TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Master Thesis

Graph-Based Competitive Clustering: A Clustering Algorithm for Hyperspectral Images in Process Analytical Technologies

carried out for the purpose of obtaining the degree of

Diplom-Ingenieur

submitted at TU Wien,
Faculty of Mechanical and Industrial Engineering

by

Benedikt Steindl

Mat.Nr.: 01025978
Alois-Prager-Straße 18
A-2283 Obersiebenbrunn

under the supervision of

Ao.Univ.Prof. Mag. Dr. Johann Lohninger
Institute of Chemical Technologies and Analytics

Vienna, March, 2018

Benedikt Steindl

I confirm, that going to press of this thesis needs the confirmation of the examination commitee.

Affidavit

I declare in lieu of oath, that I wrote this thesis and performed the associated research myself, using only literature cited in this volume. If text passages from sources are used literally, they are marked as such.
I confirm that this work is original and has not been submitted elsewhere for any examination, nor is it currently under consideration for a thesis elsewhere.

Vienna, March, 2018

Benedikt Steindl

Abstract

Recent developments in process analytical technologies have introduced hyperspectral imaging as a novel tool for quality assessment and process control. This technology allows to capture spectral information in conjunction with spatial features and thus enables us to analyze products in their heterogeneous state. As a consequence hyperspectral data is complex to process and necessitates the use of modern machine learning algorithms. Clustering is one such technique that can be applied to detect interesting spatial features which otherwise could easily be overlooked because of the sheer amount of information. In this thesis a novel clustering algorithm is presented which is designed to cope with some characteristics of hyperspectral images. As clustering is a widely applied technique in data-oriented research fields the proposed algorithm is discussed in the light of other established methods on the algorithmic level as well as by comparing experimental results of artificial datasets and hyperspectral images.

Kurzfassung

Eine neue Entwicklung im Bereich der Prozessanalytik ist das Anwenden von Hyperspectral Imaging als Methode der Qualitätssicherung und Prozesskontrolle. Diese Technologie erlaubt die Erfassung von ortsaufgelösten spektroskopischen Daten, wodurch Produkte in ihrer heterogenen Form analysiert werden können. Wegen ihrer komplexen Struktur sind für die Analyse von hyperspektralen Daten moderne Machine Learning Algorithmen notwendig. Eine dieser Methoden ist die Clusteranalyse, welche die Detektion interessanter örtlicher Merkmale ermöglicht, die sonst aufgrund der großen Menge an Information leicht übersehen werden können. Inhalt dieser Arbeit ist die Präsentation eines neuartigen Cluster-Algorithmus, welcher auf gewisse Eigenschaften von Hyperspectral Images optimiert ist. Da Clusteranalyse eine weit verbreitete Technik in datenlastigen Forschungsfeldern ist, wird der Algorithmus in Bezug auf andere etablierte Methoden auf dem algorithmischen Level sowie durch Vergleich von experimentellen Ergebnissen künstlicher Datensätze und Hyperspectral Images untersucht.

Acknowledgements

First I would like to thank Hans Lohninger who supervised this thesis and guided me through the long process of its making. Among his greatest achievements I count the creation of a social environment where one feels valued and understood. This and his mentoring in all matters of chemistry, programming and chemometrics provided me with the fundamentals and inspiration needed for this work.

I would also like to thank the attendees of the weekly 'Jour Fixe' Wolfgang Ganglberger and Andreas Cremer for their critical questions which enabled me to find and fix some design flaws of the proposed method.

Furthermore my thanks go to Elisabeth Renner for her tutoring in the Einstein summation convention which helped me to pass some of the more difficult exams of the masters program.

Ladies and Gentlemen, it has been a pleasure.

This thesis and my studies would not have been possible without the long lasting support of my parents Andrea and Manfred. I thank them deeply for inspiring me to study the natural sciences and engineering.

Last but not least there remains one individual I would like to name whose contribution to this thesis should not be underestimated. Its my dog Graham who has been a steadfast companion and the most faithful of friends during the long hours of programming and studying.

For Tina

Contents

List of Figures

List of Tables

1 Introduction

Modern-day chemical engineering has evolved into a discipline where computational statistics, machine learning and data mining gain ever more importance. This trend is partially due to the demand for better sensor equipment to automate and control ever more complex chemical or pharmaceutical processes. Bakeev (2010) summarizes various applications of spectroscopic techniques in the chemical industry starting with mass spectroscopy in the 1970s, followed by infrared spectroscopy in the 1980s and raman spectroscopy in the 2000s. The data that we acquire from these technologies are usually high-dimensional and thus tend to be very complex to analyze. Depending on the resolution of the applied sensors the data might contain up to 1000 variables or higher.

The complexity of the data might increase even further if the analyte is spatially distributed over a solid matrix of multiple chemical components (e.g. a solid dosage form with pharmaceutical ingredients, fillers, binders, disintegrants, lubricants and other materials). A mere bulk analysis of the dissolved product thus only yields a mixed spectrum of the components and overall concentrations within the product, whereas their spatial distribution and particle size is completely lost.

Recent developments have introduced *hyperspectral imaging* (HSI) as a novel tool in process analytical technologies to overcome these issues. HSIs are acquired by using a suitable spectroscopic technique to spatially resolve a sample. Each measurement thus has certain x- and y-coordinates which are used to form an image. An illustration of this concept is given in figure 1. A key difference in this form of data is that the compositional heterogeneity of the sample is preserved within the spatial relations of the pixels. This makes the analysis of such datasets distinct from the ones that are obtained through bulk analysis because we no longer deal with a single mixed spectrum of all components, but a set of spectra that represent individual and sometimes quite pure components.

In order to gain the most information from HSIs it is thus necessary to perform some sort of image segmentation. This means that spectral information which is scattered across a multitude of layers is combined and thereby reduced to a single map of spatial features. Each numerical value of that map then stands for a certain label which corresponds to a specific chemical ingredient. The spatial boundary information can then be used to measure the size distribution and form of the particles, or chemical information such as purity.

A well known chemometric technique that can be used for image segmentation is *classification*, which is commonly applied in many other process engineering applications (e.g. sorting-machines). The goal of classification is to assign an observation to one of many classes. In our case this could be to select a pixel of the HSI and assign it to a certain class of chemical compound. Whether an observation belongs to a class is determined by a statistical model which is derived from a training set. This training set is labeled according to a *ground truth*, which means that the class of each member of this set is

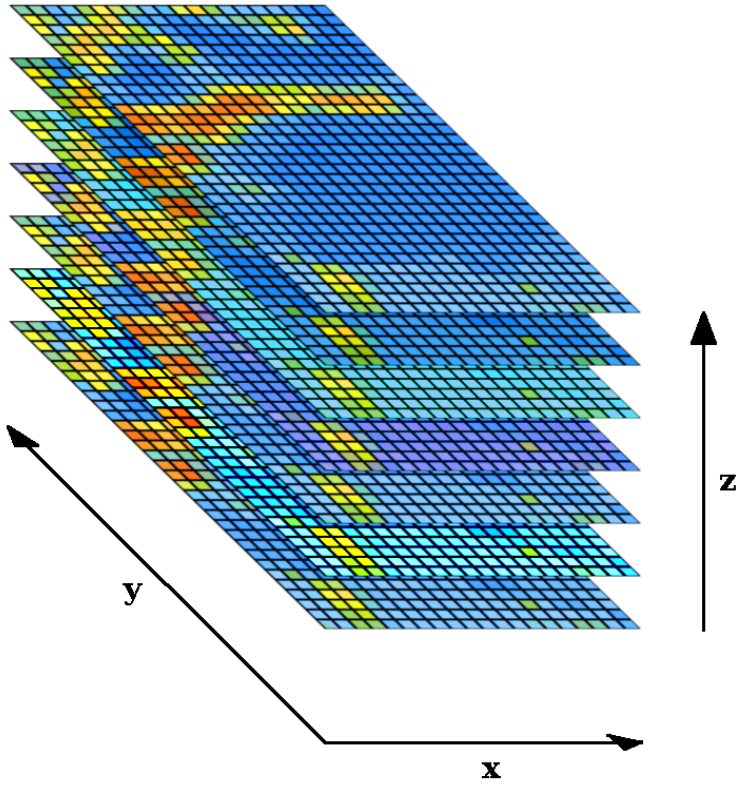


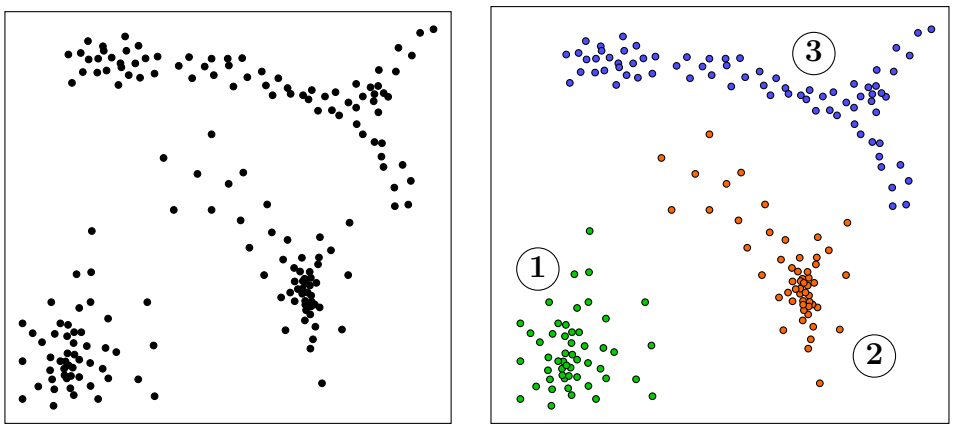
Figure 1: Conceptual drawing of a hyperspectral image. Each pixel contains the spectral information which was acquired at a certain x- and y-position on the sample. The z-coordinate reflects the frequency λ or $\frac{m}{z}$-value of the corresponding layer.

already known.

In a spectroscopic application the training set could be based on a database of reference spectra, though this approach can come with many problems and pitfalls:

- The product of a chemical process may contain various byproducts or contaminations which also have to be part of the training set.
- If the compounds do not form clear grain boundaries or particles, mixture effects have to be expected. This means that spectra of different mixing ratios have to be labeled as well to give a clear decision boundary for the classification.
- Reference spectra are usually measured under optimal laboratory conditions with respect to the sensor equipment and the preparation of the sample. In practical applications there can be various causes why the quality of the measured spectra is very low (e.g. signal saturation of the sensor, baseline distortion due to Mie scattering, deterioration of the equipment, impurities, poor calibration, ...).

These implications finally lead to the conclusion that a training set is best derived from the measured data. With respect to HSIs this means that a qualified person has to gather



(a) (b)

Figure 2: A simple clustering example. (a) depicts an unlabeled set of dots. A human being that looks at this illustration will unconsciously group or respectively categorize the dots to arrive at something like in (b), where the color coding corresponds to a perceived cluster.

a representative set of spectra from one or more images and label each according to its class affiliation.

The spatial and spectral resolution of todays HSIs however can make this a very challenging process which may take many weeks. The amount of information a human being has to deal with when studying HSIs can be quite overwhelming. One might try to select one of the many layers and use the spatial intensity features that are visible at this specific wavelength to gather the desired training data. The problem that comes with this approach is that the visibility of spatial features can vary with different wavelengths. Inevitably some features that might be vital for the construction of a training set thus will be overlooked.

Again there arises the need to have an image segmentation algorithm that can crunch the HSI into peaces that are more comprehensible for a human being. One such technique, which will be the central issue of this thesis, is *clustering*. Contrary to classification this method does not require *a priori* information about the data. Instead it produces its own labeled set based on a mathematical heuristic. Clustering has been applied in almost any field of studies that has to deal with the interpretation of large amounts of complex data. The variety of backgrounds where this technique has been applied has lead to a multitude of different algorithms and more are developed every day to meet the needs of an ever more data oriented society.

The aim of this thesis is to present a novel clustering algorithm which is designed to work on high-dimensional spectroscopic data.

2 Cluster Analysis

In order to get a quick impression what clustering is about consider figure 2a, where a scatter plot of black dots is depicted. Even before we are fully aware of it our brain subconsciously recognizes a structure within this image and organizes the seemingly meaningless dots into groups or so-called *clusters*. One possible clustering could be something like the one depicted in figure 2b, where three clusters are highlighted in green (1), orange (2) and blue (3). In the view of the author this categorization is meaningful, though that does not imply that others see it the same way. One might argue that (2) and (3) should yet be partitioned into two smaller clusters or even that there is no such 'natural' grouping at all.

Jain et al. (1999) define clustering (or *cluster analysis* as it is also known) as

> [...] the organization of a collection of patterns [...] into clusters based on similarity.

Similarity is often measured by using a distance metric such as the Euclidean distance. Two points in a data space are then considered more similar if they are closer together than other observations of the dataset. If we take two adjacent dots from the blue cluster and one from the orange cluster we come to the conclusion that the blue ones are more similar to each other than the orange dot. On the other hand if we consider that some dots of the blue cluster are more distant to each other than to one dot of the orange cluster we suddenly see that this criterion alone does not suffice to formulate a clustering algorithm. Xu and Wunsch (2005) see the goal of clustering as

> [...] to separate a finite unlabeled dataset into a finite and discrete set of 'natural', hidden data structures [...].

