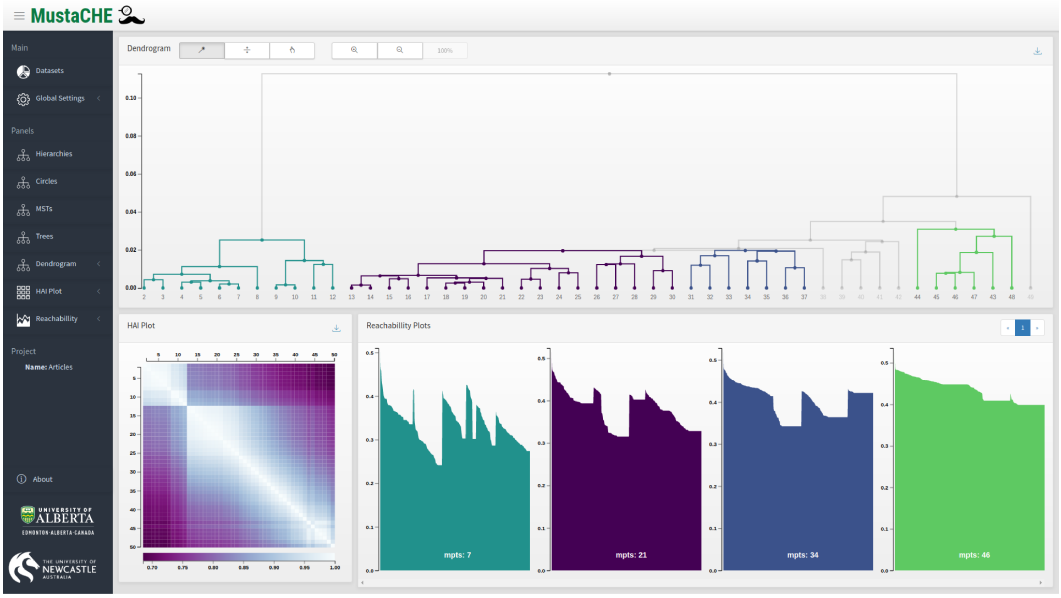
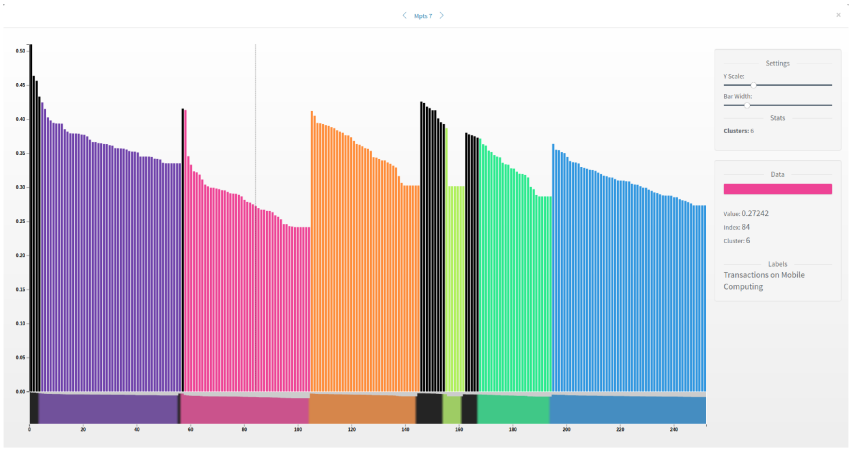


Figure 5.13: Journal documents ("Articles-1442-5") results.

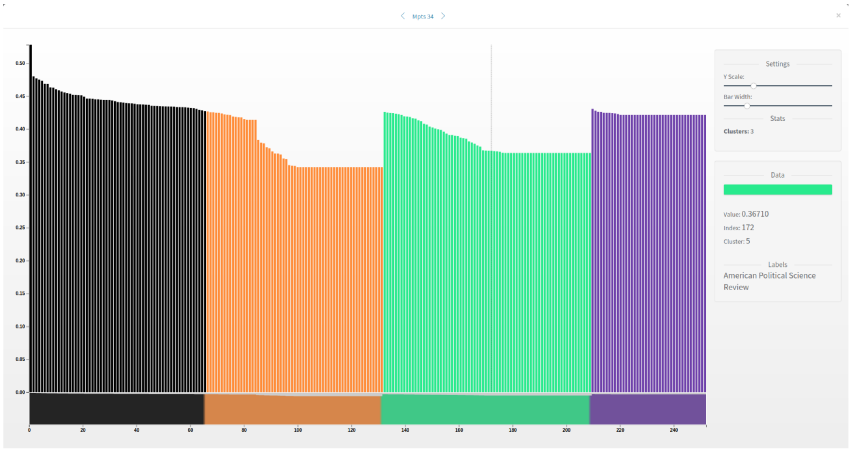
The HAI similarity matrix in Figure 5.13 shows that there are two main ranges of *mpts* values for which the hierarchies are more similar among themselves. Overall, the hierarchies in the larger range ($mpts \in [16, 50]$) have a weaker degree of similarity than the ones in the smaller range ($mpts \in [2, 15]$), although there are subranges of larger values within which similarity is more prominent, *e.g.* around [16, 27] and around [42, 49]. Notice that in a dataset with these many dimensions, selecting a meaningful value of *mpts* would be particularly challenging.

Figure 5.13 also shows the meta-hierarchy for this dataset, with meta-clusters (as detected by FOSC) highlighted in different colors. The reachability plots of each meta-cluster's medoid are shown at the bottom half of Figure 5.13. Figure 5.14 shows the detailed 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 c_4 (Brittish Food Journal) 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 ε. A relevant aspect that can be learned from this is that clusters c_1 (DNA Research), c_2 (Transactions on Mobile Computing) and c_5 (American Political Science) can still be detected for $mpts = 34$, whereas clusters c_3 (Monthly Weather Review) and c_4 (British Food Journal) become mostly noise at this level. This shows that clusters c_1, c_2 and c_5 are more stable than the others, as they persist across varied parameter ranges and hierarchies.



(a) $mpts = 7$



(b) $mpts = 34$

Figure 5.14: Selected plots for the "Articles-1442-5" dataset.

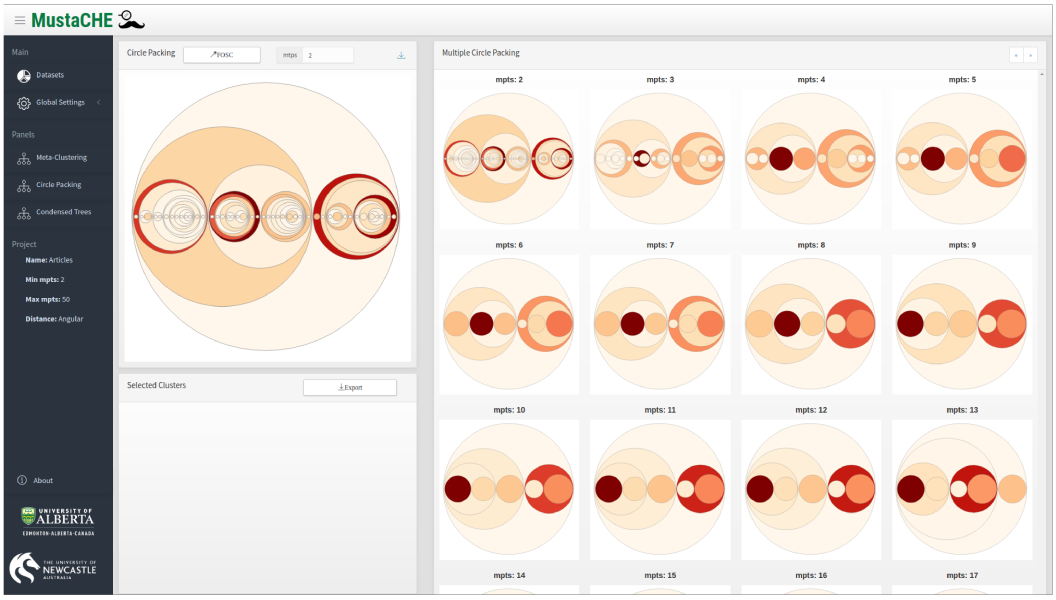


Figure 5.15: "Articles-1442-5" - Circle Packing

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 at the bottom of Figure 5.13 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.

Additionally, users are also able to visualize the hierarchical structures of the data with the *circle packing* (Figure 5.15) and *condensed cluster tree* (Figure 5.16) visualizations. One can see from both modes that the number of clusters in the clustering hierarchies for this dataset decreases for larger values of *mpts*. It is also possible to visualize how the stability of clusters behave w.r.t. the values of *mpts* used for constructing the clustering hierarchies, as well as the size of the clusters in each hierarchy.

These modes offer a different perspective on the analysis of multiple clustering hierarchies. Instead of looking at just a small set of *significantly different* hierarchies, users are able to visualize how the value of *mpts* affects the cluster structures found by HDBSCAN* in a more incremental way.

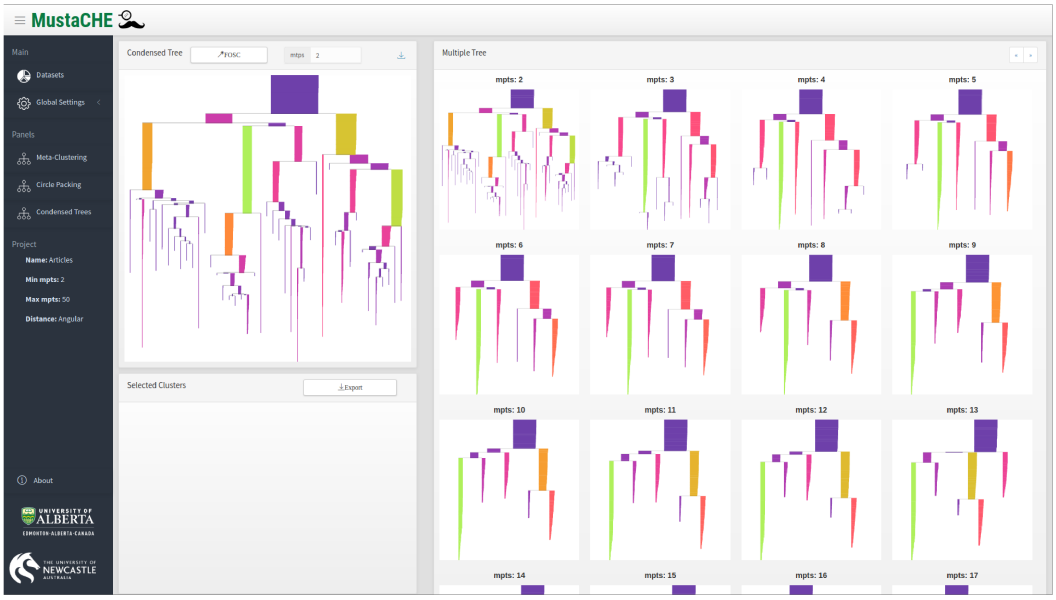


Figure 5.16: "Articles-1442-5" - Condensed Trees

5.4.2 Case Study #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 7000 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. The HAI similarity matrix in Figure 5.17 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, four more prominent meta-clusters are identified, as shown in the dendrogram in Figure 5.17.

