2021

K-NEAREST NEIGHBORS DENSITY-BASED CLUSTERING

Avory C. Bryant
Virginia Commonwealth University

DISSERTATION K-NEAREST NEIGHBORS DENSITY-BASED CLUSTERING

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

by

AVORY C. BRYANT

B.S., Virginia Commonwealth University, USA - August 1999 to May 2003

M.S., Virginia Commonwealth University, USA - August 2003 to May 2008

Director: Dissertation Krzysztof Cios,

Professor and Chair, Department of Computer Science

Virginia Commonwewalth University

Richmond, Virginia

August, 2021

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ALGORITHMS

Abstract

DISSERTATION *K*-NEAREST NEIGHBORS DENSITY-BASED CLUSTERING

By Avory C. Bryant

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at Virginia Commonwealth University.

Virginia Commonwealth University, 2021.

Director: Dissertation Krzysztof Cios,

Professor and Chair, Department of Computer Science

Traditional density-based clustering approaches rely on a distance-based parameter to define data connectivity and density. However, an appropriate value of this parameter can be difficult to determine as it is highly dependent on the underlying distribution of the data. In particular, distribution parameters affect the scale of inter-group distances (e.g., variance); this dependence leads to a well-known inability to simultaneously detect clusters at varying levels of density. In this work, connectivity and density are defined according to the rank-order induced by the distance metric (i.e., invariant to the expected scale of the distances). Connectivity by k-nearest neighbors and density by the number of reverse k-nearest neighbors (i.e., vertex in-degree in the directed k-nearest neighbors graph).

Two novel density-based clustering algorithms are proposed, the non-hierarchical *RNN-DBSCAN* and its hierarchical generalization *Hk-DC*. The advantage of *RNN-DBSCAN* is that it requires a single parameter k and is robust to varying levels of cluster density, whereas *Hk-DC* provides an efficient solution for producing a hierarchical clustering of *RNN-DBSCAN* solutions over k for a fixed density threshold.

Importantly, heuristics are proposed for selecting k and density threshold for *RNN-DBSCAN* and *Hk-DC*, along with a method for extracting a flat clustering solution from the hierarchy. Additionally, a cluster-dependent solution for handling noise is proposed.

CHAPTER 1

INTRODUCTION

1.1 Overview

Clustering is an important and challenging problem in machine learning [1] when there is no additional information about data elements (no labels). It is generally described as the process of grouping data elements, often according to the assumption that intra-element pairs (within-group) are more similar than inter-element pairs (between groups). More formally, assuming the data are a sample drawn i.i.d. from a mixture of component densities, clustering is the task of uncovering each component's support (a subset of sample elements). Applications of clustering can be found in almost any discipline; examples are image segmentation [2], document organization [3], analysis of gene expression data [4, 5, 6, 7, 8], business [9], and engineering [10, 11].

Clustering algorithms can be categorized according to differences in their definition of a cluster and their strategy to search for clusters (brute-force being intractable due to combinatorial explosion). A commonly used categorization identifies five, non-mutually exclusive, types of clustering algorithms: hierarchical [12], centroid-based [13], density-based [14], model-based [15], and grid-based [16, 17]. Additionally, clustering algorithms can be categorized according to the type of grouping performed, such as strict partitioning (elements belong to exactly one cluster) or overlapping (elements may belong to more than one cluster). This work focuses on a density-based clustering that groups data elements into a strict partitioning while also identifying noise (elements not belonging to any cluster).

In density-based clustering, clusters are defined as high-density regions separated by regions of low-density. Specifically, clusters are assumed to be disjoint, compact, and connected subsets of the embedding space for a given density level. Several desirable properties of density-based clustering include handling noise, discovering clusters with arbitrary shapes, and automatically discovering the number of clusters. Hierarchical clustering is represented as a rooted tree of clusters (tree vertices). The tree's root is a cluster containing all elements of a dataset, with descendant clusters being proper subsets of their ancestors. In other words, the hierarchy consists of multiple clusterings, such that as level (distance from the root) increases, clusterings become more refined (number of clusters increases).

This work focuses on non-parametric, level-set density-based clustering, which can be broken up into two tasks. First, non-parametric density estimation is applied to identify elements lying within dense regions. Second, traversals of dense elements define dense regions (clusters) using a definition of element reachability restricted to dense regions. For example, one such approach, popularized by *DBSCAN* [14], defines clusters as connected components of dense elements in the undirected ϵ-neighbors graph, density defined by vertex degree. A common criticism of such an approach is an inability to simultaneously discover clusters appearing at varying levels of density. This inability, in part being due to the use of a single distance-based parameter (ε) to define connectivity and density, ignoring differences in the scale of intra-group distances (i.e., local cluster density).

In contrast to *DBSCAN*, Reverse Nearest Neighbor-DBSCAN (*RNN-DBSCAN*) [18] is proposed, which defines connectivity and density by the directed k-nearest neighbor (kNN) graph and vertex in-degree (i.e., number of reverse k-nearest neighbors (RkNN) or k occurrences). The primary reasoning is that kNN graph connectivity and the in-degree density of a cluster are invariant to the scale of its intra-cluster

distances. Instead, it depends on the ordering induced by the distance metric. Additional reasoning being that k is generally considered easier to choose and does not tend to lead to disconnected graphs (as opposed to ε), though it is less geometrically intuitive. Finally, the size of the parameter input domain and solution range is reduced making exploratory analysis more tractable. Thus, the primary goal of this research can be described as investigating the benefits of kNN over ε-neighbors in density-based clustering, in addition to proposing novel kNN-based variants of existing ε-neighbors-based clustering algorithms.

As an appropriate choice of k remains sensitive to varying levels of cluster support (size), an extension of *RNN-DBSCAN*, Hierarchical k-Density Clustering (*Hk-DC*), is also proposed. Advantages of the hierarchical approach include the ability to discover cluster structure at varying levels of k, an improved ability to handle noise, and an overall better tool for exploratory data analysis. Beginning with a weighted, directed kNN graph (edges weighted according to kNN rank), *Hk-DC* defines density by in-degree, which is monotonically increasing for k. The k-density of an element is defined as the minimum value $k' \leq k$ at which an element is dense (i.e., the smallest k'NN graph in which the element is dense). For any value $k' \leq k$, a k-density clustering can be defined as the set of connected components in the dense-vertex subgraph consisting of vertices/edges whose k-density and edge-weight are greater than or equal to k' (i.e., the *RNN-DBSCAN* clustering at k'). Finally, hierarchical clustering is performed to produce all k-density clusterings over $k' = [1..k]$ (i.e., *Hk-DC* is a hierarchical clustering over connectivity parameter k). Thus, clusters of variable size are discovered at differing levels of k (e.g., larger clusters at larger values of k and vice versa).

In the remainder of this Chapter, a thorough discussion on the underlying assumptions and limitations of *DBSCAN* is given in Section 1.2, contributions of this

work are highlighted in Section 1.3, and underlying assumptions and limitations of the proposed clustering algorithms are presented in Section 1.4. A review of the related literature is given in Chapter 2, while Chapters 3 and 4 present the proposed clustering algorithms and experimental results. Finally, the findings of this work are summarized in Chapter 5.

1.2 Reconsidering *DBSCAN*

DBSCAN models data by connectivity/density parameter ε and density threshold parameter $MINPTS$, ε defining the neighborhood of an element (i.e., within an ε distance radius), and $MINPTS$ the minimum number of neighbors required for a dense element (i.e., considered to be lying within a dense region). Assuming a symmetric distance $d(\cdot,\cdot)$, let $G = (V, E)$ define the undirected ε-neighbors graph where $\forall\{u, v\} \in E : d(u, v) \leq \varepsilon$. The set of core (dense) elements is defined as the subset of V such that all members have at least $MINPTS$ neighbors in G, $Core = \{v \in V : |\{\{u, v\} \in E\}| \geq MINPTS\}$.

A clustering of core elements $\mathcal{C} = \{C_1, ..., C_\ell\}$ is defined by the set of connected components in the core-vertex-induced subgraph of G, $(Core, \{\{u, v\}) \in E : u, v \in Core\}) \subseteq G$ (i.e., $\mathcal{C}$ is a partitioning of $Core$). Clusters in $\mathcal{C}$ are expanded by border elements defined as the set of non-core elements belonging to the ε-neighborhood of any core element, $Border = \{v \in V \setminus Core : \cup_{\{u,v\}\in E}\{u\} \cap Core \neq \emptyset\}$. A border element $v \in Border$ is assigned to a cluster $C_i \in \mathcal{C}$ where $\exists u \in Core : u \in C_i \wedge \{u, v\} \in E$. As multiple such C_i's may exist for v, *DBSCAN* is non-deterministic for border elements with ties broken at random (dependent on data ordering). Remaining elements are identified as noise, $Noise = V \setminus (Core + Border)$. Ignoring the non-deterministic nature of assigning border elements to clusters, the minimum size of a cluster $C_i \in \mathcal{C}$ is $MINPTS$.

Recall *DBSCAN*'s inability to simultaneously discover clusters at varying levels of density. To simplify the discussion of this inability, let us begin by ignoring the existence of border or noise elements, in which case a ground truth clustering $\mathcal{C}$ is said to be simultaneously discoverable iff $\mathcal{C}$ is strictly simultaneously discoverable (Definition 1).

Definition 1 (Strictly Simultaneously Discoverable)**.** For ground truth clustering $\mathcal{C} = \{C_1, .., C_\ell\}$, $\mathcal{C}$ is strictly simultaneously discoverable by *DBSCAN* (with density function $f_\varepsilon(\cdot)$) iff ε and $MINPTS$ exists such that:

1. $\varepsilon \geq \varepsilon_{lb}$ where $\varepsilon_{lb} = \max(\varepsilon_{C_1}, .., \varepsilon_{C_\ell})$ and ε_{C_i} is equal to the maximum edge weight in the minimum spanning tree of C_i (i.e., the minimum value of ε such that all members of cluster C_i belong to the same connected component in the ε-neighbors graph).
2. $\varepsilon < \varepsilon_{ub}$ where $\varepsilon_{ub} = \min\{d(\boldsymbol{x}, \boldsymbol{y}) : (\boldsymbol{x} \in C_i \wedge \boldsymbol{y} \in C_{j \neq i}) \wedge C_i, C_j \in \mathcal{C}\}$ (i.e., no two clusters belong to the same connected component in the ε-neighbors graph)
3. $MINPTS \leq min\{\boldsymbol{x} \in \mathcal{C} : f_\varepsilon(x)\}$ (i.e, all elements are dense or $\forall \boldsymbol{x} \in \mathcal{C} : f_\varepsilon(x) \geq MINPTS$)

It follows from Definition 1 that $\mathcal{C}$ is strictly simultaneously discoverable by *DBSCAN* iff $\varepsilon_{ub} > \varepsilon_{lb}$ for $\varepsilon \in [\varepsilon_{lb}, \varepsilon_{ub})$ and $MINPTS \leq min\{\boldsymbol{x} \in \mathcal{C} : f_\varepsilon(x)\}$. In other words, the minimum inter-cluster distance must be larger than the maximum of the minimum intra-cluster distances required to connect all member elements of each cluster. Associating ε_{C_i} with the density of C_i (e.g., higher values of ε_{C_i} correspond to lower density), one can say that discoverability is dependent on the lowest density cluster in $\mathcal{C}$. Specifically, the lowest density cluster defines the threshold for inter-cluster distances (i.e., the minimum distance at which two clusters are considered

separable). As density and this threshold are negatively correlated (i.e., as density decreases, the threshold will increase), this dependence becomes increasingly troublesome as the range of cluster densities increases. For example, two highly dense clusters that are separable relative to their densities may be otherwise non-separable given the existence of a third cluster of lower density.

Commonly, parameter selection of *DBSCAN* is performed over ε for some fixed value of $MINPTS$, implicitly assuming the existence of an appropriate value of ε for any $MINPTS$. In Definition 1, one sees the reasoning behind such an assumption given the dependence between $MINPTS$ and ε (i.e., $MINPTS \leq min\{\boldsymbol{x} \in \mathcal{C} : f_\varepsilon(x)\}$). However, one also sees the flaw in this assumption as $MINPTS$ is bounded within the range defined by $\varepsilon \in [\varepsilon_{lb}, \varepsilon_{ub})$. Figure 1 highlights this fact where *DBSCAN* clustering performance in several cases is highly dependent on the choice of $MINPTS$ independent of ε. In any case, an intuitive positive correlation between ε and $MINPTS$ exists given that $\forall \boldsymbol{x} \in \mathcal{C} : \varepsilon \leq \varepsilon' \longleftrightarrow f_\varepsilon(x) \leq f_{\varepsilon^\varepsilon}(x)$. Note that the domain of ε may be defined discretely according to the pairwise distances of $\mathcal{C}$ as connectivity and density are only affected at these distances.

From Definition 1, one can also see the dependence of *DBSCAN* on the scale of intra-cluster distances of $\mathcal{C}$. Specifically, consider the minimum distance ε_C of a cluster $C \in \mathcal{C}$. Assume that distance $d(\cdot, \cdot)$ is absolute homogeneous (e.g., Euclidean) such that $\forall \boldsymbol{x}, \boldsymbol{y} \in C$ and $\alpha \in \mathbb{R}$, $|\alpha| d(\boldsymbol{x}, \boldsymbol{y}) = d(\alpha \boldsymbol{x}, \alpha \boldsymbol{y})$ where α controls the scale of intra-cluster distances in C. Note the orderings induced by $d(\cdot, \cdot)$ on C are invariant to the choice of α (i.e., $\forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in C : d(\boldsymbol{x}, \boldsymbol{y}) \leq d(\boldsymbol{y}, \boldsymbol{z}) \longleftrightarrow |\alpha| d(\boldsymbol{x}, \boldsymbol{y}) \leq |\alpha| d(\boldsymbol{y}, \boldsymbol{z})$). Consequently, the set of edges defining the minimum spanning tree of C are likewise invariant to α (assuming identical orderings in the case of ties), though $|\alpha|$ scales their weights. Thus, for any α, C is connected at minimum distance $|\alpha| \varepsilon_C$ (i.e., dependent on the scale of intra-cluster distances of C). In contrast, the kNN graph of C is

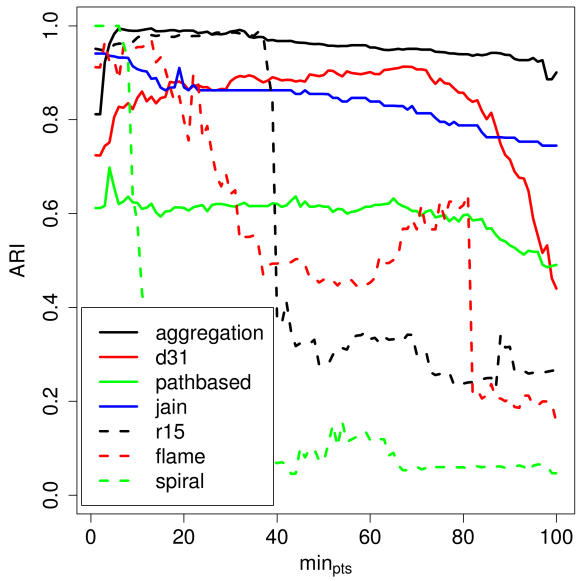


Fig. 1.: Adjusted Rand Index (ARI) performance of $DBSCAN$ for $MINPTS = 1..100$ for several synthetic datasets from Table 2. For each $MINPTS$, the value of ε was chosen by optimizing ARI.

identical for any α (i.e., invariant to the scale of intra-cluster distances of C).

Extending Definition 1, relax the strictly dense assumption by allowing for noise elements, continuing to ignore the handling of border elements. Now, a ground truth clustering $\mathcal{C}$ is said to be simultaneously discoverable iff $\mathcal{C}$ is relaxed simultaneously discoverable (Definition 2).

Definition 2 (Relaxed Simultaneously Discoverable)**.** For ground truth clustering $\mathcal{C} = \{C_1, .., C_\ell\}$, $\mathcal{C}$ is relaxed simultaneously discoverable by $DBSCAN$ (with density function $f_\varepsilon(\cdot)$) iff ε and $MINPTS$ exists such that:

1. $\mathcal{C}' = \{C'_1, .., C'_\ell\}$ such that $\forall i = 1..\ell : C'_i \subseteq C_i$ where $C'_i = \{\boldsymbol{x} \in C_i : f_\varepsilon(\boldsymbol{x}) \geq MINPTS\}$ and $C'_i \neq \emptyset$ (i.e., a non-empty subset of each cluster at density level $MINPTS$).

2. $\mathcal{C}'$ is strictly simultaneously discoverable (i.e., dense elements of $\mathcal{C}$ are strictly

simultaneously discoverable at density level $MINPTS$)

By introducing noise (Definition 2) *DBSCAN* relaxes the definition of simultaneously discoverable to allow for error (i.e., cluster elements incorrectly identified as noise, $N = \{\boldsymbol{x} \in \mathcal{C} : f_\varepsilon(\boldsymbol{x}) < MINPTS\}$). Note that if ground truth clustering $\mathcal{C}$ satisfies Definition 1, then $\mathcal{C}$ also satisfies Definition 2, else $MINPTS$ must be chosen such that $MINPTS > min\{\boldsymbol{x} \in \mathcal{C} : f_\varepsilon(x)\}$ regardless of ε. As such, $MINPTS$ controls the amount of error in a *DBSCAN* clustering (e.g., increasing $MINPTS$ increases the size of N). Following this reasoning, the optimal value of $MINPTS$ is the minimum value at which such a $\mathcal{C}'$ (Definition 2) exists (i.e., the clustering that minimizes the error or size of N). Note that even with the introduction of noise, a ground truth clustering $\mathcal{C}$ may exist such that $\mathcal{C}$ is not relaxed simultaneously discoverable (Definition 2). For example, the ground truth clustering $\mathcal{C}$ shown in Figure 2, which is not relaxed simultaneously discoverable for two reasons. First, for any ε, all elements in the green cluster are denser than ($f_\varepsilon(\boldsymbol{x})$ is greater than or equal to) all elements in the red cluster (i.e., for any value of $MINPTS$ that identifies a green element as noise, all red elements are identified as noise). Second, there exists an element $\boldsymbol{x}$ in the red cluster with the following properties: (1) $\boldsymbol{x}$ defines the minimum inter-cluster distance, and (2) for any ε, $f_\varepsilon(\boldsymbol{x})$ is greater than or equal to the density of all other elements in the red cluster.

The implicit assumption of *DBSCAN* is that by removing elements as noise in increasing order by density (e.g., by increasing $MINPTS$ or decreasing ε), one increases the upper bound of ε and/or decreases its lower bound (i.e., one assumes that a relaxed simultaneously discoverable solution is approached such that $\varepsilon_{ub} > \varepsilon_{lb}$ for $\mathcal{C}'$). Ignoring the requirement that noise elements are determined according to density, the optimal relaxed simultaneously discoverable solution (i.e., the solution

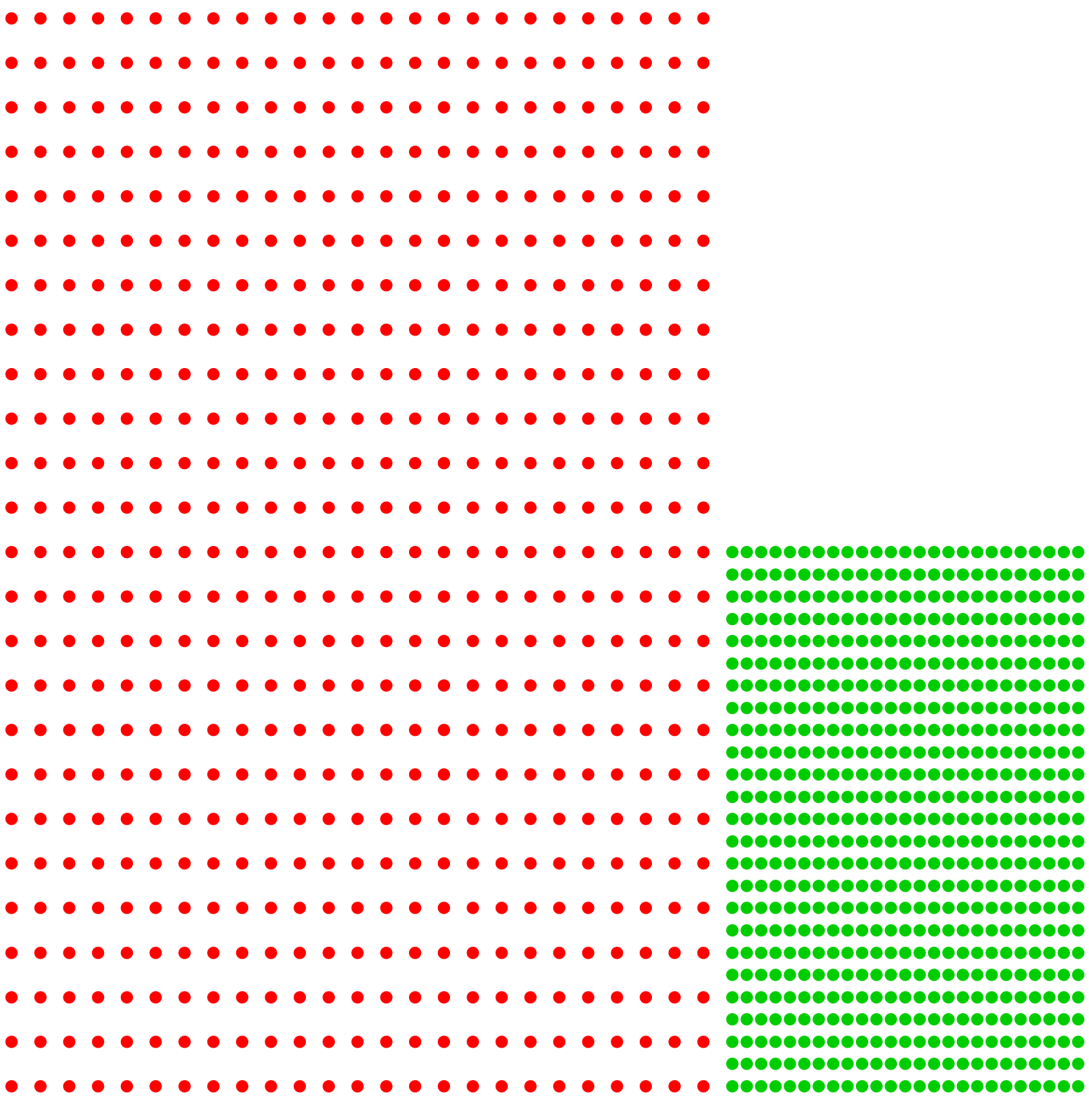


Fig. 2.: Example of a dataset with ground truth clustering $\mathcal{C}$ = (red elements, green elements) that is not relaxed simultaneously discoverable (Definition 2).

which minimizes the size of N) can be determined by directly removing elements that define the ε bounds (e.g., element $\boldsymbol{x}$ or $\boldsymbol{y}$, which define the minimum inter-cluster distance ε_{ub}). As such, in *DBSCAN*, the removal of low-density elements can be viewed as a heuristic for identifying such elements where ground truth is unknown. Expressly, one assumes that elements that define the ε bounds correspond to low-density elements.

In the remainder of this section, an argument is made for the correctness of this assumption. To simplify the discussion, as in Definition 1, assume $\varepsilon = \varepsilon_{lb} = \max(\varepsilon_{C_1}, .., \varepsilon_{C_\ell})$, $MINPTS = min\{\boldsymbol{x} \in \mathcal{C} : f_\varepsilon(x)\}$, and $\varepsilon_{lb} \geq \varepsilon_{ub}$ (i.e., $\mathcal{C}$ is not strictly simultaneously discoverable). Furthermore, assume that ε is fixed such that the amount of noise elements is increased by increasing $MINPTS$. Of course, one could fix $MINPTS$ and decrease ε, or adjust both simultaneously to obtain similar (increased noise) though non-identical results. Let $MINPTS$ be increased to $MINPTS' = MINPTS + 1$ with new upper and lower bounds ε'_{ub} and ε'_{lb}. At a minimum, one needs to make the argument that removing low-density elements (i.e., those below density level $MINPTS'$) is more likely to result in a case where $\varepsilon_{ub} < \varepsilon'_{ub}$ or $\varepsilon_{lb} > \varepsilon'_{lb}$ than removing elements at random.

For the upper bound (minimum inter-cluster distance), the argument is relatively straightforward. Note that by definition, minimum inter-cluster distance elements lie on the border between their respective clusters and that such elements should likewise, by definition, lie in regions of relatively low density (for their clusters). As such, by selecting a low-density element, one increases the likelihood of the selected element being a border element, which likewise increases the likelihood of said element defining the upper bound (i.e., those elements whose removal would increase the upper bound).

For the lower bound (maximum of the maximum edge weight in the minimum spanning tree of each cluster), the argument is more complex. Given cluster $C \in \mathcal{C}$

with minimum spanning tree $\mathcal{T}$ and maximum edge weight ε_C, removing element $\boldsymbol{x} \in \mathcal{C}$ decreases ε_C iff $\boldsymbol{x}$ belongs to a single edge in $\mathcal{T}$ whose weight is equal to ε_C, and the multiplicity of ε_C in the multiset of $\mathcal{T}$'s edge weights is equal to 1. In other words, when $\boldsymbol{x}$ belongs to multiple edges in $\mathcal{T}$, the maximum edge weight in the new minimum spanning tree is guaranteed to be greater than or equal to ε_C. However, if ε_C decreases given the removal of $\boldsymbol{x}$, then by definition, $\boldsymbol{x}$ has the largest 1-nearest neighbor distance in C. Note that density in *DBSCAN* can be equivalently defined by $MINPTS$-nearest neighbor distance, such that the lowest density element in C has the largest $MINPTS$-nearest neighbor distance. Thus, assuming a positive correlation between 1- and $MINPTS$-nearest neighbor distance (the degree of correlation decreasing as $MINPTS$ increases), one can say that removing the lowest density element is more likely to decrease the lower bound than random selection.

Similar reasoning can be applies in the kNN graph, where a clustering is discoverable where the minimum k required to ensure intra-cluster connectivity (k_{lb}) must be less than the minimum k of an inter-cluster edge (k_{ub}). Here the removal of low-density elements is expected to decrease k_{lb} or increase k_{ub}. The assumption being that those likely to have the above effect correspond to isolated vertices in the kNN graph (i.e., those with low in-degree). Finally, the handling of border elements (assignment of noise elements to clusters) can be viewed as a heuristic for further reducing error (number of noise elements) after cluster discovery. In general, the interpretation of noise should be consistent with their usage (i.e., increasing cluster separability). In other words, though they may be viewed as outliers, this interpretation is inconsistent with their usage.

1.3 Contributions

Due to the popularity of *DBSCAN* (owing to its simplicity), there exists a long list of related works seeking to improve upon its results (see Section 2.3). This work follows along this vain by suggesting two novel level-set, density-based clustering algorithms. First, *RNN-DBSCAN* is introduced, which replaces ε-neighborhoods connectivity with kNN and total-degree density by in-degree (RkNN). Novel contributions of this work include the following:

- Empirical demonstration that the model could be reduced to a single parameter k, using expected in-degree (k) as a natural choice of the density threshold.
- Introduction of a cluster-dependent expansion procedure, with expansion radius dependent on local cluster density.
- Introduction of two heuristics for the selection of an optimal k based on model stability and an internal evaluation measure.
- Empirical demonstration of the applicability of approximate kNN to improve run-time complexity with minimal impact to clustering performance.

Second, the *Hk-DC* algorithm is introduced, which provides a hierarchical solution for efficiently computing *RNN-DBSCAN*-like clusterings over a range of connectivity parameter k, reintroducing the density threshold parameter. Novel contributions of this work include the following:

- Introduction of a kNN graph structure and density measure for efficiently computing *RNN-DBSCAN*-like clusterings over a range of connectivity parameters k.
- Introduction of a non-distance-based, recursive cluster expansion procedure.

- Introduction of a heuristic for selecting a final flat clustering over a density threshold range based on cluster persistence.

1.4 Assumptions & Limitations

As is generally the case with real-world data, in clustering, one implicitly assumes that elements form natural groups, where each group is assumed to be dependent on a unique component distribution. Furthermore, in distance-based clustering, one assumes a meaningful distance measure useful in distinguishing between groups. Specifically, one assumes the scale of inter-group distances dominates that of intra-group distances (i.e., groups are separable within the data embedding). However, this assumption becomes increasingly tenuous as dimensionality increases due to various reasons (e.g., poor discrimination of distances and the presence of irrelevant or redundant features in the embedding [19]). In particular, poor discrimination due to the concentration of distances phenomenon where distance becomes indiscernible (i.e., the curse of dimensionality). For example, as dimensionality increases, an element's nearest neighbor distance approaches its farthest neighbor distance [20].

Given these concerns, a common approach is to apply unsupervised dimensionality reduction to the data. For example, using matrix factorization (PCA [21], manifold learning (UMAP [22]), or Autoencoders [23]. Note that here one primarily addresses the problem of redundant features in the embedding, the goal being to reduce the data embedding to its lower intrinsic dimensionality such that distances are meaningful. Another approach is subspace clustering, where distance becomes dependent on local feature selection/transformation of the embedding [16, 24, 25]. In this case, one primarily addresses the problem of irrelevant features, the goal being to limit distance calculations to a subset of group-dependent relevant features.

This work does not directly address these concerns, though undoubtedly, some

solutions may be directly applicable (e.g., by first applying PCA to the data). Instead, a discussion on the effects of high-dimensionality on the kNN graph is provided. In [20], the question of when nearest neighbors queries are meaningful in high dimensional embeddings is addressed, specifically, scenarios likely to retain good separation between the farthest and nearest neighbors. Such scenarios include instances where the query is increasingly close to a sample element and falls within some group. Note that both scenarios correspond to the case of clustering using the kNN graph.

Such queries remain meaningful due to the effect of distance concentration on multimodal data (i.e., assuming natural groups). Specifically, while intra-group distances do become meaningless (each group corresponding to a unimodal distribution), inter-group distances retain meaning. Thus, nearest neighbors are likely to belong to the same group, whereas the ordering of intra-group neighbors becomes increasingly meaningless (e.g., the ordering of k-nearest neighbors). However, this does affect the use of vertex in-degree to estimate density.

In [26], this phenomenon is further investigated for the distribution of RkNN (referred to as k-occurrences) as dimensionality increases. Specifically, in low-dimensions, this distribution resembles a random graph model (i.e., Erdos-Renyi, which is binomial and Poisson in the limit), whereas dimensionality increases the distribution skews to the right, becoming log-normal. The positive skew is due to the emergence of hubs, elements with a high number of RkNN (i.e., popular nearest neighbors). In multimodal data, hubs correspond to elements closer to the mean of a component distribution, expected distance to the mean becoming smaller than the distance to other elements (i.e., spatial centrality becomes amplified). Note that anti-hubs also emerge with low in-degree that are farthest away from their component's mean.