The term *natural* here refers to a structure that a human being might consider natural, but this is always in the eye of the beholder and cannot (so far) be formulated in a mathematical sense. This term becomes even more problematic to assess if we consider clusters in higher-dimensional data spaces. A human being cannot 'see' these clusters and therefore also cannot tell if their grouping is natural. Finally the Merriam-Webster Online Dictionary (2017) define cluster analysis as

> [...] a statistical classification technique for discovering whether the individuals of a population fall into different groups by making quantitative comparisons of multiple characteristics.

From the above definitions we can gather how difficult it is to grasp clustering in a mathematical term and indeed it is questionable whether a general problem-independent definition will ever be found at all. This is also the reason why clustering is a much more difficult task than classification, for the latter is a well defined problem with a loss

function which precisely formalizes what one is trying to do (Von Luxburg et al., 2012). There have been many attempts to design such validation criteria for clustering (Arbelaitz et al., 2013), which suffer from the same lack of a clear definition.

Despite these problems clustering has been successfully applied in exploratory data analysis and as a means to compress data. Nonetheless its application should be treated with care and there are various things to consider:

- Clustering *can* be applied if no *a priori* information about a dataset is known. On the other hand one usually already knows a great deal about the data even before we start to explore it. If we measure the product of a chemical process there are already some substances that we can expect to be in the data. In most cases we will also know their abundances.

- Spectroscopic data is usually high-dimensional and noisy. The majority of clustering algorithms are prone to the so-called *Curse of Dimensionality*, which is a term introduced by Bellman (1961) to describe the strange behavior of distance metrics in high-dimensional data spaces. Noise can be quite problematic for some algorithms as well and it can easily be shown that adding artificial noise to a dataset can produce an entirely different clustering result. Both problems can be tackled by using *Spectral Descriptors* (Lohninger and Ofner, 2014), which perform a dimensionality reduction as well as a noise reduction. On the other hand this also introduces assumptions about the content of the data which might lead to overlooking some important details.

- If we browse through the clustering literature of different research fields and take a look what kind of data is clustered in these papers, we also see how different the internal structures of data can be. Imagine that we select two descriptive wave numbers from an HSI and plot the values against each other in a two-dimensional scatter plot. If we do the same with datasets from different research fields, e.g. geodesics, face recognition, letter recognition, the structures that we will see might look quite different. In vibrational spectroscopy there is often a high correlation between variables which produces elongated clusters. On the other hand we might find clusters of great size and density differences.

With these things in mind one then has to choose which clustering algorithms are reasonably applicable for the current problem. To be able to so one has to know the criterion and the characteristics of the methods in question. We will now take a closer look on the differences between three of the more well established algorithms.

2.1 Notations

- An *object* (also *pattern*, *feature vector* or *observation*) is a single data item of a set of measurements used by the clustering algorithm. It typically consists of a vector

of d measurements: $\mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d})$.

- The individual scalar components $x_{i,j}$ of an object $\mathbf{x}_i$ are called *features*.
- d is the *dimensionality* of the object or the object space.
- A *set of objects* is denoted $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. In many cases a set of objects is viewed as an $n \times d$ matrix.

2.2 A Short Overview

From the viewpoint of machine learning *clustering* is an unsupervised learning method. This means that a clustering algorithm only needs the unlabeled dataset $\mathcal{X}$ to produce a result. Contrary to that *classification* is categorized as supervised learning. With respect to the result of the algorithm clustering can be divided into two categories:

Partitional Clustering attempts to form K clusters C_i in a set of objects $\mathcal{X}$, so that

- $C_i \neq \{\} \quad \forall i = 1, \ldots, K$
- $\bigcup_{i=1}^{K} C_i = \mathcal{X}$
- $C_i \cap C_j = \{\} \quad \forall i, j = 1, \ldots, K \wedge i \neq j$

Hierarchical Clustering attempts to build a nested tree structure of partitions $\mathbf{H}$, so that

- $\mathbf{H} = \{H_1, \ldots, H_q\} \quad q \leq n$
- $C_{i,m} \in H_m, C_{j,l} \in H_l, m > l$
  $\Rightarrow C_{i,m} \subseteq C_{j,l} \vee C_{i,m} \cap C_{j,l} = \{\}$

From the methodical viewpoint clustering can be further divided into grid-based, density-based, graph-based, model-based and combinatorial algorithms (Birant and Kut, 2007; Liu et al., 2012). The category graph-based contains another subcategory which is called spectral clustering (Von Luxburg, 2007). In order to distinguish spectral methods from other graph-based methods we will denote all algorithms which are not spectral as graph-oriented.

2.2.1 K-Means

K-Means is based on the *within-cluster sum of squares* criterion defined as

$$e^2(\mathcal{X}, \mathbf{C}) = \sum_{j=1}^{K} \sum_{i=1}^{|C_j|} ||\mathbf{x}_i^{(j)} - \mathbf{c}_j||^2, \tag{1}$$

where $i = 1, \ldots, |C_j|$, $\mathbf{C} = \{C_1, \ldots, C_K\}$ and $\mathbf{x}_i^{(j)} \in C_j, \mathbf{x}_i^{(j)} \notin C_k \quad \forall k \neq j$. $\mathbf{c}_j$ is the centroid of cluster C_j, which can be calculated with

$$\mathbf{c}_j = \frac{1}{|C_j|} \sum_{i=1}^{|C_j|} \mathbf{x}_i^{(j)}. \tag{2}$$

The goal of K-Means is to minimize e with respect to the user-defined number of clusters K by going through the following steps:

1. Choose the initial $\mathbf{c}_j$ randomly from the objects $\mathbf{x}_i$.
2. All $\mathbf{x}_i$ are assigned to the nearest $\mathbf{c}_j$ to form cluster C_j.
3. Compute the new centroids $\mathbf{c}_j$ from the objects $\mathbf{x}_i^{(j)}$ of C_j.
4. Repeat steps 2. and 3. until a termination condition is reached (e.g. no or minimal reassignment between clusters).

The result of the K-Means algorithm is the Voronoi decomposition of $\mathcal{X}$ with respect to the centroids $\mathbf{c}_j$ (assuming that the algorithm stops at step 2.). Since e can have multiple local minima one usually repeats the clustering algorithm with different initial conditions in hope to find the global minimum.

Figure 3 depicts two clustering results for $K = 3$ and $K = 6$ of the same dataset that was introduced in figure 2. In conjunction with the Voronoi decompositions these results reveal some of the characteristics of K-Means. In the view of the author $K = 3$ constitutes a meaningful number of clusters with boundaries corresponding to clusters (**1**), (**2**) and (**3**). However the results in figures 3a and 3b are quite different from that assumption. As can be expected from the *within-cluster sum of squares* criterion the globular cluster (**1**) is detected correctly. Clusters (**2**) and (**3**) both share some of each others data points. This is due to the large values of e when elongated clusters that are close together are concerned. If one increases to $K = 6$ cluster (**1**) is preserved while (**2**) and (**3**) are further divided.

K-Means is one of the most well established clustering algorithms and has often been used as a benchmark to compare others against it. Through the years other algorithms such as K-Medoid and Kernel K-Means (Jain, 2010) have been derived from it.

2.2.2 Hierarchical Cluster Analysis

Hierarchical Cluster Analysis (HCA) can be considered as a family of clustering algorithms that build a nested tree structure which reflects the hierarchy between the clusters with respect to a certain criterion. In general there are also partitional algorithms that can be applied consecutively on a dataset so that each resulting cluster is further divided into smaller clusters until the number of clusters reaches the number of objects.

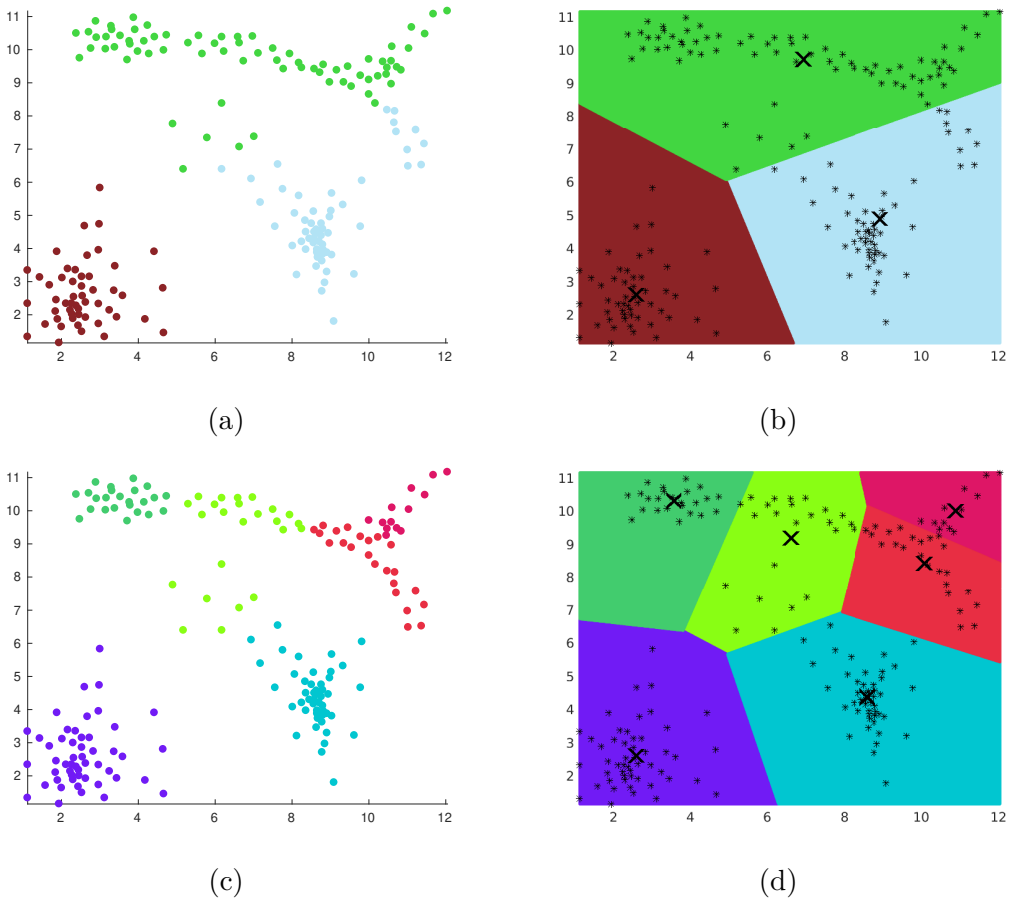


Figure 3: K-Means clustering result. (a) and (c) show the clustering results for $K = 3$ and $K = 6$. (b) and (d) depict the corresponding Voronoi decomposition with respect to the centers (indicated by 'x').

In the classical sense HCA refers to the linkage criteria listed in table 1 which are applied in an agglomerative way. This means that the observations $\mathbf{x}_i$ form the initial clusters. The selected criterion is then applied to each pair of observations to find the one which will form the cluster of the next hierarchical level. This process is repeated until all clusters are merged together. The threshold at which the criterion merges a pair of clusters is depicted in the dendrogram to highlight the hierarchy between the clusters. By cutting the tree at a certain threshold level one obtains a clustering where all clusters below that threshold have been merged.

Lance and Williams (1967) discovered an equation that combines the listed criteria into a single expression which is defined as

$$\mathrm{d}_{\bar{ij}k} = \alpha_i \mathrm{d}_{ik} + \alpha_j \mathrm{d}_{jk} + \beta \mathrm{d}_{ij} + \gamma |\mathrm{d}_{ik} - \mathrm{d}_{jk}|. \tag{3}$$

Here d_{ij}, d_{ik} and d_{jk} represent the distances between clusters C_i, C_j and C_k whereas $\mathrm{d}_{\bar{ij}k}$ denotes the distance between the merged cluster $C_i \cup C_j$ and C_k respectively. The

Table 1: Linkage criteria used in hierarchical cluster analysis.

Algorithm	Criterion	Characteristics
Single Linkage	Minimum distance between the nearest patterns of two clusters	Forms bigger clusters. Outliers are isolated
Complete Linkage	Minimum distance between the most distant patterns of two clusters	Forms smaller clusters
Average Linkage	The mean of the distances of all patterns of the two clusters	The real structure of the data set is reflected well
Ward's Method	Formation of the cluster, where the variance of all distances increases the least	The real structure of the data set is reflected well, if the clusters are of equal size

associated parameter settings for each linkage method are given in table 2.

Figure 4 depicts two results of Ward's Method for $K = 3$ in 4a and $K = 6$ in 4c. To emphasize the difference between the linkage criteria this figure also shows a result from Single Linkage in 4e. What strikes the eye is that Single Linkage can produce the same result which the author deems meaningful. Though this should not be understood in a way that Single Linkage is superior to other methods. If one moves the threshold just a little bit lower the clusters will fall apart into dense central regions surrounded by many small clusters in the outskirts.

Table 2: Lance-Williams parameters for hierarchical cluster analysis. n_l denotes the number of objects in cluster C_l.

Algorithm	α_i	α_j	β	γ
Single Linkage	0.5	0.5	0	-0.5
Complete Linkage	0.5	0.5	0	0.5
Average Linkage	0.5	0.5	0	0
Ward's Method	$\frac{n_i+n_k}{n_i+n_j+n_k}$	$\frac{n_j+n_k}{n_i+n_j+n_k}$	$\frac{-n_k}{n_i+n_j+n_k}$	0

2.2.3 DBSCAN

Density Based Spatial Clustering of Applications with Noise (DBSCAN) (Ester et al., 1996) is an algorithm that focuses on neighborhood relations rather than using a variance-oriented criterion like K-Means. Since its philosophy is somewhat related to graph-based algorithms the definitions will be discussed more in detail.