Figure 5.18 shows the reachability plots for each medoid meta-cluster colored according to the partitioning found by FOSC. For $mpts = 4$, FOSC

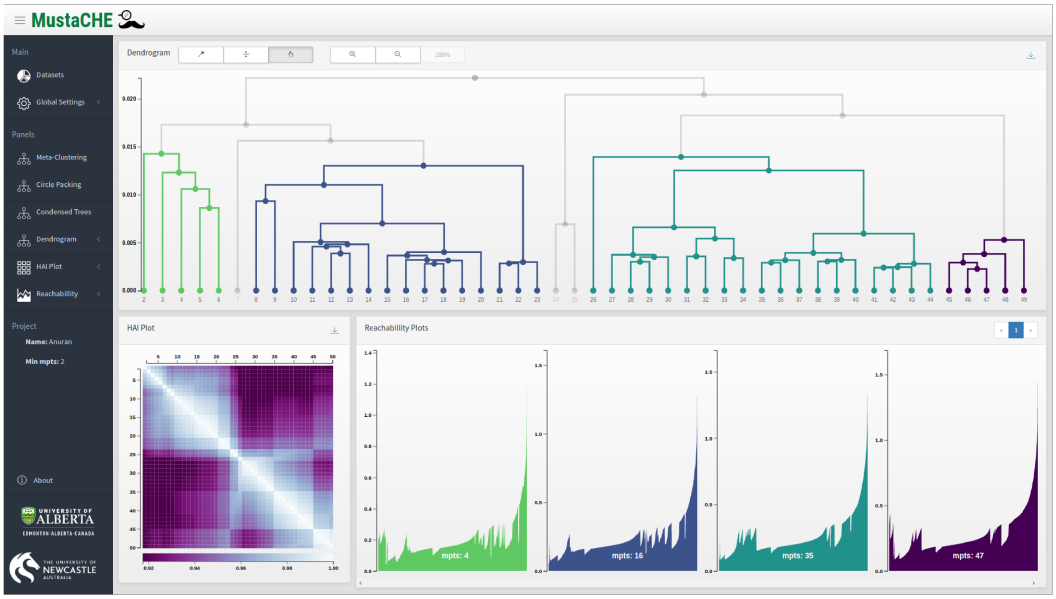
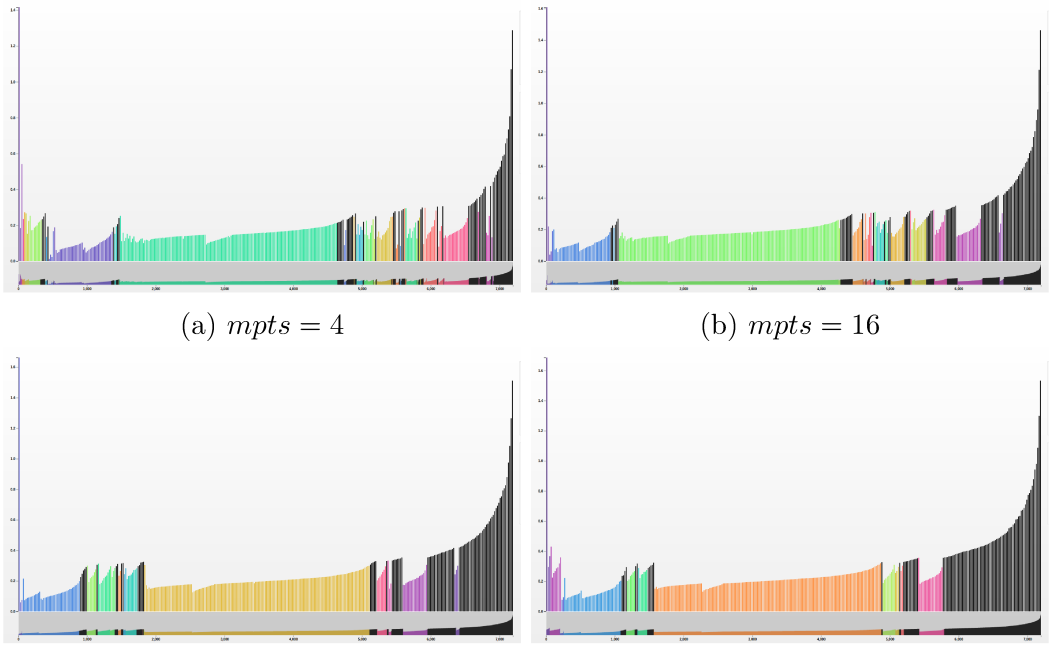


Figure 5.17: MustaCHE - Main Screen (Anuran)



(a) $mpts = 4$ (b) $mpts = 16$

(c) $mpts = 35$ (d) $mpts = 47$

Figure 5.18: Anuran - Reachability Plots

extracted 51 clusters. When compared to the true labels, we observed that clusters 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, 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 Figure 5.18 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.

When looking at the *circle packing* (Figure 5.19) and *condensed tree* (Figure 5.20) visualizations, one can observe how clusters split as values of *mpts* change. In this particular case, it is clear how part of the splits present a certain pattern – clusters split into a small component and a large component – that results in an unbalanced cluster tree. This can be interpreted as an indicator that the dataset has a great amount of noise, or that the *minimum cluster size* must be set higher to avoid these kinds of splits with very small clusters.

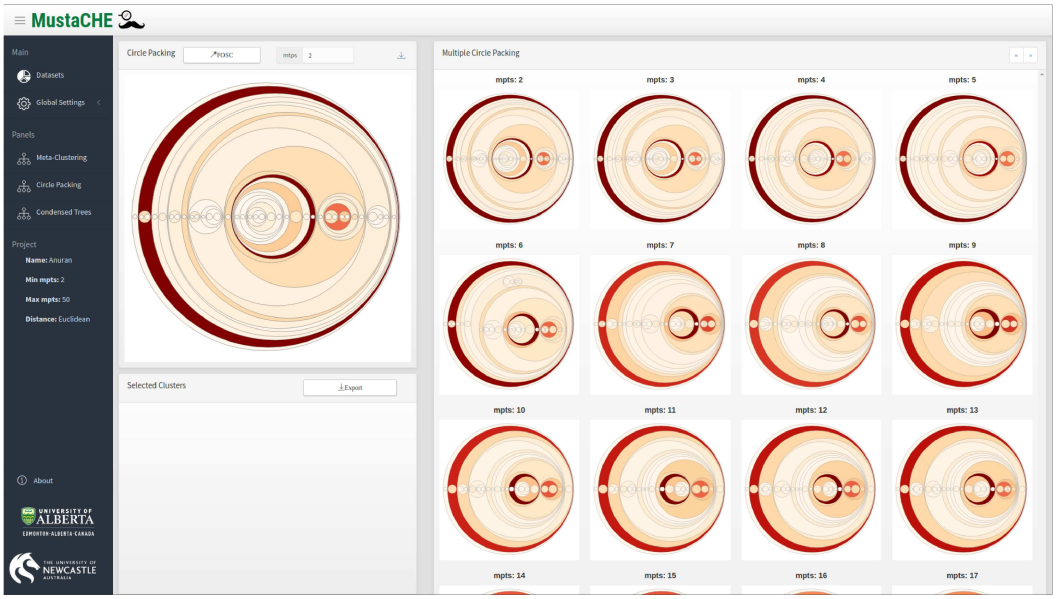


Figure 5.19: Anuran - Circle Packing

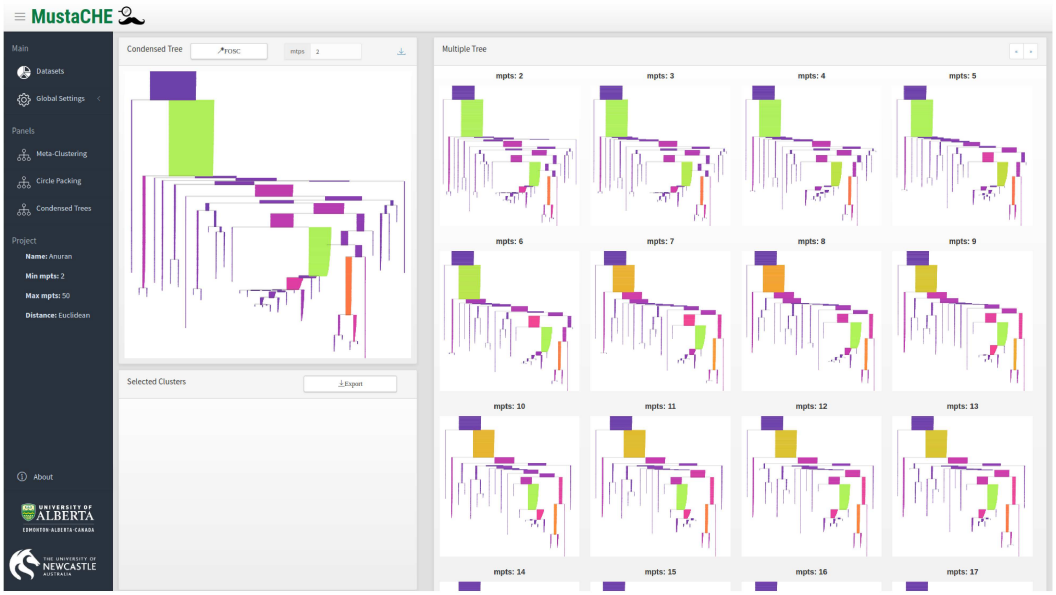


Figure 5.20: Anuran - Condensed Trees

5.5 Conclusion and Future Work

We have presented MustaCHE, a visualization tool that allows the analysis of multiple HDBSCAN* density-based clustering hierarchies in a visual and interactive way. The use of MustaCHE makes it easier for a user to have a deeper understanding of how the cluster structures in the data behave under different density levels. We have also presented case studies that showcase MustaCHE's capabilities and demonstrate how analyses can be performed through the visualizations available in the application.

As future work, we believe MustaCHE could be deployed as an end-to-end web-based "Clustering as a Service" where users can upload, cluster, visualize, analyze, archive or share (if appropriate) their data and analyses. In terms of performance, MustaCHE could also benefit from approximate solutions where results could be available for analysis faster and refined later. Regarding functionality, another direction for future work in MustaCHE is the addition of visualizations for the data (2d and 3d plots, geolocated data on maps, word clouds, etc.) being clustered. This would make it possible for users to perform more comprehensive analyses without having to recur to other tools. On the same note, the support for semi-supervised algorithms would enrich MustaCHE's capabilities, allowing users to interactive explore labels and assess results in real-time.

Chapter 6

Multiple Values of $mpts$ in a Clustering Hierarchy

6.1 Introduction

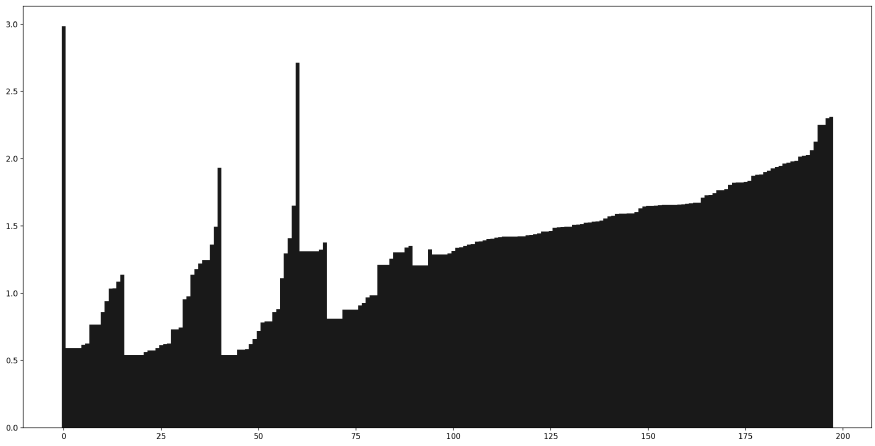
In non-parametric density estimation, one is usually faced with the task of choosing the amount of smoothing in the density estimates with the goal of getting as close as possible to what is believed to be the true density of the data. In density-based clustering, the main interest is the identification of high-density regions of the space (clusters) separated by low-density regions. However, different amounts of smoothing might reveal different clustering structures. Also, the presence of noise in the data might prevent cluster structures from being detected, or the noise points themselves might be mistakenly interpreted as clusters depending on how smooth the density estimates are. Hence, a more adaptive strategy must be applied to cope with the underlying characteristics of each cluster or region of the space.

HDBSCAN* works with a non-normalized k-NN density estimator, where the density estimate at a point p corresponds to the inverse of the distance from p to its k-th nearest neighbor in the data set. The smoothing parameter $mpts$ in HDBSCAN* acts as k and defines the minimum number of neighbors that a point in space must have in its neighborhood in order to be considered *dense*.