In distance-based clustering, this is problematic as hubs exhibit both relatively

low intra- and inter-group distances (i.e., they are close to both inter- and intra-group elements). Thus, the effect of hubs on clustering requires further investigation, specifically with respect to the distance induced ordering of their intra- versus inter-group elements.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This chapter presents a review of the related literature, with a primary focus on level-set density-based clustering and the kNN graph. Section 2.2 provides an overview of density-based clustering approaches, and Section 2.3 a more thorough discussion on level-set density-based clustering approaches. Section 2.4 discusses the related field of outlier detection, while Section 2.5 discusses kNN graph construction cost and connectivity properties. Finally, Section 2.6 provides a formal discussion on non-parametric density estimation techniques seen in level-set density-based clustering.

2.2 Density-based Clustering

Density-based clustering can be described as a two-step procedure, density estimation followed by cluster discovery (see Figure 3 for an overview of approaches). For density estimation, algorithms can be categorized as parametric or non-parametric. In parametric, density estimation is performed using a finite mixture model where component densities are assumed to be of parametric form (e.g., Gaussian). As an example, assuming some number of clusters, model parameters can be estimated using the EM algorithm [15] (density estimation), with each element being assigned to a component using Bayes rule (cluster discovery). The parametric form of density-based clustering is commonly referred to as model-based clustering [27].

In non-parametric, density estimation is not restricted by assumptions regard-

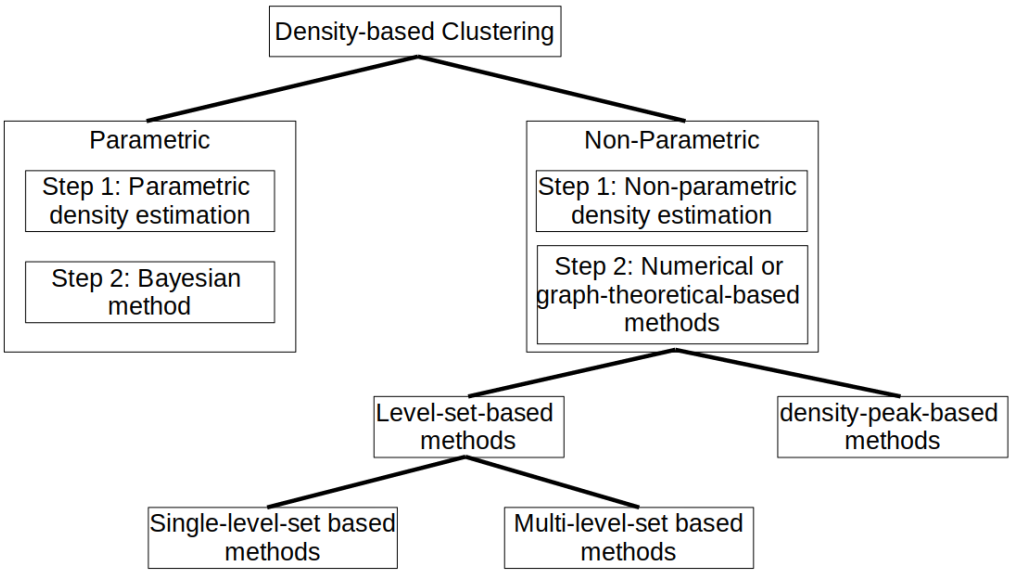


Fig. 3.: Overview of density-based clustering approaches.

ing the parametric form of the underlying density function (e.g., histograms, naive estimator, kernel estimator, or nearest neighbor method [28]). Such estimates are not parameter-free (e.g., bandwidth for kernel estimators or k for nearest neighbor methods); instead, parameters result in less rigid distributional assumptions of the data. Specifically, the data are not assumed to be drawn from a known parametric family of distributions. For example, in a kernel estimator, the bandwidth parameter controls the estimator's smoothness, a trade-off between the estimator's bias and variance. Note that for efficiency, density estimates with compact support are commonly preferred (i.e., estimates whose support size are not of order n).

Non-parametric density estimate approaches can be further categorized according to their approach to cluster discovery, identifying level-sets (density borders) or density peaks (mode seeking). In density peak-based clustering, cluster discovery is performed dynamically by iteratively moving elements along the direction of the steepest density ascent, where clusters are identified as the set of elements converging

to the same density peak. For example, gradient- and non-gradient-based algorithms seen in [29, 30, 31].

Level-set-based clustering, popularized in [14], was first observed as a solution to the chaining phenomenon in single-linkage clustering [32, 33] and later formalized in [34]. Here a cluster is defined as a set of maximally connected elements above a density threshold, with all such sets defining a clustering. In other words, given a non-parametric density estimate and definition of element connectivity (e.g., a kNN graph), a clustering is defined as the set of connected components in the dense vertex subgraph of the element connectivity graph. For example, [14] defines element connectivity by an ε-neighbors graph and element density by vertex degree. Such a density estimate is equivalent to using a uniform or boxcar kernel estimator, with bandwidth set to ε. Note that for efficiency, the connectivity graph generally defines the support of each element's density estimate.

The above description of level-set-based clustering corresponds to a single-level case, which may be extended to the multi-level case using hierarchical clustering. Naively, this can be performed by single-level-set clustering over a range of density thresholds given a fixed connectivity parameter value (e.g., ε in the case of an ε-neighborhood graph), or range of connectivity parameter values given a fixed density threshold. Efficient examples of the latter are described in [35, 36, 37].

2.3 Level-Set-Based Methods

As previously stated, level-set-based methods were initially proposed to solve the chaining phenomenon in single-linkage clustering [32, 33]. An example of the chaining phenomenon is when two clusters lying in distinct high-density regions are connected by a chain of elements traversing a low-density region, resulting in the two separate clusters incorrectly identified as a single cluster. To address this phenomenon, [32]

suggested limiting single-linkage clustering to elements satisfying a minimum density requirement (number of neighboring elements within a distance threshold). Similarly, [33] extended single-linkage clustering with the condition that both elements of a minimum distance split had to satisfy a minimum density requirement (number of neighboring elements within a distance equal to their pairwise distance).

These earlier concepts were later extended and formalized in [34], which described a density-contour clustering (Definition 3) and density-contour tree in the context of single-linkage clustering.

Definition 3 (Density-Contour Cluster)**.** Let $\boldsymbol{\mathcal{X}}$ be a set of elements and function $f(\boldsymbol{x})$ a density function for elements $\boldsymbol{x} \in \boldsymbol{\mathcal{X}}$. A density-contour cluster $\boldsymbol{C}$ at density level λ is a non-empty subset of $\boldsymbol{\mathcal{X}}$, $\boldsymbol{C} \subseteq \boldsymbol{\mathcal{X}}$, satisfying the following conditions:

1. $\forall\ \boldsymbol{x} \in \boldsymbol{C} : f(\boldsymbol{x}) \geq \lambda$ (density level)
2. $\forall\ \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{C}$: $\boldsymbol{x}$ and $\boldsymbol{y}$ are linked by a path whose members all lie in $\boldsymbol{C}$ (connected)
3. $\forall\ \boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{\mathcal{X}}$: if $\boldsymbol{x} \in \boldsymbol{C}$ and $\boldsymbol{x}$ and $\boldsymbol{y}$ are linked by a path whose members $\boldsymbol{z}$ all satisfy $f(\boldsymbol{z}) \geq \lambda$, then $\boldsymbol{y} \in \boldsymbol{C}$. (maximal)

For cluster $\boldsymbol{C}$ at density level λ, the density inside $\boldsymbol{C}$ is no less than λ, and for every path connecting $\boldsymbol{x} \in \boldsymbol{C}$ to $\boldsymbol{y} \notin \boldsymbol{C}$, the density somewhere along the path is less than λ. Thus, cluster $\boldsymbol{C}$ conforms to the requirement that $\boldsymbol{C}$ is a high-density region surrounded by lower-density regions. A density-contour clustering at density level λ is the collection of all such maximal connected subsets covering the set of elements $\{\boldsymbol{x} \in \boldsymbol{\mathcal{X}} \mid f(\boldsymbol{x}) \geq \lambda\}$ (i.e., a clustering of the λ level-set). A density-contour tree is a tree of nested density-contour clusters constructed by varying density threshold λ. Increasing λ refines the clustering while decreasing λ coarsens the clustering (i.e., a

density-contour tree defines a hierarchical density-based clustering or a multi-level-set clustering).

While Definition 3 directly assumes a given density estimate, there is an implicit assumption of connectivity required to define paths between elements, realized in practice by some definition of an element's neighborhood (e.g., ε-neighbors or kNN). Thus, Definition 3 has a simple graph interpretation given the resulting neighborhood graph where a density-contour cluster is a connected component (i.e., connected and maximal) in the λ level-set vertex-induced subgraph (i.e., density level). As the prototypical example of a single-level-set method, DBSCAN extended Definition 3 to include non-level-set elements (border) while presenting an $\mathcal{O}(n \log n)$ scan procedure for computing a density-contour clustering (complexity dependent on dimensionality ant the choice of ε). Concerning the former, level-sets simplify the clustering problem by focusing on dense elements, effectively increasing the cluster separability. However, the drawback is that non-dense elements are considered noise and remain unclustered. Thus, a typical third step is to expand a density-contour clustering with elements that border neighboring clusters (i.e., bordering elements are assigned to clusters).

Concerning complexity, the scan procedure of DBSCAN performs at most n breadth-first traversals that are restricted to elements not visited by a prior traversal. In other words, each element is visited exactly once and requires an ε-neighbors query to continue the traversal (i.e., n ε-neighbors queries). Thus, given a spatial index of $\boldsymbol{\mathcal{X}}$ with $\mathcal{O}(\log n)$ ε-neighbors query time complexity, the complexity of DBSCAN is $\mathcal{O}(n \log n)$. Given its popularity, most prior work in level-set clustering seek to improve upon DBSCAN, though they might more correctly be considered variants of density-contour clustering or trees.

A categorization of level-set clustering research includes approaches for handling varying levels of cluster density (i.e., multi-level-set), improving computational effi-

ciency [17, 29, 38, 39, 40, 41, 42, 43, 44, 45, 46], parameter reduction/selection [35, 47, 48], handling high-dimensional data [24, 25, 49], and domain-specific solutions [50, 51] (e.g., streaming or non-spatial proximity graphs). Note that these categories are not necessarily exhaustive or mutually exclusive. Furthermore, the focus here is on handling varying levels of cluster density, which can be further categorized into hierarchical [35, 52, 53, 40, 41, 36, 37, 54] and non-hierarchical [48, 51, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 18, 65, 66, 67] approaches to multi-level-set clustering.

Non-hierarchical solutions to the varying levels of cluster density problem focus on identifying homogeneous density clusters and using density estimates robust to variation in local density. Technically, the former case is not a form of level-set clustering, though it is included here as it is usually contrasted with *DBSCAN*. For this case (as described above), a cluster is redefined as a connected set of vertices of homogeneous density. Clusters are formed by neighborhood graph traversals that terminate at vertices whose inclusion significantly affects current cluster density (i.e., does not fit the cluster's density distribution). In other words, the density threshold (level-set requirement) is replaced by a threshold defining significant change in density. Here elements are generally processed by decreasing density (i.e., clusters are discovered by increasing density). Thus, clusters are identified at varying levels of density corresponding to a form of multi-level-set clustering.

For example, in [55], elements are processed in descending order by kNN distance. Like DBSCAN, a kNN graph graph traversal is performed for each unvisited element that terminates at vertices whose density is below some threshold of the cluster's center vertex density. A center vertex is the initial element of the cluster (i.e., the vertex with the highest density). As another example, in [48], a cluster is defined as a set of elements whose observed nearest neighbor distribution fits its expected probability distribution within some confidence level. Cluster traversals are performed by

ε-neighbors queries where ε is dependent on the cluster's current expected nearest neighbor distance distribution and terminates at vertices that fail the above distribution check. For a third example, in [58], a vertex's neighborhood is only used in a cluster traversal if the densities of its neighbors are within some threshold of its density (i.e., the homogeneous density check is performed locally to the neighborhood of a vertex).

A density estimate robust to variation in local density refers to an estimate that is adaptive to fluctuations in local density (i.e, intended to be comparable across an entire sample). As an example, consider some unimodal distribution; such an estimate would be one whose expected mean value is to some degree robust to changes in variance or sample size. In other words, estimates from two such distributions with different variance and sample size would remain comparable. An example of such an estimate is the number of RkNN, where an element's density depends on its neighbors' local density and is independent of its local density (see Section 2.6). Similarly, the number of mutual k-nearest neighbors is dependent on the local density of its neighbors and its local density. Additionally, for RkNN, the estimate's expected value, upper bound, and distribution are all to some degree independent of distributional parameters and sample size (e.g., the expected value is k, and the upper bound is dependent on dimensionality and k, see the kissing number problem). Note mutual k-nearest neighbors exhibit similar desirable properties for comparability (e.g., the upper bound is k).

Other robust estimates can be found in local outlier detection (Section 2.4), though most of these correspond to normalized local density measures, normalized according to the local density of neighboring elements (i.e., they are measures of density homogeneity). As examples in level-set clustering, in [62, 18] and [63, 64], an element's density is defined by the number of RkNN and mutual k-nearest neighbors

in the kNN graph. Similarly, in [59], density is defined as a function of mutual k-nearest neighbors and their ranks. Whereas in [57], density is defined by local outlier factor [68, 69], where cluster traversals are restricted by a change in vertex density (i.e., a combination of both non-hierarchical approaches).

Hierarchical approaches can be differentiated from other approaches by generating a hierarchical clustering structure - specifically, a hierarchical structure for extracting single-level-set clusterings over various densities (i.e., multi-level-set). An example of a hierarchical approach is *OPTICS* [35, 52] which generates a type of dendrogram (reachability plot) of *DBSCAN* clusterings over a range of ε, using the concepts of core and reachability distance. The core distance of an element is defined by its $MINPTS$-nearest neighbor distance (i.e., the minimum ε at which the element is in the core set). The reachability distance of element $\boldsymbol{x}$ from $\boldsymbol{y}$ is defined as the maximum of $d(\boldsymbol{x}, \boldsymbol{y})$ and the core distance of $\boldsymbol{y}$ (i.e., the minimum ε where $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the same cluster).

Given $MINPTS$ and an upper bound of ε, let $\boldsymbol{G}$ be the weighted directed graph defined by reachability distance. Like *DBSCAN*, *OPTICS* performs a scan procedure using reachability distance for each unvisited vertex v. This procedure is equivalent to computing the spanning arborescence (rooted directed out-tree) $\boldsymbol{A}$ rooted at v with minimum weight terminating at non-core vertices. An ordering of vertices is returned by the weighted path from v in $\boldsymbol{A}$, along with vertex minimum reachability distance, defined by a vertex's reachability distance to its parent in $\boldsymbol{A}$. Plotting minimum reachability distance by the returned vertex ordering (i.e., the reachability plot), clusters are identified as valleys in the plot where deeper valleys indicate increased density. A *DBSCAN* clustering at density $\varepsilon' \leq \varepsilon$ is identified as contiguous regions in the plot having a minimum reachability distance less than or equal to ε'.

An extension of *OPTICS* is *HDBSCAN* [36, 37], which introduced mutual reach-

ability distance. Mutual reachability distance between elements $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as the maximum of the $MINPTS$-nearest neighbor distance of $\boldsymbol{x}$ and $\boldsymbol{y}$ and distance $d(\boldsymbol{x}, \boldsymbol{y})$. Given $MINPTS$, let $\boldsymbol{G}$ be the weighted undirected graph defined by mutual reachability distance. A hierarchical density-based clustering over ε is produced by performing single-linkage clustering on $\boldsymbol{G}$, by computing its minimum spanning tree and iteratively removing edges in descending order by edge weight. This hierarchical clustering contains all *DBSCAN* clusterings of core elements over ε. Additionally, [37] presents a heuristic for extracting a flat clustering from a hierarchical, density-based clustering based on the concept of cluster stability. Extending [37], [70] presents a method for selecting a final flat clustering over a range of density threshold values based on hierarchy similarity.

2.4 kNN-based outlier detection

Similar to the clustering problem, outlier detection identifies outlier elements in data given some assumption describing the property(s) that an outlier exhibits [61]. For example, kNN-based outlier detection approaches use the same concept of density seen in *DBSCAN* [71, 72, 73, 74, 75]. A commonly used categorization of outlier detection approaches includes classification-, clustering-, kNN-, and statistical-based approaches [76]. This section focuses on kNN-based outlier detection, commonly referred to as local outlier detection. In [77], an overview of kNN-based local outlier detection approaches is given and defined as relative measures of local density (i.e., an element's local density relative to the local density of its neighboring elements). In other words, outlier measures that are robust to variation in local density. Hence, the relationship to density estimates that are robust to varying levels of cluster density.

As an example, the popular Local Outlier Factor (LOF) [68, 69, 78] and its variants [77, 79, 80, 81, 82], which is a measure of density homogeneity, or relative

measure of kNN distance local density (i.e., the local density of an element normalized by the local densities of its neighbors). Similar to the approach taken here, RkNN approaches are presented in *ODIN* [83] and *Antihub* [84], the latter presenting a relative RkNN measure or the mutual k-nearest neighbors approaches presented in *INFLO* [85], *SCAN* [51]. Examples of other measures include those which are rank-based (as opposed to distance) [86, 87], path-based [88], and those focused on high-dimensional data [89].

An interesting categorization of these approaches lies in their dependence on the size of the neighborhood in the kNN. For example, an element's *LOF* score is dependent on its two-hop neighborhood (i.e., paths of length two originating from the element). However, with respect to the work here, many are not monotonically increasing with k, making them difficult to model hierarchically (i.e., over a range of k).

2.5 kNN Graph Construction and Properties

As with most kNN-based algorithms, the complexity of the proposed clustering algorithms is bounded by the cost of constructing the kNN graph. This complexity is dependent on the solution to the nearest neighbor search problem and its generalization to kNN. The naive solution to this problem is a simple sequential scan which is linear $\mathcal{O}(n)$. Thus, a desirable (efficient) solution to this problem is sub-linear. Note that for the kNN graph one requires the all-pairs solution to this problem, which is naively $\mathcal{O}(kn^2)$ (i.e., a desirable solution being sub-quadratic for n).

In the case of low-dimensional data ($d \leq 2$), numerous $\mathcal{O}(n \log n)$ solutions exist for this problem [42, 43], $\mathcal{O}(kn \log n)$ for kNN. Similarly, for higher dimensions, numerous space- and data-partitioning solutions with average case $\mathcal{O}(n \log n)$ complexity exists (e.g., *R^*-tree* [90, 91], *ball-tree* [92], *kd-tree* [93] and *cover-tree* [94]).

Note that we do not differentiate between metric versus non-metric spaces, though solutions for the latter are more difficult to design given the lack of generic properties (e.g., triangle inequality). Unfortunately, the performance of such approaches degrades rapidly as dimensionality increases, rarely outperforming sequential scan in relatively low-dimensions (e.g., $d > 10$ [95]). As an example, the authors' of *DBSCAN* suggests an R^*-*tree* for an efficient solution to the related ε-neighbors search problem in the case of low-dimensional data and relatively small values of ε (small values of ε likely corresponding to cluster solutions of interest [96, 97]), making an argument for $\mathcal{O}(n \log n)$ complexity.

Approximate solutions have been presented as no efficient (sub-quadratic) solution exists to the exact all-pairs nearest neighbor problem in high-dimensions; approximate referring to the expected accuracy (recall) of a k-nearest neighbors query. For example, the greedy nearest neighbor propagation approaches introduced in [98, 99, 100, 101]. Given the significance of the problem, numerous approximate solution exist approaching or surpassing $\mathcal{O}(n \log n)$ complexity (see the benchmark [102]). Additionally, the suitability of approximate solutions to density-based clustering has been previously demonstrated [42, 43, 49, 18].

For clustering, an important question involves the expected connectivity of the kNN graph for k as n increases. Specifically, given multi-modal data of size n, one is interested in the lower bound of k such that with high probability, each group is connected (i.e., all group elements belong to a single connected component). Note that this problem may be considered for the symmetric (weakly connected), directed (strongly connected), or mutual k-nearest neighbors graphs and that the value of k increases such k symmetric $\leq k$ directed $\leq k$ mutual.

In all cases, this value is known to be of the form $k \sim \log n$ [103, 104, 105], equivalent to the value of k for the connectedness of a uni-modal dataset. However,

numerous proofs have shown $k \sim c \log n$ such that $c < 1$ under various conditions [104, 105]. Furthermore, k can be further reduced for the purposes here due to the emergence of a giant connected component, where the maximum size of isolate components (i.e., those not belonging to the connected component) have been shown to decrease rapidly as k increases. The latter point is important given a minimum cluster size (i.e., isolated components do not result clusters).

2.6 Non-Parametric Density Estimation

Level-set density-based clustering algorithms rely on non-parametric methods to estimate the density of an element from a given sample. This section presents several estimators, adapted from the univariate discussion of [106] (generalizable to multivariate), for commonly used densities seen in level-set clustering. Perhaps the most used definition of an element's density is the number of sample elements within an ε-neighbors neighborhood centered at the element. A Level-set is defined as the subset of elements with a density greater than or equal to $MINPTS$ (i.e., the $MINPTS$ density level-set). For simplicity, this definition is referred to as the *DBSCAN* density estimator.

Begin by assuming sample X of size n containing element x. The *DBSCAN* density estimator is equivalent to a kernel estimator (Equation 2.1) with a uniform (box) kernel (Equation 2.2) and bandwidth $h = \varepsilon$.

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \tag{2.1}$$

$$K(x) = \begin{cases} \frac{1}{2}, & \text{if } |x| \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{2.2}$$

For Equation 2.1, an element belongs to the $MINPTS$ density level-set if $\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right) \geq \frac{MINPTS}{2}$ (i.e., at least $MINPTS$ sample elements are within bandwidth h of x). Accordingly, $MINPTS$ density level-set indicator function can be defined as $\hat{f}(x) \geq \frac{MINPTS}{2nh}$ or simply $\hat{f}(x) \geq MINPTS$ for the unnormalized density as commonly used in level-set clustering. The unnormalized density being adequate for finding samples lying in high-density regions. Note that the use of a compactly supported kernel (e.g., uniform) has computational advantages. Specifically, the density of an element is dependent on the subset of sample elements within the bandwidth, where the kernel is 0 for all other elements.

Similarly, the *DBSCAN* density estimator can be defined as the distance to an element's kNN in the sample where $k = MINPTS$. The level-set of dense sample elements being defined as the subset with kNN distance less than or equal to ε (i.e., the ε density level-set). This interpretation is equivalent to the kNN density estimate (Equation 2.3) with $k = MINPTS$.

$$\hat{f}(x) = \frac{k}{2nd_k(x)} \tag{2.3}$$

Note in Equation 2.1 $d_k(x)$ defines the distance from x to its kNN in the sample and is inversely proportional to the minimum bandwidth required to contain k sample elements when centered at x. Furthermore, assuming a uniform kernel, Equation 2.3 is a special case of the generalized kNN estimator (Equation 2.4). Nearest neighbor estimators adapt smoothing to the local density of a sample element where k controls the degree of smoothing.

$$\hat{f}(x) = \frac{1}{nd_k(x)} \sum_{i=1}^{n} K\left(\frac{x - X_i}{d_k(x)}\right) \tag{2.4}$$

For Equation 2.3, an element belongs to the ε density level-set if $d_k(x) \leq \varepsilon$ (i.e.,

by definition, at least k sample elements are within $d_k(x)$ of x). Accordingly, the indicator function of the ε density level-set can be defined as $\hat{f}(x) \geq \frac{k}{2n\varepsilon}$ or simply $\hat{f}(x) \geq \frac{1}{\varepsilon}$ for the unnormalized density. The inversion of epsilon and reversal of the inequality is due to the inverse relationship between density and kNN distance (i.e., $\hat{f}(x) \geq \frac{1}{\varepsilon} = \frac{1}{\hat{f}(x)} \leq \varepsilon$). Note the equivalence of the $MINPTS$ and ε density level-set indicator functions when $h = \varepsilon$ and $k = MINPTS$ (i.e., $\frac{MINPTS}{2nh} = \frac{k}{2n\varepsilon}$).

For both cases, a smoothing parameter defines the neighborhood (bandwidth) of a sample element (i.e., the distance-based smoothing parameter $h = \varepsilon$ in Equation 2.1 and the size-based smoothing parameter $k = MINPTS$ in Equation 2.3). Similarly, a threshold parameter is used to define the density level-set (i.e., the size-based threshold parameter $MINPTS$ for Equation 2.1 and the distance-based threshold parameter ε for Equation 2.3). Equality of the two cases is due to the following reasons: (1) use of a uniform kernel, (2) inversion of the smoothing and threshold parameters, and (3) inversely proportional relationship between kNN distance and the minimum bandwidth required to contain k samples. In either case, density is a local estimate (i.e., dependent on the neighborhood of a sample element defined by a fixed bandwidth or size).

A less common definition of an element's density is its number of RkNN in a sample. As with the *DBSCAN* density estimator, the level-set of dense sample elements is defined as the subset with a density greater than or equal to $MINPTS$, differing by the definition of density. Again, for simplicity, this definition will be referred to as the *RNN-DBSCAN* density estimator. The *RNN-DBSCAN* density estimator is equivalent to the variable (adaptive) kernel method (Equation 2.5) with a uniform kernel and bandwidth $h = 1$.

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{hd_k(X_i)}K\left(\frac{x - X_i}{hd_k(X_i)}\right) \tag{2.5}$$

In the unnormalized case, for Equation 2.5, an element belongs to the $MINPTS$ density level-set if $\sum_{i=1}^{n} K\left(\frac{x-X_i}{hd_k(X_i)}\right) \geq MINPTS$ (i.e., x is within the kNN distance ($d_k(X_i)$) of at least $MINPTS$ sample elements X_i). Accordingly, the $MINPTS$ density level-set indicator function of the unnormalized density is $\hat{f}(x) \geq MINPTS$.

As with the *DBSCAN* density estimator, a smoothing parameter is used to define neighborhood size (i.e., k used to define the variable bandwidths). Note that the variable bandwidths are themselves local density estimates (e.g., Equation 2.3). However, unlike the *DBSCAN* density estimator, in the *RNN-DBSCAN* density estimator, a sample element's density is independent of its local density; instead, it is dependent on the local densities of the other sample elements (i.e., variable bandwidths). Thus, density is a global estimate (i.e., an element's density is dependent on a sequence of local estimates computed over the entire sample).

As a final abbreviated example, in *IS-DBSCAN*, an element's density is defined as its number of mutual k-nearest neighbors in a sample. This density is equivalent to a variable kernel, as shown in Equation 2.6 for a uniform kernel and bandwidth $h = 1$. Note that here density is a combination of a local and global element.

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h\min\left(d_k(x), d_k(X_i)\right)}K\left(\frac{x - X_i}{h\min\left(d_k(x), d_k(X_i)\right)}\right) \tag{2.6}$$

CHAPTER 3

METHODOLOGY

3.1 Introduction

In this chapter, the two proposed clustering algorithms, *RNN-DBSCAN* and *Hk-DC*, are presented. Sections 3.2, 3.3, and 3.4 formally describe the single-level-set algorithm *RNN-DBSCAN* with respect to the kNN graph, density, and level-set clustering. The Pseudo-code of *RNN-DBSCAN* with its method for handling noise is presented in Section 3.4.1, and a proposed heuristic for selecting k is presented in Section 3.4.2. The extension to a multi-level-set algorithm *Hk-DC* is formally described in Section 3.5. A proposed heuristic for extracting a flat clustering is presented in Section 3.5.1, along with a method for handling noise in Section 3.5.2.

Assume dataset $\mathcal{X} = (\boldsymbol{x}_1..\boldsymbol{x}_n)$, a tuple consisting of n elements drawn from the real coordinate space of dimension d such that $\forall i \in \{1..n\} : \boldsymbol{x}_i \in \mathbb{R}^d$. Let $N = \{1..n\}$ define the set of implicit element indices of $\mathcal{X}$ such that $i \in N \iff \boldsymbol{x}_i \in \mathcal{X}$, i and $\boldsymbol{x}_i$ are used interchangeably throughout to refer to the i^{th} element in $\mathcal{X}$. A clustering is defined as a strict partitioning of $\mathcal{X}$, or an equivalent partitioning of N, with noise. In the latter case, each noise element may be viewed as either belonging to a single noise partition (i.e., containing all noise elements) or its own singleton partition (i.e., one singleton partition for each noise element).

3.2 kNN Graph

First, for all elements $i, j \in N$, assume a symmetric distance function $d : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ such that $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i)$. Next, let rank order function $r(i, j)$

(Definition 4) define the rank of j for i such that there exist no more than $r(i,j)$ elements in N with distance to i smaller than $d(\boldsymbol{x}_i, \boldsymbol{x}_j)$. In other words, $r(i,j) = k$ indicates that j is the kNN of i. According to Definition 4, an element is not the nearest neighbor of itself, and distance ties are broken at random (i.e., an ordinal ranking).

Definition 4. *Rank Order* Let function $r : N \times N \to N$ define the rank orders of $\mathcal{X}$ where $r(i,\cdot)$ is the rank order of i, according to the order statistics of sample $d(\boldsymbol{x}_i, \cdot)$, such that:

1. $\forall i,j,j' \in N : r(i,j) \leq r(i,j') \iff d(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq d(\boldsymbol{x}_i, \boldsymbol{x}_{j'})$ where $j, j' \neq i$ (ascending order by distance)
2. $\forall i \in N : \sum_{j \in N \setminus i} \{r(i,j)\} = N \setminus n$ (ordinal ranking)
3. $\forall i \in N : r(i,i) = \infty$ (rank of an element wrt. itself is ignored)

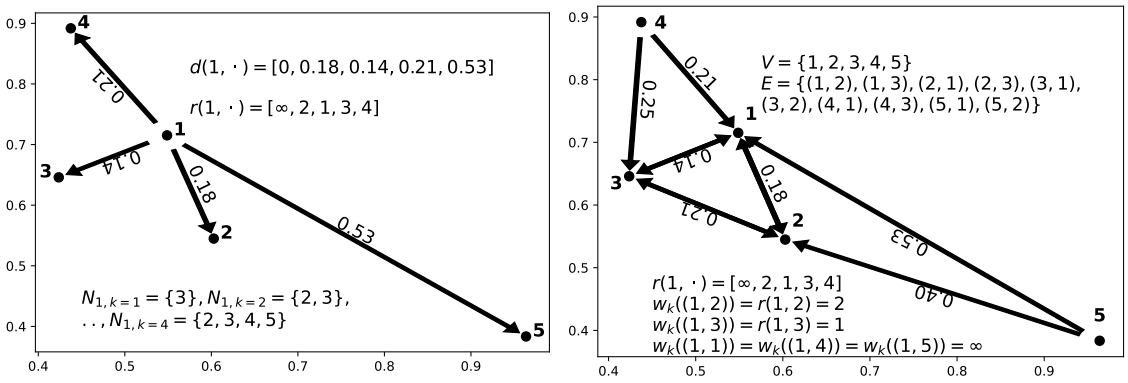
Given rank order function $r(i,j)$, the set of kNN of i (i.e., i's k-nearest neighborhood) can be defined as:

$$N_{i,k} = \{j \in N : r(i,j) \leq k\} \tag{3.1}$$

For example, Figure 4a shows the rank order and k-nearest neighborhood(s) of a small dataset. Recalling that an element is not considered the nearest neighbor of itself, the neighborhood's size is restricted to $k \in [1..n-1]$. In practice, kNN definitions may vary (e.g., wrt. the handling of distance ties and the inclusion of an element as the nearest neighbor of itself).