DBSCAN requires the user to define Eps, which can be interpreted as a radius that is drawn around each point and MinPts, which is a minimum number of other points that

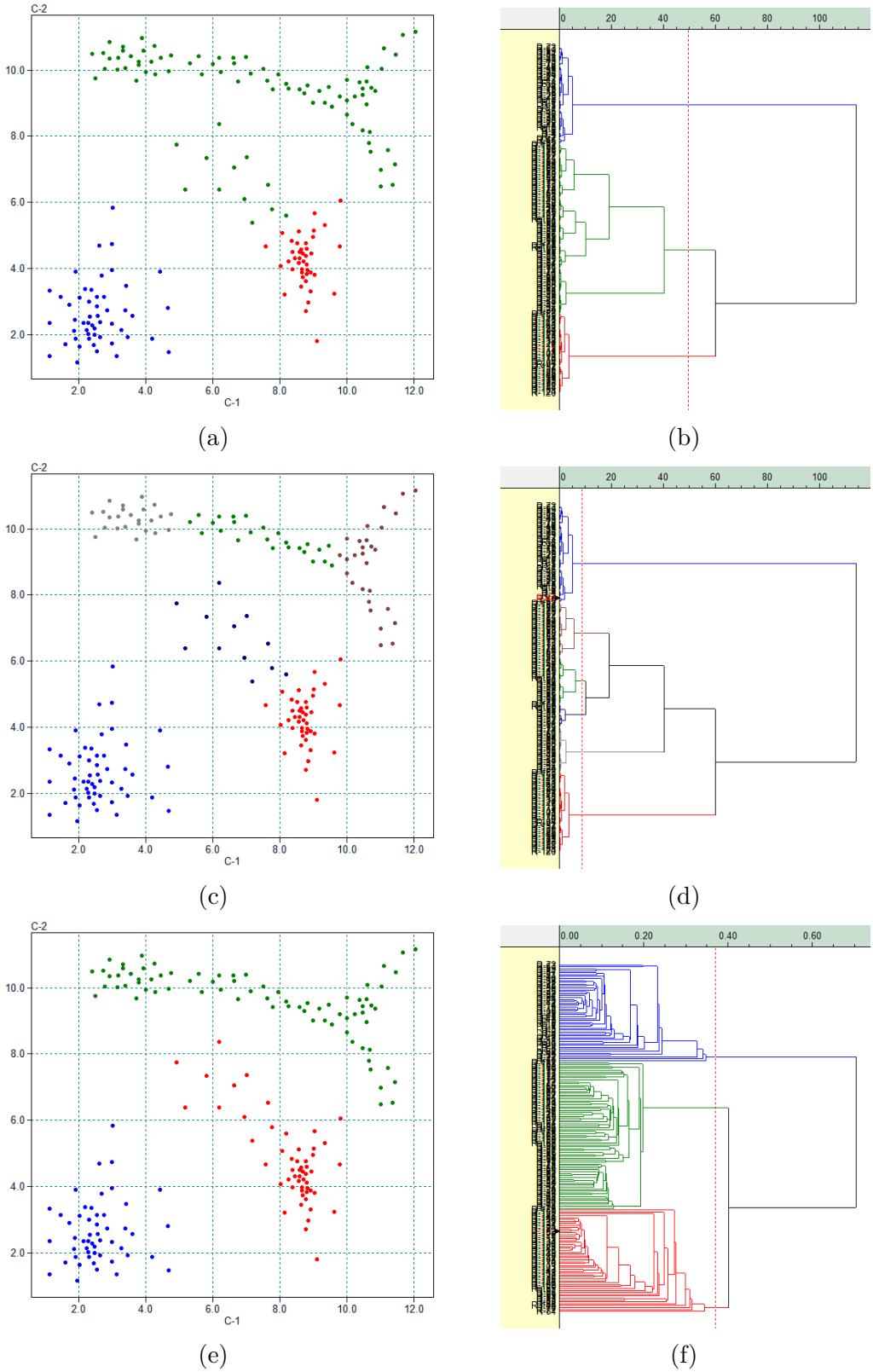


Figure 4: HCA clustering result. (a) and (c) show the clustering results for $K = 3$ and $K = 6$ using 'Ward's Method'. (b) and (d) depict the corresponding dendrograms. (e) and (f) show the resulting clusters when using 'Single Linkage'. The vertical red line in the dendograms show where the hierarchical tree was cut to receive the result.

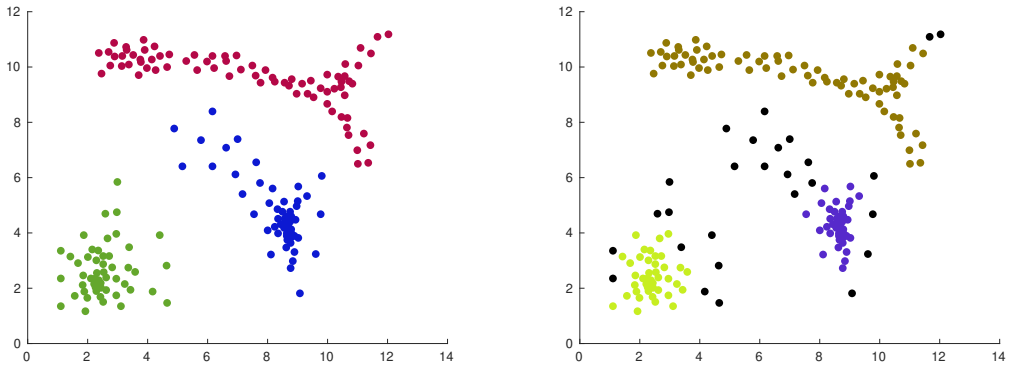
have to be within that radius. The Eps-neighborhood of a point p, denoted by $N_{\text{Eps}}(p)$, is then defined by $N_{\text{Eps}}(p) = \{q \in \mathcal{X} \mid \text{D}(p, q) \leq \text{Eps}\}$. A point p is *directly density-reachable* from a point q if $p \in N_{\text{Eps}}(q)$ and $|N_{\text{Eps}}(q)| \geq \text{MinPts}$. The latter definition is also known as the core point condition because points at the border of a cluster will usually not satisfy this condition due to a lack of close enough neighbors.

A point p is considered *density-reachable* from a point q if there is a chain of points $p_1, \ldots, p_n$, $p_1 = q$, $p_n = p$ such that p_{i+1} is directly density-reachable from p_i. Two border points of the same cluster are possibly not density reachable from each other because the core point condition might not hold for both of them. A point p is *density-connected* to a point q is there is a point o such that both p and q are density-reachable from o. Finally a cluster C_i has to satisfy the following conditions:

- $\forall p, q$: if $p \in C_i$ and q is density-reachable from p then $q \in C_i$.
- $\forall p, q \in C_i$: p is density connected to q.

All points which are not part of a cluster are considered noise. A result for Eps = 1.2 and MinPts = 4 is depicted in figure 5a. As can be seen DBSCAN can correctly identify the three clusters as seen by the author. If one reduces Eps to 0.7 DBSCAN identifies all points not satisfying this criterion as noise which is depicted as the black dots in figure 5b.

The ability to reject noise is sometimes an important criterion for selecting a suitable clustering algorithm for a given problem. K-Means and HCA do not distinguish noise as a different cluster.



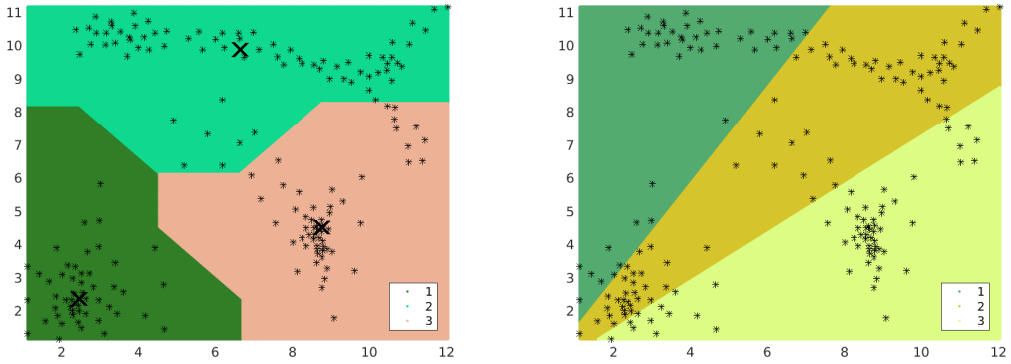
(a) (b)

Figure 5: DBSCAN clustering result for MinPts = 4 and Eps = 1.2 in (a) and MinPts = 4 and Eps = 0.7 in (b).

2.3 Metrics and other Similarity Measures

In the previously described algorithms it was assumed that the Euclidean distance is used as a similarity measure. Table 3 lists some other commonly used measures and metrics

that can be applied to define similarity. The different metrics can change the clustering result quite dramatically as can bee seen in figure 6 where the city block metric and the cosine similarity have been used to perform a K-Means clustering. Through the rest of this thesis the Euclidean metric is used.



(a) City block metric (b) Cosine similarity

Figure 6: K-Means results using different similarity measures.

2.4 Clustering Problems

As we saw in the previous section different clustering algorithms can sometimes produce quite contradicting results. This is of course due to the different mathematical criteria used to obtain the clustering. Clustering is also not a domain-independent discipline. In each research field there are data structures which are of more importance to detect than in others and therefore there are some specifics that can be problematic for certain algorithms.

Zhong et al. (2010) define the clusters in figures 7a-b as *well-separated*. This means that any pair of points within one cluster is closer together that a pair of points from both clusters. Let a and b be the two points from both clusters which are closest together. If the local point density around these points is high with respect to the distance between the two points then the problem is called *distance-separated*. If the local point density of the two points differs significantly then the problem is called *density-separated*. Under these definitions figures 7a-e would then be considered as distance-separated whereas figure 7f would be a density-separated problem. Figures 7g-h constitute *touching* problems as these clusters are connected through a 'neck'. The removal of the neck should produce two clusters which are substantially larger than the neck itself.

Furthermore clusters can be categorized into *compact* and *connected*. The former is a set of points such that the distance between any point in the cluster and the representative (this can be the centroid or medoid) is less than the distance between the point and any representative of other clusters. The latter is a set of points such that for every point in

Table 3: Metrics and quasi-metrics for the measurement of similarity

Measures	Forms	Comments
Euclidean	$D_{i,j} = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2}$	Special case of Minkowski metric at $n = 2$. This metric is the most widely used. It tends to form hyperspherical clusters.
Squared Euclidean	$D_{i,j} = \sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2$	Puts more emphasis on patterns with a greater distance.
Tchebycheff	$D_{i,j} = \max_{1\leq k\leq d} \lvert x_{i,k}, x_{j,k}\rvert$	Special case of Minkowski metric at $n \to \infty$.
Minkowski	$D_{i,j} = (\sum_{k=1}^{d} \lvert x_{i,k} - x_{j,k}\rvert^{1/N})^N$	Features with large values and variances tend to dominate over other features.
Mahalanobis	$D_{i,j} = (\mathbf{x}_i - \mathbf{x}_j)^{\mathrm{T}}\mathbf{S}^{-1}(\mathbf{x}_i - \mathbf{x}_j)$	$\mathbf{S}$ is the within-group covariance matrix. This measure is invariant to any nonsingular linear transformation. It tends to form hyperellipsoidal clusters.
Pearson Correlation	$D_{i,j} = \frac{1 - r_{i,j}}{2}$	$r_{i,j}$ is the Pearson correlation coefficient. This measure is not a metric. It is unable to detect the magnitude of differences of two variables. Widely used for analyzing gene expression data.
Point Symmetry	$D_{i,r} = \min_{\substack{j=1,\ldots,n \\ j\neq i}} \frac{\lvert\lvert(\mathbf{x}_i - \mathbf{x}_r) + (\mathbf{x}_j - \mathbf{x}_r)\rvert\rvert}{\lvert\lvert\mathbf{x}_i - \mathbf{x}_r)\rvert\rvert + \lvert\lvert\mathbf{x}_j - \mathbf{x}_r)\rvert\rvert}$	Not a metric. $\mathbf{x}_r$ is a reference point to the observation $\mathbf{x}_i$. $D_{i,r}$ is minimized when a symmetric pattern exists.
Cosine Similarity	$S_y = \cos\alpha = \frac{\mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j}{\lvert\lvert\mathbf{x}_i\rvert\rvert\lvert\lvert\mathbf{x}_j\rvert\rvert}$	Most commonly used measure in document clustering.
Gaussian Similarity	$D_{i,j} = \exp(-\lvert\lvert\mathbf{x}_i - \mathbf{x}_j\rvert\rvert^2/(2\sigma^2))$	Often applied in spectral clustering to derive similarity graphs.