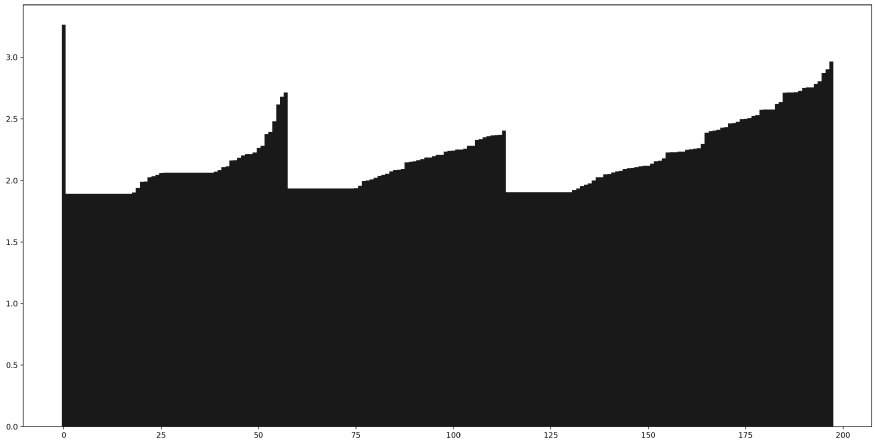
Even though the use of a k-NN density estimator already makes the density estimates more adaptive w.r.t. the local neighborhood of a point, there are still situations where different values of $mpts$ reveal different structures. Figure 6.1 shows the reachability plot representation of two hierarchies generated from the same dataset w.r.t. $mpts = 5$ and $mpts = 18$, respectively. The dataset contains 200 points in a 6-dimensional space, forming a total of 5 clusters. Three of these clusters are well defined and well separated, while the other two clusters have a certain amount of noise between them, which makes them hard to separate from each other at low values of $mpts$. We can see this clearly in Figure 6.1a, which shows the three small clusters and a large one (with some spurious sub-clusters) with $mpts = 5$, while Figure 6.1b shows three larger structures for $mpts = 18$, one of them corresponding to the three smaller clusters seen in Figure 6.1a that were merged into a larger cluster as a result of the larger amount of smoothing. In fact, this dataset is an example where there is no single value of $mpts$ that will make all five main cluster structures in the data detectable in the same hierarchy.

DBSCAN finds clusters w.r.t. a single density level, defined by the two parameters $mpts$ and ε (see Chapter 2), while HDBSCAN* constructs a complete hierarchy of clusterings w.r.t. all density levels corresponding to a *single* value of $mpts$ and *multiple* values of ε. This use of a single value of $mpts$ can be regarded as a global control of smoothing in HDBSCAN*. Ideally, it might be advantageous to be able to *fully* adapt the amount of smoothing according to the properties of the local neighborhood of each point. In HDBSCAN*'s context, this means using different values of $mpts$ for different subsets of the data.

Let us first look at the $\varepsilon \times mpts$ space and try to understand what it means to vary or fix these parameters. Figure 6.2 shows the possible combinations of ε and $mpts$ w.r.t. variability. When both ε and $mpts$ are fixed, they define



(a) $mpts = 5$



(b) $mpts = 18$

Figure 6.1: Example

a density level that corresponds to what is found by DBSCAN. Note that this also corresponds to a horizontal cut at a level ε in a hierarchy generated by HDBSCAN* w.r.t. $mpts$. When ε varies and $mpts$ is fixed, they define a hierarchical organization of density levels as produced by HDBSCAN*. In that case, each level of the hierarchy is defined by a value of ε. A cluster extraction might consider different values of ε in different branches of the hierarchy, which is already supported by HDBSCAN* by using FOSC [8].

Fixing ε and varying $mpts$ also defines a hierarchical organization of dif-

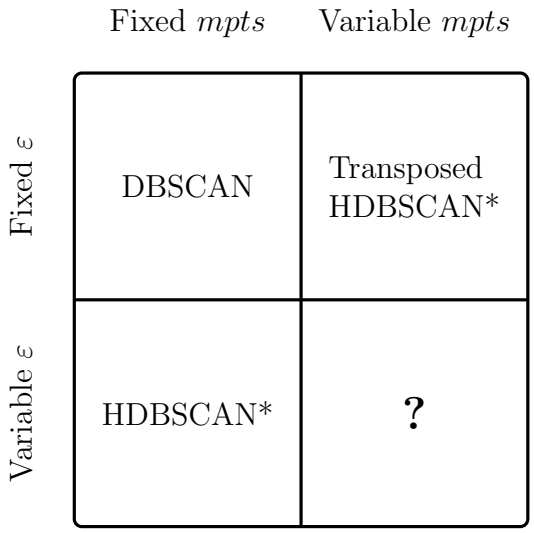


Figure 6.2: $\varepsilon \times mpts$

ferent density levels similar to the one produced by HDBSCAN*. We refer to this possible algorithm as Transposed HDBSCAN*. Essentially, this algorithm counts how many neighbors each point has in its ε-neighborhood, and these counts determine the levels at which the points are dense, *i.e.*, each level of the hierarchy corresponds to a value of $mpts$. Thus, the clusters at a level μ correspond to connected components of the density level defined by ε and μ, similarly to HDBSCAN*. Even though this might not be a very practical method on its own, its conceptual existence helps us understand how the $\varepsilon \times mpts$ space is organized.

The missing piece in Figure 6.2 corresponds to the case where both ε and $mpts$ can vary in the same hierarchy, which is equivalent to the use of multiple values of $mpts$ in HDBSCAN*. In order to develop a solution for this case, we investigate (1) how HDBSCAN* can be adapted to accommodate multiple values of $mpts$ in the same hierarchy, and (2) what is the interpretation of the final results.

In Section 6.2, we discuss adaptive density estimation methods and compare them to HDBSCAN*. In Section 6.3 we discuss theoretical aspects related

to using different values of *mpts* in a single hierarchy, and in Section 6.4 we discuss the practical aspects. In Section 6.5, we discuss the effects of multiple values of *mpts* in the cluster extraction process. In Section 6.6 we present our conclusion and discuss possible future work following this line of research.

6.2 Related Work

To the best of our knowledge, the problem of fully adaptive density-estimation with HDBSCAN* has not been investigated in the literature. In a broader sense, the goal we want to achieve in this chapter relates directly to the problem of estimating density with different amounts of smoothing in different regions of the data. In fact, classic nonparametric methods for density estimation such as *histograms* and *kernel density estimators* have been adapted to accommodate such scenarios. In *histograms* [38], the amount of smoothing in the estimates is controlled by the width of the bins placed over the samples of data. In its adaptive version, the width of the bins can vary to better represent the data distribution and avoid bins with a very low number of points. In [12], for instance, the authors propose a method for estimating the best histogram with k equiprobable bins to approximate a probability density function. Even though k is an input to the problem, the width of the bins vary to adjust to the local density of the data.

In kernel density estimation (KDE) [37], the smoothing of the estimates is controlled by the bandwidth parameter h (Equation 6.1).

$$\hat{f}(y) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x_i - y}{h}\right) \tag{6.1}$$

In order to deal with cases where it is believed that different amounts of smoothing should be applied to different regions of the space, one might recur to variable-bandwidth approaches. In [44], Terrell and Scott discuss two main adaptive methods, namely the *balloon estimator* and the *sample-point estima-*

tor. In both methods, a function $h(\cdot)$ is used to determine the bandwidth of the kernels. In the *balloon estimator* (Equation 6.2), the amount of smoothing is determined by the point where the density is being measured, *i.e.* the bandwidth of the kernel functions placed over the samples in the data, determined by $h(y)$, only depends on the point y.

$$\hat{f}(y) = \frac{1}{nh(y)} \sum_{i=1}^{n} K\left(\frac{x_i - y}{h(y)}\right) \tag{6.2}$$

On the other hand, the *sample-point estimator* (Equation 6.3) adapts the amount of smoothing according to each sample in the data. Thus, the bandwidth of the kernels placed over the samples x_i, determined by $h(x_i)$, only depends on x_i and is not influenced by the point y where the density is being estimated.

$$\hat{f}(y) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h(x_i)} K\left(\frac{x_i - y}{h(x_i)}\right) \tag{6.3}$$

The authors argue that the *sample-point estimator* has several advantages over the *balloon estimator*. For instance, the *sample-point estimator* is able to produce a proper probability density function, and performs better than fixed-bandwidth approaches in high-dimensional scenarios. A function commonly adopted as $h(\cdot)$ is the distance from a point to its k-th nearest neighbor. This makes the density estimator more flexible as there is no global bandwidth h, but rather a value k that can yield different bandwidth values in different regions of the data. Note that with an appropriate kernel function, one is able to construct a proper probability density function even with different values of k for different points.

The k-Nearest Neighbor density estimator is another successful and powerful method for adaptive density estimation, where a positive integer value k is fixed and the density at a point x is estimated based on k and the distance from x to its k-th nearest neighbor. Note, however, that a global parameter k still controls the amount of smoothing applied to the final estimates, which

might still cause situations as the ones discussed in Section 6.1.

While these methods offer some flexibility w.r.t. the smoothing of the estimates, works in the literature usually focus their approach in finding the best global smoothing factor (*e.g.* [41], [42]) to approximate the density function from which the data was sampled. Our goal, however, is a user-centered exploratory analysis process that should provide information on the hierarchical organization of the data under multiple parameter settings and in which users can interactively select the structures that seem more prominent according to application-specific requirements. Therefore, we focus on using different smoothing factors rather than trying to find a globally "best" value.

6.3 Multiple $mpts$ values: Theory

Currently HDBSCAN* uses an "unnormalized" measure, the *core-distance*, to estimate the density at each point p. The *core-distance* represents the radius of a ball around p that contains $mpts$ points from the data set. Hence, the density at individual points is based on the intuition that, the further away a point is from its $mpts$-th nearest neighbor, the lower its density. In other words, the density at a point is assumed to be inversely proportional to its *core-distance*, as expressed in Equation 6.4.

$$density(a) \propto \frac{1}{c_{mpts}(a)} \tag{6.4}$$

Moreover, HDBSCAN* estimates density at all points according to a *single* value of $mpts$, which leads to the following interpretation of the resulting clustering hierarchy: a point p is considered dense at a level ε if the ball of radius ε around p contains at least $mpts$ many points; two points p and q are considered *mutually reachable* w.r.t. $mpts$ at a level ε if both p and q have at least $mpts$ many points in their ε-neighborhood *and* they are both in each other's ε-neighborhood; two points p and q are part of the same cluster at a

level ε if there is a chain of points that are *mutually reachable* w.r.t. *mpts* at a distance ε, and p and q are part of that chain.

Note that the interpretation relies on the value of *mpts* to determine whether points are dense and part of the same connected component (cluster) at a certain level. In order to accommodate the use of multiple values of *mpts* in a single hierarchy, the first thing one has to consider is that each point p in the data must be associated with *one* value of *mpts*, which we denote as $mpts_p$. For simplicity of notation, we also consider that the values of *mpts* for each point are stored in an array $\mathcal{M}$ that can be seen as an assignment of *mpts* to points in the data, in a way that $\mathcal{M}(p) = mpts_p$. Based on that, we offer the following interpretation of a clustering hierarchy: a point p is considered dense at a level ε if the ball of radius ε around p contains $mpts_p$ many points; two points p and q are considered *mutually reachable* at a level ε if p and q have, respectively, $mpts_p$ and $mpts_q$ many points in their ε-neighborhood and both are in each other's ε-neighborhood; two points p and q are part of the same cluster at a level ε if there is a chain of points that are *mutually reachable* w.r.t. their respective values of *mpts* at a level ε, and p and q are part of that chain.

The interpretation described above is achieved with the definition of *mutual reachability distance* expressed in Equation 6.5, that considers that each point can be associated with a different value of *mpts*.

$$mrd_{\mathcal{M}}(a,b) = max\left\{c_{\mathcal{M}(a)}(a), c_{\mathcal{M}(b)}(b), d(a,b)\right\} \tag{6.5}$$

This definition determines the smallest radius ε at which both points are *dense* w.r.t. their own values of *mpts* and are in each other's ε-neighborhood. Thus, based on Equation 6.5, one is able to construct a clustering hierarchy where the density at different points or subsets of points can be estimated w.r.t. different values of *mpts*. Note that the formulation in Equation 6.5 is a generalization

of the scenario where a single value of $mpts$ is used. It is easy to see that when $mpts_a = mpts_b$, Equation 6.5 reduces to Equation 2.2.

6.4 Multiple $mpts$ values: Practice

In practical terms, our ultimate goal is to provide the necessary support to allow the construction of hierarchies with clusters generated with different values of $mpts$. In order to achieve this goal we discuss (1) how one can assign a value of $mpts$ to a subset of the data and then (2) how to construct the clustering hierarchy according to this assignment.

6.4.1 Assignment of $mpts$ values

Ideally, the assignment of a value of $mpts$ to a data point would be done in a way that makes the final density estimates at that point as close as possible to the true density of the data. However, the lack of ground truth information in unsupervised tasks makes it hard to validate which hierarchies, or branches of hierarchies, are objectively better than others. Besides the accuracy of the density estimates, an assignment $\mathcal{M}$ of $mpts$ values to (subsets of) data points must have other properties to make the construction of a clustering hierarchy w.r.t. $\mathcal{M}$ possible. An assignment $\mathcal{M}$ must *cover* all points in the data, *i.e.* for every point p in the data, $\mathcal{M}(p)$ must correspond to a value of $mpts$. If a point p is not associated with any value of $mpts$, the density at p cannot be estimated as defined in HDBSCAN*. Moreover, an assignment $\mathcal{M}$ must also guarantee that each point in the data is associated with *exactly one* value of $mpts$. Different values of $mpts$ may result in a different density estimates for a point, but the definitions of HDBSCAN* use a single value for each point. Building an assignment that satisfies these requirements without any objective criteria is not trivial.