Given Equation 3.1, the kNN graph of $\mathcal{X}$ is defined as:

$$\boldsymbol{G}_k = (V = N, E = \{(u,v) \in V \times V : v \in N_{u,k}\}) \tag{3.2}$$



(a) rank order (b) edge weight

Fig. 4.: (a) Rank order $r(i, \cdot)$ and nearest neighborhoods $N_{i,k}$ of element $i = 1$, given the order statistics of sample $d(i, \cdot)$, for $k \in \{1..4\}$. (b) kNN graph ($k = 2$) with edge weights $w((i, \cdot))$ for element $i = 1$.

where $\boldsymbol{G}_k$ is the directed graph with vertices equal to N and a directed edge from each vertex to each of its kNN. Assume for any graph functions v and e (e.g., $\mathrm{v}(\boldsymbol{G}_k)$ and $\mathrm{e}(\boldsymbol{G}_k)$) that refer to a graph's set of vertices and edges, respectively.

Finally, $\boldsymbol{G}_k$ is extended to be edge-weighted according to weighting function w_k, such that for all edges $(u, v) \in \mathrm{v}(\boldsymbol{G}_k) \times \mathrm{v}(\boldsymbol{G}_k)$, w_k is defined as:

$$w_k((u, v)) = \begin{cases} r(u, v) & (u, v) \in \mathrm{e}(\boldsymbol{G}_k) \\ \infty & \text{otherwise} \end{cases} \tag{3.3}$$

where the weight of edge $(u, v) \in \mathrm{e}(\boldsymbol{G}_k)$ is equal to $k' \in [1..k]$ such that v is the k'NN of u (i.e., the minimum value of k such that v is in the k-nearest neighborhood of u). As an example, Figure 4b shows a kNN graph with edge weights.

3.3 k-Density

Given kNN graph $\boldsymbol{G}_k$, for all vertices $v \in \mathrm{v}(\boldsymbol{G}_k)$, let function $f_k(v)$ (Definition 5) define a neighborhood density estimate of v such that $f_k(v)$ is dependent on a local

subgraph of $\boldsymbol{G}_k$ centered at v. More intuitively, one may consider f_k a measure of local vertex centrality or isolation (e.g., degree-based centrality).

Definition 5. *Neighborhood Density Estimate* Let function $f_{\boldsymbol{G}_k} : N \to \mathbb{R}_{\geq 0}$ (or simply $f_k = f_{\boldsymbol{G}_k}$) define a mapping from vertices in N to a neighborhood density estimate such that:

1. $\forall k \in \{1..n-1\}$ and $v \in \mathrm{v}(\boldsymbol{G}_k) : f_k(v)$ is dependent on the proper subset of vertices and edges in $\boldsymbol{G}_k$ within some graph distance from v (compact support)
2. $\forall k, k' \in \{1..n-1\}$ and $v \in N : f_{k'}(v) \leq f_k(v) \longleftrightarrow k' \leq k$ (monotonically increasing wrt. k)

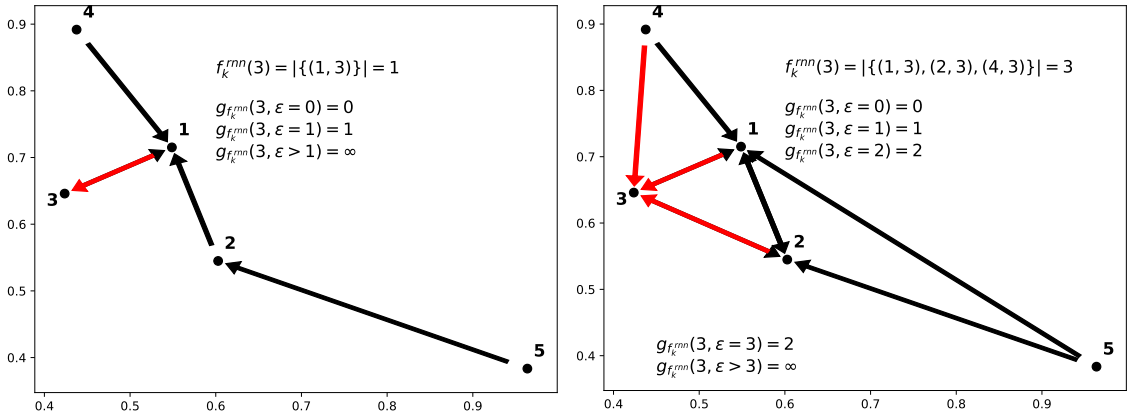
As a simple yet effective example of density function f_k, let $f_k(v)$ be defined as the number of RkNN of v in $\boldsymbol{G}_k$ (i.e., the in-degree of v):

$$f_k^{rnn}(v) = |\{(u, w) \in \mathrm{e}(\boldsymbol{G}_k) : w = v\}| \tag{3.4}$$

In the remainder of this work, f_k assumes Equation 3.4 (i.e., $f_k = f_k^{rnn}$), though other valid examples of Definition 5 exist (e.g., the number of mutual k-nearest neighbors). Given f_k and a density threshold $\varepsilon \in [0, n-1]$, let a vertex's k-density be defined as the minimum value $k' \in [1..k]$ such that $f_{k'}(v) \geq \varepsilon$ (or ∞ if no such k' exists for k):

$$g_{f_k}(v, \varepsilon) = \begin{cases} \arg\min_{k' \in [1..k]} f_{k'}(v) \geq \varepsilon & f_k(v) \geq \varepsilon \\ \infty & \text{otherwise} \end{cases} \tag{3.5}$$

In other words, for $g_{f_k}(v, \varepsilon) = k'$, k' defines the smallest kNN graph (i.e., subgraph $\boldsymbol{G}_{k'} \subseteq \boldsymbol{G}_k$) at which v is dense for ε. As an example, Figure 5 shows kNN graphs, RkNN densities, and k-densities. Note the following properties of k-density function g_{f_k}:



(a) $k = 1$ (b) $k = 2$

Fig. 5.: kNN graphs ((a) $k = 1$ and (b) $k = 2$) showing the RkNN density and k-density of vertex $i = 3$ (reverse nearest neighbors indicated by red arrows).

Remark 1. $\forall k \in \{1..n-1\}, v \in \mathrm{v}(\boldsymbol{G}_k)$, and $\varepsilon, \varepsilon' \in [0, n-1] : g_{f_k}(v, \varepsilon') \leq g_{f_k}(v, \varepsilon) \longleftrightarrow \varepsilon' \leq \varepsilon$ (g_{f_k} is monotonically increasing for ε).

Remark 2. $\forall v \in N, \varepsilon \in [0, n-1]$, and $k, k' \in \{1..n-1\} : g_{f_k}(v, \varepsilon) \geq g_{f_{k'}}(v, \varepsilon) \longleftrightarrow k \leq k'$ (g_{f_k} is monotonically decreasing for k, specifically, for cases where $g_{f_k}(v, \varepsilon) = \infty$).

Remark 3. $\exists k \in \{1..n-1\}$ such that $\forall v \in \mathrm{v}(\boldsymbol{G}_k)$ and $\varepsilon \in [0, n-1] : g_{f_k}(v, \varepsilon) \leq n-1$ (there exists a k at which all vertices satisfy the ε density threshold).

Remark 4. $\forall k \in \{1..n-1\}$ and $\varepsilon \in [0, n-1] : \{v \in \mathrm{v}(\boldsymbol{G}_k) : f_k(v) \geq \varepsilon\} = \{v \in \mathrm{v}(\boldsymbol{G}_k) : g_{f_k}(v, \varepsilon) \leq k\}$ (g_{f_k} can replace f_k in identifying ε-dense vertices).

Finally, given k-density function g_{f_k} and density threshold $\varepsilon \in [0, n-1]$, the set of ε-dense vertices in $\boldsymbol{G}_k$ is defined as the subset of vertices whose local density is greater than or equal to ε (or equivalently whose k-density is less than or equal to k):

$$D_{f_k,\varepsilon} = \{v \in \mathrm{v}(\boldsymbol{G}_k) : g_{f_k}(v, \varepsilon) \leq k\} \tag{3.6}$$

3.4 k-Density Clustering

Given definitions of density and connectivity, let a k-density cluster be defined as a set of elements C such that: (1) all elements in C are dense, (2) all elements in C are connected by a path whose members all lie in C, and (3) C is maximal. Thus, a k-density clustering is a set of all clusters C in $\mathcal{X}$ with non-dense elements identified as noise.

Given a neighborhood graph and set of dense vertices, let a k-density clustering be defined as the set of connected components in the graph's dense-vertex-induced subgraph. In other words, a k-density clustering of $\mathcal{X}$ is a partitioning defined as the set of connected components in the dense-vertex-induced subgraph (e.g., the set of weakly or strongly connected components given a directed neighborhood graph).

Formally, given neighborhood graph $\boldsymbol{G} = \boldsymbol{G}_k$ and dense vertex set $D = D_{f_k,\varepsilon}$, let the dense-vertex-induced subgraph of $\boldsymbol{G}$ be defined as:

$$\boldsymbol{G}(D) = (V = D, E = \{(u, v) \in \mathrm{e}(\boldsymbol{G}) : u, v \in D\}) \tag{3.7}$$

Assuming dense-vertex-induced subgraph $\boldsymbol{G} = \boldsymbol{G}(D)$, let partitioning function $p(\cdot)$ (Defintion 6) define the set of connected components of $\boldsymbol{G}$:

Definition 6. *Partitioning* Let function $p : \boldsymbol{G} \to \boldsymbol{\mathcal{P}}$ define a partitioning of $\boldsymbol{G}$ such that:

1. (1) $\emptyset \notin \boldsymbol{\mathcal{P}}$, (2) $\bigcup_{P \in \boldsymbol{\mathcal{P}}} P = \mathrm{v}(\boldsymbol{G})$, and (3) $\forall P, P' \in \boldsymbol{\mathcal{P}} : P \cap P' = \emptyset \longleftrightarrow P \neq P'$: (partitioning).
2. $\forall P \in \boldsymbol{\mathcal{P}}$ and $u, v \in P : u$ is reachable from v in $\boldsymbol{G}$ (connected).
3. $\forall P, P' \in \boldsymbol{\mathcal{P}}$ where $P' \neq P : \nexists v \in P'$ reachable from all $u \in P$ in $\boldsymbol{G}$ (maximal).

Note that for vertices $u, v \in \mathrm{v}(\boldsymbol{G})$, u is reachable from v if there exists an undi-

rected path from v to u in $\boldsymbol{G}$. In which case, $p(\boldsymbol{G})$ is the set of weakly connected components when $\boldsymbol{G}$ is directed. Likewise, one could define the set of strongly connected components by requiring a directed path from v to u, or $\boldsymbol{G}$ could be undirected (e.g., the mutual kNN graph).

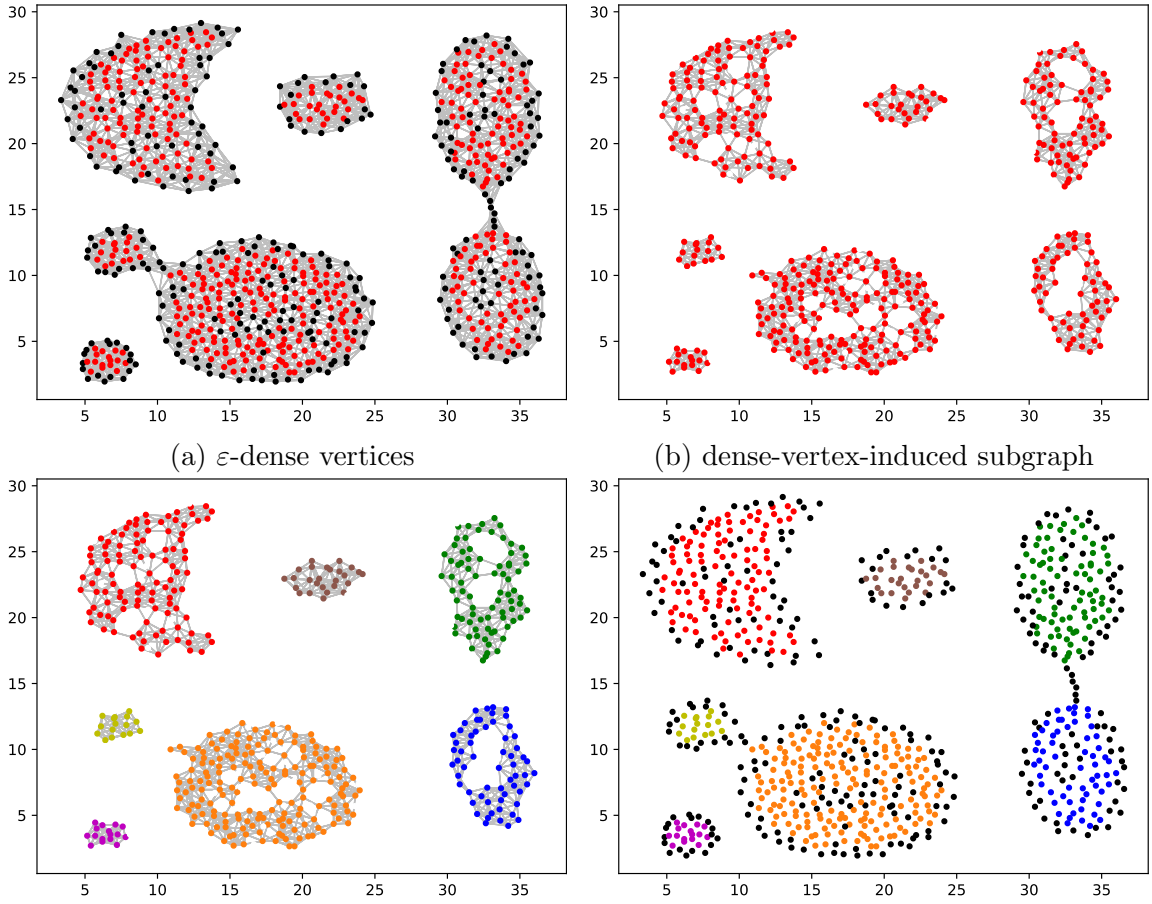
To summarize, a k-density clustering of $\mathcal{X}$ is defined as a partitioning of the ε-dense vertices in $\boldsymbol{G}_k$ ($\boldsymbol{\mathcal{P}} = p(\boldsymbol{G}_k(D_{f_k,\varepsilon}))$), the set of non-dense vertices being identified as noise $O = \mathrm{v}(\boldsymbol{G}_k) \setminus D_{f_k,\varepsilon}$. Note that if $O \neq \emptyset$ then $\boldsymbol{\mathcal{P}} + O$ defines a partitioning of $\mathcal{X}$, else $\boldsymbol{\mathcal{P}}$ defines a partitioning of $\mathcal{X}$. As an example, Figure 6 shows the ε-dense vertices, dense-vertex-induced subgraph, connected components, and resulting k-density clustering for a sample dataset.

3.4.1 *RNN-DBSCAN*

k-density clustering is an example of single-level-set clustering, where connectivity is defined by the kNN graph and density by the k-density function. In the case of single-level-set clustering, k-density is not strictly required as the level-set can be identified directly using its underlying density function (i.e., $\{v \in \mathrm{v}(\boldsymbol{G}_k) : g_{f_k}(v,\varepsilon) \leq k\} = \{v \in \mathrm{v}(\boldsymbol{G}_k) : f_k(v) \geq \varepsilon\}$). The advantages of k-density are shown in Section 3.5 for the case of multi-level-set clustering.

In *RNN-DBSCAN* [18], we proposed a single-level-set clustering algorithm equivalent to the k-density clustering described above (i.e., using kNN graph connectivity and RkNN density). In addition, a fixed minimum density threshold of $\varepsilon = k$ was suggested to simplify parameter selection, representing the expected density of an element drawn randomly from dataset $\mathcal{X}$. Algorithm 1 lists the pseudo-code of *RNN-DBSCAN*, presented as a scanning procedure similar to that of *DBSCAN*.

For dataset $\mathcal{X}$ and parameter k, elements are traversed in arbitrary order. If the current (seed) element has yet to be assigned to a cluster and is a core (dense)



(a) ε-dense vertices

(b) dense-vertex-induced subgraph

(c) subgraph connected components

(d) k-density clustering

Fig. 6.: (a) Set of ε-dense vertices (colored red), (b) dense-vertex-induced subgraph, (c) subgraph connected components (indicated by color), and (d) resulting k-density clustering (clusters indicated by color with black elements indicating noise) for the kNN graph of the aggregate dataset (see Table 2), $k = 10$ and $\varepsilon = k$.

element, it is assigned to a new cluster. The new cluster is expanded by a breadth-first traversal of the undirected kNN graph starting at the current element, traversals terminating at non-core elements (Algorithm 2). For core elements, the returned clustering ($assign$) corresponds to the partitioning defined by the connected components in the core-vertex-induced undirected kNN graph of $\mathcal{X}$ (i.e., is equivalent to the k-density clustering). Clustered non-core elements correspond to elements that are within the kNN of a core element. Note that the cluster assignment of non-core elements is non-deterministic and is conceptually equivalent to the cluster expansion seen in *DBSCAN*.

In addition to the above expansion of non-core elements, *RNN-DBSCAN* further expands clusters by the local distance-based density of each cluster. For the returned cluster assignments $assign$, let the resulting partitioning be defined by $\boldsymbol{\mathcal{P}} = (P_1..P_\ell)$ where $\ell = \max assign$ and $\forall P_i \in \boldsymbol{\mathcal{P}} : P_i = \{j = 1..n : assign[j] = i\}$, along with the set of noise elements $O = \{i = 1..n : assign[i] = NOISE\}$. The distance-based density of partition $P \in \boldsymbol{\mathcal{P}}$ is defined as the maximum pairwise distance between elements in P, restricted to the set of pairs occurring as edges in the kNN graph (Equation 3.8).

$$d(P) = \max\{d(\boldsymbol{x}_i, \boldsymbol{x}_j) : i, j \in P \wedge ((i,j) \in \mathrm{e}(\boldsymbol{G}_k) \vee (j,i) \in \mathrm{e}(\boldsymbol{G}_k))\} \tag{3.8}$$

Algorithm 1 *RNNDBSCAN*

Input: $\mathcal{X}, k$
Output: *assign*

Compute kNN graph $\boldsymbol{G}_k$ of $\mathcal{X}$
$assign[\forall v \in \mathrm{v}(\boldsymbol{G}_k)] = UNCLASSIFIED$
$cluster = 1$
for $v \in \mathrm{v}(\boldsymbol{G}_k)$ **do**
    if $assign[v] = UNCLASSIFIED$ **then**
        if $f_k^{rnn}(v) < k$ **then**
            $assign[\boldsymbol{x}] = NOISE$
        else
            initialize empty queue *seeds*
            $neighbors = RNNDBSCAN\_Neighborhood(\boldsymbol{G}_k, v)$
            $seeds.enqueue(neighbors_v)$
            $assign[v + seeds] = cluster$
            while $seeds \neq \emptyset$ **do**
                $w = seeds.dequeue()$
                if $f_k^{rnn}(w) \geq k$ **then**
                    $neighbors = RNNDBSCAN\_Neighborhood(\boldsymbol{G}_k, w)$
                    for $z \in neighbors$ **do**
                        if $assign[z] = UNCLASSIFIED$ **then**
                            $seeds.enqueue(z)$
                            $assign[z] = cluster$
                        else if $assign[z] = NOISE$ **then**
                            $assign[z] = cluster$
            $cluster = cluster + 1$
$ExpandClusters(\boldsymbol{G}_k, assign)$
return *assign*

Algorithm 2 *RNNDBSCAN_Neighborhood*

Input: $\boldsymbol{G}, v$
Output: *neighbors*

$neighbors = \{u \in \mathrm{v}(\boldsymbol{G}) : (v, u) \in \mathrm{e}(\boldsymbol{G})$ or $((u, v) \in \mathrm{e}(\boldsymbol{G})$ and $f_k^{rnn}(u) \geq k)\}$
return *neighbors*

Given distance-based density function $d(\cdot)$, Algorithm 3 lists the pseudo-code for this additional cluster expansion step. For all noise elements $o \in O$, o is assigned to partition $P \in \boldsymbol{\mathcal{P}}$ if a kNN of o is a core element in P whose distance to o is less than or equal to $d(P)$, $\exists p \in P : (o,p) \in \mathrm{e}(\boldsymbol{G}_k) \wedge f_k(o) \geq k \wedge d(\boldsymbol{x}_o, \boldsymbol{x}_p) \leq d(P)$. In the case of multiple assignee partitions, o is assigned to the nearest partition P defined by minimum distance to core element p, $P = \arg\min_{\{P \in \boldsymbol{\mathcal{P}}, p \in P : (o,p) \in \mathrm{e}(\boldsymbol{G}_k) \wedge f_k(o) \geq k \wedge d(\boldsymbol{x}_o, \boldsymbol{x}_p) \leq d(P)\}} d(\boldsymbol{x}_o, \boldsymbol{x}_p)$.

Algorithm 3 *RNNDBSCAN_ExpandClusters*

Input: $\boldsymbol{G}, assign$
Output: $assign$

```
for cluster = 1..max assign do
    den[i] = d(P_i)
for v ∈ v(G) do
    if assign[v] = NOISE then
        neighbors = {u ∈ v(G) : (v,u) ∈ e(G)}
        mincluster = NOISE
        mindist = ∞
        for u ∈ neighbors do
            cluster = assign[u]
            dist = d(v,u)
            if f_k^rnn(u) ≥ k & dist ≤ den[i] & dist < mindist then
                mincluster = cluster
                mindist = dist
        assign[v] = mincluster
return assign
```

3.4.2 *RNN-DBSCAN*: Choice of k

Given the dependence of *RNN-DBSCAN* on k, in this section, two heuristics are presented to aid in selecting an appropriate value of k for a given dataset. The first heuristic is based on the assumption that a correct clustering solution is stable with respect to the model parameter(s) (i.e., stable over a range of k). A simple yet effective strategy for observing this stability is to examine the number of clusters

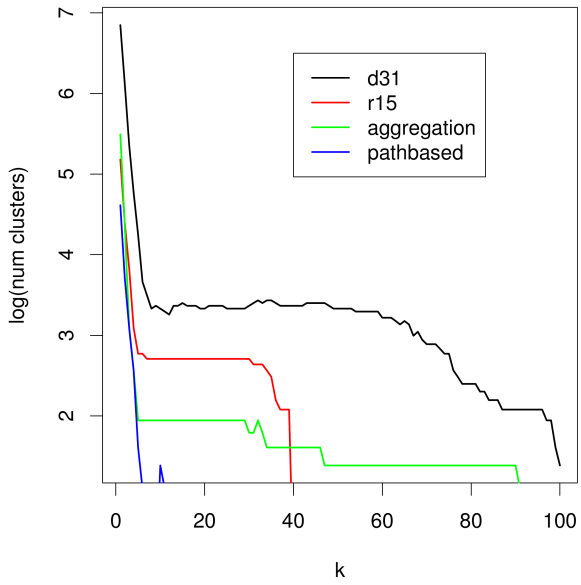


Fig. 7.: Parameter k versus the number of clusters (log) for *RNN-DBSCAN* clusterings produced over the range $k = 1..100$ for several datasets from Table 2.

produced by *RNN-DBSCAN* versus k (i.e., stable wrt. the number of clusters).

Recall that *RNN-DBSCAN* returns partitioning $\boldsymbol{\mathcal{P}} = (P_1..P_\ell)$, along with a set of noise elements O, with the number of partitions (clusters) $\ell = |\boldsymbol{\mathcal{P}}|$. Let $\boldsymbol{\mathcal{P}}_k$ and O_k be the *RNN-DBSCAN* clustering for parameter k. As shown in Figure 7, the number of clusters is not monotonically decreasing for k (i.e., no guarantee that $|\boldsymbol{\mathcal{P}}_k| \geq |\boldsymbol{\mathcal{P}}_{k+1}|$). However, a strong negative correlation exists such that as $k \to n-1 : |\boldsymbol{\mathcal{P}}_k| \to 1$. Additionally, one can observe elbows in the plot, after which the number of clusters maintains some degree of stability as k increases.

Assuming that the correct clustering solution exists at the elbows and is stable, they can be identified by spikes in the histogram of $|\boldsymbol{\mathcal{P}}_k|$ over an appropriate range of k, dependent on the size of the dataset. Specifically, by ordering the histogram in decreasing order by the number of clusters, we propose the first spike in the histogram observed at the number of clusters c^* (i.e., the assumed correct number of

clusters). As by definition, multiple values of k should result in c^* clusters, we further propose selecting the correct value of k, k^*, as the minimum value of k with c^* clusters, $k^* = \arg\min_k |\boldsymbol{\mathcal{P}}_k| = c^*$ (i.e., the k corresponding to the first elbow). Note that by examining the histogram, instead of the plot, no strict definition of the stability's range is required (i.e., a sequence of some length), relying instead on the negative correlation between k and the number of clusters. Similarly, given the negative correlation, k^* should correspond to the first elbow in the plot.

This heuristic is driven by the assumption that the correct choice of k is related to the stable solution, which simultaneously maximizes the number of clusters while minimizing k. In Section 4.2.1, empirical evidence is presented in support of this heuristic. As a second heuristic, we considered *DBVC* [107], an internal validation measure for density-based clustering. Specifically, we consider a simplification of *DBVC* [43]. *DBCV* is based on the assumption that elements in a cluster should be tightly connected while elements belonging to different clusters are well separated. First, defining 'tightly connected', let function $d(\cdot)$ define the density of a partition $P \in \boldsymbol{\mathcal{P}}_k$ as the maximum distance in the minimum spanning tree of core-elements in P. Next, to define cluster separability, let function $s(\cdot, \cdot)$ define the separability of two partitions $P, P' \in \boldsymbol{\mathcal{P}}_k$ as the minimum pairwise distance between core-elements from P to P'.

For partitioning $\boldsymbol{\mathcal{P}} = \{P_1, ..., P_l\}$, the validity of partition $P \in \boldsymbol{\mathcal{P}}$ is defined by function v (Equation 3.9), which compares the density of P to its minimum separability.

$$v(P) = \frac{\min_{P' \in \boldsymbol{\mathcal{P}} \setminus P} s(P, P') - d(P)}{\max\left(\min_{P' \in \boldsymbol{\mathcal{P}} \setminus P} s(P, P'), d(P)\right)} \tag{3.9}$$

The validity index of partitioning $\boldsymbol{\mathcal{P}}$ is defined by function vi (Equation 3.10) as

the weighted average of the validity indices of all partitions in $\boldsymbol{\mathcal{P}}$:

$$vi(\boldsymbol{\mathcal{P}}) = \sum_{i=1}^{|\boldsymbol{\mathcal{P}}|} \frac{|P_i|}{n} \times v(P_i) \tag{3.10}$$

where $vi(\boldsymbol{\mathcal{P}}) \in [-1, +1]$ with larger values indicating a better clustering. Thus, the assumed correct choice of k, k^*, is the k which maximizes Equation 3.10, $k^* = \arg\max_k vi(\boldsymbol{\mathcal{P}}_k)$. Empirical evidence in support of this approach is presented in Section 4.2.1.

3.4.3 *RNN-DBSCAN* Complexity

The complexity of *RNN-DBSCAN* is dependent on the cost of computing the exact or approximate kNN graph, along with the choice of k. Here will assume that the kNN graph is given (see Section 2.5 for a discussion on kNN graph complexity). Given $\boldsymbol{G}_k$, the RkNN density function (Equation 3.4) has complexity $\mathcal{O}(1)$, with an overall complexity of $\mathcal{O}(n)$ to compute the density of all elements. Algorithm 1 is equivalent to the cost of performing ℓ breadth-first searches of core-elements in $\boldsymbol{G}_k$, equivalent to the cost of computing the core-element connected components in $\boldsymbol{G}_k$ with the number of components equal to ℓ. Thus, the complexity of Algorithm 1 is $\mathcal{O}(n + nk)$, where n is the number of vertices and nk is the number of edges in $\boldsymbol{G}_k$. Of course, this complexity becomes quadratic when $k \approx n$, though in practice, the correct choice of k is such that $k << n$.

Algorithm 3 has complexity $\mathcal{O}(kn)$, as the k-neighborhood of each noise element is searched for candidate partitions. Note that this complexity covers the cost of computing the density of each partition (i.e., the maximum pairwise distance between core-elements for edges in $\boldsymbol{G}_k$, which is likewise $\mathcal{O}(kn)$). Thus, the cost of *RNN-DBSCAN* is $\mathcal{O}(n + nk)$ plus the cost of computing the kNN graph, which will

assuredly dominate the complexity (e.g., $\mathcal{O}(kn \log n)$ or $\mathcal{O}(kn^2)$).

With respect to the heuristics for choosing k, given some maximum value k_{max}, note that the k_{max}NN graph contains all nearest neighbor graphs in the range of $k = 1..k_{max}$. Furthermore, for each k, the $(k-1)$NN graph can be obtained by removing n edges from the kNN graph. Thus, the cost of computing all kNN graphs is equal to the cost of computing the k_{max}NN graph plus $\mathcal{O}(nk_{max})$ (i.e., the cost of removing n edges k_{max} times, assuming $\mathcal{O}(1)$ cost of edge removal). Additionally, *RNN-DBSCAN* (Algorithm 1) is run on each kNN graph for a total complexity of $\mathcal{O}((n + nk_{max}) + (n + n(k_{max} - 1)) + .. + (n + n))$. Note that as neither heuristic is dependent on Algorithm 3, it need not be considered.

3.5 Hierarchical k-Density Clustering (*Hk-DC*)

Recall that k-density clustering uses two parameters to define the connectivity (k) and density level-set (ε) of all elements in $\mathcal{X}$. Inspired by [36], for a fixed density threshold of $\varepsilon \in [1, n-1]$ and connectivity $k \in [1, n-1]$, an efficient hierarchical k-density clustering algorithm is proposed to compute all k'-density clusterings over the range $k' \in [1, k]$ within a single hierarchical clustering.