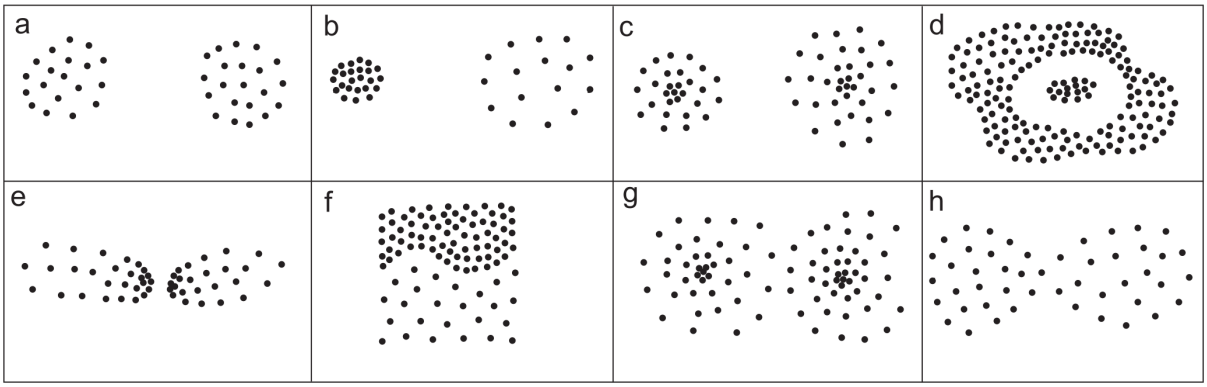


Figure 7: Clustering problems as defined by Zahn (1971).

the cluster there exists at least one other point in the cluster so that the distance between them is less than the distance between this point and any point not in the cluster. Figure 8a constitutes a compact problem whereas figure 8b is a connected problem.

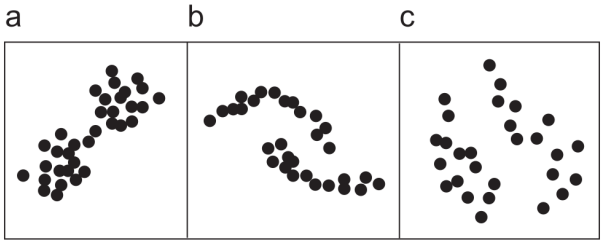


Figure 8: Clustering problems as defined by Handl and Knowles (2007).

The illustrations of figures 7 and 8 show the clusters with clear boundaries so that it is unambiguous where one cluster ends and the other begins. In chemical datasets there usually occur mixture effects because compounds diffuse into each other or are inhomogeneous. As a two-dimensional example one could illustrate this problem like in figure 9. For a human being it is easy to see the three clusters though it is not obvious where to draw the decision boundaries to separate them.

Another problem that is not mentioned in the above definitions is that clusters can vary greatly in size and density and can even be nested. One might argue that this falls under the definition of density-separated, but this would require some notion of a cluster boundary. In figure 10 such a problem is illustrated to highlight the difference. At first glance there is only one cluster in figure 10a. If one looks closer there seems to be a second compact cluster within the rectangle. Figure 10b shows a zoomed-in view of the rectangle which again seems to contain only one cluster. If one zooms in towards the second rectangle the single cluster separates again as in figure 10c which makes a total of three clusters that reside within this dataset.

At this point one should ask how we can so easily recognize the clusters of figures 9 and 10 even though there are no clear boundaries between them. The answer lies

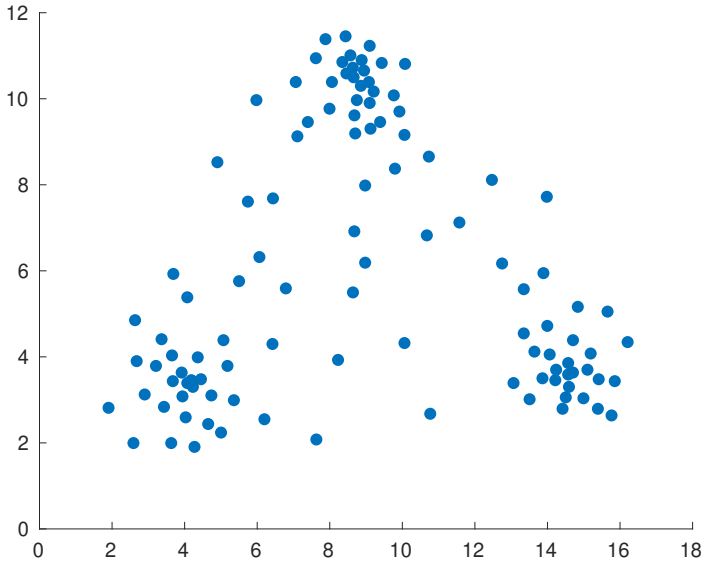


Figure 9: The Mixing Clusters dataset consists of three clusters that have no clear boundary between them.

in the density distribution of the datasets. The human perception is very sensitive to gradients and there are obviously some regions where the density of data points increases or decreases. The author will denote problems which are separable using data gradients as *gradient-separable*. This form of separability stands at the center of this thesis and will be discussed in more depth in later chapters.

2.5 Graph-Based Algorithms

2.5.1 Fundamentals of Graph Theory

A graph $\mathcal{G} = (V, E)$ is an abstraction where the relations between objects are represented by the presence or absence of connecting edges. V represents a non-empty set of vertices and E a set of edges. With respect to the dataset the objects $\mathbf{x}_i, \mathbf{x}_j \in \mathcal{X}$ thus become vertices $v_i, v_j \in V$ and the edge that connects v_i and v_j is denoted as $e_{ij} \in E$. Each edge e_{ij} has an associated weight $w_{ij} \geq 0$ that describes the similarity between vertices v_i and v_j. The weighted adjacency matrix of the graph is the matrix $\mathbf{W} = (w_{ij})$. If $w_{ij} = 0$ then v_i and v_j are not connected. If $w_{ij} = w_{ji} \quad \forall i, j = 1, \ldots, n$ then $\mathbf{W}$ is symmetric and the graph is called undirected. The opposite would be a directed graph where a connection from v_i to v_j does not imply a connection from v_j to v_i. Figure 11 depicts three common examples of similarity graphs that were derived from the introductory dataset in figure 2 using the euclidean distance as a metric to measure the edge weight.

The Delaunay Triangulation $\mathrm{DT}(\mathcal{X})$ is a triangulation of the dataset $\mathcal{X}$ such that no point is inside the circumcircle of any triangle of $\mathrm{DT}(\mathcal{X})$. For a d-dimensional space it is

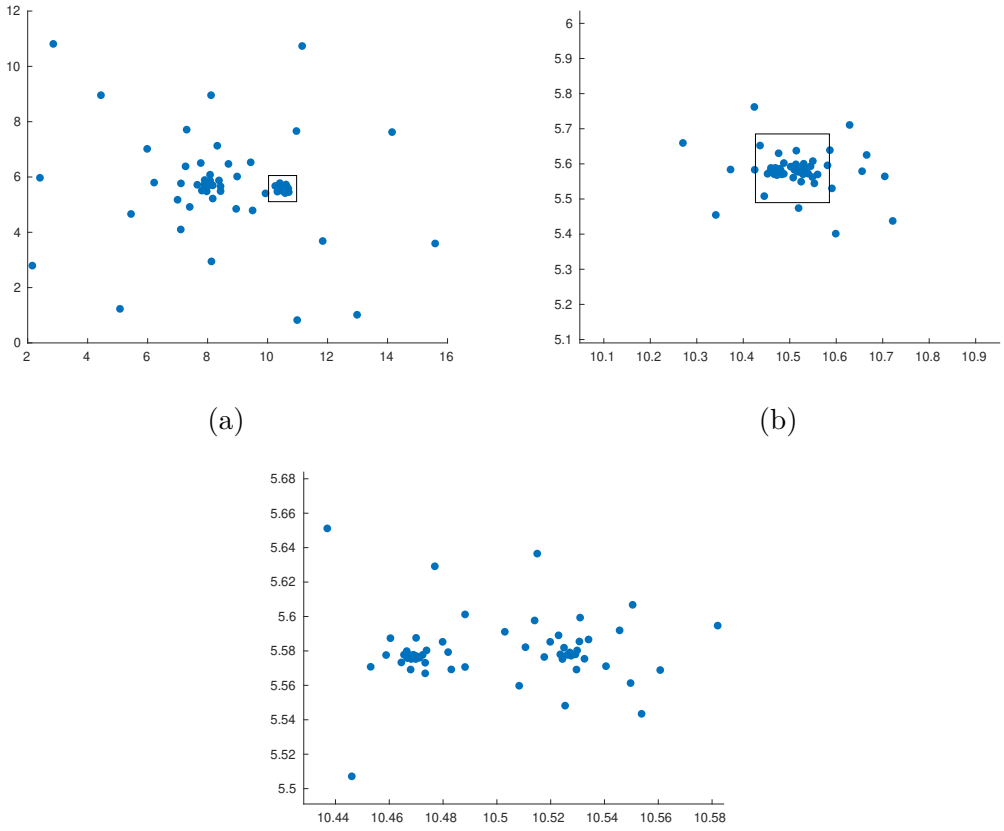


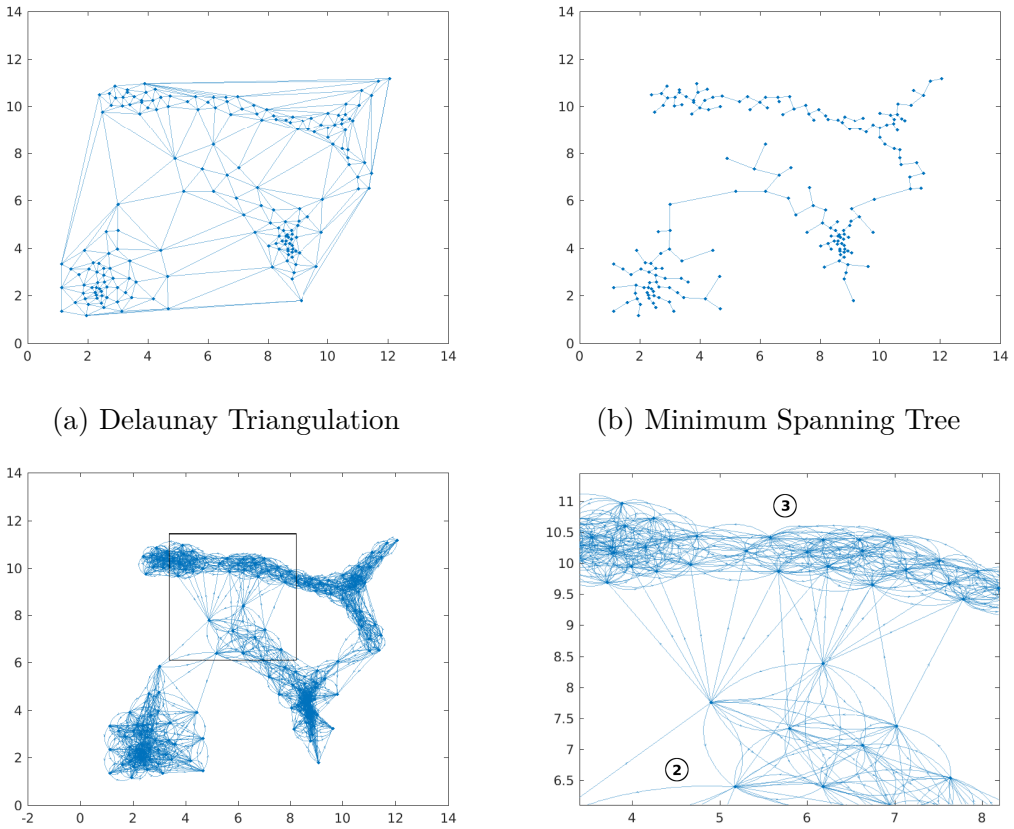
Figure 10: The Russian Dolls dataset consist of a series of three nested globular clusters that differ greatly in their size. The rectangle in (a) represents the frame of (b) and so on.

defined as a triangulation $\mathrm{DT}(\mathcal{X})$ such that no point of $\mathcal{X}$ is inside the circum-hypersphere of any d-simplex in $\mathrm{DT}(\mathcal{X})$.

The Minimum Spanning Tree (MST) is a graph that connects all vertices with the minimum possible total edge weight and without any cycles. In most cases the MST is unique but there can be more than one MST with the same total weight under certain circumstances.

The k-nearest neighbor graph (kNN) is computed by performing a k nearest neighbor search for each object $\mathbf{x}_i$. Each object $\mathbf{x}_j$ among the k-neighborhood is then linked to $\mathbf{x}_i$ through an edge e_{ij}. In general the kNN graph is directed because $\mathbf{x}_j$ being in the K-neighborhood of $\mathbf{x}_i$ does not imply that $\mathbf{x}_i$ is also in the k-neighborhood of $\mathbf{x}_j$. However some algorithms require the graph to be undirected. One way to achieve this is to set all $w_{ji} := w_{ij}$ if $w_{ij} \neq 0$ and $w_{ji} = 0$. The so called *mutual* k-nearest neighbor graph is deduced by setting all $w_{ij} := 0$ if $w_{ij} \neq w_{ji}$.

If one compares the different graph types in figure 11 with the original dataset of figure 2 there are some fundamental differences in how the resulting graph stores neighborhood relations and what information is lost. The Delaunay triangulation in figure 11a connects



(a) Delaunay Triangulation

(b) Minimum Spanning Tree

(c) KNN with $K = 13$

(d) KNN zoomed in

Figure 11: Examples of graph types. The rectangle in (c) corresponds to the zoomed-in view in (d). Arched edges represent a bidirectional connection.

all three clusters through their border points regardless of the distance between them. On the other hand the intra-cluster relations are reduced to the neighbors which are part of a triangle. The MST in figure 11b reduces the number of edges even further as there are only two edges that connect the three clusters. The kNN graph clearly shows its undirected nature if one takes a closer look at the zoomed-in view in figure 11d. Here cluster (2) is connected through directed edges to (3) but not vice versa. Another interesting property of kNN graphs is that if k is set low enough the clusters can be fully disconnected from each other simply because there is no member of a different cluster among any of the k-neighborhoods of the objects $\mathbf{x}_i$.