A general and safe approach for assessing which values of $mpts$ are more

appropriate for different subsets of the data consists of manual/visual examination of the clustering hierarchies by users as part of an exploratory analysis. Therefore, equipped with a proper visualization tool like MustaCHE [35] (see Chapter 5), users can manually inspect clustering hierarchies and choose the clusters or branches they see fit according to any application or domain requirements. Implicitly, the selection of a cluster from a hierarchy w.r.t. $mpts = i$ can be seen as an indication that i is an adequate value of $mpts$ for estimating density at all points in that cluster.

In order to formally define a selection of a cluster, we introduce the following definitions and notations. Let X be a dataset with n points, and let $\mathcal{H} = \{H_1, H_2, ..., H_{k_{max}}\}$ be a set of hierarchies of X computed w.r.t. $mpts \in [1, k_{max}]$. Let $\{C_{i,1}, C_{i,2}, ..., C_{i,m_i}\}$ be the set of clusters in the hierarchy H_i, and $\phi(C) \in \{0, 1\}$ indicate whether a cluster C is selected ($\phi(C) = 1$) or not ($\phi(C) = 0$). Thus, a selection of clusters S can be defined as follows:

$$S = \{C \mid \phi(C) = 1\}$$

for which we require that the intersection of the selected clusters is empty:

$$\bigcap_{C \in S} C = \varnothing$$

In practice, by using the *condensed tree* visualization available in MustaCHE, users can select the clusters they want to include in S and MustaCHE *disables* the clusters that are not available for selection based on the overlapping with clusters already in S. This can be done in an interactive manner, allowing a *real-time* visual assessment of the cluster selection. Then, the cluster selection S can be transformed into an assignment of $mpts$ values to data points based on how clusters in S *cover* different subsets of the data. In situations where a cluster that a user wants to select slightly overlaps with the clusters in S, MustaCHE offers the possibility to users to select the levels of

that cluster at which there is no overlap with the clusters in S, if these levels exist. This possibility ultimately helps with the construction of a selection that is overlap-free and covers as much as possible of the clusters that the user wants to select.

In an *ideal* scenario, a selection S would cover each point in the data and a point x gets assigned the value of $mpts$ corresponding to the cluster in S that contains x. However, factors such as the presence of noise in the data or simply the preferences of users when constructing S might result in points not being covered by the selection. A natural interpretation of the selection of clusters/subtrees from different hierarchies by a user is that it is the intention of the user to focus on the selected clusters and "ignore", i.e. smooth out, the remaining points that are not included in the selection as much as possible. Since larger values of $mpts$ lead to smoother estimates, then the value of $mpts$ that corresponds to this interpretation, within the set for which we have pre-computed density estimates for each point, is the maximum value k_{max}.

Based on this interpretation, we formally define an assignment $\mathcal{M} : X \rightarrow [1, k_{max}]$ from points in a dataset X to values of $mpts$ according to a cluster selection S as follows:

$$\mathcal{M}(x) = \begin{cases} i & \text{if there is a cluster } C_{i,j} \in S \text{ such that } x \in C_{i,j} \\ k_{max} & \text{otherwise} \end{cases} \tag{6.6}$$

Note that users could choose different default values of $mpts$ to assign to data points that are not covered by a selection S. Other options include, for example, any values in the range $[1, k_{max}]$; the maximum value of $mpts$ corresponding to the clusters in S; or a more elaborate function that assigns values of $mpts$ to data points based on their proximity to the selected clusters in S.

6.4.2 Constructing the Clustering Hierarchy

Having an assignment of *mpts* values to points in the data makes it possible to build a clustering hierarchy where different amounts of smoothing are used in different parts of the data. One strategy for doing so consists of running HDBSCAN* from scratch with the formulation presented in Equation 6.5. However, one does not need to use the complete *mutual reachability graph* as originally proposed in HDBSCAN* in order to compute the complete hierarchy in this case. While our RNG-based strategy was primarily designed to speed up the computation of MSTs where all points are associated with the same value of *mpts*, we can show that any MST whose edge weights are determined by a combination of values of $mpts \in [1, k]$ can also be computed from the same RNG. We can show that, if two points are *relative neighbors* w.r.t. their respective values of *mpts* as determined by the assignment $\mathcal{M}$, they are also relative neighbors when the same value of *mpts* is used.

Theorem 3 *If an assignment $\mathcal{M}$ is defined over $[1, k]$, then $RNG_{\mathcal{M}} \in RNG_k$.*

Proof 4 *Consider two points a and b that are relative neighbors w.r.t. an arbitrary assignment $\mathcal{M}$. Then, for all other points c in the data, the following inequality is always satisfied.*

$$mrd_{\mathcal{M}}(a, b) \leq max\{mrd_{\mathcal{M}}(a, c), mrd_{\mathcal{M}}(b, c)\} \tag{6.7}$$

Since the assignment $\mathcal{M}$ will only assign values of mpts in the range $[1, k]$ and the mutual reachability distance *is monotone w.r.t. mpts, we have that $mrd_{\mathcal{M}}(a, b) \leq mrd_k(a, b)$. Therefore, we can replace the components of the max function on the right-hand side of Inequality 6.7 and get the following:*

$$mrd_{\mathcal{M}}(a, b) \leq max\{mrd_k(a, c), mrd_k(b, c)\} \tag{6.8}$$

The main idea of this proof is to transform the left hand side of Inequality 6.8 into the mutual reachability distance *between a and b w.r.t. mpts = k with*

a sequence of operations that do not *alter the validity of the inequality. Let us start by expanding the right-hand side of Inequality 6.8 and simplifying the resulting max functions into a single max function, which results in Inequality 6.9.*

$$mrd_{\mathcal{M}}(a,b) \leq max\{max\{c_k(a), c_k(c), d(a,c)\}, max\{c_k(b), c_k(c), d(b,c)\}\}$$

$$mrd_{\mathcal{M}}(a,b) \leq max\{c_k(a), c_k(c), d(a,c), c_k(b), c_k(c), d(b,c)\}$$

$$mrd_{\mathcal{M}}(a,b) \leq max\{c_k(a), c_k(b), c_k(c), d(a,c), d(b,c)\} \tag{6.9}$$

Next, by expanding the left-hand side of Inequality 6.9 we get the following:

$$max\{c_{\mathcal{M}(a)}(a), c_{\mathcal{M}(b)}(b), d(a,b)\} \leq max\{c_k(a), c_k(b), c_k(c), d(a,c), d(b,c)\} \tag{6.10}$$

In this step of the proof, we start replacing the components of the max function on the left-hand side of Inequality 6.10, while making sure that the resulting inequality is still satisfied. Note that the max function on the right-hand side of Inequality 6.10 evaluates to a value at least *as large as* $c_{k_{max}}(a)$*. In fact, if we replace* $c_{\mathcal{M}(a)}(a)$ *with* $c_{k_{max}}(a)$ *in the max function on the left-hand side of Inequality 6.10, one of the following cases must happen: (1) the result of the max function on the left-hand side of the inequality is not altered – in this case, it is trivial to see that the resulting inequality is still satisfied, as the evaluation of both max functions have not changed; (2) the max function on the left-hand side of the inequality evaluates to* $c_{k_{max}}(a)$ *– similarly, the right-hand side is still smaller than or equal to the right-hand side, and the resulting inequality is still satisfied. Therefore, we can safely replace* $c_{\mathcal{M}(a)}(a)$ *with* $c_{k_{max}}(a)$*, which results in the following inequality:*

$$max\{c_k(a), c_{\mathcal{M}(b)}(b), d(a,b)\} \leq max\{c_k(a), c_k(b), c_k(c), d(a,c), d(b,c)\} \tag{6.11}$$

Based on the same reasoning, we can replace the $c_{\mathcal{M}(b)}(b)$ *with* $c_k(b)$ *in the*

left-hand side of Inequality 6.11 and get the following:

$$max\{c_k(a), c_k(b), d(a,b)\} \leq max\{c_k(a), c_k(b), c_k(c), d(a,c), d(b,c)\} \quad (6.12)$$

Note that the left-hand side of Inequality 6.12 corresponds to the mutual reachability distance *between a and b w.r.t. mpts = k. Also, note that the max function on the right-hand side of Inequality 6.12 contains all the components of the* mutual reachability distances *between a and c and between b and c. In fact, the right-hand side of Inequality 6.12 corresponds to the simplification of the nested max functions corresponding to the* mutual reachability distances *in the right-hand side of Inequality 6.8. Thus, let us then rewrite 6.12 in terms of* mutual reachability distances *with the following sequence of simple operations:*

$$mrd_k(a,b) \leq max\{c_k(a), c_k(b), c_k(c), d(a,c), d(b,c)\}$$

$$mrd_k(a,b) \leq max\{c_k(a), c_k(c), d(a,c), c_k(b), c_k(c), d(b,c)\}$$

$$mrd_k(a,b) \leq max\{max\{c_k(a), c_k(c), d(a,c)\}, max\{c_k(b), c_k(c), d(b,c)\}\}$$

$$mrd_k(a,b) \leq max\{mrd_k(a,c), mrd_k(b,c)\} \quad (6.13)$$

With Inequality 6.13, we conclude that a and b are relative neighbors *w.r.t. mpts = k. Therefore, if two points a and b are* relative neighbors *w.r.t. an arbitrary assignment $\mathcal{M} \in [1,k]$, then a and b are also relative neighbors w.r.t. mpts = k. This means that a and b are connected with an edge in the RNG computed w.r.t. mpts = k and, therefore, the MST corresponding to the assignment $\mathcal{M}$ can also be computed from the same RNG used for compute the multiple clustering hierarchies w.r.t. mpts $\in [1,k]$.*

Finally, Figure 6.3 shows the reachability plot corresponding to the dataset from Figure 6.1 when different values of *mpts* are used for different parts of the data. More specifically, in this example *mpts* = 5 is used for the points corresponding to the three smaller clusters visible at the left side of Figure 6.1a,

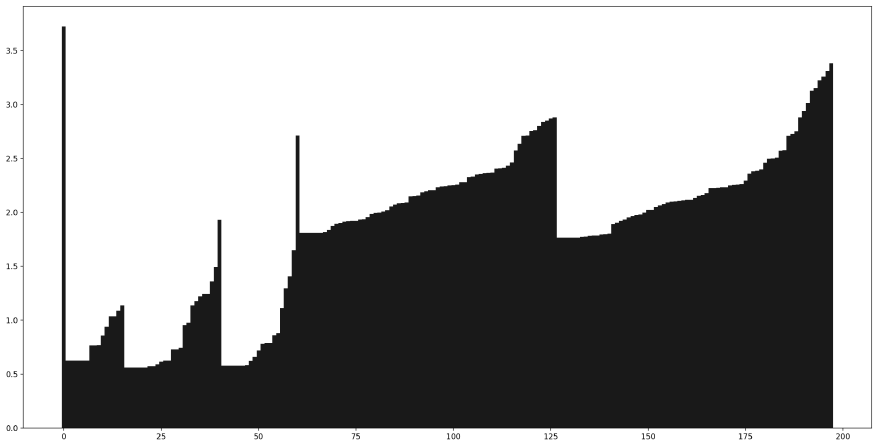


Figure 6.3: Hierarchy constructed w.r.t. multiple values of $mpts$.

and $mpts = 18$ is used for the points that form the two larger clusters on the right side of Figure 6.1b. Note that the aforementioned clusters correspond to our cluster selection S, and the value of $mpts$ corresponding to the hierarchies from which they were selected is assigned to the points in the clusters. In this case, we have computed clustering hierarchies w.r.t. $mpts \in [1, 30]$ and therefore, the points that are not part of any selected cluster in S gets assigned a value of $mpts$ equivalent to $k_{max} = 30$. As a result, one is able to visualize all major cluster structures in a single hierarchy/reachability plot.

In summary, the process of using multiple values of $mpts$ in a single clustering hierarchy with HDBSCAN* can be described as follows: (1) users select clusters from the clustering hierarchies using a visualization tool (MustaCHE); (2) each point in the data gets assigned a value of $mpts$ based on the clusters in the cluster selection; (3) a clustering hierarchy is built according to each point's assigned value of $mpts$.