For k and ε, let mutual reachability graph $\boldsymbol{G}_{f_k,\varepsilon}$ be defined as the undirected collapsed graph of $\boldsymbol{G}_k$ (i.e., $\mathrm{v}(\boldsymbol{G}_{f_k,\varepsilon}) = \mathrm{v}(\boldsymbol{G}_k)$ and $(u,v) \vee (v,u) \in \mathrm{e}(\boldsymbol{G}_k) \iff \{u,v\} \in \mathrm{e}(\boldsymbol{G}_{f_k,\varepsilon})$). Furthermore, $\boldsymbol{G}_{f_k,\varepsilon}$ is extended to be edge-weighted according to weighting function $w_{f_k,\varepsilon}$, such that for all edges $\{u,v\} \in \mathrm{v}(\boldsymbol{G}_{f_k,\varepsilon}) \times \mathrm{v}(\boldsymbol{G}_{f_k,\varepsilon})$, $w_{f_k,\varepsilon}$ is defined as:

$$w_{f_k,\varepsilon}(\{u,v\}) = \begin{cases} \max(g_{f_k}(u,\varepsilon), g_{f_k}(v,\varepsilon), & \{u,v\} \in \mathrm{e}(\boldsymbol{G}_{f_k,\varepsilon}) \\ \min\left(w_k(u,v), w_k(v,u)\right)) & \\ \infty & \text{otherwise} \end{cases} \tag{3.11}$$

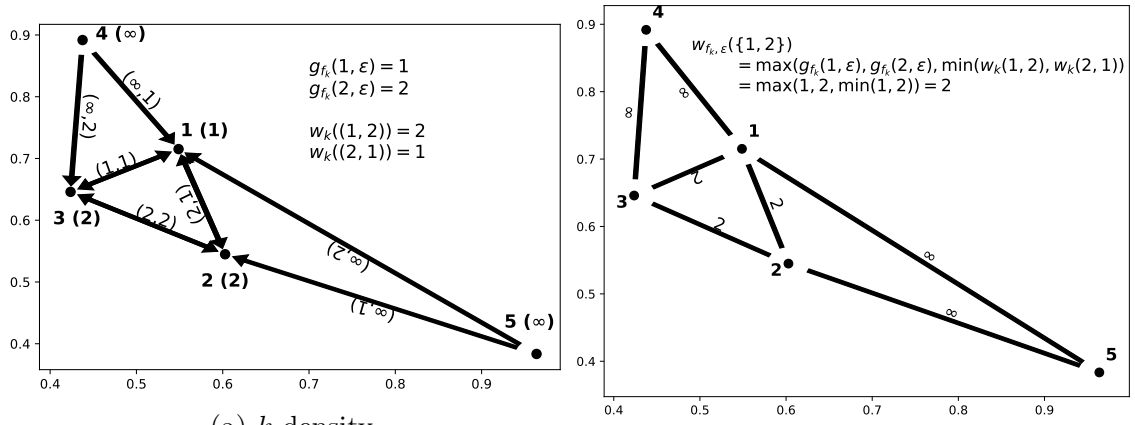
Equation 3.11 defines the weight of an edge $\{u,v\}$ in $\boldsymbol{G}_{f_k,\varepsilon}$ as the minimum $(k' \leq k)$NN graph such that: (1) both u and v are ε-dense, and (2) an undirected edge exists between u and v. In other words, for $k' = w_{f_k,\varepsilon}(\{u,v\})$, $\boldsymbol{G}_{k'} \subseteq \boldsymbol{G}_k$ is the smallest nearest neighbor graph in which u and v are both ε-dense ($g_{f_k}(u,\varepsilon) \leq k'$ and $g_{f_k}(v,\varepsilon) \leq k'$) and u is in the k'NN neighborhood of v or vice versa ($\min\left(w_k(u,v), w_k(v,u)\right) \leq k'$). This definition is similar to the mutual reachability graph of [36], inspired by the concept of mutual reachability [35]. The difference being in definitions of connectivity and density.

Figure 8 shows the resulting mutual reachability graph for the kNN graph and k-density of a sample dataset. Note that other valid definitions of the mutual reachability graph exist, including using the undirected mutual graph of $\boldsymbol{G}_k$ (as opposed to the collapsed graph) or taking the maximum edge weight between two vertices (as opposed to the minimum).

Using the subgraph notation from Equation 3.7, let $\boldsymbol{G}_{f_k,\varepsilon}(k')$ define the subgraph of $\boldsymbol{G}_{f_k,\varepsilon}$ obtained by removing all vertices and edges whose k-density and edge weight are greater than $k' \in [1..k]$:

$$\begin{aligned} \boldsymbol{G}_{f_k,\varepsilon}(k') = (V = \{v \in \mathrm{v}(\boldsymbol{G}_{f_k,\varepsilon}) : g_{f_k}(v) \leq k'\}, \\ E = \{\{u,v\} \in \mathrm{e}(\boldsymbol{G}_{f_k,\varepsilon}) : u,v \in V \wedge w_{f_k,\varepsilon}(\{u,v\}) \leq k'\}) \end{aligned} \tag{3.12}$$

The resulting k' mutual reachability subgraph $\boldsymbol{G}_{f_k,\varepsilon}(k')$ has the following prop-



(a) k-density

(b) mutual reachability graph

Fig. 8.: (a) k-density and (b) mutual reachability graph for a kNN graph ($k = 2$). Note here $\varepsilon = k$ and graph edges show k-density and mutual reachability graph edge weight.

erties:

Remark 5. $\forall k' \in [1..k]$: partitioning $\mathcal{P}_{k'} = p(\boldsymbol{G}_{f_k,\varepsilon}(k'))$ is equal to $p(\boldsymbol{G}_{k'}(D_{f_{k'},\varepsilon}))$ (all k'-density clusterings can be obtained from $\boldsymbol{G}_{f_k,\varepsilon}$).

Remark 6. $\forall k', k'' \in [1..k]$ such that $k'' \leq k'$: $p(\boldsymbol{G}_{f_{k''},\varepsilon}(k'')) = p(\boldsymbol{G}_{f_{k'},\varepsilon}(k'')) = p(p(\boldsymbol{G}_{f_k,\varepsilon}(k''))$ (all k''-density clusterings can be obtained from $\boldsymbol{G}_{f_k,\varepsilon}(k')$).

Remark 7. $\forall k', k'' \in [1..k]$ such that $k'' \leq k'$: partitioning $\mathcal{P}_{k''}$ is a refinement of $\mathcal{P}_{k'}$ ($\mathcal{P}_{k'}$ and $\mathcal{P}_{k''}$ represent cuts in a single hierarchical partitioning).

Consequently, all k'-density clusterings for $k' \in [1..k]$ can be produced in a nested, hierarchical way by removing edges in decreasing order of weight from the minimum spanning tree of $\boldsymbol{G}_{f_k,\varepsilon}$ [36]. This procedure is equivalent to divisive single-linkage clustering of a neighborhood graph (e.g., $\boldsymbol{G}_{f_k,\varepsilon}$). The k'-cut of the resulting dendrogram being equal to the k'-density clustering.

Let $\boldsymbol{H}$ be a matrix of size $n \times (k+1)$ representing the hierarchical k-density

clustering. Matrix entry $\boldsymbol{H}_{i,j}$ equal to the i^{th} element's cluster assignment at the j^{th}-cut in the hierarchical clustering. In other words, $\boldsymbol{H}_{i,j}$ is the k-density clustering assignment of element i where $k = j$. Note that $k+1$ is done strictly for convenience when the k-density clustering does not yield a single cluster. It represents the root of a divisive hierarchical clustering where all elements are assigned to a single cluster. In practice, $\boldsymbol{H}$ might be defined more compactly as a cut might not exist for all values of k' (e.g., when k' is not an edge weight in the minimum spanning tree of $\boldsymbol{G}_{f_k,\varepsilon}$). Algorithm 4 lists the pseudo-code of hierarchical k-density clustering (Hk-DC) returning matrix $\boldsymbol{H}$. Inputs include dataset $\mathcal{X}$, connectivity parameter k, density threshold ε, and minimum cluster size m (elements of clusters smaller than m identified as noise).

Lines 1-3 compute the weighted mutual reachability graph $\boldsymbol{G}_{f_k,\varepsilon}$ as discussed above. Line 4 computes the minimum spanning tree $\boldsymbol{M}$ of the mutual reachability graph, used in place of the mutual reachability graph to perform the hierarchical clustering, improving efficiency while producing identical results, a common technique used in single linkage clustering. As the existence of a minimum spanning tree is not guaranteed for all instances of $\boldsymbol{G}_{f_k,\varepsilon}$ (i.e., $\boldsymbol{M}$ may be a minimum spanning forest), lines 5-6 convert $\boldsymbol{M}$ to a tree by adding a minimum set of edges with weight ∞. Line 7 adds self-edges for all vertices in $\boldsymbol{M}$ of weight equal to the k-density of each vertex, as done in [36]. Self-edges distinguishes between the case of a singleton cluster versus noise when $m = 1$. Finally, line 8 initializes the root vertex, assigning all vertices to a single cluster.

Lines 9-23 perform the divisive hierarchical clustering by removing $\boldsymbol{M}$'s edges in decreasing order by weight (from k to 1). At each iteration k', this is performed by first removing all edges from $\boldsymbol{M}$ with weight greater than k' (lines 10 and 11) and then computing the k'-density clustering (partitioning $\mathcal{P}$) using $\boldsymbol{M}$ (line 12). Next, $\mathcal{P}$ is used to compute $\boldsymbol{H}_{\cdot,k'}$ by updating the $(k'+1)$-density clustering in $\boldsymbol{H}_{\cdot,k'+1}$ to

Algorithm 4 *HkDC*

Input: $\mathcal{X}, k, \varepsilon, m$

Output: $\boldsymbol{H}$

Compute weighted k-nearest neighbor graph $\boldsymbol{G}_k$ of $\mathcal{X}$
Compute k-density ($g_{f_k}(v, \varepsilon)$) for all vertices $v \in \mathrm{v}(\boldsymbol{G}_k)$
Compute weighted mutual reachability graph $\boldsymbol{G}_{f_k,\varepsilon}$ of $\boldsymbol{G}_k$ and $g_{f_k}(\cdot, \varepsilon)$
Compute minimum spanning tree $\boldsymbol{M}$ of $\boldsymbol{G}_{f_k,\varepsilon}$
if $\boldsymbol{M}$ is a forest **then**
    Randomly add # of trees - 1 edges of weight ∞ such that $\boldsymbol{M}$ is a tree
Extend $\boldsymbol{M}$ by adding a self-edge for each vertex $v \in \mathrm{v}(\boldsymbol{M})$ with edge weight equal to $g_{f_k}(v, \varepsilon)$
Set c_id=0 and initialize matrix $\boldsymbol{H}_{\cdot,k+1} = c\_id$
for $k' = k$ to 1 **do**
    Compute set of edges E in $\boldsymbol{M}$ with edge weight greater than k'
    Remove edges E from $\boldsymbol{M}$, $\boldsymbol{M} = \boldsymbol{M} \setminus E$
    Compute k' partitioning $\mathcal{P} = p(\boldsymbol{M})$
    Compute set of $k' + 1$ clusters inter-connected by an edge in E, $C = \bigcup_{\{u,v\} \in E} \{\boldsymbol{H}_{u,k'+1}\}$
    For all clusters $c \in C$, compute number of non-spurious partitions n_c of c in partitioning $\mathcal{P}$
    for $P \in \mathcal{P}$ **do**
        $c = \boldsymbol{H}_{v,k'+1}$ for any $v \in P$
        if $c \notin C$ or $n_c < 2$ **then**
            $\boldsymbol{H}_{v \in P,k'} = c$
        else if $|P| < m$ or ($m == 1$ and the current degree of $v \in P$ in $\boldsymbol{T}$ is zero) **then**
            $\boldsymbol{H}_{v \in P,k'} = -1$
        else
            $c\_id = c\_id + 1$
            $\boldsymbol{H}_{v \in P,k'} = c\_id$
return $\boldsymbol{H}$

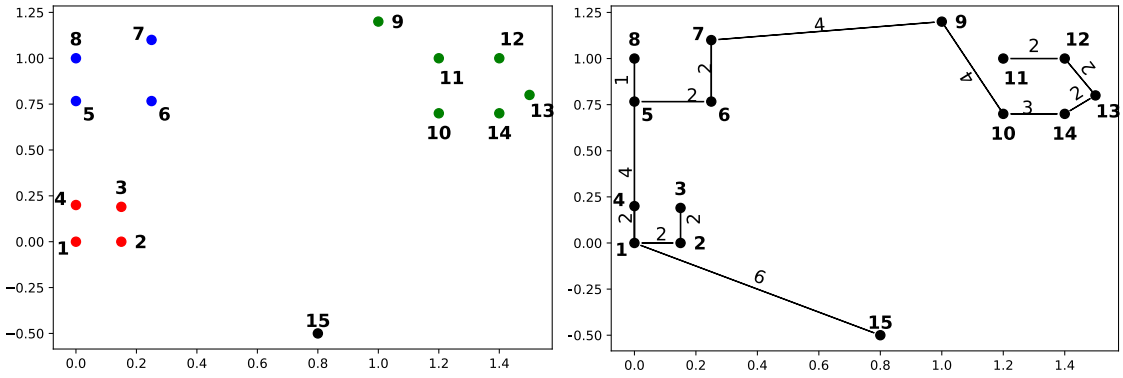
the k'-density clustering (lines 13-23). Note that only clusters in the $(k'+1)$-density clustering with vertices adjacent to the set of removed edges require updating. Line 13 computes this set of clusters C. Next, for each cluster $c \in C$, the number of non-spurious partitions n_c is computed in line 14. A non-spurious partition of cluster c is a partition $P \in \mathcal{P}$ such that P is a subset c and $|P| \geq m$.

$\boldsymbol{H}_{\cdot,k'}$ is updated in lines 15-27 by iterating over all partitions $P \in \mathcal{P}$, and assigning all vertices $v \in P$ to one of the following three values: (1) v's $(k'+1)$-density cluster ($\boldsymbol{H}_{v,k'+1}$), (2) noise (-1), or (3) a new cluster (c_id+1). In the first case (lines 17-18), v's $(k'+1)$-density cluster c (line 16) has not changed or, P is the only non-spurious partition of the partitioning of c. For the second case (lines 19-20), P is a spurious partition or a singleton partition of v where v is not ε-dense. Finally, in the third case (lines 21-23), P is a new cluster as P is one of several non-spurious partitions of the partitioning of c.

For example, for the small dataset and corresponding minimum spanning tree (computed from the weighted mutual reachability graph) shown in Figure 9, Table 1 shows matrix $\boldsymbol{H}$ produced by Hk-DC, with the dendrogram representation of $\boldsymbol{H}$ shown in Figure 10a.

3.5.1 *Hk-DC*: Extracting a Flat Clustering

One straightforward solution for extracting a flat clustering is to select a single cut in the hierarchy (e.g., a k'-cut in the k-density hierarchical clustering). However, this assumes the correct clustering is discoverable using a single global cut. Unfortunately, such a solution may fail to simultaneously detect clusters at varying levels of density (i.e., significant differences in k-density), which would require multiple local cuts (at different levels). Furthermore, recall that *HK-DC* is dependent on density threshold ε. Thus, in addition to discovering a final flat clustering, discovering an appropriate



(a) sample dataset (b) minimum spanning tree

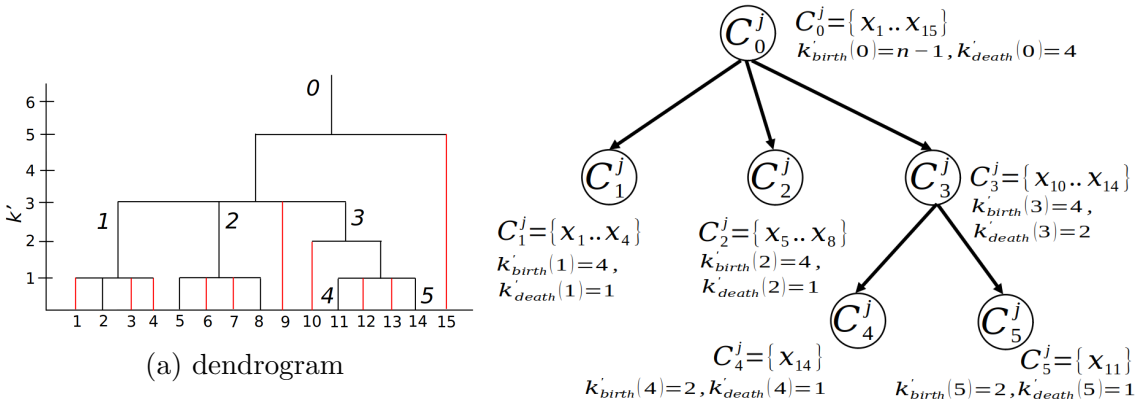
Fig. 9.: (a) Sample dataset and (b) corresponding minimum spanning tree of the weighted mutual reachability graph for $k = n - 1$ and $\varepsilon = 2$. In (a), element colors represent a notional clustering with black elements representing noise.

Table 1.: Matrix $\boldsymbol{H}$ Returned by Hk-DC†‡

k'	1	2	3	4	5	6	...	14
$\boldsymbol{x}_1$	-1	1	1	0	0	0	...	0
$\boldsymbol{x}_2$	1	1	1	0	0	0	...	0
$\boldsymbol{x}_3$	-1	1	1	0	0	0	...	0
$\boldsymbol{x}_4$	-1	1	1	0	0	0	...	0
$\boldsymbol{x}_5$	2	2	2	0	0	0	...	0
$\boldsymbol{x}_6$	-1	2	2	0	0	0	...	0
$\boldsymbol{x}_7$	-1	2	2	0	0	0	...	0
$\boldsymbol{x}_8$	2	2	2	0	0	0	...	0
$\boldsymbol{x}_9$	-1	-1	-1	0	0	0	...	0
$\boldsymbol{x}_{10}$	-1	-1	3	0	0	0	...	0
$\boldsymbol{x}_{11}$	5	3	3	0	0	0	...	0
$\boldsymbol{x}_{12}$	-1	3	3	0	0	0	...	0
$\boldsymbol{x}_{13}$	-1	3	3	0	0	0	...	0
$\boldsymbol{x}_{14}$	4	3	3	0	0	0	...	0
$\boldsymbol{x}_{15}$	-1	-1	-1	-1	-1	0	...	0

† For the sample dataset with corresponding minimum spanning tree shown in Figure 9.

‡ Note that $k = n-1$, $\varepsilon = 2$, and $m = 1$, also, column $k+1$ has been excluded as a singleton cluster exists at k.



(a) dendrogram

(b) directed rooted tree

Fig. 10.: (a) Dendrogram of matrix $\boldsymbol{H}$ shown in Table 1. Note that the x-axis represents elements, y-axis values of k', and red lines indicate element transitions to noise. (b) Directed rooted tree of matrix $\boldsymbol{H}$ shown in Table 1.

value of ε is essential. A heuristic solution is proposed below for extracting a final flat clustering addressing both issues.

We being by defining a rooted directed trees for each *Hk-DC* matrix $\boldsymbol{H}$ over a range of density threshold ε values. Let $\boldsymbol{e}$ define a range of density thresholds (ε's), $\boldsymbol{e} = [1..\varepsilon_{max}]$ for maximum value ε_{max}. For $j = [1..|\boldsymbol{e}|]$, let $\boldsymbol{T}_j$ represent the directed rooted tree of the *Hk-DC* matrix $\boldsymbol{H}$ for $\varepsilon = \boldsymbol{e}_j$ (indicated by $\boldsymbol{H}^j$). Vertices of $\boldsymbol{T}_j$ are defined as the set of clusters in $\boldsymbol{H}^j$, $\mathrm{v}(\boldsymbol{T}_j) = \{0..\max(\boldsymbol{H}^j)\}$ where $\max(\boldsymbol{H}^j)$ is the largest cluster index in $\boldsymbol{H}^j$. For each vertex $v \in \mathrm{v}(\boldsymbol{T}_j)$, let C_v^j define the set of member elements of cluster v, $C_v^j = \{i \in N : \exists k' \text{ such that } \boldsymbol{H}^j_{i,k'} = v\}$.

To define the edges of $\boldsymbol{T}_j$, for each vertex $v \in \mathrm{v}(\boldsymbol{T}_j)$, let $k'_{birth}(v)$ define the maximum value of k' below which cluster v exists in $\boldsymbol{H}^j$, $k'_{birth}(v) = (\arg\max_{k'} \boldsymbol{H}^j_{\cdot,k'} = v)+1$. Similarly, let $k'_{death}(v)$ define the minimum value of k' at which cluster v exists in $\boldsymbol{H}^j$, $k'_{death}(v) = \arg\min_{k'} \boldsymbol{H}^j_{\cdot,k'} = v$. A directed edge (u, v) exists iff cluster u is a parent of cluster v in $\boldsymbol{H}^j$ (i.e., $\boldsymbol{T}_j$ is an out-tree with edges pointing away from the

root vertex), $\mathrm{e}(\boldsymbol{T}_j) = \{(u, v) \in \mathrm{v}(\boldsymbol{T}_j) \times \mathrm{v}(\boldsymbol{T}_j) : C_v^j \subset C_u^j \text{ and } k'_{death}(u) = k'_{birth}(v)\}$. As an example, Figure 10b shows the directed rooted tree of the *Hk-DC* matrix shown in Table 1.

Next, we define the set of all flat clusterings within each rooted directed tree. Let $\boldsymbol{\mathcal{F}}^j$ define the set of all flat clusterings in $\boldsymbol{T}_j$ such that $\forall c = [1..|\boldsymbol{\mathcal{F}}^j|] : \boldsymbol{\mathcal{F}}_c^j \subseteq \mathrm{v}(\boldsymbol{T}_j)$. A flat clustering $\boldsymbol{\mathcal{F}}_c^j$ requires that all paths from the root vertex to the leaf vertices in $\boldsymbol{T}_j$ traverse exactly one vertex in $\boldsymbol{\mathcal{F}}_c^j$. This requirement ensures that $\boldsymbol{\mathcal{F}}_c^j$ clusters are non-overlapping and fully cover the elements in $\boldsymbol{T}_j$ (i.e., $\boldsymbol{\mathcal{F}}_c^j$ is a partitioning, excluding noise elements). For example, the directed rooted tree $\boldsymbol{T}_j$ in Figure 10b contains three flat clusterings, $\boldsymbol{\mathcal{F}}^j = \{\{C_0^j\}, \{C_1^j, C_2^j, C_3^j\}, \{C_1^j, C_2^j, C_4^j, C_5^j\}\}$. As the size of $\boldsymbol{\mathcal{F}}$ increases exponentially for the depth of $\boldsymbol{T}$ (see Section 3.5.3), the first $m_{\boldsymbol{\mathcal{F}}}$ flat clusterings are selected by greedily traversing $\boldsymbol{T}$ by vertex size $|C|$ (Algorithm 5).

Algorithm 5 *HkDC_flat_clusterings*

Input: $\boldsymbol{T}, m_{\boldsymbol{\mathcal{F}}}$
Output: $\boldsymbol{\mathcal{F}}$

```
Initialize priority queue Q (sorted in descending order by vertex size |C|)
Insert root vertex of T into Q
Initialize F with the root vertex clustering
while |Q| > 0 and |F| < m_F do
    v = dequeue Q
    S = successors of v in T
    enqueue S into Q
    for C ∈ F where v ∈ C do
        Insert (C \ v) + S into F
        if |F| ≥ m_F then
            break
return F
```

Intuitively, given $\boldsymbol{T}_1, .., \boldsymbol{T}_{|\boldsymbol{e}|}$, the problem of extracting the correct flat clustering involves identifying the most prominent clusters. For a cluster at density threshold j, prominence is defined as the cluster's persistence over a local window centered at j. Given local window size $w \geq 1$, the persistence of cluster $v \in \mathrm{v}(\boldsymbol{T}_j)$ is defined as the

average of the maximum similarity (Jaccard index) between v and all clusters within the window of trees $\boldsymbol{T}_{\ell \in W(w,j)}$, $W(w,j) = \{0 \leq \ell \leq |\boldsymbol{e}| : j - w \leq \ell < j \text{ or } j < \ell \leq j + w\}$:

$$ls(v) = \frac{1}{|W(w,j)|} \sum_{\ell \in W(w,j)} \max_{u \in \boldsymbol{T}_\ell} \frac{C_v^j \cap C_u^\ell}{C_v^j \cup C_u^\ell} \tag{3.13}$$

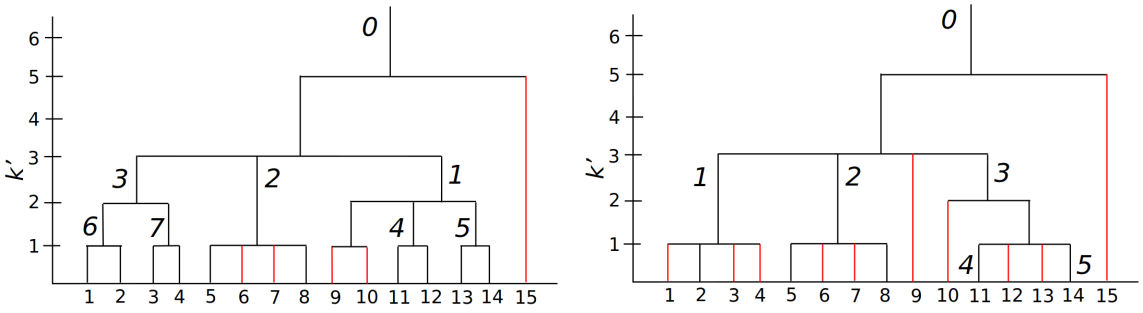
Note that the persistence of cluster $v \in \mathrm{v}(\boldsymbol{T}_j)$ is a measure of local similarity (i.e., local with respect to the choice of density threshold ε). For example, consider the directed rooted trees $\boldsymbol{T}_j$ ($j = [1..4]$) shown as dendrograms in Figure 11. For window size $w = 2$, the persistence score of cluster two at $j = 2$ ($C_2^2 = \{5,6,7,8\}$) is $(1 + 1 + 1/8)/3$ as the window of j is $W = \{1,3,4\}$ and the most similar clusters to C_2^2 in W are $C_2^1 = \{5,6,7,8\}$, $C_3^3 = \{5,6,7,8\}$, and $C_1^4 = \{1,2,3,4,5,6\}$.

Next, cluster persistence is aggregated to compute a flat clustering's overall persistence. For a flat clustering $\boldsymbol{\mathcal{C}} \in \boldsymbol{\mathcal{F}}^j$, the aggregate persistence of $\boldsymbol{\mathcal{C}}$ is defined as the minimum cluster persistence for all clusters $v \in \boldsymbol{\mathcal{C}}$:

$$als(\boldsymbol{\mathcal{C}}) = \min_{v \in \boldsymbol{\mathcal{C}}} ls(v) \tag{3.14}$$

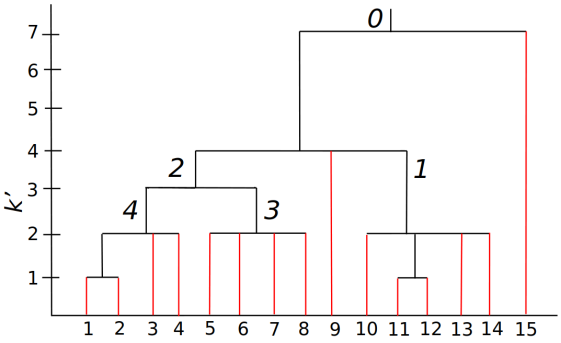
Now, one might consider selecting the flat clustering with maximum aggregate persistence. However, such a solution would not be of interest as aggregate persistence is biased towards flat clusterings consisting of fewer clusters (e.g., the root vertex cluster is guaranteed a maximum value of one). In general, a cluster's persistence can be expected to decrease as the distance from the root vertex increases, which correlates to an increase in the size of the flat clustering (i.e., the number of clusters).

Thus, a flat clustering should be selected by simultaneously maximizing aggregate persistence and clustering size. We propose plotting the relationship between aggregate persistence and clustering size to select an appropriate size c^* (number

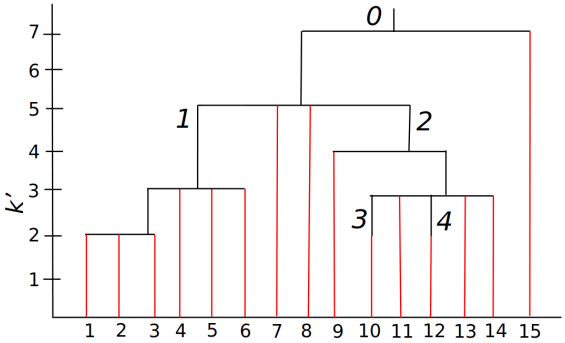


(a) $j = 1$

(b) $j = 2$



(c) $j = 3$



(d) $j = 4$

Fig. 11.: *Hk-DC* dendrograms for the dataset shown in Figure 9a. Note that $\boldsymbol{e} = [1..4]$ ((a)-(d)), $k = n - 1$, $m = 1$, x-axis represents elements, y-axis values of k', and red lines indicate element transitions to noise.

of clusters). Given c^*, the final flat clustering solution is assumed to be the solution of size c^* that maximizes aggregate persistence. Let c^{max} define the size of the largest flat clustering in $\boldsymbol{T}_1, .., \boldsymbol{T}_{|\boldsymbol{e}|}$, $c^{max} = \max_{1 \leq j \leq |\boldsymbol{e}|, \mathcal{C} \in \boldsymbol{\mathcal{F}}^j} |\mathcal{C}|$. For each flat cluster size $c \in [2..c^{max}]$, maximum aggregated persistence for size c ($mals_c$) in $\boldsymbol{T}_1, .., \boldsymbol{T}_{|\boldsymbol{\varepsilon}|}$ is defined as:

$$mals_c = \max_{1 \leq j \leq |\boldsymbol{\varepsilon}|, \mathcal{C} \in \boldsymbol{\mathcal{F}}_c^j} als(\mathcal{C}) \tag{3.15}$$

where $\boldsymbol{\mathcal{F}}_c^j$ is the subset of flat clusterings in $\boldsymbol{\mathcal{F}}^j$ of size c, $\boldsymbol{\mathcal{F}}_c^j = \{\mathcal{C} \in \boldsymbol{\mathcal{F}}^j : |\mathcal{C}| = c\}$. Note that singleton clusterings ($c = 1$) are ignored. Thus, c^* is selected by observing the maximum aggregated local similarity at each size c.

In general, the ($mals_c$ versus c)-plot can be expected to decrease as c increases. However, sharp decreases in this plot at the ground truth number of clusters have been consistently observed (see Section 4.3.3). These results suggest that the ($mals_c$ versus c)-plot can be used to select the number of clusters c^*. Additionally, given c^*, the solution with c^* clusters that maximizes aggregate persistence has been observed to be strongly correlated with the optimal solution (see Section 4.3.3).