2.5.2 Spectral Clustering

Spectral Clustering is a family of algorithms that use standard clustering methods such as K-Means to cluster the eigenvectors of the Laplacian matrix $\mathbf{L}$ (Von Luxburg, 2007; Nascimento and De Carvalho, 2011). Different kinds of Laplacians can be deduced from the weighted adjacency matrix $\mathbf{W}$ and the degree matrix $\mathbf{D}$ of a graph $\mathcal{G} = (V, E)$. The

degree of a vertex v_i is defined as $d_i = \sum_{j=1}^{n} w_{ij}$, where n denotes the number of vertices. $\mathbf{D}$ is then defined as the diagonal matrix $\mathbf{D}_{ii} = d_i$. Given a subset of vertices $A \subset V$ and its complement $\bar{A} = V \setminus A$ the indicator vector $\mathbb{1}_A = (f_1, \ldots, f_n)^{\mathrm{T}} \in \mathbb{R}^n$ is defined as the vector with entries $f_i = 1$ if $v_i \in A$ and $f_i = 0$ otherwise. $A \subset V$ is connected if any two vertices in A can be joined by a path such that all intermediate points also lie in A. A is called a connected component if it is connected and if there are no connections between vertices in A and $\bar{A}$. The nonempty sets $A_1, \ldots, A_K$ form a partition of the graph if $A_i \cap A_j = \varnothing$ and $A_1 \cup \cdots \cup A_K = V$.

In the following $\mathcal{G}$ is assumed to be undirected. Eigenvectors are not necessarily assumed to be normalized which means that the constant vector $\mathbb{1}$ and a multiple $a\mathbb{1}$ for $a \neq 0$ will be considered as the same eigenvectors. Eigenvalues will be ordered increasingly, respecting multiplicities. The first K eigenvectors therefore correspond to the K smallest eigenvalues.

The *unnormalized* graph Laplacian is defined as

$$\mathbf{L} := \mathbf{D} - \mathbf{W} \tag{4}$$

and has the following properties:

- $\mathbf{f}^{\mathrm{T}}\mathbf{L}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 \quad \forall \mathbf{f} \in \mathbb{R}^n$.
- $\mathbf{L}$ is symmetric and positive semi-definite.
- The smallest eigenvalue of $\mathbf{L}$ is 0 and the corresponding eigenvector is the constant vector $\mathbb{1}$.
- $\mathbf{L}$ has n non-negative, real-valued eigenvalues $0 = \lambda_1 \leq \cdots \leq \lambda_n$.

A property of the unnormalized graph Laplacian which is important for spectral clustering is that the multiplicity K of the eigenvalue 0 of $\mathbf{L}$ equals the number of connected components $A_1, \ldots, A_K$ in the graph. The eigenspace of eigenvalue 0 is spanned by the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_K}$ of those components.

This property can be proven by considering the case $K = 1$ which means that $\mathcal{G}$ is a single connected component. Let $\mathbf{f}$ be an eigenvector with eigenvalue 0. Then we know that

$$\frac{1}{2}\sum_{i,j=1}^{n} w_{ij}(f_i - f_j)^2 = 0 \tag{5}$$

which can only be true if all the terms $w_{ij}(f_i - f_j)^2$ vanish. Since $w_{ij} \geq 0$ this means that all $f_i = f_j$. Thus $\mathbf{f}$ is the constant vector $\mathbb{1}$ which is also the indicator vector of the connected component.

Now let us consider the case of $K > 1$ connected components. Without loss of generality we order the vertices of the graph Laplacian $\mathbf{L}$ so that vertices of the same

component are next to each other. We thus get a matrix

$$\mathbf{L} = \begin{pmatrix} \mathbf{L}_1 & & & \\ & \mathbf{L}_2 & & \\ & & \ddots & \\ & & & \mathbf{L}_K \end{pmatrix} \tag{6}$$

of block diagonal form where each $\mathbf{L}_i$ is the graph Laplacian of the i^{th} connected component. Every $\mathbf{L}_i$ has an eigenvalue of 0 with a multiplicity of 1. The corresponding eigenvector is the constant vector $\mathbb{1}$ with respect to $\mathbf{L}_i$. Thus the matrix $\mathbf{L}$ has as many eigenvalues of 0 as there are connected components and the corresponding eigenvectors are the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_K}$ with respect to $\mathbf{L}$.

There are two matrices which are called *normalized* graph Laplacians. They are defined as

$$\mathbf{L}_{\text{sym}} := \mathbf{D}^{\frac{1}{2}}\mathbf{L}\mathbf{D}^{\frac{1}{2}} = \mathbf{I} - \mathbf{D}^{\frac{1}{2}}\mathbf{W}\mathbf{D}^{\frac{1}{2}} \tag{7}$$

and

$$\mathbf{L}_{\text{rw}} := \mathbf{D}^{-1}\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}, \tag{8}$$

and have the following properties:

- $\mathbf{L}_{\text{sym}}$ is a symmetric matrix and $\mathbf{L}_{\text{rw}}$ is closely related to a random walk.
- $\mathbf{f}^{\text{T}}\mathbf{L}_{\text{sym}}\mathbf{f} = \frac{1}{2}\sum_{i,j=1}^{n} w_{ij}\left(\frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}}\right)^2 \quad \forall \mathbf{f} \in \mathbb{R}^n$.
- λ is an eigenvalue of $\mathbf{L}_{\text{rw}}$ with eigenvector $\mathbf{u}$ if and only if λ is an eigenvalue of $\mathbf{L}_{\text{sym}}$ with eigenvector $\mathbf{w} = \mathbf{D}^{\frac{1}{2}}\mathbf{u}$.
- λ is an eigenvalue of $\mathbf{L}_{\text{rw}}$ with eigenvector $\mathbf{u}$ if and only if λ and $\mathbf{u}$ solve the generalized eigenproblem $\mathbf{L}\mathbf{u} = \lambda\mathbf{D}\mathbf{u}$.
- 0 is an eigenvalue of $\mathbf{L}_{\text{rw}}$ with the constant vector $\mathbb{1}$ as eigenvector. 0 is an eigenvalue of $\mathbf{L}_{\text{sym}}$ with eigenvector $\mathbf{D}^{\frac{1}{2}}\mathbb{1}$.
- $\mathbf{L}_{\text{sym}}$ and $\mathbf{L}_{\text{rw}}$ are positive semi-definite and have n nonnegative real-valued eigenvalues $0 = \lambda_1 \leq \cdots \leq \lambda_n$.

As is the case with the unnormalized Laplacian $\mathbf{L}$ the multiplicity K of the eigenvalue 0 of both $\mathbf{L}_{\text{sym}}$ and $\mathbf{L}_{\text{rw}}$ equals the number of connected components $A_1, \ldots, A_K$. For $\mathbf{L}_{\text{rw}}$ the eigenspace of 0 is spanned by the indicator vectors $\mathbb{1}_{A_i}$ of those components. For $\mathbf{L}_{\text{sym}}$ the eigenspace of 0 is spanned by the vectors $\mathbf{D}^{\frac{1}{2}}\mathbb{1}_{A_i}$.

Based on the properties of graph Laplacians we can now formulate the method of *unnormalized spectral clustering* which goes through the following steps:

1. Compute $\mathbf{L} \in \mathbb{R}^{n \times n}$ and define the desired number of clusters K.

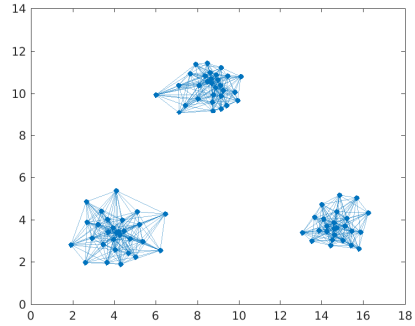
2. Compute the first K eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$ of $\mathbf{L}$.

3. Let $\mathbf{U} \in \mathbb{R}^{n \times K}$ be the matrix containing the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K$ as columns. Let $\mathbf{y}_i \in \mathbb{R}^K$ be the vector corresponding to the i^{th} row of $\mathbf{U}$. Cluster the points $\mathbf{y}_i$ using the K-Means algorithm into clusters $A_1, \ldots, A_K$.

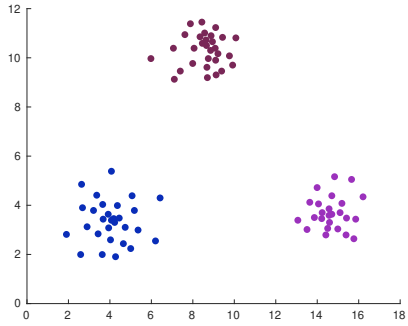
Normalized spectral clustering as defined by Shi and Malik (2000) differs in step 2 as the K eigenvectors are computed from the generalized eigenproblem $\mathbf{Lu} = \lambda\mathbf{Du}$ which means that $\mathbf{L}$ is replaced by $\mathbf{L}_{\text{rw}}$. Ng et al. (2002) proposed to use the matrix $\mathbf{L}_{\text{sym}}$. The rows of the resulting matrix $\mathbf{U}$ are then normalized to gain the matrix $\mathbf{T} = (t_{ij})$ where $t_{ij} = u_{ij}/(\sum_{k=1}^{K} u_{ik}^2)^{1/2}$. From there the procedure continues as in step 3 with $\mathbf{T}$ replacing $\mathbf{U}$. In step 3 the K-Means algorithm is used though other clustering algorithms are applicable as well.

In the case of K connected components the eigenvectors of the eigenvalue 0 are the indicator vectors $\mathbb{1}_{A_1}, \ldots, \mathbb{1}_{A_K}$. Thus each vertex $v_j \in A_i$ is mapped to the same point which makes it a very trivial clustering problem. An example of such a clustering task is given in figure 12, where a kNN graph with $k = 13$ was computed on three well-separated clusters. As can be seen in figure 12a all three clusters form their own connected component. The graph was clustered using the graph Laplacian $\mathbf{L}_{\text{rw}}$. As expected the first three eigenvalues in figure 12c are 0 which corresponds to the number of connected components. The corresponding eigenvectors are the indicator vectors $\mathbb{1}_{A_1}, \mathbb{1}_{A_2}, \mathbb{1}_{A_3}$ in 12d-f. Note that the indicator vectors are not normalized and thus $\mathbb{1}_{A_i}$ and a multiple $a\mathbb{1}_{A_i}$ for $a \neq 0$ are considered as the same eigenvectors. The remaining eigenvectors would contain information about the structures within the connected components but as can bee seen in 12g-h there seems to be no clustering tendency.

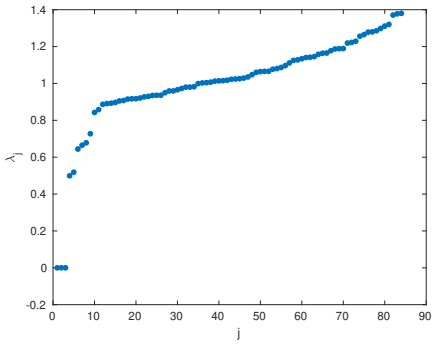
The appearance of the eigenvectors changes completely however, if the graph in question is a single connected component. The Mixing Clusters dataset of figure 9 was derived from the three clusters in figure 12 by adding points into the empty space between them. As can bee seen in figure 13a these additional points serve as a bridge for the 13-nearest neighbor graph to connect the three clusters. An interesting feature which is visible in figure 13c is the so-called eigengap between the eigenvalues λ_3 and λ_4. The distance $|\lambda_3 - \lambda_4|$ is obviously significantly larger than the preceding ones. The eigengap can be used as a heuristic for choosing the number of clusters K if the data contains well-pronounced clusters. Figure 13d depicts the constant eigenvector for eigenvalue 0. Seen up close the values vary within a small margin which is due to errors in the numerical computation of the eigenvectors. The eigenvectors $\mathbf{u}_{j>1}$ in 13e-f are no longer constant but their values still indicate the affiliation of vertices to certain subsets. Here it becomes clear why only the first K eigenvectors are used for the final clustering step as the information content of $\mathbf{u}_4$ and $\mathbf{u}_5$ is already very small.