6.5 Cluster Extraction

Many classical clustering methods try to find a flat partitioning of the data into different groups/clusters instead of a hierarchical organization of nested clusters. Even though HDBSCAN* is intrinsically a hierarchical clustering algorithm, its framework offers a method for automatic cluster selection and extraction called FOSC (Framework for Optimal Selection of Clusters) [8]. Given a suitable objective criterion, FOSC finds the set of clusters that optimize this criterion over all possible non-overlapping combinations of clusters in the clustering hierarchy.

A suitable criterion suggested by the authors, called the *relative excess of mass*, tries to capture which clusters have a more stable structure in terms of size and lifetime. The lifetime of a cluster can be measured by the difference between the density levels at which a cluster appears and disappears in a hierarchy. Thus, clusters that live longer with a large number of points are deemed more stable than clusters that have shorter lifetimes or are much smaller. The lifetime of a cluster and its size depend directly on the amount of smoothing applied in the density estimation, which is controlled by the parameter $mpts$. If the value of $mpts$ is too low, the estimates are more spiky and sets of smaller clusters tend to have a large combined relative excess of mass. If the value of $mpts$ is too high, the smaller clusters are smoothed out and larger clusters are favoured in the cluster selection. Therefore, one can argue that the value of $mpts$ has a significant influence on the cluster extraction performed by FOSC w.r.t. *relative excess of mass*, as it affects which clusters are more prominent in the clustering hierarchy. In the following, we discuss how the adaptations to HDBSCAN*, allowing multiple values of $mpts$, affect cluster extraction done by FOSC according to the cluster stability measure.

The stability of a cluster C w.r.t. its *relative excess of mass* can be ex-

pressed as follows:

$$S(C) = \sum_{x \in C} \left(\lambda_{max}\left(x, C\right) - \lambda_{min}\left(C\right) \right) \tag{6.14}$$

where $\lambda_{max}(x, C)$ corresponds to the maximum *density level* at which a data point x belongs to cluster C, and $\lambda_{min}(C)$ corresponds to the minimum *density level* at which cluster C exists in the clustering hierarchy.

In the simple scenario where all points in the data are assigned the same value of $mpts$, the stability values and the selected clusters are not affected when compared to HDBSCAN*. However, in the more general scenario where multiple values of $mpts$ might be assigned to different subsets of the data, the stability of clusters might be affected. When clusters from different hierarchies are combined in a single hierarchy, the density level at which they are connected might be higher or lower when compared to the value in their original hierarchy due to the use of different values of $mpts$ determining the *mutual reachability distance* that separates the data points in the different clusters. Thus, as stability takes into account the level at which a cluster first appears in the hierarchy, clusters that would be extracted by FOSC in their original hierarchy might not get selected when combined with clusters from different hierarchies. Note that, in a hierarchy built from branches of different hierarchies, *only* the clusters at the root of the selected branches will have their stability affected, and not all the clusters in each branch.

Figure 6.4 illustrates the cluster extraction performed by FOSC on the same dataset we use as example throughout this chapter for $mpts = 5$ and $mpts = 18$. The extracted clusters are represented with different colors and the bars in black correspond to points that are considered noise by the extraction method. Thus, four clusters are extracted from the hierarchy represented in Figure 6.4a, and three clusters are extracted from the hierarchy represented in Figure 6.4b. Note that in both cases some points are considered noise

and, consequently, are not part of any extracted cluster. However, there is no single value of *mpts* that can be used to detect and extract, simultaneously, the three leftmost clusters in Figure 6.4a and the two rightmost clusters in Figure 6.4b. On the other hand, when using multiple values of *mpts* for different subsets of the data, the 5 aforementioned clusters can not only be detected simultaneously, but also extracted simultaneously, as shown in Figure 6.4c.

6.6 Conclusion and Future Work

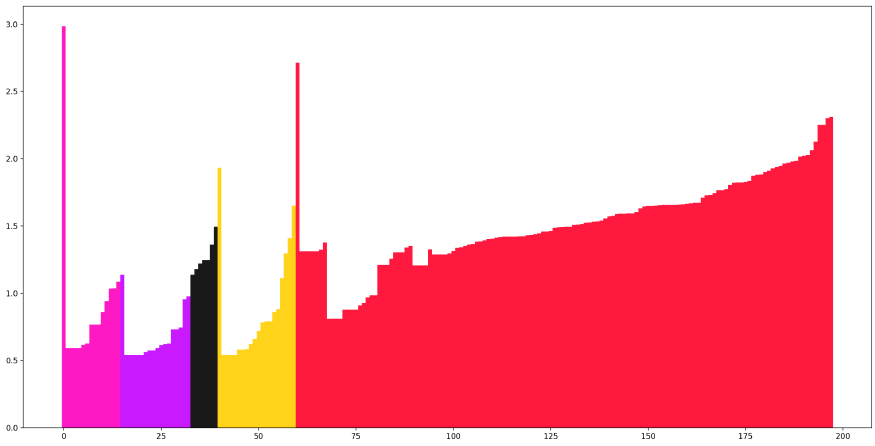
We have proposed and discussed the use of different values of *mpts* to construct a single hierarchy with HDBSCAN*. One of the advantages of this new approach over the density estimates used in the original formulation of HDBSCAN*, is the flexibility that allows users to apply different amounts of smoothing in different regions of the space independently.

We believe that this new flexibility opens opportunities for new measures or strategies to select different smoothing factors for different subsets of the data. Moreover, the use of different values of *mpts* makes the extraction of clusters more flexible, since one can extract clusters from different regions of the space without a global parameter setting that controls the amount of smoothing applied to all regions of the space.

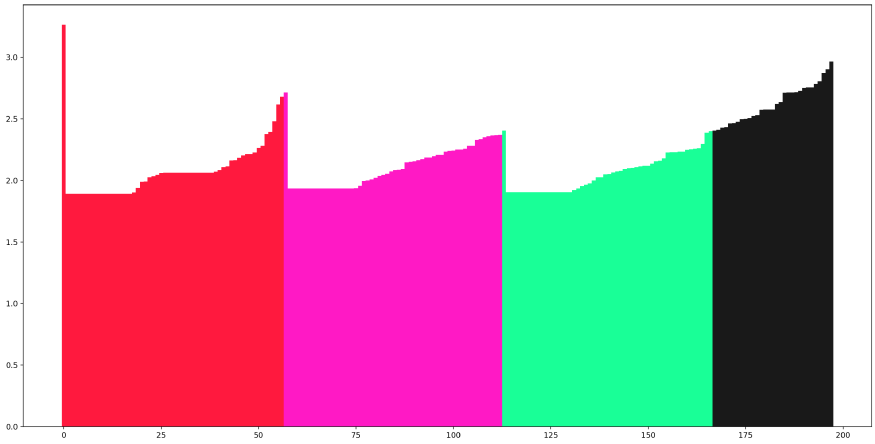
Note that the assignment of values of *mpts* to data points does not need to be done necessarily via a selection of clusters. One could, for instance, devise a way to determine the value of *mpts* of each point based on its local neighborhood rather than on how prominent clusters appear in the hierarchies. The investigation of methods for automatic estimation of "good" values of *mpts* for different regions of the data is an interesting direction for future research.

We also believe that modeling the density estimates in HDBSCAN* as real probability density function might open some doors for further research. While

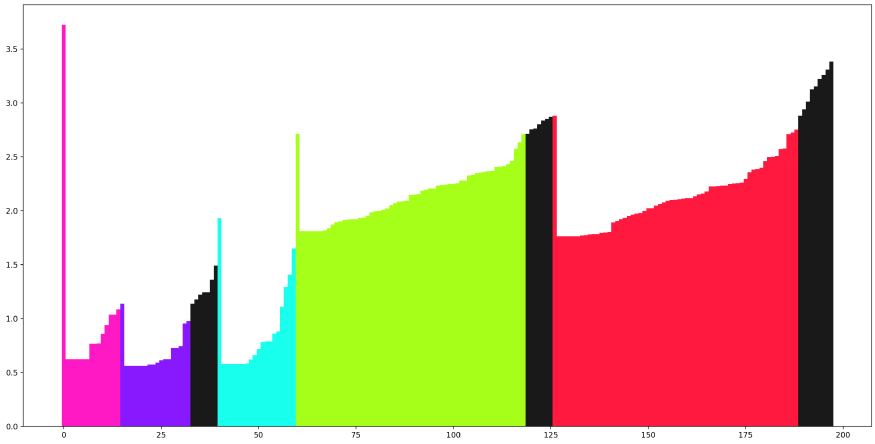
the use of kernel functions in HDBSCAN* has been investigated in [28], using an adaptive bandwidth control has not been considered. Furthermore, when using a proper probability density function, comparing density and stability values will be more straightforward. However, this implies also investigating efficient methods to compute hierarchies w.r.t. multiple values of *mpts* using a kernel density estimator, which may prove to be difficult.



(a) $mpts = 5$ (FOSC)



(b) $mpts = 18$ (FOSC)



(c) Merged Hierarchy (FOSC)

Figure 6.4: Cluster Extraction

Chapter 7

General Framework

7.1 Introduction

The contributions presented so far in this thesis have been developed in the context of unsupervised hierarchical density-based clustering with HDBSCAN*. The ability to efficiently compute clustering hierarchies for a range of *mpts* values with our RNG-based approach has enabled other research directions that were otherwise limited by the cost of exploring the space of parameter values in HDBSCAN*.

In this chapter, we will show that our approach is not restricted to an unsupervised setting but can be helpful in supporting density-based semi-supervised clustering and semi-supervised classification, where users might have to recur to a *trial-and-error* approach in order to explore which parameter settings produce the best results for a given application. In a recent study [18], Getrudes et al. have thoroughly analyzed unsupervised and semi-supervised clustering algorithms, as well as semi-supervised classification algorithms, from the point of view of density-based clustering, and proposed a unified framework that subsumes existing algorithms in the literature. When presenting the building blocks of the unified framework, the authors argue that these algorithms can be conceptually interpreted as a (direct or indirect) processing of Minimum Spanning Trees (MST) constructed in the space of

reachability distances, and that the MST construction can be decoupled from these algorithms for performance and generalization purposes.

Similarly to HDBSCAN*, the computation of multiple MSTs from the complete graph in the space of *mutual reachability distances* represents a performance bottleneck that prevents one from efficiently exploring multiple parameter settings for a comprehensive data analysis. As both the unsupervised and semi-supervised settings share the same performance issue related to the computation of MSTs, it is reasonable to argue that the contributions made for the unsupervised scenario can also be adapted or directly transferred to the semi-supervised scenario.

In this chapter, we discuss how the semi-supervised classification and semi-supervised clustering algorithms presented in [18] can benefit from the techniques presented so far in this thesis, and how our contributions go beyond the purpose to which they were developed.

In Section 7.2, we present a general overview of the framework proposed by Gertrudes et al. [18]. In Section 7.3, we discuss how the contributions of this thesis can be adapted or directly applied to semi-supervised classification and semi-supervised clustering algorithms. In Section 7.4, we present our conclusions and discuss directions for future work.

7.2 Background: Unified Framework [18]

Semi-supervised learning algorithms are designed to address problems where ground truth information is available for only a small portion of the data, and to make use of this information to improve its results. In clustering applications, for instance, ground truth can be expressed in the form of pre-labeled objects or pairwise constraints of the type *should-link* and *should-not-link*.

In [18], the authors present a framework that unifies *semi-supervised clustering* and *semi-supervised classification* techniques based on their common

characteristics. Even though both categories have slightly different purposes – for instance, in semi-supervised clustering the number of groups found can be larger than the number of groups in the set of labels available, while in semi-supervised classification the groups found are an expansion of the set of labels given as input to the algorithm – many of these algorithms can be defined according to the same set of fundamental concepts. Next we discuss the components of the framework for semi-supervised clustering and semi-supervised classification, and the algorithms they encompass.

7.2.1 Unified Density-based Clustering Framework

After a thorough analysis of several unsupervised and semi-supervised density-based clustering techniques in the literature – DBSCAN* [7], [9], DBSCAN [16], OPTICS [2], SSDBSCAN [32], HISSCLU (k-cluster) [4] and HDBSCAN* [7], [9] – the authors observed that these algorithms can all be interpreted as a (direct or indirect) processing of the *minimum spanning tree* computed in the space of reachability distances. That observation allowed the authors to generalize and group these techniques under the same framework.