In addition to maximum aggregate persistence, to capture the persistence of flat clusterings of size c over the global range of $\boldsymbol{e}$, each c in the ($mals_c$ versus c)-plot is colored according to the average maximal aggregate persistence over $\boldsymbol{T}_1, .., \boldsymbol{T}_{|\boldsymbol{e}|}$ for size c:

$$gmals_c = \sum_{1 \leq j \leq |\boldsymbol{e}|} \max_{\mathcal{C} \in \boldsymbol{\mathcal{F}}_c^j} als(\mathcal{C}) / |\boldsymbol{e}| \tag{3.16}$$

In Equation 3.16, maximal aggregate persistence for size c is computed separately for each $\varepsilon \in \boldsymbol{e}$ and averaged. Thus, $gmals_c$ is a measure of how persistent size c is across the global range of ε. Similar to $mals_c$, $gmals_c$ decreases as c increases, with

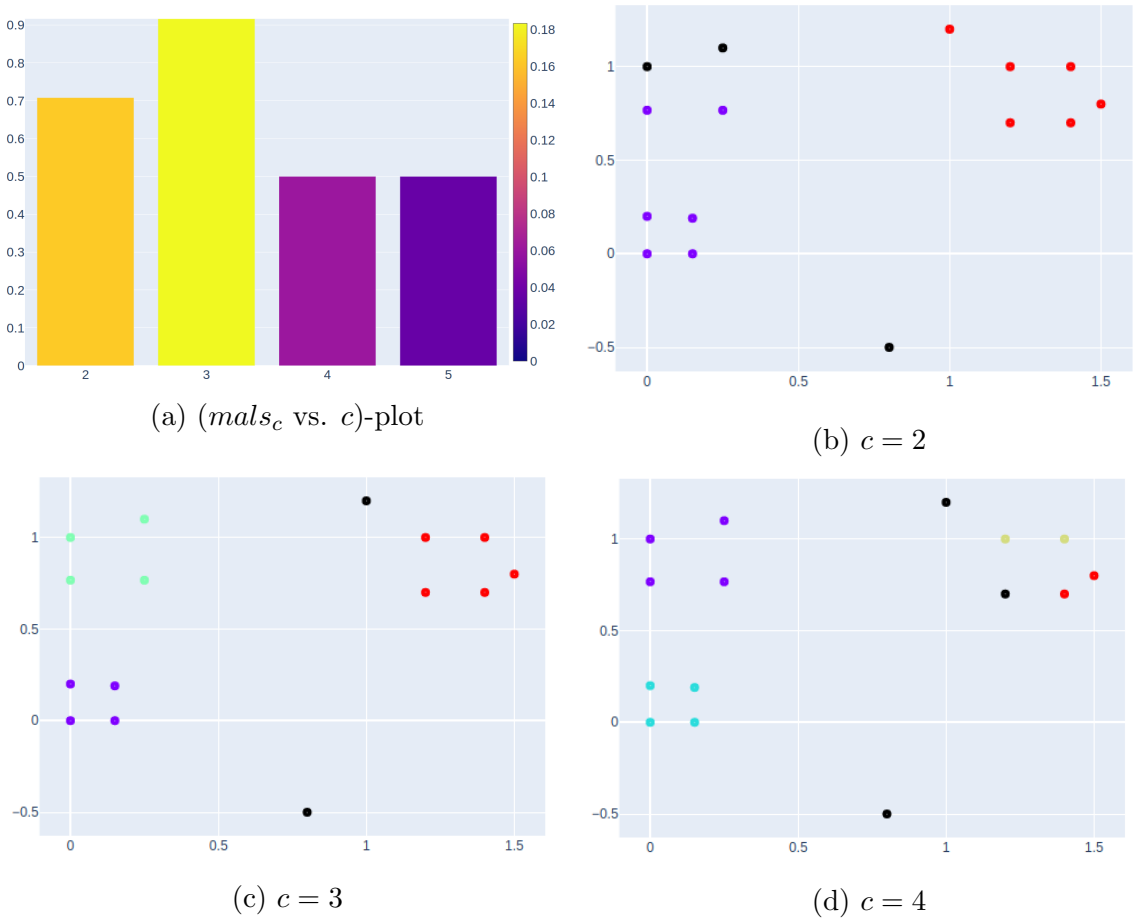
sharp decreases observed at the ground truth number of clusters (see Section 4.3.3).

For example, Figure 12 shows the ($mals_c$ versus c)-plot for the dataset in Figure 9a. The correct number of clusters ($c^* = 3$) is identifiable by the significant decrease in $mals_c$ and $gmals_c$ observed at $c = 4$. Given this observation, the maximal aggregate persistence flat clustering of size $c^* = 3$ correlates with the ground truth.

3.5.2 *Hk-DC*: Cluster Expansion

Recall a clustering is defined as a partitioning of a subset of $\mathcal{X}$, $\boldsymbol{\mathcal{P}}$, with remaining elements identified as noise, O. If O is empty, $\boldsymbol{\mathcal{P}}$ covers $\mathcal{X}$. However, as discussed in Section 1.2, the purpose of identifying noise is to increase separability, where clustering error can be reduced by the assignment of noise to neighboring clusters. In other words, for noise element $o \in O$, o might be better described as lying on the boundary of a cluster $P \in \boldsymbol{\mathcal{P}}$ (lying within a low-density region of P). Thus, o should be assigned to cluster P (i.e., P should be expanded by o) in such cases. As an example, see Figure 12c, which identifies two noise elements, one correctly and the other incorrectly.

From Section 3.5.1, recall $k'_{birth}(P)$ defines the maximum value of k' below which cluster P exists in a *Hk-DC* solution, $k'_{death}(P)$ defining the smallest. Consequently, $k'_{birth}(P)$ defines the largest kNN graph ($k = k'_{birth}(P)$) in which k-density cluster P exists. Thus, a natural upper bound for expanding cluster P is to limit its expansion to the $(k'_{birth}(P) - 1)$NN of elements in P. In other words, noise $o \in O$ should be assigned to cluster P if o is a $(k'_{birth}(P) - 1)$NN of any element in P, $\exists v \in P : w((v, o)) \leq k'_{birth}(P) - 1$. In the case of ties, o is assigned to the P maximizing the inverse sum of nearest neighbor edge weights:



(a) ($mals_c$ vs. c)-plot

(b) $c = 2$

(c) $c = 3$

(d) $c = 4$

Fig. 12.: (a) ($mals_c$ versus c)-plot (colored by $gmals_c$) for the dataset shown in Figure 9, with maximal aggregate persistence flat clusterings for $c = [2..4]$ ((b)-(c) color indicating clusters and black points indicating noise). Note that in (a), x-axis is $mals_c$, y-axis is c, and color is $gmals_c$; and that the parameters $w = 1$, $k = n - 1$, and $m = 1$ were used.

$$nnw(P, o) = \sum_{v \in P : w((v,o)) \leq k'_{birth}(P)-1} \frac{1}{w((v, o))} \tag{3.17}$$

Note that this expansion of clusters $P \in \boldsymbol{\mathcal{P}}$ with noise O can be applied recursively. For example, assuming noise $o \in O$ is assigned to P, noise in o's $(k'_{birth}(P) - 1)$NN could likewise be assigned to P. This is being performed iteratively, terminating when no new noise assignments exist, or O has been exhausted. Note that each iteration increases the graph distance from P's original elements by one in the $(k'_{birth}(P) - 1)$NN graph. Thus, ones confidence in o belonging to P should decrease after each iteration.

Algorithm 6 lists the pseudo-code for cluster expansion, returning an updated clustering $(\boldsymbol{\mathcal{P}}, O)$. Inputs include the previously selected *Hk-DC* flat clustering $(\boldsymbol{\mathcal{P}}, O)$, kNN graph $\boldsymbol{G_k}$, and boolean indicator *recursive*. Lines 3-5 compute the inverse sum of nearest neighbor edge weights (Equation 3.17) for each cluster-noise pair (P, o). This value is zero when o is not in P's $(k'_{birth}(P) - 1)$NN. Lines 6-10 assigns noise o to cluster P that maximizes the inverse sum of nearest neighbor weights if this value is greater than zero and removes o from the set of noise. This process is continued (Line 11) if *recursive* is true, the set of noise is nonempty, and new assignments were performed in the previous iteration.

As an example, Figure 13 shows an expansion of the flat clustering in Figure 12c. Element $\boldsymbol{x}_9$ is assigned to cluster 3 because it is the second nearest neighbor of $\boldsymbol{x}_{11}$ in 3, while element $\boldsymbol{x}_{15}$ remains noise as it is not in the third nearest neighborhood of any cluster element. Further discussion of this expansion procedure is provided in Section 4.3.3.

Algorithm 6 *HkDC_cluster_expansion*

Input: $\boldsymbol{\mathcal{P}}, O, \boldsymbol{G}_k, recursive$
Output: $\boldsymbol{\mathcal{P}}, O$

do
  $\boldsymbol{W}_{|\boldsymbol{\mathcal{P}}|\times|O|} = 0$
  for $P \in \boldsymbol{\mathcal{P}}$ **do**
    for $o \in O$ **do**
      $\boldsymbol{W}_{P,o} = nnw(P, o)$
  for $o \in O$ **do**
    $P = \arg\max_P \boldsymbol{W}_{P,o}$
    if $\boldsymbol{W}_{P,o} > 0$ **then**
      $P = P + o$
      $O = O \setminus o$
while *recursive* and $|O| > 0$ and $max(\boldsymbol{W}) > 0$
return P, O

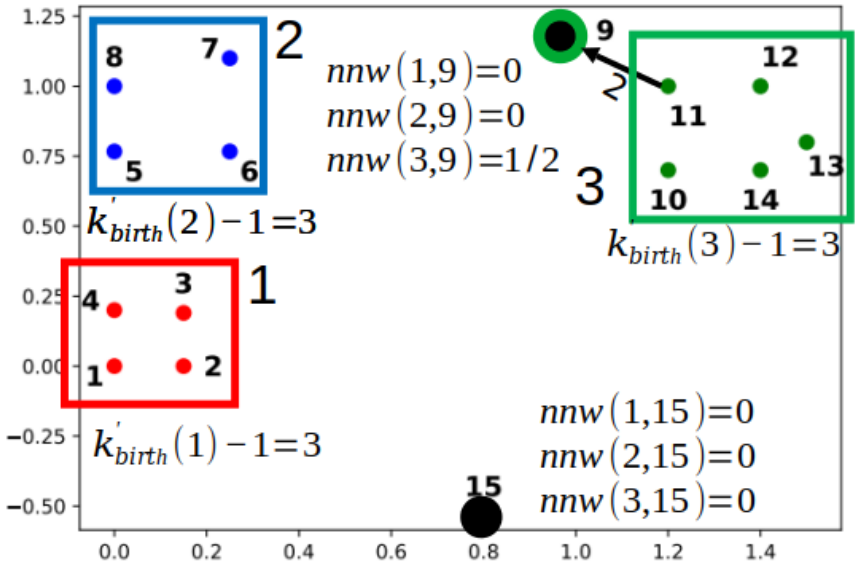


Fig. 13.: Expansion of the flat clustering (colors indicating clusters and black points indicating noise) shown in Figure 12c for $recursive = true$. Note that the directed edge indicates the assignment of noise (9) through a cluster element (11).

3.5.3 *Hk-DC*: Complexity

Hk-DC's complexity depends on the cost of computing the exact or approximate kNN graph, along with the choice of k. Here will assume that the kNN graph is given (see Section 2.5 for a discussion on kNN graph complexity). Then, given $\boldsymbol{G}_k$, edge weights of $\boldsymbol{G}_k$ (Equation 3.3) require kNN to be sorted by distance, which has a complexity of $\mathcal{O}(kn \log k)$, using an efficient sorting method such as quicksort [108] or mergesort [109]. Note that this cost may be built into the cost of computing the kNN graph.

The RkNN density function (Equation 3.4) has complexity $\mathcal{O}(1)$. The complexity of k-density is $\mathcal{O}(k \log k)$, as k-density (Equation 3.5) can be efficiently computed by sorting incoming edges by edge weight and selecting the weight at index ε (i.e., the minimum value of k with at least ε reverse nearest neighbors). Thus, overall k-density complexity is $\mathcal{O}(kn \log k)$. This complexity may increase depending on the chosen density function (e.g., densities with greater than one-hop neighborhood support).

Construction of the mutual reachability graph $\boldsymbol{G}_{f_k,\varepsilon}$ requires computing $\boldsymbol{G}_k$'s collapsed graph, $\mathcal{O}(kn)$. *Hk-DC* (Algorithm 4) computes the minimum spanning tree of $\boldsymbol{G}_{f_k,\varepsilon}$ and sorts the resulting tree's edges by weight. The minimum spanning tree is $\mathcal{O}(kn \log n)$ using an efficient version of the Prim [110] or Kruskal [111] algorithm, while sort is $\mathcal{O}(n \log n)$ (minimum spanning tree edges may be returned in order by weight). The divisive clustering cost depends on the height of the resulting hierarchical clustering tree, which is at most k, along with the cost of computing the set of connected components after each split. Thus, complexity is $\mathcal{O}(kn)$, where at each level of the tree, computing connected components cost at most $\mathcal{O}(n)$ as at most n elements exist at each level, and a vertex belonging to each component is known (vertices incident to the removed edge). Thus, the cost of *Hk-DC* is $\mathcal{O}(kn \log k)$ plus

the cost of computing the kNN graph, which may dominate the complexity (e.g., $\mathcal{O}(kn^2)$).

The cost of running *Hk-DC* over the range of density threshold $\boldsymbol{e}$ is $\mathcal{O}(|\boldsymbol{e}|kn \log k)$. Without loss of generality, assume that hierarchical clustering tree $\boldsymbol{T}$ is a binary tree as an arbitrary m-ary tree can be converted to a binary tree of equivalent height ($\mathcal{O}(\log_2 |\boldsymbol{T}|) \equiv \mathcal{O}(\log_m |\boldsymbol{T}|)$). Computing cluster persistence (Equation 3.13) is worst-case $\mathcal{O}(|\boldsymbol{e}|w4^k)$ as a perfect binary tree of depth k contains $2(2^k) - 1 = \mathcal{O}(2^k)$ vertices with $\mathcal{O}((2^k)^2) = \mathcal{O}(4^k)$ pairs. The number of flat clusterings $|\boldsymbol{\mathcal{F}}|$ in $\boldsymbol{T}$ is equal to the number of full rooted subtrees in $\boldsymbol{T}$. For a perfect binary tree $\boldsymbol{T}$ of depth $d_{\boldsymbol{T}}$, the number of full rooted subtrees is equal to the number of full rooted subtrees (minus one) of a perfect binary tree of depth $d_{\boldsymbol{T}} - 1$ squared (e.g., $|\boldsymbol{\mathcal{F}}| = 1, 4, 25, 676, 458329, 210066388900, ..$ for $d_{\boldsymbol{T}} = 1, 2, 3, 4, 5, 6, ..$). Thus, computing all flat clusterings $\boldsymbol{\mathcal{F}}$ of $\boldsymbol{T}$ is intractable for even small values of k. Instead, a subset $\boldsymbol{\mathcal{F}}_{m_{\boldsymbol{\mathcal{F}}}} \subseteq \boldsymbol{\mathcal{F}}$ such that $|\boldsymbol{\mathcal{F}}_{m_{\boldsymbol{\mathcal{F}}}}| \leq m_{\boldsymbol{\mathcal{F}}}$ is used, where $\boldsymbol{\mathcal{F}}_{m_{\boldsymbol{\mathcal{F}}}}$ is computed by greedily traversing $\boldsymbol{T}$ by vertex size.

The complexity of computing the aggregate cluster persistence scores (Equations [3.14,3.15,3.16]) is $\mathcal{O}(|\boldsymbol{e}|m_{\boldsymbol{\mathcal{F}}}2^k)$ as the maximum size of a flat clustering in a perfect binary tree of depth k is equal to 2^k (number of leaf vertices). Finally, expansion (Algorithm 6) has a complexity of $\mathcal{O}(nk)$, where each edge in $\boldsymbol{G}_k$ is traversed. Note that minimum cluster size m can be used to decrease complexity by significantly decreasing the size of $\boldsymbol{T}$. For example, given a perfect binary tree $\boldsymbol{T}$ of depth k with uniform vertex size, $m = n/2^k$ decreases the size of $\boldsymbol{T}$ by 2^k vertices (effectively reducing the depth of $\boldsymbol{T}$ by one). Similarly, increasing the value of density threshold ε decreases the size of $\boldsymbol{T}$. Let $p(g_{f_k}(v, \varepsilon) \leq k)$ be the probability that the k-density of element $v \in \mathrm{v}(\boldsymbol{G}_k)$ is less than or equal to k for ε. As $\forall v \in \mathrm{v}(\boldsymbol{G}_k) : p(g_{f_k}(v, \varepsilon) \leq k) \geq p(g_{f_k}(v, \varepsilon + 1) \leq k)$, increasing ε decreases the probability that any element will

be k-dense, decreasing the number of elements at depth k in $\boldsymbol{T}$.

CHAPTER 4

RESULTS & DISCUSSION

4.1 Introduction

In this chapter, the two proposed clustering algorithms, *RNN-DBSCAN* and *Hk-DC*, are evaluated, and their performance is discussed. Section 4.2 presents the results for *RNN-DBSCAN*, with performance evaluation in Section 4.2.3. Additionally, Section 3.4.2 presents results on the choice of k heuristic, while Section 4.2.2 discusses the effect of n on k. Finally, Section 4.2.4 investigates the effect of using an approximate kNN graph on performance. Section 4.3 presents the results for *Hk-DC*, with performance evaluation in Section 4.3.3. Additionally, Sections 4.3.1 and 4.3.2 discuss the effect of n and m (minimum cluster size) on k.

The artificial datasets in Table 2 and the real-world datasets in Table 3 were used to evaluate *RNN-DBSCAN* and *Hk-DC*. Short descriptions of the real-world datasets are banknote (banknote authentication), ctg (cardiotocography fetal state), digits (optical recognition of handwritten digits), ecoli (ecoli protein localization sites), htru2 (pulsar candidates), iris (iris plant type), seeds (varieties of wheat kernels), farm (farm satellite image), and house (individual household electric power consumption).

Clustering performance was measured using external evaluation metrics Adjusted Rand Index (ARI) [120] and Normalized Mutual Information (NMI) [121, 122]. ARI is a similarity measure between two clusterings adjusted for chance that is related to accuracy, while NMI quantifies the amount of information obtained about one clustering through the other (i.e., the mutual dependence between the two). In the

Table 2.: Artificial Datasets

Name	**Size**	**# Classes**	**# Dimensions**
aggregation [112]	788	7	2
blobs [113]	1K,10K,100K,1M	5	3
can3147 [63]	3147	4	2
can473 [63]	473	8	2
circle [113]	1K,10K,100K,1M	2	2
clust3_2d [114]	330	3	2
clust3_3d [114]	330	3	3
compound [112]	399	6	2
d31 [112]	3100	31	2
fire [112]	1025	2	2
flame [112]	240	2	2
grid	1250	2	2
jain [112]	373	2	2
moons [113]	1K,10K,100K,1M	2	2
pathbased [112]	300	3	2
r15 [112]	600	15	2
spiral [112]	312	3	2
swissroll [113]	1K,10K,100K,1M	2	3
t4[†][115]	8000	6	2
t7[†][115]	10000	9	2
t8[†][115]	8000	8	2

[†] Original dataset was unlabeled requiring elements to be labeled manually.

Table 3.: Real-World Datasets

Name	**Size**	**# Classes**	**# Dimensions**
banknote [116]	1372	2	4
ctg [116]	2126	10	19
digits [116]	1,797	10	64
ecoli [116]	336	8	7
farm[†][117, 118]	3,627,086	-	4
house[†][117, 116]	2,049,280	-	6
htru2 [116, 119]	17,898	2	8
iris [116]	150	3	4
seeds [116]	210	3	7

[†] Datasets are unlabeled.

case of elements being identified as noise, each noise element was treated as a distinct singleton cluster for both ARI and NMI. Additionally, clustering Purity was used, a weighted average of the percentage of elements belonging to the dominant class in each cluster.

Recall that *RNN-DBSCAN* requires one model parameter defining the k-nearest neighbor graph size (k), while *Hk-DC* requires additional model parameters: density threshold (ϵ) and minimum cluster size (m). In both cases, a Euclidean distance measure was used to compute the k-nearest neighbor graph.

4.2 *RNN-DBSCAN*

4.2.1 *RNN-DBSCAN*: Choice of k

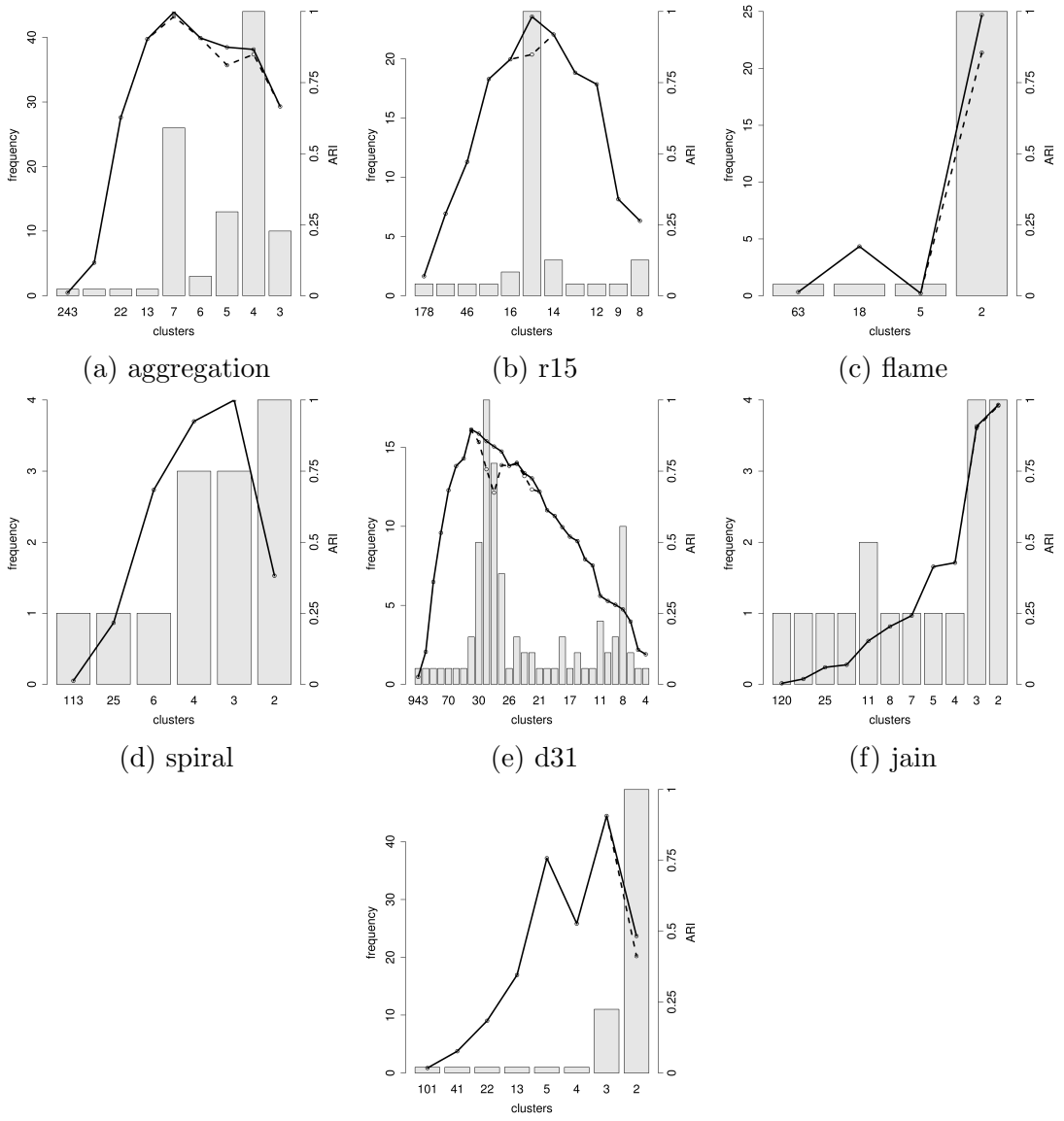
Two heuristics for selecting k based on clustering stability and the internal evaluation metric *DBVC* were proposed in Section 3.4.2. First, for clustering stability, it was assumed that a correct choice of k could be discovered by examining the histogram of clustering size for a range of k and selecting the first spike in the histogram (in descending order by cluster size). Empirical evidence for this assumption is shown in Figure 14, where optimal clustering performance is highly correlated with this first spike. However, this only suggests an appropriate number of clusters that may be obtained using multiple values of k.

Additionally, Figure 14 shows maximum ARI performance (solid line) and ARI performance at the minimum value of k (dashed line). Here a strong correlation is observed between the two measures suggesting that the minimum value of k is an appropriate choice for the selected number of clusters. Results for two unlabeled datasets are shown in Figure 15, where ARI performance is unknown. Here one sees the resulting cluster solutions corresponding to the minimum value of k at the first spike in the histogram.

Finally, Figure 16 shows the correlation between *DBCV* (gray line) and ARI (black line), suggesting it can also be used to select an appropriate value of k (i.e., k, which maximizes *DBVC*). Though in some cases, high values of *DBCV* result in relatively low values of ARI wrt. the optimal.

4.2.2 *RNN-DBSCAN*: Effect of dataset size n on k

As the computational complexity of *RNN-DBSCAN* is dependent on the cost of constructing the kNN graph, the effect of n on k is considered here. Intuitively,

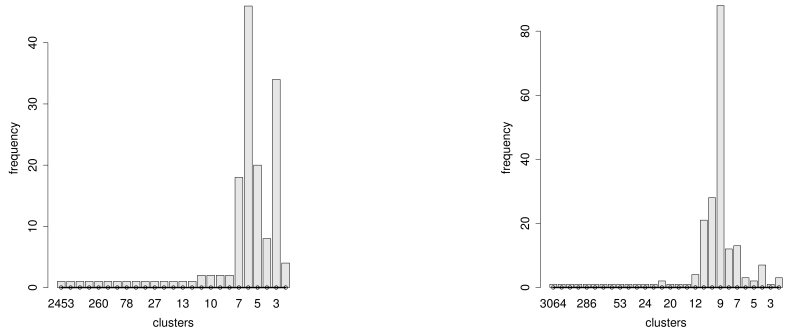


(a) aggregation (b) r15 (c) flame

(d) spiral (e) d31 (f) jain

(g) path-based

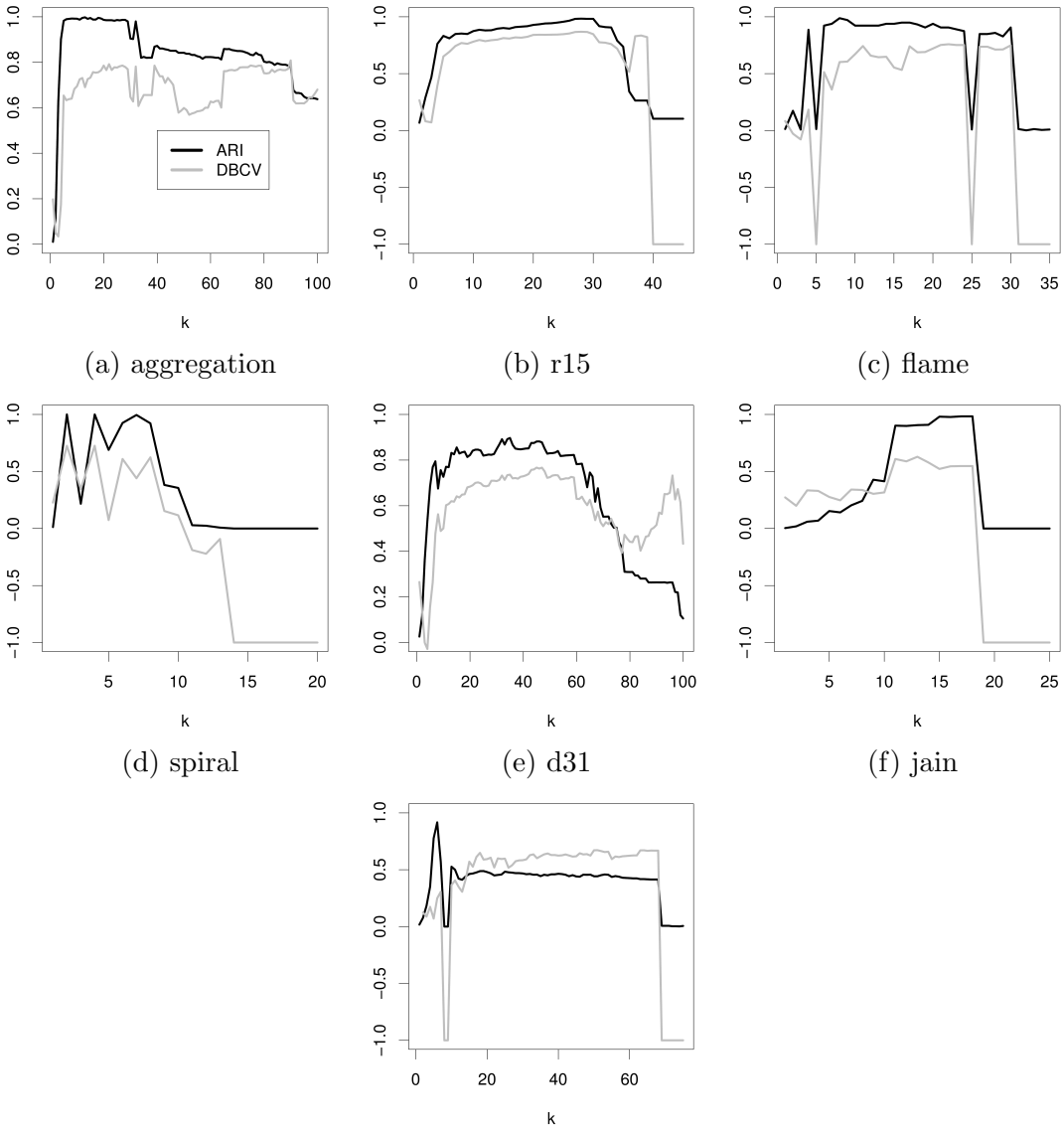
Fig. 14.: Number of clusters histogram (bars) and ARI performance (maximum solid line and at minimum k dashed line) for *RNN-DBSCAN* clusterings produced over the range $k = [1..100]$ using several datasets from Table 2. Note that occurrences of number of clusters equal to 1 are not shown.



(a) (b)

(c) (d)

Fig. 15.: Number of clusters histograms for *RNN-DBSCAN* clusterings produced over the range $k = [1, 200]$ for the (a) *t4* and (b) *t7* datasets from Table 2. Note that occurrences of number of clusters equal to 1 are not shown. *RNN-DBSCAN* clustering results (number of clusters, 6 and 9, determined from (a) and (b) and corresponding minimum k values used) for the (c) *t4* and (d) *t7* datasets.