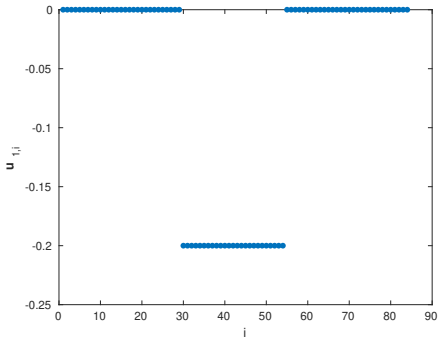
(a) Undirected 13-nearest neighbor graph



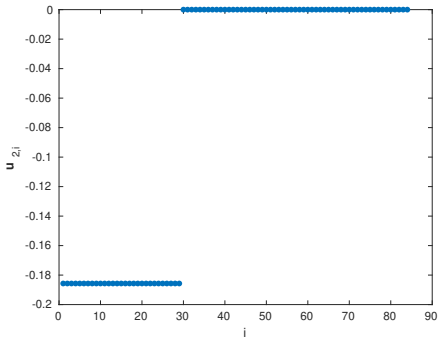
(b) Clustering result



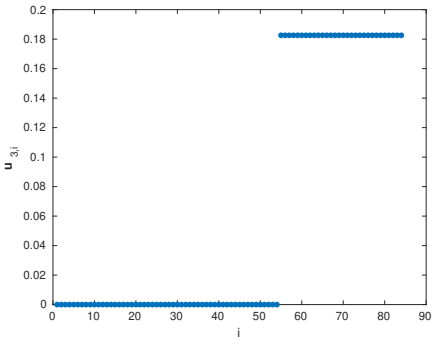
(c) Eigenvalues λ_j



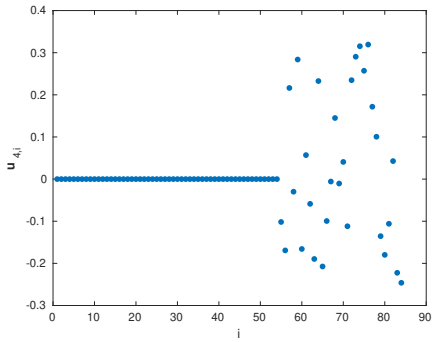
(d) Eigenvector $\mathbf{u}_1$



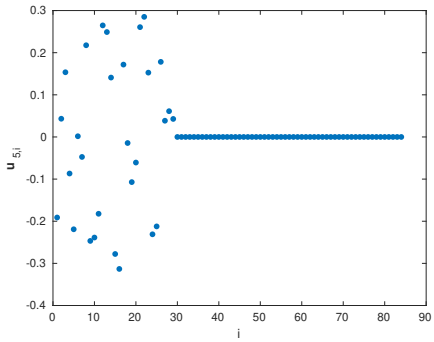
(e) Eigenvector $\mathbf{u}_2$



(f) Eigenvector $\mathbf{u}_3$

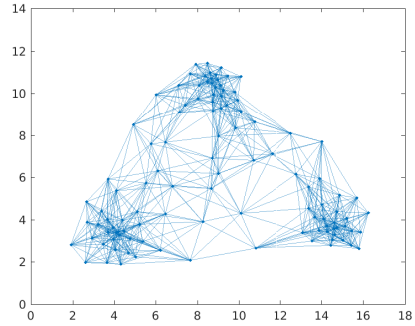


(g) Eigenvector $\mathbf{u}_4$

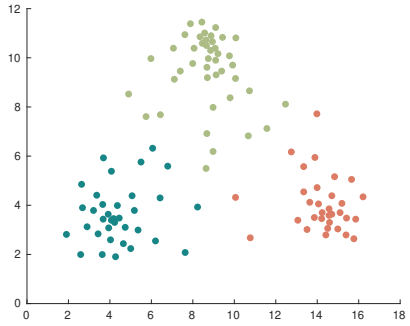


(h) Eigenvector $\mathbf{u}_5$

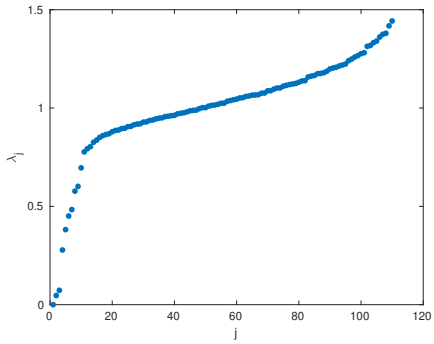
Figure 12: Spectral clustering of three connected components.



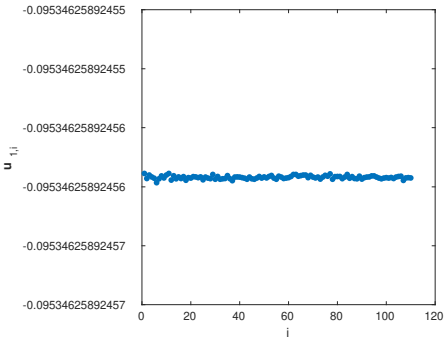
(a) Undirected 13-nearest neighbor graph



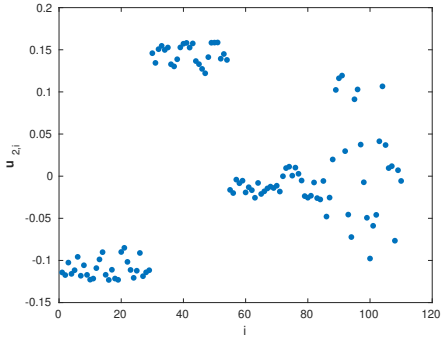
(b) Clustering result



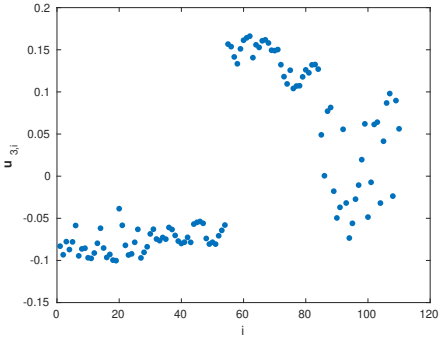
(c) Eigenvalues λ_j



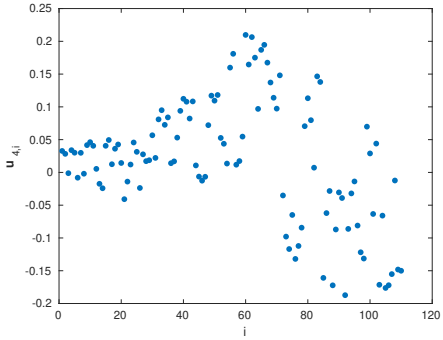
(d) Eigenvector $\mathbf{u}_1$



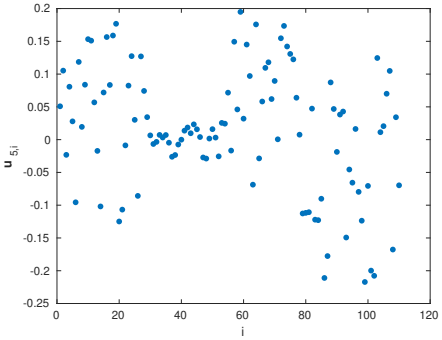
(e) Eigenvector $\mathbf{u}_2$



(f) Eigenvector $\mathbf{u}_3$



(g) Eigenvector $\mathbf{u}_4$



(h) Eigenvector $\mathbf{u}_5$

Figure 13: Spectral clustering of a single connected components.

2.5.3 Graph-Oriented Clustering

Apart from spectral clustering there is another family of clustering algorithms that heavily rely on data being represented as graphs. These methods distinguish themselves from spectral clustering as they do not cluster the eigenvectors of a graph Laplacian. In fact they are more comparable to algorithms such as DBSCAN. Depending on the construction method graphs have some advantages over using the dataset for clustering. A computationally intensive part of the clustering process is the calculation of the similarity measure. As objects or pairs of them are usually iteratively revisited the similarity measure thus has to be recomputed over and over again. If one stores the pairwise distances as weights of a graph the associated adjacency matrix $\mathbf{W}$ thus becomes a look-up table which can greatly increase performance. For non-graph-oriented algorithms this means to compute the *fully connected* graph, which is the graph that links every object $\mathbf{x}_i$ with any other object $\mathbf{x}_j$, where $i \neq j = 1, \ldots, n$.

Especially in high-dimensional data spaces this transforms an expensive repeated computation of the similarity measure between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ to the much cheaper task of reading the weight w_{ij} of the weight matrix $\mathbf{W}$ from memory. Many implementations of clustering algorithms use this approach though it comes with a great drawback: For a dataset of n objects the storage requirement for a fully connected graph using double precision is n^2`*sizeof(double)`. For a set of $n = 250000$ data points this corresponds to a memory requirement of 465.7 GiB, which is clearly out of range for most computing devices. This obstacle encourages the study of algorithms which use less memory intensive graphs.

Another aspect of graphs such as the ones presented in figure 11 is that they are an abstraction of the dataset in which the applied algorithm focuses on local neighborhood relations rather than using the distance information of the entire dataset. This makes graph-based methods well-suited for dealing with connected clustering problems. In the following some concepts of graph-oriented algorithms will be discussed in brief. Most of them use the same criteria as in other non-graph-oriented algorithms with the sole difference that these are applied to vertices and edges instead of objects.

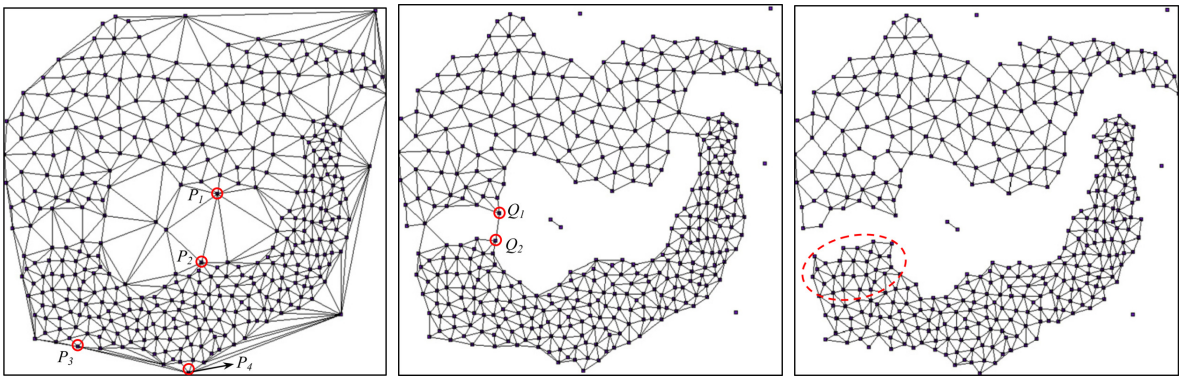
Liu et al. (2012) used the Delaunay triangulation which is depicted in figure 14a as a model of the spatial proximity relationships among objects. Some edges are then removed by applying an global edge constraint which is defined as

$$\mathrm{GDC}(v_i) = \mu_w + \frac{\mu_w}{\mu(v_i)}\sigma_w, \tag{9}$$

where μ_w is the average edge weight, σ_w its standard deviation and $\mu(v_i)$ the average weight of all edges originating from v_i. This produces the graph in figure 14b. In a second step a local edge constraint produces the graph in figure 14c, which is defined as

$$\mathrm{LDC}(v_i) = \mu^{2^{\mathrm{nd}}}(v_i) + \beta\mu_\sigma^{2^{\mathrm{nd}}}(v_i), \tag{10}$$

where $\mu^{2^{\text{nd}}}(v_i)$ is the mean of the 2^{nd}-order neighborhood, $\mu_\sigma^{2^{\text{nd}}}(v_i)$ the mean of the standard deviation and $\beta > 1$. The final graph is then clustered with a density-based clustering algorithm which is comparable to DBSCAN to obtain the result.



(a) (b) (c)

Figure 14: The edges of the Delaunay triangulation in (a) are subjected to a global edge constraint which produces the graph in (b), followed by a local edge constraint resulting in graph (c). (Liu et al., 2012)

The removed links are often referred to as 'inconsistent' edges. Minimum Spanning Tree clustering also uses this method were edges e_{ij} of an MST, such as the one in figure 11b, are removed if the condition

$$|\mu^{2^{\text{nd}}}(v_i) - w_{ij}| > \zeta \mu_\sigma^{2^{\text{nd}}}(v_i) \tag{11}$$

holds, where ζ is a user-defined value. This approach has the drawback that only very little information is used to determine the removal of an edge which can lead to an over-segmentation of the clusters.