While DBSCAN*, DBSCAN and OPTICS are originally unsupervised techniques, SSDBSCAN and HISSCLU are intrinsically semi-supervised. As for HDBSCAN*, even though it was originally proposed as an unsupervised technique in [7], its first semi-supervised version was proposed in [9] with a strategy for extracting clusters from the clustering hierarchy based on a measure that takes into account the ground truth received as input. In fact, one can obtain different semi-supervised clustering algorithms based on different measures for extraction of clusters from the clustering hierarchies produced by HDBSCAN*. Therefore, the authors present the following semi-supervised variations of HDBSCAN* based on the the measures used for cluster extraction: HDBSCAN*(CON), HDBSCAN*(MixCON), HDBSCAN*(BC) and

HDBSCAN*(MixBC); enlarging the body of algorithms that fall under the proposed framework. The first two algorithms, HDBSCAN*(CON) and HDBSCAN*(MixCON), are based on the original publication of semi-supervised clustering with HDBSCAN* [9], where the ground-truth information is expressed in terms of pairwise constraints of the type *should-link* and *should-not-link* – in HDBSCAN*(CON), the cluster extraction is performed solely based on the pairwise constraints, and in HDBSCAN*(MixCON) the cluster extraction is performed based on a combination of cluster stability (relative excess of mass) and pairwise constraints. The last two algorithms, HDBSCAN*(BC) and HDBSCAN*(MixBC), rely on label-based information to select clusters from the clustering hierarchy computed by HDBSCAN*. The authors use the F_{B^3} measure as the optimization criterion for cluster selection with FOSC to capture how much the clusters in the hierarchy conform with the labels given as input. In HDBSCAN*(BC), the selection is performed based only on the F_{B^3} measure, and in HDBSCAN*(MixBC) the selection is performed according to a combination of the F_{B^3} measure and the stability w.r.t. to the relative excess of mass of each cluster.

Note that the only aspect that differentiates these algorithms from each other is the criterion used for extracting clusters from the density-based clustering hierarchy. In fact, as these algorithms are fundamentally similar in their processing, it is expected that they perform similarly in terms of running time. This expectation is is confirmed by the experimental evaluation conducted in [18] that shows virtually no difference in the running time of all algorithms.

7.2.2 Unified Density-based Classification Framework

The algorithms under the unified framework for density-based clustering share a common characteristic that is the processing of the MST computed in the space of *mutual reachability distances*. Additionally, each of these algorithms

can be further decomposed into "building blocks" that can be recombined to create new algorithms. Thus, in order to construct a new framework for classification under the perspective of density-based clustering, Gertrudes et al. extended HDBSCAN* based on the addition of the following optional "building blocks" decoupled from the algorithms in Subsection 7.2.1: (1) definition of core- and reachability-distances; (2) MST computation; (3) label expansion; and (4) preprocessing of the distances based on the labels.

In the first block, the authors consider two definitions of core-distances for estimating the density at points, and consequently, the mutual reachability distance between points. They consider the original definition of *core-distance* already available in HDBSCAN*, and the *all-points core-distance* (Equation 7.1) proposed in [33] in the context of density-based cluster validation.

$$d_{aptsCore}(x) = \left(\frac{\sum_{x_i \in X \setminus \{x\}} \left(\frac{1}{d(x,x_i)} \right)^d}{n-1} \right)^{-\frac{1}{d}} \tag{7.1}$$

Note that the definition of *all-points core-distance* does not depend on the parameter *mpts*, instead its definition corresponds to the *generalized mean* of the inverse of the distances between a point and all the other points in the data. The second building block corresponds to the computation of the MST in the space of mutual reachability distances, and the third building block corresponds to the expansion of labels to obtain a class label for each object in the data. The fourth building block corresponds to a label-based distance weighting that can be applied to the distances between the objects in the data as performed by the HISSCLU algorithm. This processing is applied to increase separation between objects of different classes guided by the labels available for a subset of the data.

The algorithms resulting from different combinations of these building

blocks are listed in the diagram[1] in Figure 7.1. The algorithms are named according to the used *core-distance* and the applied label-based distance weighting. The algorithms based on the original definition of *core-distance* are marked with "cd", and the ones based on the definition of *all-points core-distance* are marked with "ap". Moreover, when label-based distance weighting is applied directly to the distance matrix, the techniques are marked with "wPWD", and when the weighting is applied only to the edges of the MST, they are marked with "wMST". The experimental evaluation presented in [18] has shown that these techniques behave similarly with regard to runtime performance.

7.3 Applicability of our Contributions

In the following subsections we discuss how the contributions of this thesis, originally developed for unsupervised clustering with HDBSCAN*, can be extended to the collection of algorithms presented in [18] for semi-supervised clustering and semi-supervised classification. Our goal is to highlight the relevance of our techniques in different applications other than the one that initially motivated our work.

7.3.1 RNG-based Approach

One of the main contributions of this thesis is the use of the relative neighborhood graph along with HDBSCAN* to speed-up the computation of a collection of clustering hierarchies w.r.t. to a range of values of $mpts$. Our approach outperforms the naive strategy of running HDBSCAN* for each value of $mpts$ in the given range, and guarantees the correctness of the results when compared to the original algorithm.

While our RNG-based strategy has been primarily proposed to speedup

[1]This is a reproduction of the diagram presented in [18].

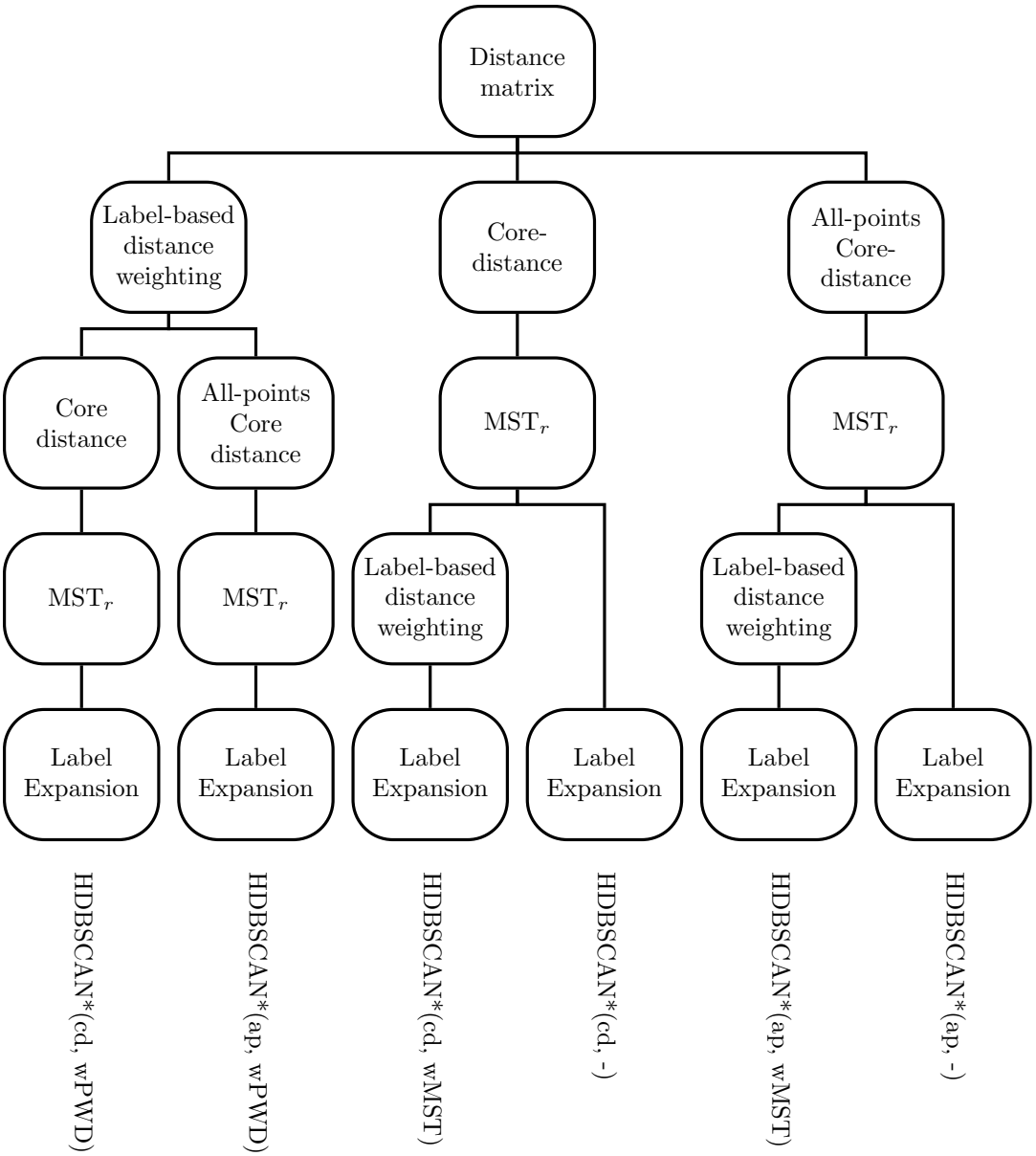


Figure 7.1: Unified Framework for Semi-supervised Classification [18]

HDBSCAN*, its applicability goes beyond its original purpose. Due to the developments made in [18], our strategy can be regarded as a more general approach that can be used with several other algorithms in the literature to aid in the process of exploring multiple parameter scenarios more efficiently.

The unified framework for density-based clustering presented in [18] is based on the processing of an MST in the space of mutual reachability dis-

tances determined by a value of *mpts*. Similarly to HDBSCAN*, the value of *mpts* used to estimate the density at the points in a dataset can influence the cluster structures found by the algorithms and, consequently, users would have to explore multiple parameter settings in order to gain a deeper understanding of the cluster structures in the data and select a value of *mpts* that is suitable for a given application. Since all algorithms in the density-based clustering framework are based on the same fundamental data structure – the MST in the space of mutual reachability distances – our RNG-based strategy can be applied to help in the efficient computation of multiple clustering results w.r.t. different parameter settings in all algorithms that use the original *core-distance*. Since the *all-points core-distance* does not have a *mpts* parameter, only one hierarchy is created in these cases and our strategy trivially does not apply.

According to the experimental evaluation conducted in [18], all clustering algorithms in the framework present a similar performance in terms of runtime. This runtime is largely determined by the time spent in the computation of the *minimum spanning tree*, while the extraction of the clusters – the differential aspect between these algorithms – can be performed quite efficiently. In fact, as the most time-consuming part of all the algorithms is the computation of the MST, it is safe to claim that the impact of our RNG-based strategy in these algorithms are expected to be comparable to what is presented and discussed for RNG-HDBSCAN* in Chapter 4.

In the semi-supervised classification framework, the algorithms presented are based on extensions of HDBSCAN* with building blocks decoupled from the algorithms in the density-based clustering framework. Even though the MST is a key building block that is a part of all the algorithms in this framework, the remaining blocks also have an influence on whether our RNG-based approach can be applied to these algorithms.

In HDBSCAN*(cd,wPWD), the authors apply the label-based weighting to the input distance matrix, similarly to HISSCLU. After weighting the distance matrix, the final values corresponding to the distances do not necessarily follow the triangle inequality. Since our approach for computing the RNG relies on the triangle inequality, one cannot guarantee that the resulting graph will be *correct* and, consequently, that the classification results computed from such a graph are consistent with the classification results computed with the original algorithm. Thus, the computation of multiple classification results with the algorithm HDBSCAN*(cd,wPWD) cannot be sped up with the use of our RNG-based strategy without losing correctness guarantees of the results.

Moreover, even though the label-based weighting block is part of the algorithm HDBSCAN*(cd,wMST), the weighting is applied only to the edges of a previously computed MST rather than to the input distance matrix. Hence, all MSTs can be computed with our RNG-based strategy before the weighting of the distances takes place. Trivially, the HDBSCAN*(cd,-) algorithm can also benefit from our RNG-based strategy, since it is based solely on the original definition of *core-distances* and no distance weighting is applied.

Thus, among the algorithms in the semi-supervised classification framework, the algorithms HDBSCAN*(cd,wMST) and HDBSCAN*(cd,-) can take advantage of our RNG-based strategy to speed up the computation of a collection of classification results for different values of $mpts$. Similarly to the algorithms in the density-based clustering framework, the semi-supervised classification algorithms are also expected to be similarly impacted by our RNG-based strategy, since their performance in terms of runtime are virtually equivalent and are determined by the computation of the MST.

7.3.2 MustaCHE

In Chapter 5, we have presented MustaCHE, a visualization tool developed for the analysis and exploration of multiple clustering hierarchies computed w.r.t. multiple values of *mpts*. Even though MustaCHE was primarily developed based on the unsupervised setting, its visualizations can be adapted to consider the algorithms discussed in [18].