(a) aggregation (b) r15 (c) flame

(d) spiral (e) d31 (f) jain

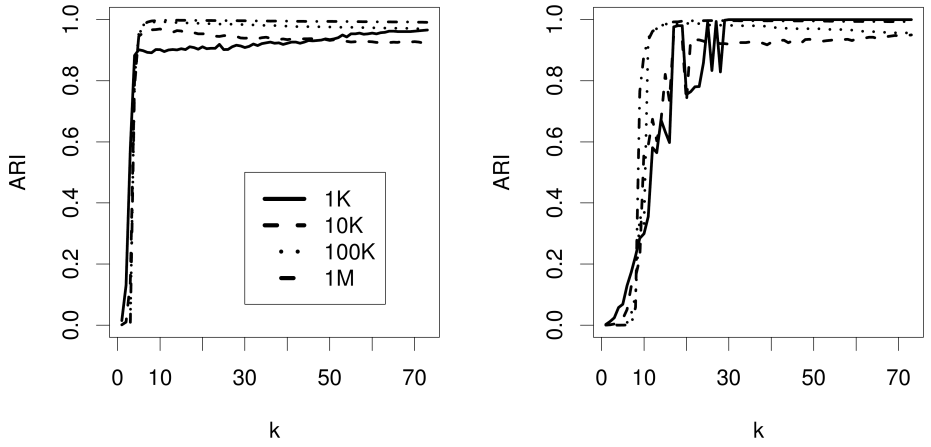
(g) path-based

Fig. 16.: Clustering performance (ARI (black) and *DBCV* (gray)) vs. k for *RNN-DBSCAN* clusterings produced over the range $k = [1..100]$ using several datasets from Table 2.

one would expect the correct choice of k to increase as n increases, for example, at a logarithmic rate $k \approx \log n$ which at a high probability ensures intra-cluster connectivity. However, for reasons discussed in Section 2.5, a much smaller value of k may be chosen. As empirical evidence of this phenomenon, Figure 17 shows plots of ARI performance over a range of k for several artificial datasets, for which the data generation process remained constant while varying the number of drawn elements ($n = 1K, 10K, 100K, 1M$).

Surprisingly, Figure 17 suggests some degree of independence between k and n. For example, in all cases, the lower bound of k at performance convergence ($ARI \approx 1$) occurs at the largest sample (i.e., at around $k \approx 10$ for the $n = 1M$ element datasets). This perceived independence is likely due to finite sample size, along with several other factors, such as the effect of expansion, minimum cluster size given k, and the expected intra-cluster connectivity of elements with high in-degree (i.e., likely belonging to the emerging giant connected component of each cluster).

Less unexpectedly, one observes less variability in clustering performance as sample size increases. One observes relatively stable performance over the selected range of k after convergence for samples of sufficient size. This stability provides some evidence supporting tying the density threshold to k (i.e., increasing the density threshold as k increases). Additionally, an appropriate value of k may exist over a large range of possible values; another interesting observation is that performance decreases slightly as k increases and sample size decreases. This decrease is likely due to more identified noise elements (relative to sample size) as the sample size decreases and k increases.



(a) blobs (b) circle

(c) moons (d) swissroll

Fig. 17.: ARI performance vs k for *RNN-DBSCAN* clusterings produced over the range $k = [1..75]$ using several datasets from Table 2 of sizes $1K$ (solid), $10K$ (dash), $100K$ (dot), and $1M$ (dot-dash).

4.2.3 *RNN-DBSCAN*: Performance Evaluation

To evaluate *RNN-DBSCAN* we compared it to *RECORD* [62], *IS-DBSCAN* [63], and *ISB-DBSCAN* [64]. The latter two model density by mutual nearest neighbors and connectivity by the mutual k-nearest neighbor graph. *DBSCAN* [14] and *OPTICS* [35] were also used. For *OPTICS*, the authors' proposed approach for automatically extracting a clustering from the reachability plot was used.

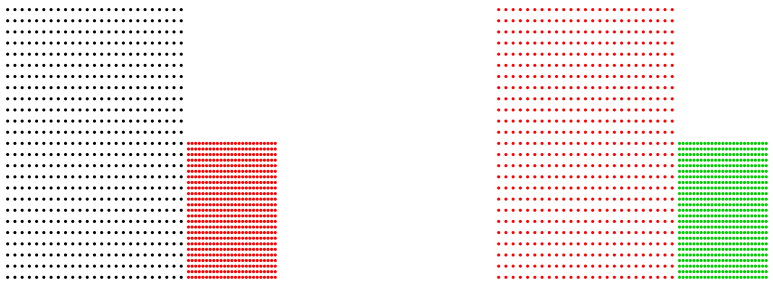
For model parameters, for each of the kNN graph algorithms, k was chosen from the range $k \in [1..100]$. For *DBSCAN*, $MINPTS$ was chosen from the set $\{1, 5, 10, 20\}$, and ε was chosen over the set of ε values defined by the $MINPTS$-nearest neighbor distance of each element. Like *DBSCAN*, *OPTICS* $MINPTS$ was chosen from the same set of values with the maximum value of ε set to the maximum $MINPTS$-nearest neighbor distance, and steepness parameter ξ chosen from the range $\xi \in [0, 1]$. For each algorithm, two sets of parameters were selected, which maximized ARI and NMI, respectively. Note that the element order was fixed for each dataset.

Table 4 shows the ARI performance, Purity, number of clusters, and number of elements identified as noise for *RNN-DBSCAN* (RNN), *RECORD* (REC), *IS-DBSCAN* (IS), *ISB-DBSCAN* (ISB), *DBSCAN* (DBS), and *OPTICS* (OPT) on the artificial datasets. In Table 4, *RNN-DBSCAN* performance is optimal in six of eight datasets (tied for first in one case) while coming in second for the other two (two tied for first in one case). Importantly, in each case, *RNN-DBSCAN* identified the underlying classes of each dataset (i.e., at the maximum ARI solution number of clusters was equal to the ground truth), whereas other approaches failed at this task in at least one case.

DBSCAN performs poorly on the grid dataset by design, whereas each other

Table 4.: Performance of *RNN-DBSCAN* on Artificial Datasets as Measured by ARI and Purity

Data		**RNN**	**REC**	**IS**	**ISB**	**DBS**	**OPT**
aggregate	ARI	**0.998**	0.752	0.872	0.914	0.994	0.979
	# clusters	7	7	6	6	7	8
	Purity	0.999	1.0	0.956	0.956	0.999	0.987
	# noise	0	163	34	0	2	0
d31	ARI	**0.896**	0.539	0.71	0.739	0.868	0.874
	# clusters	31	38	34	43	31	60
	Purity	0.975	0.928	0.901	0.861	0.982	0.95
	# noise	167	1051	492	244	286	0
flame	ARI	**0.971**	0.631	0.682	0.215	0.944	0.928
	# clusters	2	2	2	23	2	3
	Purity	0.996	0.995	0.981	1.0	0.992	0.983
	# noise	2	43	31	33	4	0
jain	ARI	0.983	0.417	0.819	**1.0**	0.941	**1.0**
	# clusters	2	2	2	2	4	2
	Purity	1.0	1.0	1.0	1.0	1.0	1.0
	# noise	2	115	34	0	1	0
pathbased	ARI	**0.917**	0.763	0.759	0.789	0.655	0.684
	# clusters	3	3	5	5	10	7
	Purity	0.99	1.0	0.986	0.989	0.986	0.957
	# noise	11	50	21	16	11	0
r15	ARI	0.984	0.751	0.807	**0.993**	0.979	0.956
	# clusters	15	14	15	15	15	16
	Purity	0.995	0.932	0.986	0.997	0.995	0.977
	# noise	3	103	91	0	6	0
spiral	ARI	**1.0**	**1.0**	0.947	**1.0**	**1.0**	0.653
	# clusters	3	3	3	3	3	6
	Purity	1.0	1.0	1.0	1.0	1.0	0.888
	# noise	0	0	11	0	0	0
grid	ARI	**1.0**	0.922	0.994	0.997	0.5	0.997
	# clusters	2	2	2	2	1	3
	Purity	1.0	1.0	0.999	0.999	1.0	0.999
	# noise	0	50	2	0	625	0



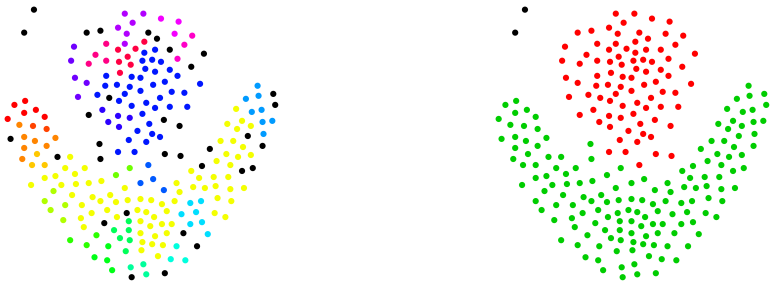
(a) (b)

Fig. 18.: (a) *DBSCAN* and (b) *RNN-DBSCAN* clustering results (maximum ARI solution) for the *grid* dataset from Table 2. Note that elements colored black were identified as noise by the clusterings.

approach can identify each grid. For example, Figure 18 shows that *DBSCAN* incorrectly identifies one grid as noise, while *RNN-DBSCAN* correctly identifies both grids. Overall, *DBSCAN* performs well across all the datasets, excluding *grid* and *pathbased*, though it does require the choice of two parameters ($MINPTS$ and ε). Note that the *pathbased* dataset consists of three classes (two blobs and one chain with varying densities).

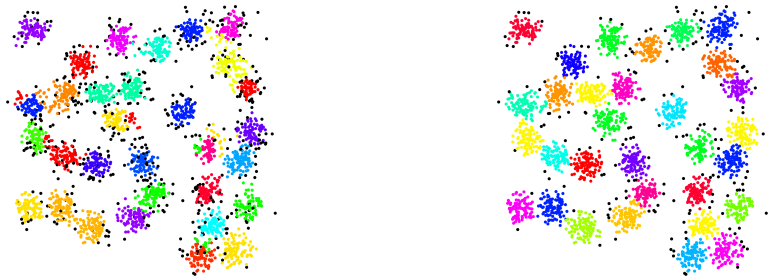
IS-DBSCAN and *ISB-DBSCAN* fail in several cases that can be attributed to one of two reasons. First, the influence-space-based approach cannot uncover the underlying class structure as shown in Figure 19 (splits classes amongst multiple clusters) and Figure 20 (splits classes amongst multiple clusters and merges classes into single clusters). Second, for *IS-DBSCAN*, many elements are incorrectly identified as noise, as shown in Figure 20 and Table 4. Like *IS-DBSCAN*, the main shortcoming of *RECORD* is that many elements are incorrectly identified as noise, as seen in Table 4.

For *OPTICS*, using the automated technique to extract clusters by identifying dents in the reachability should be considered. For example, performing a single cut



(a) (b)

Fig. 19.: (a) *ISB-DBSCAN* and (b) *RNN-DBSCAN* clustering results (maximum ARI solution) for the *flame* dataset from Table 2. Note that elements colored black were identified as noise by the clustering.



(a) (b)

Fig. 20.: (a) *IS-DBSCAN* and (b) *RNN-DBSCAN* clustering results (maximum ARI solution) for the *d31* dataset from Table 2. Note that elements colored black were identified as noise by the clustering.

in the reachability is identical to *DBSCAN*. However, the automated technique is an early form of multi-level-set density-based clustering, explaining the improvement over *DBSCAN* (e.g., see *OPTICS* results on the *grid* dataset compared to *DBSCAN*). Table 4 shows that the automated version of *OPTICS* tends to overestimate the number of clusters. Additionally, the number of noise elements is shown to always be zero due to the maximum $MINPTS$-nearest neighbor distance used as the maximum ε value).

Table 5.: Performance of *RNN-DBSCAN* on Artificial Datasets as Measured by ARI

Data	**RNN**	**REC**	**IS**	**ISB**	**DBS**	**OPT**
aggregate	**0.996**	0.742	0.888	0.954	0.991	0.969
d31	**0.934**	0.772	0.844	0.879	0.911	0.921
flame	**0.931**	0.54	0.567	0.362	0.869	0.875
jain	0.97	0.376	0.709	**1.0**	0.862	**1.0**
pathbased	**0.872**	0.706	0.735	0.772	0.704	0.686
r15	0.988	0.881	0.871	**0.994**	0.984	0.964
spiral	**1.0**	**1.0**	0.917	**1.0**	**1.0**	0.685
grid	**1.0**	0.824	0.983	0.991	0.301	0.991

Purity performance is good overall for each clustering approach. However, Purity must be considered for the number of clusters and noise elements. Hence, the discrepancies in ARI/NMI versus Purity performance. Table 5 shows NMI performance results on the set of artificial datasets. *RNN-DBSCAN* performance is identical to those discussed for ARI (i.e., optimal for six of eight datasets and second for the other two).

Table 6 shows ARI performance, Purity, number of clusters, and number of noise elements for *RNN-DBSCAN* (RNN), *RECORD* (REC), *IS-DBSCAN* (IS), *ISB-DBSCAN* (ISB), *DBSCAN* (DBS), and *OPTICS* (OPT) on the real-world datasets. *RNN-DBSCAN* ranks first in three, second in three, and third in one of the datasets. On the other hand, *DBSCAN* ranks first in four of the seven datasets, though ranking no better than third (fourth in one case) on the remaining datasets. Additionally, for two of the dataset *DBSCAN* ranks first in, both *RNN-DBSCAN* and *DBSCAN* perform relatively well (*ctg*) or poorly (*htru*).

Similar to the artificial datasets, *RECORD* performs the worst, and *ISB-DBSCAN* slightly outperforms its predecessor *IS-DBSCAN*. In the case of *RECORD*

and *IS-DBSCAN*, from Table 6 one can again see that both methods have issues with identifying noise observations. Finally, Table 7 shows NMI performance results on the same set of real-world datasets. Again, one sees that these results correlate with the ARI observations.

Table 8 shows comparison tests using ARI performance for each pair of clustering algorithms following the approach in [123]. The results suggest that *RNN-DBSCAN* performance is significantly better ($p-value \leq 0.05$) than all other evaluated methods except *DBSCAN*. In the case of *DBSCAN*, no significant conclusions can be drawn, though *RNN-DBSCAN* is optimal in more cases. However, this is still significant as in *RNN-DBSCAN*, the density threshold is fixed, whereas in *DBSCAN*, the threshold is varied. Additionally, the connectivity parameter is selected from a smaller input domain. Given the difficulty in handling noise of the prior kNN approaches, we suggest that the performance gains in *RNN-DBSCAN* are due to better noise handling. Furthermore, we suggest this is not observed in *DBSCAN* due to its larger connectivity input domain. In other words, cluster boundaries defined by the region enclosed by a set of intersecting hyper-spheres may be more precisely adjusted by ε.

4.2.4 *RNN-DBSCAN*: Approximate k Nearest Neighbor Results

An approximate solution is investigated, given the reliance of *RNN-DBSCAN* on computing the kNN graph and the associated high computational cost to solve this problem exactly in high dimensions. The *NN-DESCENT* [98] algorithm produces an approximate kNN graph based on the assumption that a neighbor of a neighbor is also likely a neighbor. *NN-DESCENT* assumes that an approximate kNN graph can be incrementally improved by exploring an element's neighbors' neighborhoods.

Given approximate kNN graph $\boldsymbol{G}_k$, let $B(\boldsymbol{x})$ define the neighborhood of element $\boldsymbol{x} \in \mathrm{v}(\boldsymbol{G}_k)$ as the union of the kNN and RkNN of $\boldsymbol{x}$, $B(\boldsymbol{x}) = N_{\boldsymbol{x},k} \cup \{(u, w) \in \mathrm{e}(\boldsymbol{G}_k) :$

Table 6.: Performance of *RNN-DBSCAN* on Real-World Datasets as Measured by ARI and Purity

Data		**RNN**	**REC**	**IS**	**ISB**	**DBS**	**OPT**
banknote	ARI	**0.771**	0.086	0.596	0.594	0.558	0.225
	# clusters	3	4	2	3	8	35
	Purity	0.985	0.828	0.894	0.896	0.896	0.98
	# noise	34	580	25	10	1	0
ctg	ARI	0.951	0.057	0.883	0.902	**0.992**	0.892
	# clusters	10	14	6	9	13	17
	Purity	1.0	0.372	0.999	0.999	1.0	0.995
	# noise	91	796	409	179	5	0
digits	ARI	**0.739**	0.011	0.462	0.695	0.684	0.315
	# clusters	34	3	25	18	21	29
	Purity	0.936	0.245	0.957	0.977	0.983	0.733
	# noise	104	1524	564	298	355	0
ecoli	ARI	0.526	0.14	0.474	0.46	**0.639**	0.591
	# clusters	8	2	4	3	3	5
	Purity	0.736	0.538	0.714	0.711	0.582	0.708
	# noise	10	89	63	55	100	0
htru2	ARI	0.334	0.146	0.147	0.166	**0.552**	0.146
	# clusters	204	56	310	204	4	26
	Purity	0.976	0.915	0.981	0.949	0.977	0.976
	# noise	236	1909	4206	2270	2289	0
iris	ARI	0.644	0.289	0.566	0.568	**0.703**	0.643
	# clusters	4	2	2	2	7	4
	Purity	0.963	0.674	0.671	0.667	0.978	0.847
	# noise	16	55	1	0	16	0
seeds	ARI	**0.617**	0.416	0.383	0.361	0.491	0.498
	# clusters	4	3	2	9	4	6
	Purity	0.898	0.903	0.653	0.888	0.95	0.857
	# noise	4	65	34	22	51	0

Table 7.: Performance of *RNN-DBSCAN* on Real-World Datasets as Measured by NMI

Data	**RNN**	**REC**	**IS**	**ISB**	**DBS**	**OPT**
banknote	**0.68**	0.213	0.59	0.585	0.579	0.363
ctg	0.934	0.399	0.803	0.886	**0.99**	0.902
digits	**0.824**	0.47	0.648	0.775	0.77	0.67
ecoli	0.569	0.538	0.551	0.571	**0.6**	0.55
htru2	0.195	0.116	0.117	0.109	**0.25**	0.109
iris	0.683	0.445	0.723	**0.734**	**0.734**	**0.734**
seeds	**0.618**	0.48	0.487	0.495	0.533	0.525

Table 8.: Wilcoxon Signed-Rank Test p-values for ARI Performance in Tables 4 and 6

	RNN	**REC**	**IS**	**ISB**	**DBS**	**OPT**
RNN		**0.0005485**	**3.052e-05**	**0.001586**	0.1292	**0.002136**
REC	0.9996		0.999	0.994	0.9958	0.995
IS	1	**0.001312**		0.9696	0.9908	0.8349
ISB	0.9987	**0.007177**	**0.03452**		0.9063	0.5279
DBS	0.8835	**0.005029**	**0.0107**	0.1046		0.06027
OPT	0.9983	**0.006018**	0.1796	0.5	0.9465	

$w = \boldsymbol{x}\}$. A basic implementation of *NN-DESCENT* is described as follows. Begin with a random approximation of the kNN graph. Then, for each element $\mathbf{x}$, $\mathbf{x}$ is compared with its neighbors' neighbors $\mathbf{z}$ such that $\mathbf{z} \in B(\mathbf{y})$ and $\mathbf{y} \in B(\mathbf{x})$. At each comparison attempting to update the kNN of $\mathbf{x}$ with $\mathbf{z}$. This process is repeated over all elements until no updates of the current k-nearest neighbor graph approximation are found (i.e., no new kNN are discovered).

In addition to the basic implementation of *NN-DESCENT*, improvements are also presented in [98] based on local join, incremental search, sampling, and early termination. These improvements reduce the number of comparisons and improve data locality for distributed implementations. Two parameters, ρ and δ, are introduced in this improved implementation with ρ defining the neighborhood sampling rate and δ defining early termination (i.e., the minimum number of updates). The complexity of each iteration of *NN-DESCENT* is $\mathcal{O}(\rho \times n \times k^2)$, with the number of required iterations to convergence being low (empirically observed to be less than 12 iterations in [98]).

Table 9 shows *NN-DESCENT*'s performance along with the ARI performance of *RNN-DBSCAN* using *NN-DESCENT* on several large artificial and real-world datasets using parameters of $k = 100$, $\rho = 0.1$, and $\delta = 0.001$ were used. Note that scan rate is defined as the number of distance calculations made relative to the number of comparisons required to compute an exact solution, $(n \times (n-1))/2$, whereas recall is the average percentage of correctly identified kNN in the approximate solution. ARI performance is computed by comparing *RNN-DBSCAN* clustering results of the approximate solution versus the exact solution (i.e., the *RNN-DBSCAN* clustering of the exact kNN graph is considered the ground truth). Recall from Section 4.2.2 that the ARI performance of *RNN-DBSCAN* for the listed artificial datasets of sample size $1M$ is approximately one at both $k = 10$ and $k = 100$.

Table 9.: Performance of *RNN-DBSCAN* for an Approximate kNN Graph as Measured by ARI

Data	**Scan Rate**	**Recall**	$\mathbf{ARI}_{k=10}$	$\mathbf{ARI}_{k=100}$
blobs	0.0038	0.996	0.998	0.999
circle	0.0042	0.997	0.973	0.998
moons	0.0042	0.997	0.982	0.998
swissroll	0.0038	0.997	0.982	0.997
farm	0.0013	0.999	0.942	0.982
house	0.0022	0.996	0.978	0.988

In Table 9, nearly perfect ARI performance is observed at $k = 100$, with a slight performance decrease at $k = 10$. Note that better ARI performance might be obtained for both cases by increasing the value of k used in *NN-DESCENT*. Additionally, different values of ρ and δ may likewise improve performance. Concerning complexity, scan rates indicate an observed complexity of $n^{1.5}$ (i.e., subquadratic), though again, this might be improved by adjusting ρ and δ. Overall, from these results, we conclude that using an approximate kNN graph has a minimal effect on *RNN-DBSCAN* performance.

4.3 *Hk-DC*

4.3.1 *Hk-DC*: Choice of k for the kNN graph

Recall that a cluster is defined by a connected component of $(k' \leq k)$-dense elements in the k'NN subgraph of the kNN graph. The effect of k is such that no clusters split at a value greater than k are discoverable. In other words, by increasing k, a new coarser flat clustering is potentially discoverable. For the fully connected kNN graph ($k = n - 1$), all possible flat clusterings are discoverable and contained in the k-density hierarchical clustering. However, it is desirable for efficiency if $k << n$.

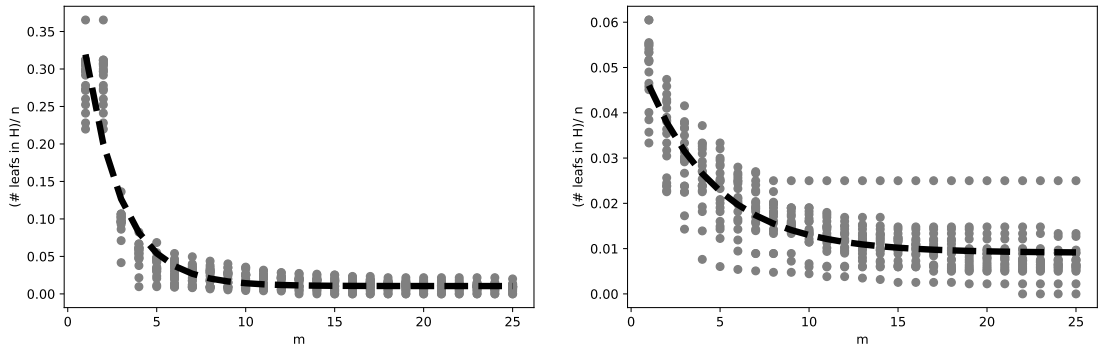
Note that k must be chosen linear to n to ensure connectivity in the kNN graph given well-separated multimodal data. Thus, clusters only discoverable at $k = \mathcal{O}(n)$ may be considered uninteresting and safely ignored, as they correspond to the joining of well-separated groups in the data. In other words, they correspond to highly significant clusters which define a suitable minimum level of coarseness in the data.

A good choice of k should ensure intra-group connectivity of the well-separated modalities within the data, know to be $k = \mathcal{O}(\log n)$. In fact, in practice, with a high probability, this value has been proven to be of the order $\alpha \log n$ where $\alpha < 1$. Furthermore, as density threshold ε increases, assuming a minimum cluster size $m = \varepsilon$, a minimal value of α has been observed to ensure intra-group connectivity of the ε-dense vertices (see Section 4.2.2). This decrease is due to the emergence of a giant connected component in the group as k increases. Dense vertices being more likely to belong to the giant connected component given their high vertex degree. Similarly, the giant connected component is likely to remain connected given the removal of k. Thus, in practice, k may be chosen much smaller than $\log(n)$.

4.3.2 *Hk-DC*: Choice of min cluster size m

Minimum cluster size m ensures that no cluster of size less than m, pre-expansion, is discoverable by Hk-DC. Increasing m reduces the effect of chaining in the hierarchical clustering (splits resulting in spurious clusters) while improving overall computational efficiency (reducing the number of vertices in the resulting tree). Spurious clusters are better characterized as noise resulting in cluster shrinkage as opposed to an actual split. In practice a good heuristic for m is ε, equal to the minimum size of a connected component containing an ε-dense vertex.

In the best case, one selects the maximum value of m such that no non-spurious clusters are removed. Figure 21 shows the effect of m on the number of vertices in the



(a) $\varepsilon = 1$ (b) $\varepsilon = 10$

Fig. 21.: Number of leaf vertices (scaled by n) versus m in the k-density hierarchical clustering ($\boldsymbol{H}$) of datasets in Tables 2 and 3 for $\varepsilon = \{1, 10\}$. The dashed curve representing the fit over all datasets using $k = n - 1$.

k-density hierarchical clustering. For $\varepsilon = 1$, Figure 21(a) shows that tree size relative to n decreases exponentially as m increases. Figure 21(b) shows that this effect is dampened as ε increases. This damping is because increasing ε also decreases the size of the tree.

For most of the datasets in Tables 2 and 3, the choice of $m = [1..25]$ had a minimal effect on the optimal ARI performance shown in the second column of Table 10 (minimum optimal ARI within 90% of the maximum). This minimal effect is likely due to the choice of $\varepsilon > 1$ having a similar effect as m (see Figure 21). Figure 22 shows the effect of m on the remaining three datasets on ARI performance. All cases show a decrease in ARI performance as m increases, likely due to removing non-spurious clusters given the small size of the datasets (See Table 3).

Overall, we suggest a value of m less than $\log(n)$, which may be replaced with $m = \varepsilon$ as ε increases (i.e., the minimum of the two values).

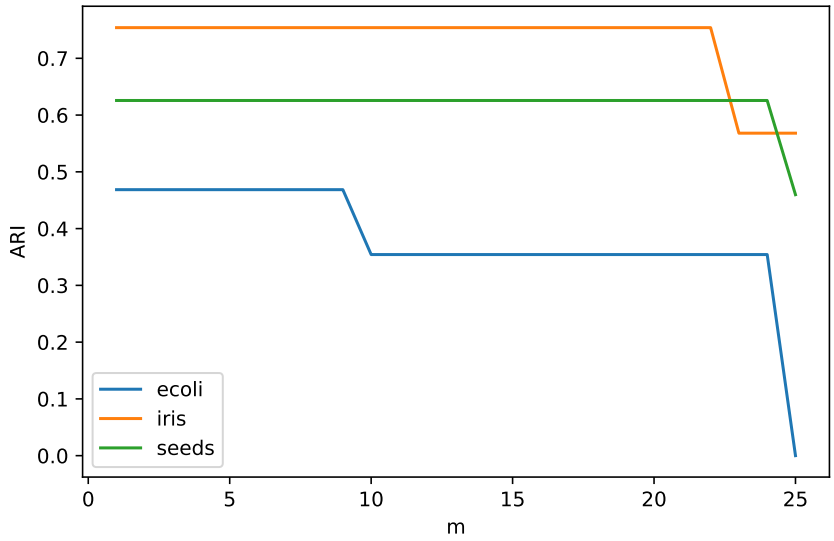


Fig. 22.: Optimal ARI performance versus m for datasets in Table 3 with the large deviation in performance over m (minimum optimal ARI less than 90% of the maximum). The ε value used corresponds to the optimal solutions by ARI shown in the second column of Table 10.

4.3.3 *Hk-DC*: Performance Evaluation

In this section, *Hk-DC*'s performance is investigated using the collection of datasets shown in Tables 2 and 3. As a hierarchical solution is generated over a range of k, *Hk-DC*, performance is dependent on the density threshold parameter ε and the extraction of a flat clustering. Thus, the flat clustering extraction heuristic presented in Section 3.5.1 is evaluated, attempting to address both of these concerns. Additionally, the performance of *Hk-DC* is compared with a hierarchical algorithm using the ε-neighbors graph, *HDBSCAN* [36, 37].

Recall from Section 3.5.1 the ($mals_c$ versus c)-plot is used to select an appropriate number of clusters c^*, where the flat clustering is the solution with c^* clusters with maximal aggregate persistence over a range of density threshold ε. For the following results, the density threshold range was set to $\varepsilon = 1..50$, window size to $w = 5$, maximum number of flat clusterings to $m_{\mathcal{F}} = 2000$, and min cluster size m to value relative to n.

Table 10 shows the optimal ARI performance of *Hk-DC* for each dataset. Optimal referring to the flat clustering solution that maximizes ARI performance, identified by exhaustive search over ε and all flat clusterings. These optimal results are used to evaluate the proposed flat clustering extraction heuristic's performance. Table 11 shows *Hk-DC*'s ARI performance using the sub-optimal flat clustering heuristic. For comparison, Table 11 also lists the performance of *HDBSCAN*. As *Hk-DC* results were generated through exploratory analysis, the reported *HDBSCAN* performance is the optimal solution over ε, referred to as the $MINPTS$ parameter in *HDBSCAN*.

In Tables 10 and 11, one observes that the heuristic selects a flat clustering whose ARI is within 6% of the optimal solution in all but four cases. Additionally, the correct number of clusters is identifiable in all but five cases, with most errors

Table 10.: Optimal Performance of *Hk-DC*† as Measured by ARI

dataset	opt ARI w/ exp_1	opt ARI w/ exp_r
aggregation	0.996	0.996
banknote	0.829	0.847
blobs	1.0	1.0
can3147	0.978	0.981
can473	0.769	0.759
clust3_2d	0.976	0.986
clust3_3d	0.945	0.918
compound	0.907	0.926
d31	0.927	0.944
digits	0.560	0.833
ecoli	0.469	0.702
fire	0.999	0.997
flame	0.962	1.0
iris	0.745	0.811
moon	1.0	1.0
r15	0.993	0.993
seeds	0.626	0.800
spiral	1.0	1.0
t4	0.945	0.912
t7	0.951	0.893
t8	0.966	0.941

† With cluster expansion (single (exp_1) and recursive (exp_r) iteration).