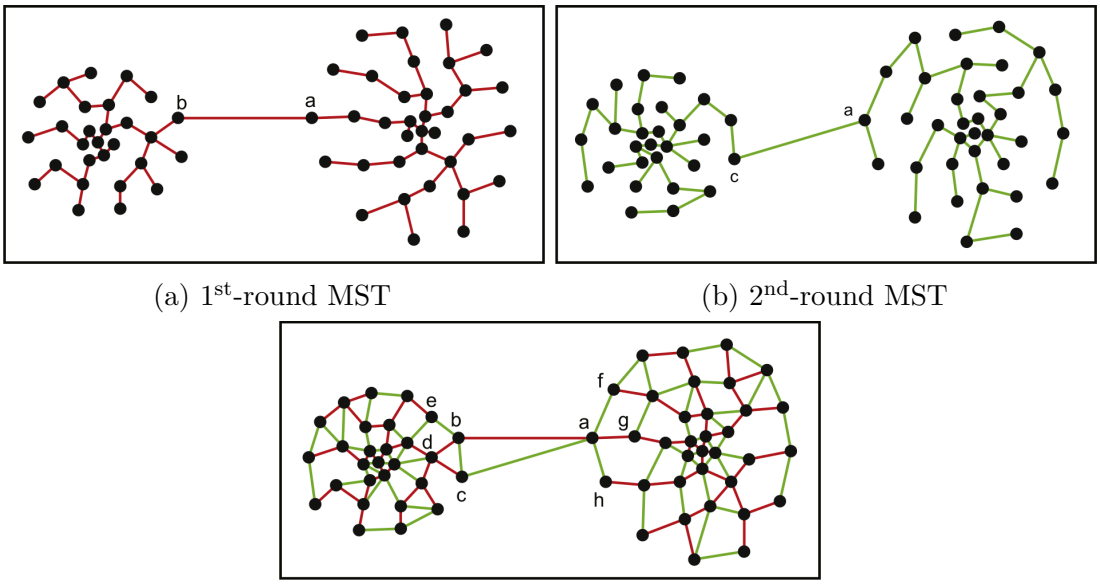
Zhong et al. (2010) therefore proposed to use the Two-Rounds MST which is depicted in figure 15c. This graph is a combination of two computations where the first MST T_1 is conducted on the fully connected graph of the dataset and the second T_2 on the remaining edges after all edges belonging to the first MST are removed. The weights w_{ij} are then redefined as

$$w_{ij}^{\text{new}} := \frac{w_{ij} - \min(\text{avg}(E_i \setminus e_{ij}), \text{avg}(E_j \setminus e_{ji}))}{w_{ij}}, \tag{12}$$

where $\text{avg}(E_i \setminus e_{ij})$ refers to the average weight of edges originating from v_i excluding edge e_{ij}. In the next step the edges are sorted in descending order with respect to w_{ij}^{new}. The clustering then happens by consecutively removing edges until a cut is achieved and a subset of the Two-Rounds MST thus becomes a connected component. Whether the cut is valid is determined by

$$\text{Ratio}(B) = \frac{\min(|B \cap T_1|, |B \cap T_2|)}{|B|} \geq \lambda, \tag{13}$$

where B is the set of removed edges and $\lambda = 1/3$. For each new connected component this procedure is repeated until all further cuts are invalid. The idea behind this validity criterion is the following: If two separated clusters are tested the cardinalities $|B \cap T_1|$ and $|B \cap T_2|$ will be almost equal, thus resulting in a value around 0.5. For a connected component that should not be further divided the ratio will be more skewed as the cardinality of $|B \cap T_2|$ is much larger.



(a) 1st-round MST

(b) 2nd-round MST

(c) Two-Rounds MST

Figure 15: The Two-Rounds MST proposed by Zhong et al. (2010) is a combination of the 1st and 2nd-round minimum spanning tree.

2.6 Common Evaluation Approaches and their Deficiencies

The previous section gave a brief overview of different kinds of clustering algorithms and showed that some procedures can behave very differently depending on the structure of the dataset and the shapes of the clusters. This leads to a fundamental problem when the task of clustering has to be performed on a high-dimensional dataset. Since our perception is limited to three dimensions the visual assessment of the clustering in the feature space is no longer possible. As a consequence the evaluation of the results is no longer trivial.

Scanning the literature one can find many different attempts to evaluate such clusterings. These methods are often used to argue that a certain algorithm successfully clusters the data in a meaningful way or that one method outperforms another. The problem that comes with this evaluation procedures is that they are insufficient and often completely misleading (Von Luxburg et al., 2012). The following paragraphs will discuss these approaches and their issues more closely:

Evaluation using artificial datasets. The clustering algorithm is applied on an artificial dataset which is created by drawing samples from a mixture of probability functions. Thereby the samples can be assigned to a 'ground truth' which makes it easy to evaluate whether the clusters were correctly recognized. This procedure makes sense if one wants to assess the statistical performance of a clustering algorithm under certain assumptions but it cannot be used to evaluate the usefulness of the clustering.

Evaluation using classification benchmark datasets. The clustering algorithm is applied to a classification dataset. These datasets have the property that each observation already has a class affiliation which is then treated as the ground truth against which the clustering result is compared. Well-known datasets of this type are the Iris Flower dataset by Fisher (1936) or the Wine dataset by Forina et al. (1986). The former is a dataset of 150 samples that contains the features 'sepal width', 'sepal length', 'petal width' and 'petal length' of three species of Iris, namely *I. setosa*, *I. virginica* and *I. versicolor*.

Using such a dataset for evaluation can be particularly misleading because we assume that the class labels coincide with the cluster structure. In the case of the Iris Flower dataset the classes *virginica* and *versicolor* overlap and form a cluster which cannot be separated in a way that the result coincides with the labels. This sophism becomes even more apparent if one considers the origin of the dataset. Edgar Anderson collected the data by using his domain-knowledge as a botanist to cluster the three species *in-vivo*. It seems unlikely that Anderson decided on the class affiliation based on the four features mentioned above.

Evaluation using real world datasets. The clustering is applied to a real world dataset and the validity of the clustering is assessed through the domain-knowledge of the researcher. This approach suffers from the same problem as the evaluation on classification benchmarks. In the case of the Iris Flower dataset the researcher would thus conclude that the method produces a bad result since the classes of *virginica* and *versicolor* could not be correctly reproduced even though these two plants are 'obviously' different.

Evaluation using cluster validity indices Different clustering results are subjected to a cluster validity index which is a score that is computed from the dataset and the associated cluster labels. One of the best known indices is probably the Davies-Bouldin index (Davies and Bouldin, 1979) which is a sum over the ratio of intra-cluster and inter-cluster distances. In theory this index should become minimal at the optimal number of clusters for a given algorithm. More about this topic can be found in Arbelaitz et al. (2013), who have conducted an extensive comparison of validity indices.

Such indices make sense on the level of the algorithm itself but they tell little about the usefulness of the clustering. Here one tries to prove the validity of a heuristic using a heuristic. Furthermore these indices are not comparable between clustering algorithms.

3 Graph-Based Competitive Clustering

3.1 Concept

As the previous sections gave an overview about different clustering algorithms and clustering problems we will now take a closer look on the motivation behind a novel clustering algorithm which will be presented in the following sections.

- Hyperspectral Images (HSI) tend to be high-dimensional even if methods of dimensionality reduction are applied. In addition they will also tend to contain many objects as their number corresponds to the number of pixels. The computation of a clustering can thus become a very time consuming task if no graph-based method is used. This is due to the multiple recalculation of distance measures between the objects. On the other hand one cannot simply use a fully connected graph to solve this problem as the storage requirement for such a data structure would indeed be very large. This motivates to use graphs which require less memory such as the ones in figure 11. This also comes with an additional advantage: Once a graph is computed and stored it can be reused to assess different clustering results for different parameters.

- One aim of using clustering as an image segmentation tool is to find lateral features which are distinct from each other. Often only very few pixels contain information which is of interest. The rest is usually background, noise, or compounds which are expected to be present. Thus the objects which contain relevant information will only be present in small numbers. The clusters of these compounds can be very small and nested within larger clusters, which is comparable to the problem in figure 10. Many clustering algorithms tend to recognize the 'bulk' structures within a dataset and have difficulties in dealing with large differences in scale. Small variations of densities are thus overlooked. This motivates the construction of an algorithm which is invariant to cluster scales.

The proposed method is based around the following idea which is depicted in figure 16. Let two clusters be connected by a path $\mathcal{P}$ that starts from vertex A which is in the densest region of the left cluster and ends at vertex O which is in the densest region of the right cluster. If one begins a walk along the edges of $\mathcal{P}$ starting at vertex A the weights of the edges increase monotonically until one reaches vertex I. From there the following edge n will be shorter than m and the consecutive edges will decrease monotonically until one reaches vertex O.

If one would have to make a decision where to separate the two clusters along the path $\mathcal{P}$ then the edge m would be the obvious choice. A simple algorithm to solve this task would be the following:

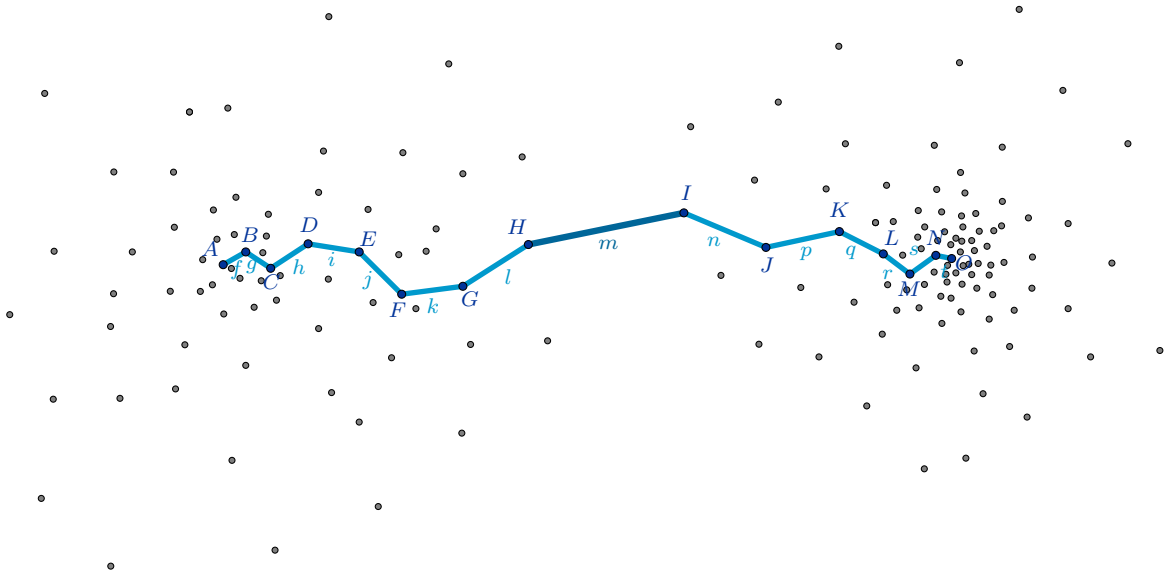


Figure 16: Fundamental concept.

1. Choose either vertex A or O as a starting point and label it with a unique cluster ID.

2. Move to the next vertex.

3. Label the vertex with the cluster ID.

4. Move to the next vertex if one of the following holds:

   Condition 1: The next vertex is still unlabeled and the weight of the previous edge is less than the weight of the next edge.

   Condition 2: The next vertex is labeled with a different cluster ID, condition 1 holds and the weight of the edge following the next vertex is greater than the weight of the edge between the current and the next vertex.

5. Repeat steps 3 and 4 until none of the conditions apply. Then repeat from step 1 with the other vertex as starting position.

This procedure will label the vertices from A to H with one ID and the vertices from O to I with a different ID. This can be easily checked by starting with vertex A. Condition 1 will let us proceed until we reach vertex I where edge n is shorter than m. If we now start the second loop with O we advance until we reach vertex J. From there condition 1 and condition 2 holds and we overwrite the label of vertex I. Now neither condition 1 or condition 2 are satisfied as m is longer than l. Thus the algorithm stops. If one starts with O in the first loop we will arrive at the same result.

The concept of this algorithm is based on the assumption that clusters can be separated by following the distance gradient along the edges until a maximum is reached. The decreasing distances following the maximum thus indicate the presence of a second cluster which should stop the advance in this direction. If the other cluster reaches the maximum edge it has the chance to reclaim the vertex that was wrongly allocated to the first cluster. In the end both clusters will enclose the maximum edge.

Even though the path is correctly separated by the algorithm this problem is rather trivial as it is only one-dimensional. Now the question arises whether this concept can be enhanced so that it can be used to cluster graphs. Then one will run into the following obstacles:

- The starting positions of the algorithm have to be determined.
- A vertex can have multiple connections that can either be bidirectional or unidirectional depending on whether the graph is undirected or directed. The algorithm will therefore have to consider multiple edges per each vertex.
- There are a multitude of paths that connect the densest areas of two clusters. Therefore the algorithm has to deal with an arbitrary number of these paths at the same time.
- In general a clustering problem will contain more than two clusters.

In the following sections these problems will be addressed to construct a clustering algorithm which will be dubbed *Graph-Based Competitive Clustering* (GBCC).

3.2 Data Structures

The proposed clustering algorithm requires a flexible data structure where internal parameters can be stored. The graph can be represented by the $(n \times n)$ weighted adjacency matrix $\mathbf{W} = (w_{ij})$, where w_{ij} corresponds to the weight of edge e_{ij} that connects vertices v_i and v_j for $i, j = 1, \ldots, n$. If v_i and v_j are not connected then $e_{ij} \notin E$. Nonetheless a value has to be assigned to w_{ij} which by convention is set to 0. From a computational point of view this can lead to a large amount of unnecessarily allocated memory as the $w_{ij} > 0$ will only sparsely populate the array that represents the matrix $\mathbf{W}$. Indeed the memory requirement for also storing the zeros would be the same as when storing a fully connected graph. As this problem often occurs in computational mathematics special data structures called *sparse matrices* have been implemented which can represent large matrices where only a small amount of values are not zero. A trivial example of such a structure is the Dictionary of Keys that maps pairs of row and column indices to the value of the element. Other common formats are the List of Lists or the Yale format.