In unsupervised clustering, the visual aspects of the clustering hierarchies combined with the user expertise and any requirements of the analysis being performed can help with the identification of interesting results among a collection thereof. When some ground truth is available, even if for a small subset of the data, users can still make use of a visualization tool like MustaCHE along with the ground truth information to guide their analysis. Even though the semi-supervised clustering and semi-supervised classification algorithms presented in [18] do not produce a hierarchical structure as a final output, their core concepts are based on hierarchical structures that can be expressed through the visualizations available in MustaCHE.

In a straightforward application, the set of labels available for a subset of the data can be used to colour the reachability plots and guide users towards results where the *valleys* visible in the plots (representing dense regions in the data) match the labels given in the input or the clustering/classification results found by the algorithms.

In a different direction, MustaCHE could be adapted to be used as a tool that allows users to label subsets of data in order to produce semi-supervised clustering results. Thus, users would be able to start their analysis with an unsupervised setting and later switch to a semi-supervised setting without the need to re-run any algorithms from scratch while assessing results in real-time.

As for the visualizations based on the cluster trees available in MustaCHE,

they can be used to help users inspect the stability of clusters according to measures such as the F_{B^3} or its combination with with *relative excess of mass*. As the algorithms are mostly based on the same data structure, MustaCHE could be used as a tool for comparison between different algorithms, where users could seamlessly switch between techniques and compare their differences.

We note that, while these features can be integrated into MustaCHE, it is not the focus nor is in the scope of this thesis the support for semi-supervised algorithms in MustaCHE. We mainly wanted to make the case that MustaCHE is a tool that *can* be employed in other contexts other than unsupervised clustering. In fact, we believe that leveraging visualizations to improve the analysis of data in semi-supervised settings, along with user interactivity and efficient computation of results, is a topic of research worth of further investigation.

7.3.3 Multiple values of *mpts*

In Chapter 6, we have discussed the benefits of using different values of *mpts* in different subsets of the data for constructing hierarchies with HDBSCAN*, along with the theoretical and practical aspects of doing so. In the semi-supervised setting, a similar argument can be used to justify the use of different values of *mpts* for different subsets of the data – different values of *mpts* can produce different edge weights or even a different structure (in terms of edges) of the MST, which, consequently, affects the clustering/classification results found by the algorithms.

In semi-supervised clustering, a large amount of smoothing will make clusters less separable and the resulting clusters found by the algorithms might not conform with the set of labels available. On the other hand, a small amount of smoothing will result in a large number of clusters and points with the same label might be placed into different clusters. In semi-supervised classification,

the values of the edge weights have a direct effect on how the labels given as input are expanded to produce a classification of the data points. Analogously to the semi-supervised clustering case, different amounts of smoothing might lead to different classification results.

When parts of the data require more smoothing than others, the use of multiple values of *mpts* might improve the final clustering/classification results, as one can adapt the amount of smoothing of the estimates to the local density of the data points *and* to the subset of labels available. In summary, the arguments and concepts presented in Chapter 6 along with adaptations to the visualizations available in MustaCHE can be potentially useful in semi-supervised clustering and classification tasks.

7.4 Conclusions and Future Work

In this chapter, we have discussed the applicability of our contributions, primarily developed for unsupervised density-based clustering with HDBSCAN*, in the semi-supervised setting. More specifically, we discuss how our RNG-based approach for computing multiple unsupervised clustering hierarchies (Chapter 4) can also be applied to a collection of semi-supervised algorithms generalized in [18]. As a consequence of the advancements made in [18], our approach can now be regarded as a more general strategy that is applicable to other methods other than HDBSCAN*. We also discuss how can be used to support data analysis in semi-supervised settings.

As for future work, we believe that the integration of semi-supervised techniques into MustaCHE for analyzing multiple parameter settings would have a positive impact on the ability of users to perform density-based clustering and classification analyses. MustaCHE could then be used as a tool for navigating, exploring and comparing not only different parameter settings, but also different algorithms.

Chapter 8

Conclusions

In this thesis we investigated problems related to the smoothing factor of the hierarchical density-based clustering algorithm HDBSCAN*, defined by its input parameter *mpts*. We observed that different values of *mpts* can potentially reveal different structures in the data, and that exploring a range of *mpts* values by running HDBSCAN* multiple times can be computationally expensive and impractical for users.

Our first contribution is RNG-HDBSCAN*, a strategy for efficient computation of multiple clustering hierarchies w.r.t. a range of values of *mpts*. We replaced the complete graph originally used in HDBSCAN* with a much smaller graph, the Relative Neighborhood Graph (RNG), and proved that the results obtained by our strategy are correct in comparison to the results computed by HDBSCAN*. Moreover, we showed that one is able to compute the RNG once for the highest value of *mpts* in a given range, and all clustering hierarchies for smaller values of *mpts* can be computed from the same RNG. Our experimental evaluation showed that our RNG-based strategy is able to compute over 100 hierarchies in the same time that the naive strategy of running HDBSCAN* would compute only about 2 hierarchies.

Next, we presented MustaCHE, a visualization tool that allows the exploration of a collection of clustering hierarchies in an interactive manner.

MustaCHE is composed of a series of visualizations that allow the analysis of multiple hierarchies w.r.t. several *mpts* values, and makes it easier for a user to have a deeper understanding of how the cluster structures in the data behave under different density levels. We also presented case studies with real datasets from different domains to showcase how the analysis of a collection of clustering hierarchies can be performed with MustaCHE.

Subsequently, we presented an approach for constructing clustering hierarchies containing structures found with different values of *mpts* with HDBSCAN*. Our strategy for selecting which values of *mpts* are more suitable to the different subsets of the data is based on the visual aspects of the clustering hierarchies that can be explored by users through MustaCHE. Users can select clusters whose structure they find interesting and these clusters are transformed into an assignment of values of *mpts* to points in the data. This possibility offers flexibility to users for building hierarchies where different amounts of smoothing can be applied to different subsets of the data.

Later, we have discussed how our contributions made originally in the context of unsupervised clustering with HDBSCAN* can be extended to a class of semi-supervised clustering and semi-supervised classification algorithms. Due to the advancements made in [18], a collection of semi-supervised algorithms that are based on the computation of an MST in the space of *mutual reachability distances* can also benefit from our RNG-based strategy and, consequently, from our visualization tool for computation and exploration of semi-supervised clustering/classification results.

Last, we presented an analysis of HDBSCAN*'s density estimates and observed that the estimates along the straight paths between points can be overestimated or underestimated. In an attempt to reduce these effects and to possibly make HDBSCAN* more robust to its parameter *mpts*, we investigated a theoretically improved definition of *mutual reachability distance* that offers

guarantees about the density along the straight paths between data points and results in a better interpretation of the resulting clustering hierarchies. However, our preliminary tests have shown that this new definition presents no practical advantage over HDBSCAN* and does not allow an efficient method to explore results w.r.t. multiple values of *mpts*. These results suggest that the *mutual reachability distance* used in HDBSCAN* is not only competitive compared to other density estimates in terms of resulting cluster hierarchies, but may offer the unique advantage that it is now feasible to explore and integrate clustering results for a whole range of parameter values (which other density estimates will still have) - given the contributions made in this thesis.

8.1 Future Work

In our analysis, we have identified that the use of HDBSCAN' does not imply in significant changes in the conclusions that can be drawn from the clustering results when compared with HDBSCAN*. However, the investigation of more precise density estimators that can be efficiently explored as the one originally available in HDBSCAN* and provide a better representation of the density in the data is still an open research problem.

Moreover, we believe that the framework proposed in this thesis based on the use of the relative neighborhood graph for efficient computation of clustering hierarchies can be extended to consider the use of approximate solutions for achieving higher levels of runtime performance for cluster analysis. This extension requires a thorough investigation on the trade-off between the quality of the clustering results obtained in this fashion and the efficiency of this strategy when compared to the results and performance of HDBSCAN* and our RNG-based strategy. In the same context, another future research direction is the study of measures that can be applied along RNG-HDBSCAN* to evaluate the quality or stability of clustering hierarchies computed according

to a range of *mpts* values. Closely related, the investigation of alternative similarity measures for estimating how similar two clustering hierarchies are is also an interesting future work direction.

As for the visualization of clustering results, we believe that MustaCHE could be deployed as an end-to-end web-based "Clustering as a Service" where users can upload, cluster, visualize, analyze, archive or share (if appropriate) their data and analyses. In terms of performance, MustaCHE could also benefit from approximate solutions where results could be available for analysis faster and refined later. Another direction for future work in MustaCHE is the addition of visualizations for the data (2d and 3d plots, geolocated data on maps, word clouds, etc.) being clustered. This would make it possible for users to perform more comprehensive analyses without having to recur to other tools.

Regarding the use of multiple values of *mpts* in a single clustering hierarchy, we believe that this new flexibility opens opportunities for new measures or strategies to select different smoothing factors for different subsets of the data. For instance, the investigation of methods for automatic estimation of "good" values of *mpts* for different regions of the data is an interesting direction for future research. Furthermore, we believe that modeling the density estimates in HDBSCAN* as real probability density function might open some doors for further research. While the use of kernel functions in HDBSCAN* has been investigated in [28], using an adaptive bandwidth control has not been considered. When using a proper probability density function, comparing density and stability values across different hierarchy is more straightforward. However, this implies also investigating efficient methods to compute clustering hierarchies w.r.t. multiple values of *mpts* using a kernel density estimator, which may prove to be difficult.

As for the application of our contributions in semi-supervised tasks, we

believe that the integration of semi-supervised techniques into MustaCHE for analyzing multiple parameter settings can have a positive impact on the ability of users to perform semi-supervised density-based clustering and classification analyses. MustaCHE could then be used as a tool for navigating, exploring and comparing not only different parameter settings, but also different algorithms. Moreover, one can also make use of the set of labels available for part of the data and measure, in a more objective manner, which hierarchies are more or less conforming with such labels. This strategy has the potential of helping users with the task of determining the appropriate amount of smoothing in the density estimates.

References

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

[2] M. Ankerst, M. M. Breunig, H. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, ACM Press, 1999, pp. 49–60.

[3] M. A. Bender and M. Farach-Colton, "The LCA problem revisited," in *Proceedings of the 4th Latin American Theoretical Informatics (LATIN)*, 2000, pp. 88–94.

[4] C. Böhm and C. Plant, "HISSCLU: a hierarchical density-based method for semi-supervised clustering," in *Proceedings of the 11th International Conference on Extending Database Technology (EDBT)*, vol. 261, 2008, pp. 440–451.

[5] P. B. Callahan, "Dealing with higher dimensions: The well-separated pair decomposition and its applications," PhD thesis, 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," *Journal of the 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 *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD*, ser. Lecture Notes in Computer Science, vol. 7819, Springer, 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 Min. Knowl. Discov.*, vol. 27, no. 3, pp. 344–371, 2013.

[9] ——, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 1, 5:1–5:51, 2015.

[10] G. Cattaneo, P. Faruolo, U. F. Petrillo, and G. F. Italiano, "Maintaining dynamic minimum spanning trees: An experimental study," *Discrete Applied Mathematics*, vol. 158, no. 5, pp. 404–425, 2010.

[11] A. Cavalcante Araujo Neto, J. Sander, R. Campello, and M. Nascimento, "Efficient computation and visualization of multiple density-based clustering hierarchies," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–1, 2019. DOI: 10.1109/TKDE.2019.2962412.

[12] S. Chan, I. Diakonikolas, R. A. Servedio, and X. Sun, "Near-optimal density estimation in near-linear time using variable-width histograms," in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, 2014, pp. 1844–1852.

[13] X. Chen, Y. Min, Y. Zhao, and P. Wang, "GMDBSCAN: multi-density DBSCAN cluster based on grid," in *Proceedings of the IEEE International Conference on e-Business Engineering (ICEBE)*, 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 E. Karra Taniskidou, *UCI machine learning repository*, 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 *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996, pp. 226–231.

[17] K. R. Gabriel and R. R. Sokal, "A new statistical approach to geographic variation analysis," *Systematic Biology*, vol. 18, pp. 259–278, 1969.

[18] J. C. Gertrudes, A. Zimek, J. Sander, and R. J. G. B. Campello, "A unified view of density-based methods for semi-supervised clustering and classification," *Data Min. Knowl. Discov.*, vol. 33, no. 6, pp. 1894–1952, 2019.

[19] J. C. Gower and G. J. S. Ross, "Minimum spanning trees and single linkage cluster analysis," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 18, no. 1, pp. 54–64, 1969.

[20] J. Handl and J. Knowles, *Cluster generators for large high-dimensional data sets with large numbers of clusters*, 2005.

[21] J. A. Hartigan, *Clustering Algorithms*, 99th. USA: John Wiley & Sons, Inc., 1975, ISBN: 047135645X.