Table 11.: Performance of *Hk-DC*[†‡] as measured by ARI

dataset	c^*	*Hk-DC* w/o exp	*Hk-DC* w/ exp_1	*Hk-DC* w/ exp_r	*HDBSCAN*
aggregation	7	0.550	**0.990**	**0.990**	0.838
banknote	2	0.012	0.011	0.011	**0.327**
blobs	5	**1.0**	**1.0**	**1.0**	**1.0**
can3147	3	0.769	0.924	**0.927**	0.912
can473	8	0.605	0.659	0.650	**0.868**
clust3_2d	3	0.904	**0.971**	0.945	0.956
clust3_3d	3	0.724	**0.922**	0.860	0.920
compound	5	0.810	0.844	0.853	**0.913**
d31	31	0.105	0.923	**0.941**	0.603
digits	10	0.020	0.521	**0.791**	0.526
ecoli	3	0.008	0.457	**0.651**	0.399
fire	2	0.805	**0.993**	0.758	0.948
flame	2	0.325	**0.967**	0.955	0.824
iris	2	**0.558**	**0.558**	**0.558**	0.548
moon	2	0.988	**1.0**	**1.0**	**1.0**
r15	15	0.810	**0.982**	**0.982**	0.957
seeds	3	0.011	0.500	**0.744**	0.257
spiral	3	**1.0**	**1.0**	**1.0**	**1.0**
t4	6	0.457	0.933	0.892	**0.969**
t7	9	0.668	0.935	0.885	**0.977**
t8	6	0.493	0.691	0.664	**0.940**

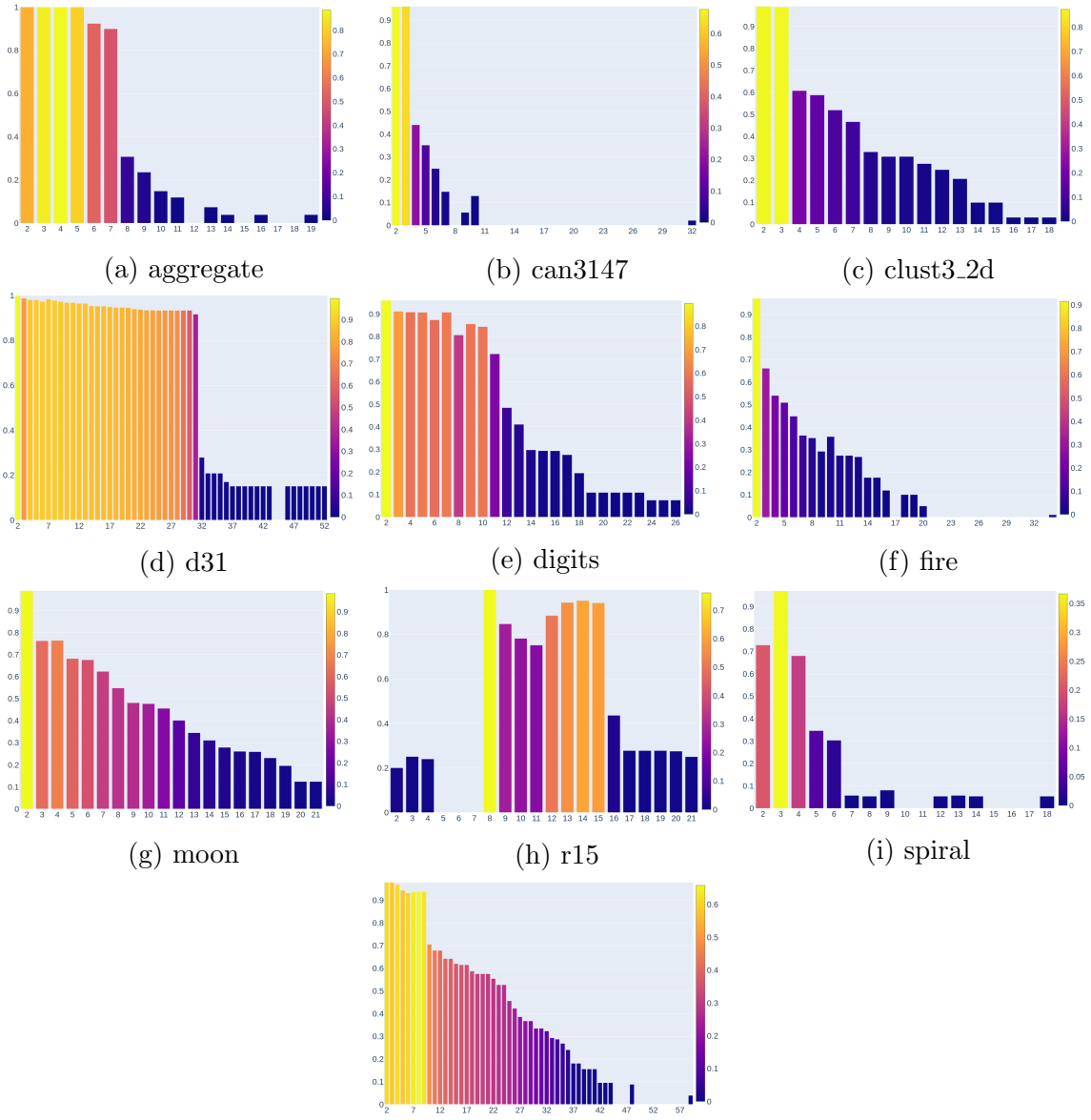
[†] With and without cluster expansion (single (exp_1) and recursive (exp_r) iteration).

[‡] Note that c^* corresponds to the number of clusters selected by examining the ($mals_c$ versus c)-plot.

within $+/-1$ cluster of the ground truth. Thus, using this heuristic, in most cases, one can use the heuristic to select the correct number of clusters and flat clustering whose performance resembles that of the optimal solution.

Figure 23 shows ($mals_c$ versus c)-plots of several datasets from Table 2. In all but two cases, the ground truth number of clusters is discoverable using the ($mals_c$ versus c)-plots, indicated by sharp decreases in $mals_c$ as c increases (e.g., at $c = 31$ for the d31 dataset or $c = 7$ for the aggregate dataset). Note that in the case of multiple elbows, preference is given to the larger solution. Similarly, the spiral and fire datasets show a sharp decrease in the $gmals_c$ score (indicated by color). One deviation in the heuristic is seen for the can3147 dataset, where the plot indicates three as opposed to the ground truth of four clusters. However, Figure 24 shows that the selection of three clusters is not a poor choice. Finally, the selection of ten clusters (vs. eleven) is less evident for the digits dataset. Here ten was selected due to the sharp decrease in $gmals_c$.

Returning to Table 11, one observes that cluster expansion improves ARI performance in all but one case. However, no meaning can be derived from the deviating case (banknote) as overall performance is low. Of interest are the deviations in ARI performance between the two expansion methods (single iteration (exp_1) and recursive (exp_r)). First, recall that recursive expansion is a multi-iteration variant of single iteration expansion (i.e., the number of expanded elements for recursive is strictly greater than or equal to the case of single iteration). Unsurprisingly, the more aggressive expansion method (exp_r) performs better on datasets without noise. In contrast, exp_1 performs better on datasets containing noise (see ARI performances results in Tables 10 and 11). As an example, Figure 25 shows expansion for the d31 and fire datasets. Here a more aggressive expansion is preferred given a lack of noise (d31), whereas the less aggressive strategy is preferred given noise.



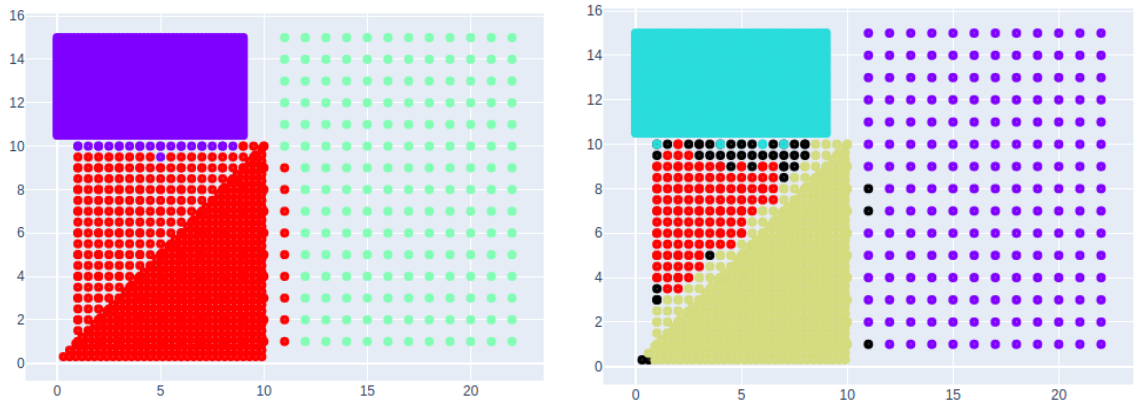
(a) aggregate (b) can3147 (c) clust3_2d

(d) d31 (e) digits (f) fire

(g) moon (h) r15 (i) spiral

(j) t7

Fig. 23.: ($mals_c$ versus c)-plots with number of cluster c shown in the x-axis, $mals_c$ in the y-axis, and colored by $gmals_c$ for ten datasets from Table 2.

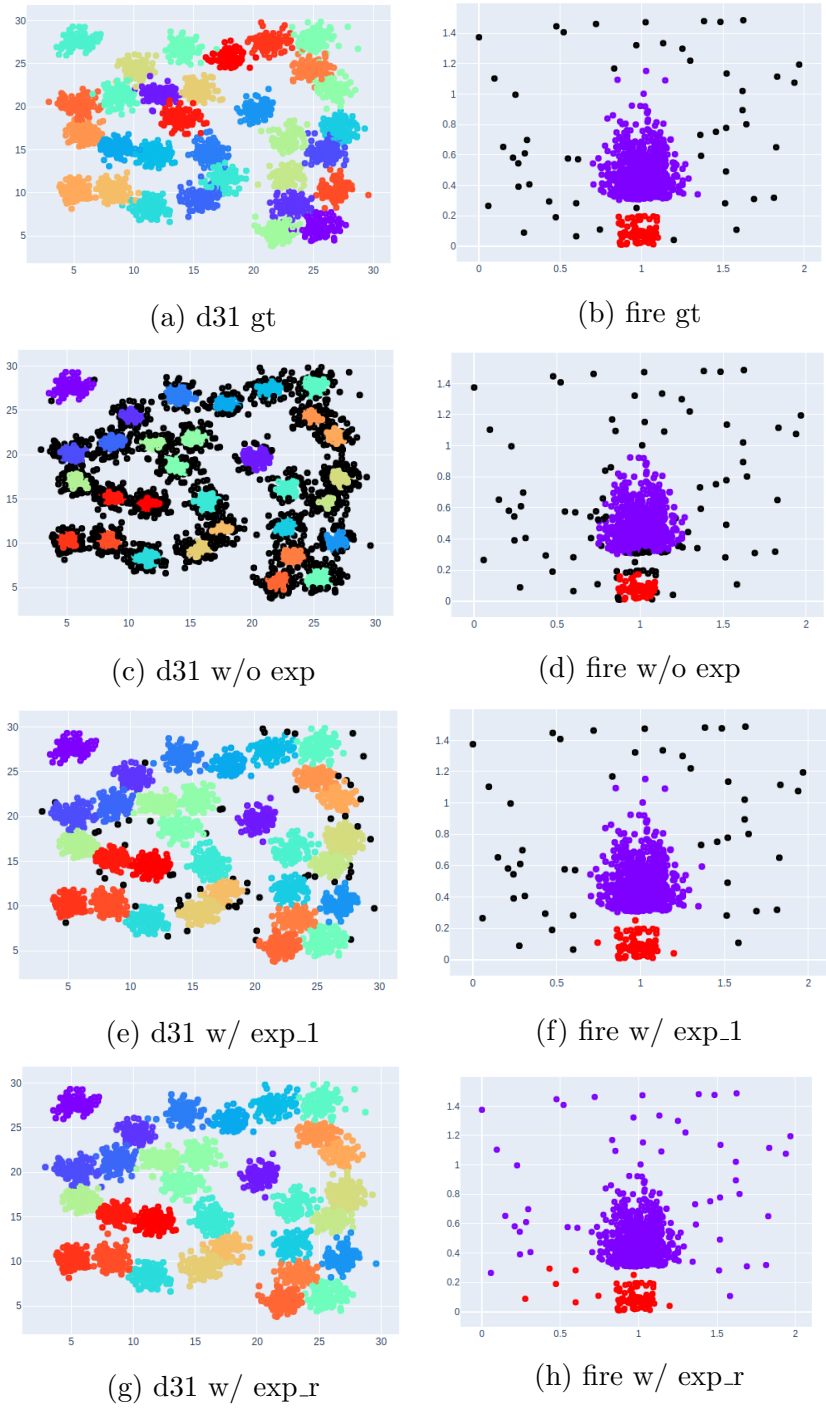


(a) c=3 (b) c=4

Fig. 24.: Maximum aggregate persistence flat clustering solutions (with recursive cluster expansion) for the can3147 dataset at number of clusters c equal to three and four. Clusters are indicated by color and noise by black elements.

Interestingly, the difference in the performance of the two expansion methods is less pronounced in the optimal solutions (see Tables 10 and 11). This result is likely due to ε's effect on cluster birth/death in the tree, which is monotonically increasing. Specifically, increasing ε increases the amount of noise. Thus, the optimal solutions likely correspond to those with the correct number of clusters and minimal value of ε. Overall, given the increased performance on the real datasets (see Tables 10 and 11), coupled with the fact that a complete clustering of the data is generally preferred, we suggest that preference should be given to the recursive expansion procedure.

Next, we more closely examine the banknote and iris dataset for failures in the heuristic on the real-world data. Note that as the dimensionality of these datasets was greater than two, we apply dimensionality reduction (t-SNE [124]) to aid in the visual interpretation of the results. For the banknote dataset, Figure 26 shows the maximal aggregate persistence clustering solutions for $c^* = 2$ (selected due to the sharp decrease in $gmals_c$) and $c = 3$ (whose performance closely aligns with that of



(a) d31 gt (b) fire gt

(c) d31 w/o exp (d) fire w/o exp

(e) d31 w/ exp_1 (f) fire w/ exp_1

(g) d31 w/ exp_r (h) fire w/ exp_r

Fig. 25.: Ground truth (gt) clustering of the d31 and fire datasets, and the maximal aggregate persistence clustering (with and without cluster expansion) for d31 at $c^* = 31$ and fire at $c^* = 2$. Note clusters are indicated by color and noise by black elements.

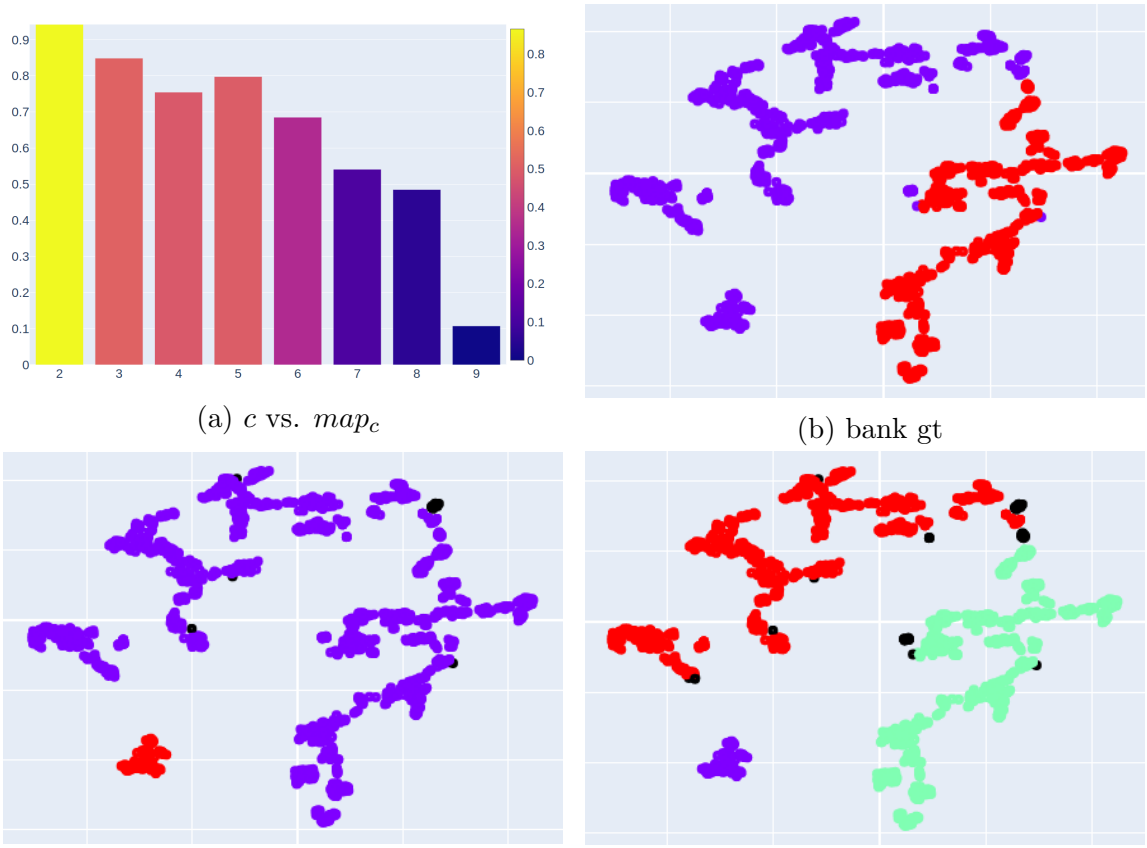
Table 12.: Wilcoxon Signed-Rank Test p-values for ARI Performance in Table 11

Hk-DC w/o exp	*Hk-DC* w/ exp_1	*Hk-DC* w/ exp_r	*HDBSCAN*	
Hk-DC w/o exp		0.9999	0.9997	0.9999
Hk-DC w/ exp_1	**0.000127**		0.3649	0.3316
Hk-DC w/ exp_r	**0.0003407**	0.6584		0.448
HDBSCAN	**8.406e-05**	0.684	0.5692	

the optimal solution reported in Table 10). The two classes of the banknote dataset are correctly split at $c = 3$, where one of the classes is further refined into two clusters. Here the failure of the heuristic is likely because one of the clusters appears to be significantly separated from the remaining data.

Figure 27 shows results for the iris dataset. Here two clusters were selected due to the sharp decrease in $mals_c$ and $gmals_c$ at $c = 2$. One observes that one of the iris classes is correctly split at $c = 2$ and further refined at $c = 3$. Not shown is the $c = 4$ solution, which splits the remaining two classes. Note that a $c = 3$ flat clustering does exist that performs the correct order of splits (as indicated by the increased performance in Table 10). However, this solution does not have maximal aggregate persistence. Here the failure of the heuristic is likely because the data is well separated into two clusters.

Finally, Table 12 shows comparison tests using ARI performance for each pair of clustering algorithms. Results suggest that neither *Hk-DC* nor *HDBSCAN* are significantly better, though *Hk-DC* does outperform *HDBSCAN* in more cases. However, this is still significant as it shows that *Hk-DC* can still discover cluster structure (comparable to *HDBSCAN*) in its smaller input domain. Additionally, one observes that expansion significantly improves the performance of *Hk-DC*. This result is likely because clusters are discoverable within a reduced range of k (e.g., in the range of



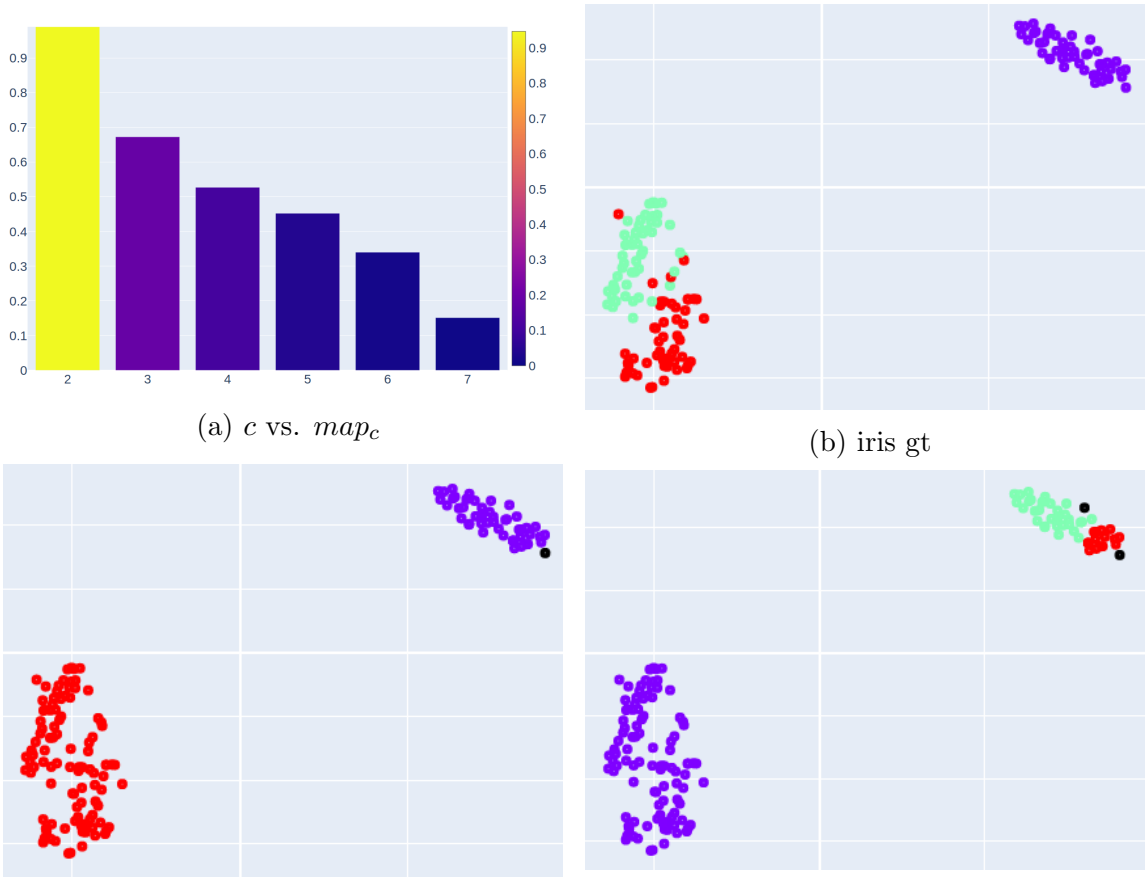
(a) c vs. map_c

(b) bank gt

(c) c=2

(d) c=3

Fig. 26.: ($mals_c$ versus c)-plot of the banknote dataset with number of cluster in the x-axis, $mals_c$ in the y-axis, and colored by $gmals_c$. The ground truth (gt) and maximal aggregate persistence clustering at $c = 2$ and $c = 3$ are shown using a t-SNE projection of the data. Note that recursive expansion was used and that clusters are indicated by color and noise by black elements.



(a) c vs. map_c

(b) iris gt

(c) c=2

(d) c=3

Fig. 27.: ($mals_c$ versus c)-plot of the iris dataset with number of cluster in the x-axis, $mals_c$ in the y-axis, and colored by $gmals_c$. The ground truth (gt) and maximal aggregate persistence clustering at $c = 2$ and $c = 3$ are shown using a t-SNE projection of the data. Note that recursive expansion was used and that clusters are indicated by color and noise by black elements.

$1..\log(n)$). This reduced range reduces the size of the tree (as compared to *HDB-SCAN*), where more elements are likely to be identified as noise at the cluster splits.

CHAPTER 5

CONCLUSIONS

This work investigated the use of the kNN graph with RkNN density in level-set, density-based clustering. Two novel clustering algorithms, *RNN-DBSCAN* and *Hk-DC*, were proposed and their performance analyzed. In *Hk-DC*, rank-based edge weights and k-density were introduced to perform *RNN-DBSCAN* over a range of k resulting in a hierarchical clustering solution. Both algorithms exhibited statistically equivalent performance to approaches using the more popular ε-neighbors approach. This result being significant as kNN-based approaches have a much smaller solution space, where k is easier to select than ε. Additionally, *RNN-DBSCAN* was shown to improve performance over prior non-hierarchical approaches using kNN significantly, *Hk-DC* being unique in its use of kNN in hierarchical level-set, density-based clustering.

For both algorithms, novel approaches for handling noise and heuristics for parameter selection were proposed. For handling noise, in *RNN-DBSCAN*, a hybrid kNN- and ε-neighborhood-based traversal approach (ε being cluster dependent), and for *Hk-DC*, a recursive kNN-based traversal approach (k being cluster dependent). These noise handling techniques lead to significant improvement in performance over prior approaches and the case of ignoring the noise. An important finding of this work is the increased importance of handling noise using kNN compared with ε-neighbors. This result being due to the reduced solution space of kNN, where elements are more likely to be identified as noise at values of k compared with ε. Given this importance, suggested future work includes investigating new stopping criteria for the recursive

approach.

Novel elbow method heuristics for selecting parameters whose clustering solutions produced an appropriate number of clusters were proposed for both algorithms. The problem of selecting clustering resolution (number of clusters) being the most challenging in clustering analysis. The proposed approach is based on the assumption that the correct number of clusters is the largest solution that is persistent (stable) over the range of parameters. Exploratory analysis showed that the heuristics identified significant grouping within the datasets, leading to the optimal solution in most cases. Suggested future work includes developing an internal measure of evaluation based on k (e.g., the ratio between intra-cluster and inter-cluster connectivity) and combining the internal measure with our concept of persistence. Finally, the complexity of both heuristics can be reduced by considering dynamic solutions for the connected components and minimum spanning tree problems [125, 126, 127, 128, 129, 130, 131].

The overall conclusion of this work is that kNN approaches offer several key advantages over ε-neighborhoods approaches while producing comparable results. Specifically, the reduced parameter and solution space of kNN approaches simplify parameter selection by reducing the space of the input and solutions. Thus, the problem of identifying a correct clustering resolution becomes more tractable, where one can explore the set of possible solutions more completely.

Appendix A

ABBREVIATIONS

ARI	Adjusted Rand Index
Hk-DC	Hierarchical k-Density Clustering
kNN	k-nearest neighbors
RkNN	reverse k-nearest neighbors
NMI	Normalized Mutual Information
RNN-DBSCAN	Reverse Nearest Neighbor-DBSCAN
RVA	Richmond Virginia
VCU	Virginia Commonwealth University

REFERENCES

[1] Krzysztof J. Cios et al. *Data Mining: A Knowledge Discovery Approach*. Berlin, Heidelberg: Springer-Verlag, 2007. ISBN: 0387333339.

[2] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. "Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm". In: *Procedia Computer Science* 54 (2015). Eleventh International Conference on Communication Networks, ICCN 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Data Mining and Warehousing, ICDMW 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Image and Signal Processing, ICISP 2015, August 21-23, 2015, Bangalore, India, pp. 764 –771. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.06.090. URL: http://www.sciencedirect.com/science/article/pii/S1877050915014143.

[3] Claudio Carpineto et al. "A Survey of Web Clustering Engines". In: *ACM Comput. Surv.* 41.3 (July 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541884. URL: https://doi.org/10.1145/1541880.1541884.

[4] Daniel Dvorkin, Valerie Fadok, and Krzysztof Cios. "SiMCAL 1 Algorithm for Analysis of Gene Expression Data Related to the Phosphatidylserine Receptor". In: *Artif. Intell. Med.* 35.1–2 (Sept. 2005), 49–60. ISSN: 0933-3657. DOI: 10.1016/j.artmed.2005.01.010. URL: https://doi.org/10.1016/j.artmed.2005.01.010.

[5] Cao Nguyen et al. "ClusFCM: An algorithm for predicting protein function using homologies and protein interactions". In: *Journal of bioinformat-*

ics and computational biology 6 (Mar. 2008), pp. 203–22. DOI: 10.1142/S0219720008003333.

[6] Lani Wu et al. "Large-scale prediction of Saccharomyces cerevisiae gene function using overlapping transcriptional clusters". In: *Nature genetics* 31 (Aug. 2002), pp. 255–65. DOI: 10.1038/ng906.

[7] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. "Clustering Gene Expression Patterns". In: *Journal of computational biology : a journal of computational molecular cell biology* 6 (Aug. 1999), pp. 281–97. DOI: 10.1089/106652799318274.

[8] G. Kerr et al. "Techniques for clustering gene expression data". In: *Computers in Biology and Medicine* 38.3 (2008), pp. 283 –293. ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2007.11.001. URL: http://www.sciencedirect.com/science/article/pii/S0010482507001801.

[9] Der-Chiang Li, Wen-Li Dai, and Wan-Ting Tseng. "A two-stage clustering method to analyze customer characteristics to build discriminative customer management: A case of textile manufacturing business". In: *Expert Systems with Applications* 38.6 (2011), pp. 7186 –7191. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2010.12.041. URL: http://www.sciencedirect.com/science/article/pii/S0957417410014041.

[10] D.T. Pham and A.A. Afify. "- Engineering applications of clustering techniques". In: *Intelligent Production Machines and Systems*. Ed. by D.T. Pham, E.E. Eldukhri, and A.J. Soroka. Oxford: Elsevier Science Ltd, 2006, pp. 326 –331. ISBN: 978-0-08-045157-2. DOI: https://doi.org/10.1016/B978-008045157-2/50060-2. URL: http://www.sciencedirect.com/science/article/pii/B9780080451572500602.

[11] Michael Chau et al. "Uncertain Data Mining: An Example in Clustering Location Data". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Wee-Keong Ng et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 199–204. ISBN: 978-3-540-33207-7.

[12] P. H. A. Sneath. "The Application of Computers to Taxonomy". In: *Microbiology* 17.1 (1957), pp. 201–226. URL: http://mic.microbiologyresearch.org/content/journal/micro/10.1099/00221287-17-1-201.

[13] J. Macqueen. "Some methods for classification and analysis of multivariate observations". In: *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*. 1967, pp. 281–297.

[14] Martin Ester et al. "A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231. URL: http://dl.acm.org/citation.cfm?id=3001460.3001507.

[15] Arthur Dempster, Natalie Laird, and Donald B. Rubin. "Maximum Likelihood From Incomplete Data Via The EM algorithm". In: 39 (Jan. 1977), pp. 1–38.

[16] Rakesh Agrawal et al. "Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications". In: *SIGMOD Rec.* 27.2 (June 1998), 94–105. ISSN: 0163-5808. DOI: 10.1145/276305.276314. URL: https://doi.org/10.1145/276305.276314.

[17] S. Mahran and K. Mahar. "Using grid for accelerating density-based clustering". In: *2008 8th IEEE International Conference on Computer and Information Technology*. 2008, pp. 35–40. DOI: 10.1109/CIT.2008.4594646.

[18] Avory Bryant and Krzysztof Cios. "RNN-DBSCAN: A Density-based Clustering Algorithm using Reverse Nearest Neighbor Density Estimate". In: *IEEE Transactions on Knowledge and Data Engineering* (2017), pp. 1–1. DOI: 10.1109/TKDE.2017.2787640.

[19] Michael E. Houle et al. "Can Shared-Neighbor Distances Defeat the Curse of Dimensionality?" In: *Scientific and Statistical Database Management*. Ed. by Michael Gertz and Bertram Ludäscher. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 482–500. ISBN: 978-3-642-13818-8.

[20] Kevin Beyer et al. "When Is "Nearest Neighbor" Meaningful?" In: *Database Theory — ICDT'99*. Ed. by Catriel Beeri and Peter Buneman. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 217–235. ISBN: 978-3-540-49257-3.

[21] Ian Jolliffe. "Principal Component Analysis". In: *International Encyclopedia of Statistical Science*. Ed. by Miodrag Lovric. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 1094–1096. ISBN: 978-3-642-04898-2. DOI: 10.1007/978-3-642-04898-2_455. URL: https://doi.org/10.1007/978-3-642-04898-2_455.

[22] Leland McInnes, John Healy, and James Melville. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. 2020. arXiv: 1802.03426 [stat.ML].

[23] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. ISSN: 0036-8075. DOI: 10.1126/science.1127647. eprint: https://science.sciencemag.org/content/313/5786/504.pdf. URL: https://science.sciencemag.org/content/313/5786/504.