[22] K. A. Heller and Z. Ghahramani, "Bayesian hierarchical clustering," in *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, L. D. Raedt and S. Wrobel, Eds., vol. 119, ACM, 2005, pp. 297–304.

[23] M. R. Henzinger and V. King, "Maintaining minimum spanning trees in dynamic graphs," in *Proceedings of the 24th International Colloquium on Automata, Languages and Programming (ICALP)*, ser. Lecture Notes in Computer Science, vol. 1256, 1997, pp. 594–604.

[24] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD)*, 1998, pp. 58–65.

[25] G. A. II, G. Cattaneo, and G. F. Italiano, "Experimental analysis of dynamic minimum spanning tree algorithms (extended abstract)," in *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, M. E. Saks, Ed., 1997, pp. 314–323.

[26] J. W. Jaromczyk and G. T. Toussaint, "Relative neighborhood graphs and their relatives," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1502–1517, 1992.

[27] A. Karami and R. Johansson, "Choosing dbscan parameters automatically using differential evolution," *International Journal of Computer Applications (IJCA)*, vol. 91, no. 7, pp. 1–11, 2014.

[28] K. Khare, "Integration and Evaluation of Different Kernel Density Estimates in Hierarchical Density-Based Clustering," Master's thesis, University of Alberta, 2016.

[29] D. G. Kirkpatrick and J. D. Radke, "A framework for computational morphology," Machine Intelligence and Pattern Recognition, vol. 2, pp. 217–248, 1985.

[30] C. Kuo, P. B. Walker, O. T. Carmichael, and I. Davidson, "Spectral clustering for medical imaging," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2014, pp. 887–892.

[31] *Late-Breaking Developments in the Field of Artificial Intelligence*, vol. WS-13-17, AAAI Workshops, AAAI, 2013.

[32] L. Lelis and J. Sander, "Semi-supervised density-based clustering," in *Proceedings of the 9th IEEE International Conference on Data Mining (ICDM)*, 2009, pp. 842–847.

[33] D. Moulavi, P. A. Jaskowiak, R. J. G. B. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the SIAM International Conference on Data Mining*, 2014, pp. 839–847.

[34] 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.

[35] A. C. A. Neto, M. A. Nascimento, J. Sander, and R. J. G. B. Campello, "Mustache: A multiple clustering hierarchies explorer," *Proc. VLDB Endow.*, vol. 11, no. 12, pp. 2058–2061, 2018.

[36] A. C. A. Neto, J. Sander, R. J. G. B. Campello, and M. A. Nascimento, "Efficient computation of multiple density-based clustering hierarchies," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2017, pp. 991–996.

[37] E. Parzen, "On estimation of a probability density function and mode," *Ann. Math. Statist.*, vol. 33, no. 3, pp. 1065–1076, Sep. 1962.

[38] K. Pearson, "Contributions to the mathematical theory of evolution," *Philosophical Transactions of the Royal Society of London. A*, vol. 185, pp. 71–110, 1894, ISSN: 02643820. [Online]. Available: `http://www.jstor.org/stable/90667`.

[39] A. Smiti and Z. Elouedi, "Dbscan-gm: An improved clustering method based on gaussian means and dbscan techniques," in *Proceedings of the 16th IEEE International Conference on Intelligent Engineering Systems (INES)*, 2012, pp. 573–578.

[40] Spotify, *Annoy: Approximate nearest neighbors oh yeah*, Available at `https://github.com/spotify/annoy`.

[41] I. Steinwart, "Fully adaptive density-based clustering," *Ann. Statist.*, vol. 43, no. 5, pp. 2132–2167, Oct. 2015.

[42] I. Steinwart, B. K. Sriperumbudur, and P. Thomann, *Adaptive clustering using kernel density estimators*, 2017. arXiv: `1708.05254 [stat.ML]`.

[43] W. Stuetzle, "Estimating the cluster tree of a density by analyzing the minimal spanning tree of a sample," *J. Classif.*, vol. 20, no. 1, pp. 025–047, 2003.

[44] G. R. Terrell and D. W. Scott, "Variable kernel density estimation," *Ann. Statist.*, vol. 20, no. 3, pp. 1236–1265, Sep. 1992.

[45] G. T. Toussaint, "The relative neighbourhood graph of a finite planar set," *Pattern Recognition*, vol. 12, no. 4, pp. 261–268, 1980.

[46] M. D. W. and S. R. R., "Properties of gabriel graphs relevant to geographic variation research and the clustering of points in the plane," *Geographical Analysis*, vol. 12, no. 3, pp. 205–222, 1980.

[47] J. Wetherell, *Java: Algorithms and data structure*, Available at `https://github.com/phishman3579/java-algorithms-implementation`.

[48] N. Zhong, Y. Li, and S. Wu, "Effective pattern discovery for text mining," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 1, pp. 30–44, 2012.

Appendix A

Computing Multiple HDBSCAN' Hierarchies

A.1 HDBSCAN' and the RNG

As discussed in Section 3.2, the clustering hierarchies constructed with HDBSCAN' offer a better interpretation of the density estimates along the straight paths between points and results in smoother density estimates. However, similarly to HDBSCAN*, the use of different values of *mpts* with HDBSCAN' can lead to different results, and choosing an appropriate value of *mpts* can be challenging. Therefore, users might still need to compute a collection of HDBSCAN' clustering hierarchies for a range of *mpts* values.

Unfortunately, our RNG-based approach presented in Chapter 4 is not applicable to HDBSCAN'. One can show that the *relative neighborhood graph* w.r.t. to a high value of *mpts* does not contain the necessary edges to compute the hierarchies w.r.t. to lower values of *mpts*. Thus, we have devised and evaluated a sophisticated adaptation of our RNG-based strategy that uses the Gabriel Graph to reduce the computational cost of computing multiple clustering hierarchies with HDBSCAN' – the technical details of this approach can be found in Appendix A.1. In our evaluation, we have observed that the strategy of computing the graph G' only outperforms the naive strategy of running HDBSCAN' for multiple values of *mpts* in certain scenarios. For

the most part, both strategies present an equivalent performance in terms of runtime. This equivalence is due to the very small number of edges in G' when compared to the RNG used in RNG-HDBSCAN*. Since the estimates in HDBSCAN' are smoother, there are fewer *mutual reachability distances* between different points with the same value and, consequently, the number of points that are *relative neighbors* is quite low when compared to the RNG generated according to the original definition of *mutual reachability distance*. Thus, even though it takes longer to compute G' than it takes to compute a single RNG w.r.t. one value of $mpts$, computing *minimum spanning trees* from G' is faster than doing so from the RNG w.r.t. to the original definition of HDBSCAN*. However, the performance gain in the MST computation is only enough to make the strategy of computing G' comparable to the naive strategy. These observations reinforce the relevance of HDBSCAN* and of our RNG-based strategy discussed in Chapter 4. Since replacing the density estimator in HDBSCAN* does not necessarily solve the problem of selecting a value of $mpts$, having a strategy to efficiently compute and explore clustering hierarchies w.r.t. multiple parameter settings is essential, and is only possible with the original definition of *mutual reachability distance* and our RNG-based approach.

Due to the properties of mrd'_{mpts}, our RNG-based strategy cannot be directly applied as-is to HDBSCAN'. In fact, it can be shown that a single RNG computed w.r.t. the highest value of $mpts$ in a range does not contain all the edges needed to compute the MSTs/hierarchies w.r.t. the lower values of $mpts$.

Figure A.1 shows the *relative neighborhood graph* of a 2-dimensional dataset with 8 points in the space of *mutual reachability distances* w.r.t. $mpts = 2$ using the adapted definition of mutual reachability distance, mrd'_{mpts}. In this case, the *relative neighborhood graph* coincides with its *minimum spanning tree*. Similarly, Figure A.2 shows the *relative neighborhood graph* for the same

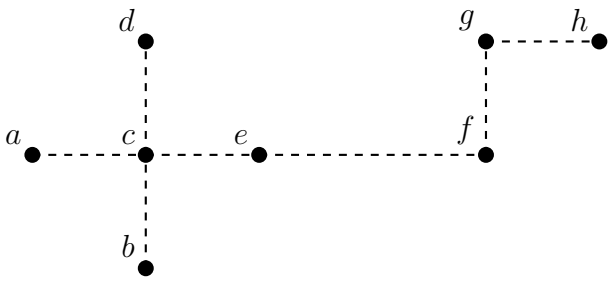


Figure A.1: RNG w.r.t. $mpts = 2$

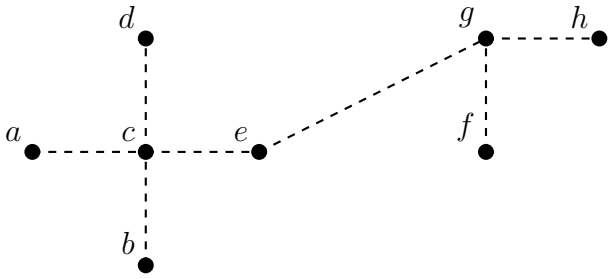


Figure A.2: RNG w.r.t. $mpts = 3$

dataset but w.r.t. $mpts = 3$ and edge weights computed using mrd'_{mpts}, which also coincides with its *minimum spanning tree*. Note that since the edge (e, f) is not in the *relative neighborhood graph* w.r.t. $mpts = 3$, one could not compute the MST for $mpts = 2$ from such *relative neighborhood graph*.

Therefore, in order to compute a collection of HDBSCAN' clustering hierarchies w.r.t. multiple values of $mpts$, one can either run HDBSCAN' multiple times, or try and compute a graph G' that corresponds to the union of the *relative neighborhood graphs* for all values of $mpts$ in a range (Equation A.1), from which all hierarchies w.r.t. to the range of $mpts$ values can be computed.

$$G' = \bigcup_{i \in [1, mpts]} RNG_i \tag{A.1}$$

However, finding all the edges of this graph can be computationally expensive and, if done naively, can be equivalent to computing the RNG once for each $mpts$ value of interest.

In RNG-HDBSCAN*, the strategy for computing the RNG consists of (1) computing a well-separated pair decomposition, (2) finding the symmetric bi-

chromatic closest neighbors (SBCN) between all well-separated sets and (3) filter the edges that do not belong to the RNG. Among these steps, the computation of the well-separated pair decomposition is the only step that does not depend on the value of *mpts* or on the definition of *mutual reachability distance*, and can be performed only once. However, the later two steps have to be performed for each value of *mpts* in the range of interest, which is computationally expensive. Most notably, the SBCN computation takes $\mathcal{O}(n^2)$-time for each value of *mpts*. Moreover, this process results in many edges between points that are not *relative neighbors* w.r.t. to any value of *mpts*, and these edges have to be later removed from the graph in step (3).

Based on the process described, one already expects that the time spent computing G' does not outweigh the performance gains from the computation of multiple MSTs w.r.t. a range of *mpts* values. As an attempt to speed up the construction of G', one can try and perform the SBCN computation for only part of the well-separated pairs while guaranteeing that all points that are potentially *relative neighbors* are connected in the resulting graph. Consider two well-separated sets A and B and their enclosing balls, as shown in Figure A.3. Before computing the SBCN between A and B, one can see if it is possible for two points $a \in A$ and $b \in B$ to be relative neighbors. This can be done by checking if there is *any* point in the region of the space between A and B, represented in Figure A.3 by the grey dotted circle. When that region is not empty, it is safe to discard the pair (A, B) as no points from these sets can be *relative neighbors*. Note that this spherical-shaped region between the sets A and B correspond to the region of the space that defines the Gabriel Graph, a super-graph of the Relative Neighborhood Graph (see Chapter 2). In our tests, we have used an approximate nearest neighbor algorithm based on random projection trees [40] to find the closest point to the center spherical region between both sets. In this case, using an approximate nearest neighbor

does not affect the correctness of the results, only the ability to discard pairs of *well-separated pairs* whose points cannot be *relative neighbors*.

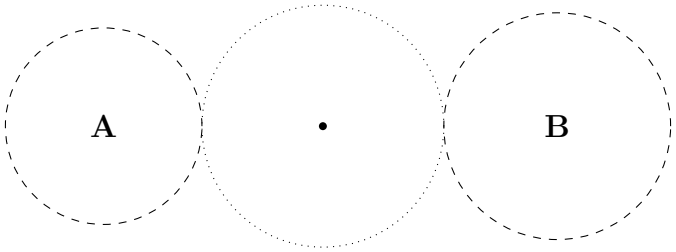


Figure A.3: Well-Separated Sets **A** and **B**