[24] Sanjay Goil, Harsha Nagesh, and Alok Choudhary. *MAFIA: Efficient and Scalable Subspace Clustering for Very Large Data Sets*. Tech. rep. Nothwestern University, 1999.

[25] C. Bohm et al. "Density connected clustering with local subspace preferences". In: *Fourth IEEE International Conference on Data Mining (ICDM'04)*. 2004, pp. 27–34. DOI: 10.1109/ICDM.2004.10087.

[26] Miloš Radovanovi, Alexandros Nanopoulos, and Mirjana Ivanovi. "Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data". In: *Journal of Machine Learning Research* 11.86 (2010), pp. 2487–2531. URL: http://jmlr.org/papers/v11/radovanovic10a.html.

[27] Volodymyr Melnykov and Ranjan Maitra. "Finite mixture models and model-based clustering". In: *Statist. Surv.* 4 (2010), pp. 80–116. DOI: 10.1214/09-SS053. URL: https://doi.org/10.1214/09-SS053.

[28] Bernard W Silverman. *Density estimation for statistics and data analysis*. Monographs on Statistics and Applied Probability. London: Chapman and Hall, 1986. URL: https://cds.cern.ch/record/1070306.

[29] Alexander Hinneburg and Daniel A. Keim. "An Efficient Approach to Clustering in Large Multimedia Databases with Noise". In: *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*. KDD'98. New York, NY: AAAI Press, 1998, pp. 58–65. URL: http://dl.acm.org/citation.cfm?id=3000292.3000302.

[30] Dorin Comaniciu and Peter Meer. "Mean shift: A robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), pp. 603–619.

[31] Teng Qiu et al. *Nearest Descent, In-Tree, and Clustering*. 2018. arXiv: 1412.5902 [cs.LG].

[32] D. Wishart. "Mode Analysis: a generalization of nearest neighbour which reduces chaining effects". In: *Numerical Taxonomy* (1969), pp. 282–311.

[33] R. F. Ling. "On the theory and construction of k-clusters". In: *The Computer Journal* 15.4 (1972), pp. 326–332. DOI: 10.1093/comjnl/15.4.326. eprint: /oup/backfile/content_public/journal/comjnl/15/4/10.1093/comjnl/15.4.326/2/150326.pdf. URL: http://dx.doi.org/10.1093/comjnl/15.4.326.

[34] John A. Hartigan. *Clustering Algorithms*. 99th. New York, NY, USA: John Wiley & Sons, Inc., 1975. ISBN: 047135645X.

[35] Mihael Ankerst et al. "OPTICS: Ordering Points to Identify the Clustering Structure". In: *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*. SIGMOD '99. Philadelphia, Pennsylvania, USA: ACM, 1999, pp. 49–60. ISBN: 1-58113-084-8. DOI: 10.1145/304182.304187. URL: http://doi.acm.org/10.1145/304182.304187.

[36] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. "Density-Based Clustering Based on Hierarchical Density Estimates". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Jian Pei et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172. ISBN: 978-3-642-37456-2.

[37] Ricardo J. G. B. Campello et al. "Hierarchical Density Estimates for Data Clustering, Visualization, and Outlier Detection". In: *ACM Trans. Knowl. Discov. Data* 10.1 (July 2015), 5:1–5:51. ISSN: 1556-4681. DOI: 10.1145/2733381. URL: http://doi.acm.org/10.1145/2733381.

[38] Y. El-Sonbaty, M. A. Ismail, and M. Farouk. "An efficient density based clustering algorithm for large databases". In: *16th IEEE International Conference on Tools with Artificial Intelligence*. 2004, pp. 673–677. DOI: 10.1109/ICTAI.2004.27.

[39] Alexander Hinneburg and Daniel A. Keim. "A General Approach to Clustering in Large Databases with Noise." In: *Knowl. Inf. Syst.* 5.4 (Mar. 14, 2005), pp. 387–415.

[40] Xin Wang and Howard J. Hamilton. "DBRS: A Density-Based Spatial Clustering Method with Random Sampling". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Kyu-Young Whang et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 563–575. ISBN: 978-3-540-36175-6.

[41] Johannes Schneider and Michail Vlachos. "Fast parameterless density-based clustering via random projections". In: *CIKM*. 2013.

[42] Junhao Gan and Yufei Tao. "DBSCAN Revisited: Mis-Claim, Un-Fixability, and Approximation". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. Melbourne, Victoria, Australia: ACM, 2015, pp. 519–530. ISBN: 978-1-4503-2758-9. DOI: 10.1145/2723372.2737792. URL: http://doi.acm.org/10.1145/2723372.2737792.

[43] Junhao Gan and Yufei Tao. "On the Hardness and Approximation of Euclidean DBSCAN". In: *ACM Trans. Database Syst.* 42.3 (July 2017), 14:1–14:45. ISSN: 0362-5915. DOI: 10.1145/3083897. URL: http://doi.acm.org/10.1145/3083897.

[44] Xiaowei Xu, Jochen Jäger, and Hans-Peter Kriegel. "A Fast Parallel Clustering Algorithm for Large Spatial Databases". In: *Data Min. Knowl. Discov.* 3.3 (Sept. 1999), pp. 263–290. ISSN: 1384-5810.

[45] Y. He et al. "MR-DBSCAN: An Efficient Parallel Density-Based Clustering Algorithm Using MapReduce". In: *2011 IEEE 17th International Conference on Parallel and Distributed Systems*. 2011, pp. 473–480. DOI: 10.1109/ICPADS.2011.83.

[46] Benjamin Welton, Evan Samanas, and Barton P. Miller. "Mr. Scan: Extreme Scale Density-based Clustering Using a Tree-based Network of GPGPU Nodes". In: *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. SC '13. Denver, Colorado: ACM, 2013, 84:1–84:11. ISBN: 978-1-4503-2378-9. DOI: 10.1145/2503210.2503262. URL: http://doi.acm.org/10.1145/2503210.2503262.

[47] Xiaoming Chen et al. "APSCAN: A parameter free algorithm for clustering." In: *Pattern Recognition Letters* 32.7 (2011), pp. 973–986.

[48] Xiaowei Xu et al. "A distribution-based clustering algorithm for mining in large spatial databases". In: *Proceedings 14th International Conference on Data Engineering*. 1998, pp. 324–331. DOI: 10.1109/ICDE.1998.655795.

[49] Alessandro Lulli et al. "NG-DBSCAN: Scalable Density-based Clustering for Arbitrary Data". In: *Proc. VLDB Endow.* 10.3 (Nov. 2016), pp. 157–168. ISSN: 2150-8097. DOI: 10.14778/3021924.3021932. URL: https://doi.org/10.14778/3021924.3021932.

[50] Feng Cao et al. "Density-based clustering over an evolving data stream with noise". In: *In 2006 SIAM Conference on Data Mining*. 2006, pp. 328–339.

[51] Xiaowei Xu et al. “SCAN: A Structural Clustering Algorithm for Networks”. In: *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD ’07. San Jose, California, USA: ACM, 2007, pp. 824–833. ISBN: 978-1-59593-609-7. DOI: 10.1145/1281192.1281280. URL: http://doi.acm.org/10.1145/1281192.1281280.

[52] Mihael Ankerst et al. “OPTICS: Ordering Points to Identify the Clustering Structure”. In: *SIGMOD Rec.* 28.2 (June 1999), pp. 49–60. ISSN: 0163-5808. DOI: 10.1145/304181.304187. URL: http://doi.acm.org/10.1145/304181.304187.

[53] Elke Achtert, Christian Böhm, and Peer Kröger. “DeLi-Clu: Boosting Robustness, Completeness, Usability, and Efficiency of Hierarchical Clustering by a Closest Pair Ranking”. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Wee-Keong Ng et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 119–128. ISBN: 978-3-540-33207-7.

[54] G. Gupta, A. Liu, and J. Ghosh. “Automated Hierarchical Density Shaving: A Robust Automated Clustering and Visualization Framework for Large Biological Data Sets”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 7.2 (2010), pp. 223–237. ISSN: 1545-5963. DOI: 10.1109/TCBB.2008.32.

[55] Ergun Biçici and Deniz Yuret. “Locally Scaled Density Based Clustering”. In: *Adaptive and Natural Computing Algorithms*. Ed. by Bartlomiej Beliczynski et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 739–748. ISBN: 978-3-540-71618-1.

[56] Anant Ram et al. “Article:A Density Based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases”. In: *International Journal of*

Computer Applications 3.6 (2010). Published By Foundation of Computer Science, pp. 1–4.

[57] L. Duan et al. "A Local Density Based Spatial Clustering Algorithm with Noise". In: *2006 IEEE International Conference on Systems, Man and Cybernetics*. Vol. 5. 2006, pp. 4061–4066. DOI: 10.1109/ICSMC.2006.384769.

[58] B. Borah and D. K. Bhattacharyya. "DDSC: A Density Differentiated Spatial Clustering Technique". In: *Journal of Computers* 3.2 (2008).

[59] Levent Ertoz, Michael Steinbach, and Vipin Kumar. "A New Shared Nearest Neighbor Clustering Algorithm and its Applications". In: *Workshop on Clustering High Dimensional Data and its Applications at 2nd SIAM International Conference on Data Mining*. 2002.

[60] Levent Ertoz, Michael Steinbach, and Vipin Kumar. "Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data". In: *Proceedings of the Third SIAM International Conference on Data Mining (SDM 2003)*. Ed. by Daniel Barbara and Chandrika Kamath. Vol. 112. Proceedings in Applied Mathematics. Society for Industrial and Applied Mathematics, 2003. URL: http://www.siam.org/meetings/sdm03/proceedings/sdm03_05.pdf.

[61] Y. Tao and D. Pi. "Unifying Density-Based Clustering and Outlier Detection". In: *2009 Second International Workshop on Knowledge Discovery and Data Mining*. 2009, pp. 644–647. DOI: 10.1109/WKDD.2009.127.

[62] S. Vadapalli, S. R. Valluri, and K. Karlapalem. "A Simple Yet Effective Data Clustering Algorithm". In: *Sixth International Conference on Data Mining (ICDM'06)*. 2006, pp. 1108–1112.

[63] Carmelo Cassisi et al. “Enhancing Density-based Clustering: Parameter Reduction and Outlier Detection”. In: *Inf. Syst.* 38.3 (May 2013), pp. 317–330. ISSN: 0306-4379.

[64] Yinghua Lv et al. “An Efficient and Scalable Density-based Clustering Algorithm for Datasets with Complex Structures”. In: *Neurocomput.* 171.C (Jan. 2016), pp. 9–22. ISSN: 0925-2312.

[65] Lian Duan et al. “A local-density based spatial clustering algorithm with noise.” In: *Inf. Syst.* 32.7 (Sept. 14, 2007), pp. 978–986.

[66] S. Vadapalli, S. R. Valluri, and K. Karlapalem. “A Simple Yet Effective Data Clustering Algorithm”. In: *Sixth International Conference on Data Mining (ICDM'06)*. 2006, pp. 1108–1112. DOI: 10.1109/ICDM.2006.9.

[67] P. Liu, D. Zhou, and N. Wu. “VDBSCAN: Varied Density Based Spatial Clustering of Applications with Noise”. In: *2007 International Conference on Service Systems and Service Management*. 2007, pp. 1–4. DOI: 10.1109/ICSSSM.2007.4280175.

[68] Markus M. Breunig et al. “LOF: Identifying Density-based Local Outliers”. In: *SIGMOD Rec.* 29.2 (May 2000), pp. 93–104. ISSN: 0163-5808. DOI: 10.1145/335191.335388. URL: http://doi.acm.org/10.1145/335191.335388.

[69] Markus M. Breunig et al. “LOF: Identifying Density-based Local Outliers”. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: ACM, 2000, pp. 93–104. ISBN: 1-58113-217-4. DOI: 10.1145/342009.335388. URL: http://doi.acm.org/10.1145/342009.335388.

[70] A. Cavalcante Araujo Neto et al. “Efficient Computation and Visualization of Multiple Density-Based Clustering Hierarchies”. In: *IEEE Transactions on Knowledge and Data Engineering* (2019), pp. 1–1. DOI: 10.1109/TKDE.2019.2962412.

[71] Edwin M. Knorr and Raymond T. Ng. “A Unified Notion of Outliers: Properties and Computation”. In: *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*. KDD’97. Newport Beach, CA: AAAI Press, 1997, pp. 219–222. URL: http://dl.acm.org/citation.cfm?id=3001392.3001438.

[72] Edwin M. Knorr and Raymond T. Ng. “Algorithms for Mining Distance-Based Outliers in Large Datasets”. In: *Proceedings of the 24rd International Conference on Very Large Data Bases*. VLDB ’98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 392–403. ISBN: 1-55860-566-5. URL: http://dl.acm.org/citation.cfm?id=645924.671334.

[73] Fabrizio Angiulli and Clara Pizzuti. “Fast Outlier Detection in High Dimensional Spaces”. In: *Principles of Data Mining and Knowledge Discovery*. Ed. by Tapio Elomaa, Heikki Mannila, and Hannu Toivonen. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 15–27. ISBN: 978-3-540-45681-0.

[74] F. Angiulli and C. Pizzuti. “Outlier mining in large high-dimensional data sets”. In: *IEEE Transactions on Knowledge and Data Engineering* 17.2 (2005), pp. 203–215. ISSN: 1041-4347. DOI: 10.1109/TKDE.2005.31.

[75] Tianming Hu and Sam Y. Sung. “Detecting Pattern-based Outliers”. In: *Pattern Recogn. Lett.* 24.16 (Dec. 2003), pp. 3059–3068. ISSN: 0167-8655. DOI: 10.1016/S0167-8655(03)00165-X. URL: http://dx.doi.org/10.1016/S0167-8655(03)00165-X.

[76] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3 (July 2009), 15:1–15:58. ISSN: 0360-0300. DOI: 10.1145/1541880.1541882. URL: http://doi.acm.org/10.1145/1541880.1541882.

[77] Erich Schubert, Arthur Zimek, and Hans-Peter Kriegel. "Local outlier detection reconsidered: a generalized view on locality with applications to spatial, video, and network outlier detection". In: *Data Mining and Knowledge Discovery* 28.1 (2014), pp. 190–237. ISSN: 1573-756X. DOI: 10.1007/s10618-012-0300-z. URL: https://doi.org/10.1007/s10618-012-0300-z.

[78] A. L. M. Chiu and Ada Wai chee Fu. "Enhancements on local outlier detection". In: *Seventh International Database Engineering and Applications Symposium, 2003. Proceedings.* 2003, pp. 298–307. DOI: 10.1109/IDEAS.2003.1214939.

[79] S. Papadimitriou et al. "LOCI: fast outlier detection using the local correlation integral". In: *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*. 2003, pp. 315–326. DOI: 10.1109/ICDE.2003.1260802.

[80] Longin Jan Latecki, Aleksandar Lazarevic, and Dragoljub Pokrajac. "Outlier Detection with Kernel Density Functions". In: *Machine Learning and Data Mining in Pattern Recognition*. Ed. by Petra Perner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 61–75. ISBN: 978-3-540-73499-4.

[81] Ke Zhang, Marcus Hutter, and Huidong Jin. "A New Local Distance-Based Outlier Detection Approach for Scattered Real-World Data". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Thanaruk Theeramunkong

et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 813–822. ISBN: 978-3-642-01307-2.

[82] Hans-Peter Kriegel et al. "LoOP: Local Outlier Probabilities". In: *Proceedings of the 18th ACM Conference on Information and Knowledge Management*. CIKM '09. Hong Kong, China: ACM, 2009, pp. 1649–1652. ISBN: 978-1-60558-512-3. DOI: 10.1145/1645953.1646195. URL: http://doi.acm.org/10.1145/1645953.1646195.

[83] V. Hautamaki, I. Karkkainen, and P. Franti. "Outlier detection using k-nearest neighbour graph". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 3. 2004, 430–433 Vol.3. DOI: 10.1109/ICPR.2004.1334558.

[84] M. Radovanović, A. Nanopoulos, and M. Ivanović. "Reverse Nearest Neighbors in Unsupervised Distance-Based Outlier Detection". In: *IEEE Transactions on Knowledge and Data Engineering* 27.5 (2015), pp. 1369–1382. ISSN: 1041-4347. DOI: 10.1109/TKDE.2014.2365790.

[85] Wen Jin et al. "Ranking Outliers Using Symmetric Neighborhood Relationship". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Wee-Keong Ng et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 577–593. ISBN: 978-3-540-33207-7.

[86] Huaming Huang, Kishan Mehrotra, and Chilukuri K. Mohan. *Outlier Detection Using Modified-ranks and Other Variants*. 72. Syracuse University, College of Engineering and Computer Science, 2011.

[87] Huaming Huang, Kishan Mehrotra, and Chilukuri K. Mohan. "Rank-based outlier detection". In: *Journal of Statistical Computation and Simulation* 83.3 (2013), pp. 518–531. DOI: 10.1080/00949655.2011.621124. eprint: https:

//doi.org/10.1080/00949655.2011.621124. URL: https://doi.org/10.1080/00949655.2011.621124.

[88] Jian Tang et al. "Enhancing Effectiveness of Outlier Detections for Low Density Patterns". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Ming-Syan Chen, Philip S. Yu, and Bing Liu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 535–548. ISBN: 978-3-540-47887-4.

[89] Hans-Peter Kriegel, Matthias S hubert, and Arthur Zimek. "Angle-based Outlier Detection in High-dimensional Data". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '08. Las Vegas, Nevada, USA: ACM, 2008, pp. 444–452. ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401946. URL: http://doi.acm.org/10.1145/1401890.1401946.

[90] Antonin Guttman. "R-trees: A Dynamic Index Structure for Spatial Searching". In: *SIGMOD Rec.* 14.2 (June 1984), pp. 47–57. ISSN: 0163-5808. DOI: 10.1145/971697.602266. URL: http://doi.acm.org/10.1145/971697.602266.

[91] Norbert Beckmann et al. "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles". In: *SIGMOD Rec.* 19.2 (May 1990), pp. 322–331. ISSN: 0163-5808. DOI: 10.1145/93605.98741. URL: http://doi.acm.org/10.1145/93605.98741.

[92] S. Omohundro. "Five Balltree Construction Algorithms". In: 2009.

[93] Jon Louis Bentley. "Multidimensional Binary Search Trees Used for Associative Searching". In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007. URL: http://doi.acm.org/10.1145/361002.361007.

[94] Alina Beygelzimer, Sham Kakade, and John Langford. "Cover Trees for Nearest Neighbor". In: *Proceedings of the 23rd International Conference on Machine Learning*. ICML '06. Pittsburgh, Pennsylvania, USA: ACM, 2006, pp. 97–104. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143857. URL: http://doi.acm.org/10.1145/1143844.1143857.

[95] R. Weber, H. Schek, and S. Blott. "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces". In: *VLDB*. 1998.

[96] Erich Schubert et al. "DBSCAN Revisited, Revisited: Why and How You Should (Still) Use DBSCAN". In: *ACM Trans. Database Syst.* 42.3 (July 2017), 19:1–19:21. ISSN: 0362-5915. DOI: 10.1145/3068335. URL: http://doi.acm.org/10.1145/3068335.

[97] Junhao Gan and Yufei Tao. "DBSCAN Revisited: Mis-Claim, Un-Fixability, and Approximation". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. SIGMOD '15. Melbourne, Victoria, Australia: ACM, 2015, pp. 519–530. ISBN: 978-1-4503-2758-9. DOI: 10.1145/2723372.2737792. URL: http://doi.acm.org/10.1145/2723372.2737792.

[98] Wei Dong, Charikar Moses, and Kai Li. "Efficient K-nearest Neighbor Graph Construction for Generic Similarity Measures". In: *Proceedings of the 20th International Conference on World Wide Web*. WWW '11. Hyderabad, India: ACM, 2011, pp. 577–586. ISBN: 978-1-4503-0632-4. DOI: 10.1145/1963405.1963487. URL: http://doi.acm.org/10.1145/1963405.1963487.

[99] Kiana Hajebi et al. "Fast Approximate Nearest-neighbor Search with K-nearest Neighbor Graph". In: *Proceedings of the Twenty-Second International*

Joint Conference on Artificial Intelligence - Volume Volume Two. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, 2011, pp. 1312–1317. ISBN: 978-1-57735-514-4. DOI: 10.5591/978-1-57735-516-8/IJCAI11-222. URL: http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-222.

[100] Yury Malkov et al. "Approximate nearest neighbor algorithm based on navigable small world graphs". In: *Information Systems* 45 (2014), pp. 61 –68. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2013.10.006. URL: http://www.sciencedirect.com/science/article/pii/S0306437913001300.

[101] Y. A. Malkov and D. A. Yashunin. "Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.4 (2020), pp. 824–836. DOI: 10.1109/TPAMI.2018.2889473.

[102] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. "ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms". In: *Information Systems* 87 (2020), p. 101374. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2019.02.006. URL: https://www.sciencedirect.com/science/article/pii/S0306437918303685.

[103] Markus Maier, Matthias Hein, and Ulrike von Luxburg. "Cluster Identification in Nearest-Neighbor Graphs". In: *Algorithmic Learning Theory*. Ed. by Marcus Hutter, Rocco A. Servedio, and Eiji Takimoto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 196–210. ISBN: 978-3-540-75225-7.

[104] M.R. Brito et al. "Connectivity of the mutual k-nearest-neighbor graph in clustering and outlier detection". In: *Statistics & Probability Letters* 35.1 (1997), pp. 33–42. ISSN: 0167-7152. DOI: https://doi.org/10.1016/S0167-

7152(96)00213-1. URL: https://www.sciencedirect.com/science/article/pii/S0167715296002131.

[105] Paul Balister et al. "Connectivity of random k-nearest-neighbour graphs". In: *Advances in Applied Probability* 37.1 (2005), 1–24. DOI: 10.1239/aap/1113402397.

[106] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. London: Chapman & Hall, 1986.

[107] Jadson Castro Gertrudes et al. "A Unified Framework of Density-Based Clustering for Semi-Supervised Classification". In: *Proceedings of the 30th International Conference on Scientific and Statistical Database Management*. SSDBM '18. Bozen-Bolzano, Italy: Association for Computing Machinery, 2018. ISBN: 9781450365055. DOI: 10.1145/3221269.3223037. URL: https://doi.org/10.1145/3221269.3223037.

[108] C. A. R. Hoare. "Algorithm 64: Quicksort". In: *Commun. ACM* 4.7 (July 1961), p. 321. ISSN: 0001-0782. DOI: 10.1145/366622.366644. URL: https://doi.org/10.1145/366622.366644.

[109] Donald E. Knuth. *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching*. USA: Addison Wesley Longman Publishing Co., Inc., 1998. ISBN: 0201896850.

[110] R. C. Prim. "Shortest Connection Networks And Some Generalizations". In: *Bell System Technical Journal* 36.6 (Nov. 1957), pp. 1389–1401. DOI: 10.1002/j.1538-7305.1957.tb01515.x.

[111] Joseph B. Kruskal. "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem". In: *Proceedings of the American Mathematical*

Society 7.1 (1956), pp. 48–50. ISSN: 00029939, 10886826. URL: http://www.jstor.org/stable/2033241.

[112] University of Eastern Finland. *Clustering Datasets: Shape sets*. URL: https://cs.joensuu.fi/sipu/datasets/ (visited on 12/08/2016).

[113] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[114] Erich Schubert and Arthur Zimek. "ELKI: A large open-source library for data analysis - ELKI Release 0.7.5 "Heidelberg"". In: *CoRR* abs/1902.03616 (2019). arXiv: 1902.03616. URL: http://arxiv.org/abs/1902.03616.

[115] G. Karypis, Eui-Hong Han, and V. Kumar. "Chameleon: hierarchical clustering using dynamic modeling". In: *Computer* 32.8 (1999), pp. 68–75. DOI: 10.1109/2.781637.

[116] M. Lichman. *UCI Machine Learning Repository*. 2013. URL: http://archive.ics.uci.edu/ml.

[117] Junhao Gan and Yufei Tao. *ApproxDBSCAN Datasets*. 1999. URL: https://sites.google.com/view/approxdbscan/datasets (visited on 07/01/2017).

[118] M. Varma and A. Zisserman. "Texture classification: are filter banks necessary?" In: *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.* Vol. 2. 2003, II–691–8 vol.2. DOI: 10.1109/CVPR.2003.1211534.

[119] R. J. Lyon et al. "Fifty Years of Pulsar Candidate Selection: From simple filters to a new principled real-time classification approach". In: *Monthly Notices of the Royal Astronomical Society* 459 (1), pp. 1104–1123.

[120] L. Hubert and P. Arabie. “Comparing partitions”. In: *Journal of classification* 2.1 (1985), pp. 193–218. URL: http://scholar.google.de/scholar.bib?q=info:IkrWWF2JxwoJ:scholar.google.com/\&output=citation\&hl=de\&ct=citation\&cd=0.

[121] Alexander Strehl and Joydeep Ghosh. “Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions”. In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 583–617. ISSN: 1532-4435. DOI: 10.1162/153244303321897735. URL: http://dx.doi.org/10.1162/153244303321897735.

[122] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. New York, NY, USA: Wiley-Interscience, 1991. ISBN: 0-471-06259-6.

[123] Janez Demšar. “Statistical Comparisons of Classifiers over Multiple Data Sets”. In: *J. Mach. Learn. Res.* 7 (Dec. 2006), 1–30. ISSN: 1532-4435.

[124] Laurens van der Maaten and Geoffrey Hinton. “Visualizing Data using t-SNE”. In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[125] D. Ediger et al. “Tracking Structure of Streaming Social Networks”. In: *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. 2011, pp. 1691–1699. DOI: 10.1109/IPDPS.2011.326.

[126] A. Eldawy, R. Khandekar, and K. L. Wu. “Clustering Streaming Graphs”. In: *2012 IEEE 32nd International Conference on Distributed Computing Systems*. 2012, pp. 466–475. DOI: 10.1109/ICDCS.2012.20.

[127] Mindi Yuan et al. “Efficient Processing of Streaming Graphs for Evolution-aware Clustering”. In: *Proceedings of the 22Nd ACM International Conference*

on Information & Knowledge Management. CIKM '13. San Francisco, California, USA: ACM, 2013, pp. 319–328. ISBN: 978-1-4503-2263-8. DOI: 10.1145/2505515.2505750. URL: http://doi.acm.org/10.1145/2505515.2505750.

[128] Yossi Shiloach and Shimon Even. "An On-Line Edge-Deletion Problem". In: *J. ACM* 28.1 (Jan. 1981), 1–4. ISSN: 0004-5411. DOI: 10.1145/322234.322235. URL: https://doi.org/10.1145/322234.322235.

[129] P. M. Spira and A. Pan. "On Finding and Updating Spanning Trees and Shortest Paths". In: *SIAM Journal on Computing* 4.3 (1975), pp. 375–380. DOI: 10.1137/0204032. eprint: https://doi.org/10.1137/0204032. URL: https://doi.org/10.1137/0204032.

[130] Francis Chin and David Houck. "Algorithms for updating minimal spanning trees". In: *Journal of Computer and System Sciences* 16.3 (1978), pp. 333–344. ISSN: 0022-0000. DOI: https://doi.org/10.1016/0022-0000(78)90022-3. URL: https://www.sciencedirect.com/science/article/pii/0022000078900223.

[131] Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. "Poly-Logarithmic Deterministic Fully-Dynamic Algorithms for Connectivity, Minimum Spanning Tree, 2-Edge, and Biconnectivity". In: *J. ACM* 48.4 (July 2001), 723–760. ISSN: 0004-5411. DOI: 10.1145/502090.502095. URL: https://doi.org/10.1145/502090.502095.

VITA

Avory C. Bryant

EDUCATION

Masters of Science in Computer Science (Dec 2008), Virginia Commonwealth University, Richmond, Virginia.

Bachelors of Science in Computer Science (May 2003), Virginia Commonwealth University, Richmond, Virginia.

EMPLOYMENT

Data Scientist, Naval Surface Warfare Center Dahlgren Division, July 2001 - present. Responsible for the development of decision support systems leveraging supervised and unsupervised machine learning techniques. Primarily in the domains of text analysis and computer vision.

PUBLICATIONS

A. C. Bryant and K. J. Cios (2021), Hk-DC: Hierarchical k-Density Clustering. Manuscript submitted for publication.

A. C. Bryant and K. J. Cios (2018), RNN-DBSCAN: A Density-based Clustering Algorithm using Reverse Nearest Neighbor Density Estimates, in IEEE Transactions on Knowledge and Data Engineering, vol. PP, no. 99, pp. 1-1. doi: 10.1109/TKDE.2017.2787640.

A. C. Bryant and K. J. Cios (2017), SOTXTSTREAM: Density-based self-organizing clustering of text streams, PlosOne. doi: 10.1371/journal.pone.0180543.

C. E. Priebe, J. L. Solka, D. J. Marchette, and A. C. Bryant (2012), Quantitative Horizon Scanning for Mitigating Technological Surprise: Detecting the Potential for Collaboration at the Interface, Stat. Anal. Data Min. 5, 3, 178-186. DOI=http://dx.doi.org/10.1002/sam.11143.

J. L. Solka, A. C. Bryant, and Edward J. Wegman (2005), Text Data Mining with Minimal Spanning Trees, in Handbook of Statistics 24 on Data Mining and Visualization, C. R. Rao, Edward J. Wegman, and J. L. Solka, Eds, Elsevier North Holland.

J. L. Solka, A. C. Bryant, and E. J. Wegman (2004), Identifying cross corpus document association via minimal spanning tree, Computing Science and Statistics, 36.

PRESENTATIONS AT PROFESSIONAL MEETINGS

A. C. Bryant (2018), Fast k Nearest Neighbor Graph Construction Experiments on a Large Scientific Publication Corpus, Symposium on Data Science & Statistics.

A. C. Bryant (2015), Tracking Evolution in Text Data Streams via Online Density-Based Clustering, Joint Statistical Meetings.

J. L. Solka and A. C. Bryant (2010), Exploratory Data Analysis on Document Collections, Joint Statistical Meetings.

A. C. Bryant (2010), Quantitative Horizon Scanning for Mitigating Technological Surprise, Interface.

J. L. Solka and A. C. Bryant (2010), Multi-feature Clustering and Visualization of Large Document Collections, Interface.

A. C. Bryant (2008), Semantic Analysis of the Term-Document Matrix, U.S. Army Conference on Applied Statistics.

A. C. Bryant (2008), Cross Corpus Discovery via Nearest Neighbor Change-point Analysis, Interface.

J. L. Solka and A. C. Bryant (2007), Exploratory Data Analysis of Large Document Collections, Quantitative Methods in Defense and National Security.

J. L. Solka, A. C. Bryant, and E. J. Wegman (2004), Identifying cross corpus document association via minimal spanning tree, Interface.

J. L. Solka, A. C. Bryant, and E. J. Wegman (2004), Recursive Bipartite Spectral Clustering for Document Categorization, U.S. Army Conference on Applied Statistics.

ACADEMIC AWARDS

Outstanding Paper Award (2018), Computer Science Department, Virginia Commonwealth University.