The vertices v_i which correspond to the objects $\mathbf{x}_i$ will have to store various variables about the ongoing procedure. Depending on the used programming language they could

be implemented as an array of structs or a struct containing the arrays for each property. For illustrative purposes the v_i and $\mathbf{W}$ can be depicted conjointly like in

$$\mathbf{W} = \begin{array}{c} \\ v_1 \\ v_2 \\ \vdots \\ v_n \end{array} \begin{array}{c} \begin{array}{cccc} v_1 & v_2 & \dots & v_n \end{array} \\ \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \dots & w_{nn} \end{pmatrix} \end{array}. \tag{14}$$

If we now want to find the vertices v_j that are connected to v_i by an edge e_{ij} we simply have to find all $w_{ij} > 0$ in the i^{th} row like in

$$\begin{array}{c} \\ v_1 \\ \vdots \\ v_i \\ \vdots \\ v_n \end{array} \begin{array}{c} \begin{array}{ccccccccccc} v_1 & \dots & v_3 & \dots & v_5 & \dots & v_8 & v_9 & \dots & v_n \end{array} \\ \begin{pmatrix} & & \uparrow & & \uparrow & & \uparrow & \uparrow & & \\ & & \uparrow & & \uparrow & & \uparrow & \uparrow & & \\ \rightarrow & \rightarrow & w_{i3} & \rightarrow & w_{i5} & \rightarrow & w_{i8} & w_{i9} & & \\ & & & & & & & & & \\ & & & & & & & & & \end{pmatrix} \end{array}, \tag{15}$$

which will redirect us to the connected vertices. On the other hand if we want to find the vertices v_h that are connected to v_i by an edge e_{hi} we have to find all $w_{hi} > 0$ in the i^{th} column like in

$$\begin{array}{c} \\ v_1 \\ \vdots \\ v_4 \\ v_5 \\ \vdots \\ v_7 \\ \vdots \\ v_n \end{array} \begin{array}{c} \begin{array}{ccccc} v_1 & \dots & v_i & \dots & v_n \end{array} \\ \begin{pmatrix} & & \downarrow & & \\ & & \downarrow & & \\ \leftarrow & \leftarrow & w_{4i} & & \\ \leftarrow & \leftarrow & w_{5i} & & \\ & & \downarrow & & \\ \leftarrow & \leftarrow & w_{7i} & & \\ & & & & \\ & & & & \end{pmatrix} \end{array}. \tag{16}$$

Note that in this example v_i and v_5 are connected by a bidirectional edge as there is both a $w_{i5} > 0$ and a $w_{5i} > 0$. The above illustrations are particularly useful for understanding GBCC and should be kept in mind through the following sections.

3.3 Initialization

Before the actual clustering of the graph can begin the starting positions of the algorithm have to be determined. As depicted in figure 16 these positions should be situated at the dense areas of the dataset so that the algorithm can expand along the gradient of

increasing edge weights. Let vertices $v_j \in N_i$ be the neighborhood of vertex v_i given that $w_{ij} > 0$. If $|N_i| \geq k$ we can build the k-neighborhood of v_i by sorting the vertices v_j with respect to w_{ij} in ascending order and then forming the subset $N_{i,k} \subseteq N_i$ of the first k elements. The property vertex weight v_i^{w} is then defined as

$$v_i^{\mathrm{w}} = \begin{cases} \frac{1}{|N_{i,k}|} \sum_j w_{ij} & \forall w_{ij} : v_j \in N_{i,k} \quad \text{if } \exists N_{i,k} \\ \frac{1}{|N_i|} \sum_j w_{ij} & \forall w_{ij} : v_j \in N_i \qquad \text{if } \nexists N_{i,k} \end{cases}, \tag{17}$$

which serves as a density measure to compare different vertices against each other. If applicable the k-neighborhood will be used to calculate the average edge weight. This is necessary because the number of edges is not necessarily the same for each vertex and if the graph is constructed in a way that very distant edges are included in the neighborhood then the average will be shifted towards larger values even though the density is high around that vertex.

Using this density measure we can now define the starting positions which we will denote as *centers*. Let $N_i^q \setminus v_i$ be the q^{th}-order neighborhood of v_i which are all vertices that are reachable from v_i over $q-1$ hops. The property center v_i^{c} is then defined by

$$v_i^{\mathrm{c}} = \begin{cases} \text{true} & \text{if } v_i^{\mathrm{w}} < \min_j v_j^{\mathrm{w}} \quad \forall v_j \in N_i^q \setminus v_i \\ \text{false} & \text{else} \end{cases}. \tag{18}$$

Note that since v_i can be its own higher-order neighbor for $q > 1$ it has to be excluded from N^q or equation (18) might never be true for any vertex. The parameters k and q can now be used do determine the centers of graph $\mathcal{G}$ which serve as the initial prototypes for the evolving clusters. Therefore the number of centers is equal to the number of clusters K. The proposed method is scale invariant as the density estimation v_i^{w} is only compared within a neighborhood N_i^q. Therefore clusters can be detected independently from differences in density and extent.

Some clustering algorithms such as K-Means require the user to specify the number of clusters and in some applications the ability to predefine K might be desirable. This can be achieved by optimizing $|\hat{K} - K|$, where $\hat{K}$ is the resulting number of clusters for a given combination of k and q. The calculations of v_i^{w} and v_i^{c} are both computationally expensive for larger graphs which makes it necessary to parallelize the calculation across multiple CPU cores. If we vary k within a margin $[k_{\min}, k_{\max}]$ and q within a margin $[q_{\min}, q_{\max}]$ then the quickest approach is to first compute v_i^{w} for a $k \in [k_{\min}, k_{\max}]$ and then compute v_i^{c} for different $q \in [q_{\min}, q_{\max}]$ with respect to the current k. The winning combination should be the one that minimizes $|\hat{K} - K|$ and has the lowest k.

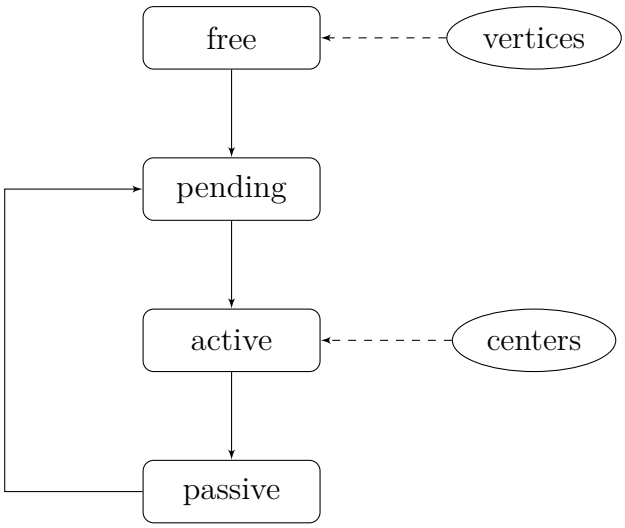


Figure 17: Flow chart of the vertex states.

3.4 Clustering

Once the centers are determined the clustering step can begin. The following rules will require a vertex to assume different states which are depicted in figure 17. The state is stored in the vertex property v_i^{s} and can take the values `free`, `pending`, `active` and `passive`. The initial state of all vertices is `free` except for the centers which are initialized as `active`. Following the concept in section 3.1 conditions 1 and 2 now have to be generalized in order to be applicable to arbitrary graphs.

Definition 1. Let v_i be an `active` vertex and v_j a vertex of its neighborhood then v_j satisfies the *gradient condition* with respect to v_i if

$$v_i^{\mathrm{a}} < w_{ij}. \tag{19}$$

Definition 2. *Allocation* is the term used to denote the process of assigning a `free` vertex to a certain cluster. If vertex v_i allocates v_j to its cluster then the properties are set to $v_j^{\mathrm{ID}} := v_i^{\mathrm{ID}}$, $v_j^{\mathrm{a}} := w_{ij}$ and $v_j^{\mathrm{s}} :=$ `pending`. A `free` vertex can be allocated only if the gradient condition holds for vertices v_i and v_j.

The vertex property v_i^{a}, which was introduced in definition 1, will be denoted as the *allocation weight*. The role that this property has in the generalization effort becomes clear if one remembers that we are no longer dealing with a one-dimensional path. Now there is no obvious 'previous' vertex to which the weight to the 'next' vertex can be compared. In conjunction definitions 1 and 2 already show how this problem can be solved. Each time a vertex is assigned to a cluster it stores the weight through which it has been allocated. This information can then be used to determine the distance gradient. On the other hand there also needs to be a way to update the allocation weight if a smaller one was found over a different path.

Definition 3. Let v_i be an `active` vertex and v_j a vertex of its neighborhood then v_j satisfies the *update condition* with respect to v_i if the gradient condition and

$$w_{ij} \leq v_j^{\mathrm{a}} \tag{20}$$

holds.

Definition 4. Let $v_i^{\mathrm{ID}} = v_j^{\mathrm{ID}}$ be two vertices of the same cluster. The term *update* then denotes the process of changing the allocation weight v_j^{a} to a smaller value if the update condition holds. Thus $v_j^{\mathrm{a}} := w_{ij}$. If v_j is a `free` or a `passive` vertex its state is set to $v_j^{\mathrm{s}} :=$ `pending`.

Finally there remains the case were a cluster has overstepped the maximum edge and the vertex has to be reassigned to a different cluster coming from the other direction.

Definition 5. Let $v_i^{\mathrm{ID}} \neq v_j^{\mathrm{ID}}$ be two vertices of different clusters then *conquering* denotes the process of reassigning vertex v_j to the cluster of v_i. Conquering can only happen if the update condition holds. The properties are then set as $v_j^{\mathrm{ID}} := v_i^{\mathrm{ID}}$ and $v_j^{\mathrm{a}} := w_{ij}$. If v_j is a `free` or a `passive` vertex its state changes to $v_j^{\mathrm{s}} :=$ `pending`.

In conjunction with the above definitions and the vertex states we can now define the GBCC-algorithm which is given as pseudocode in algorithm 1. Note that from a computational viewpoint it does not make sense to implement definitions 2, 4 and 5 separately as only definition 4 does not affect v_j^{ID}. They are thus combined in the if-clause in lines 16 to 22. We will now take a closer look on how the procedure clusters some very basic graphs to highlight its behavior under certain conditions. The star connection in figure 18a enhances the conceptual problem of figure 16 by a third cluster. The centers a, d and g are drawn as bold circles. The for-loop in lines 1 to 10 initializes all centers as `active` and with a unique cluster ID. The allocation weight is set to zero as there is no 'previous' vertex. All other vertices are initialized as `free` and their cluster ID set to zero. Line 12 in the while-loop gives us the set of the `active` vertices. For each element in that set we construct the set of the neighbors in line 14. Each neighbor is then tested whether it can be added to the cluster. This is a trivial case for vertices b, e and h since the allocation weight of the centers are all zero. The new members remain in the `pending` state until all `active` vertices have processed their neighbors. Finally the `active` vertices are set to `passive` and the `pending` vertices become `active`. The while-loop repeats in the same manner and vertices c, f and i are added to their respective clusters.

The next while-loop will reveal how conquering is implemented. Without loss of generality we assume that vertex c is the first to test its neighbors. Like in the previous loop it will add j to its cluster. If f is the next vertex in line it will test vertex j. This time j is no longer a `free` vertex but the update condition holds since $w_{ef} < w_{fj} < w_{cj}$. j is therefore conquered by the pink cluster and its allocation weight is set to w_{fj}. The

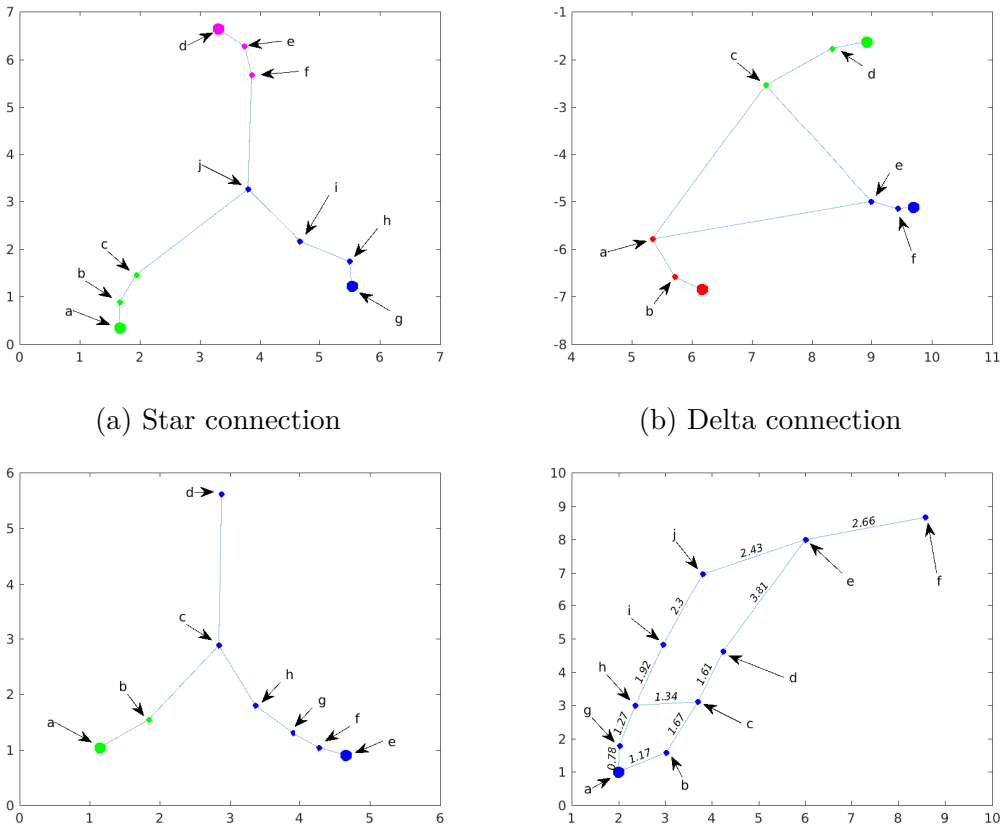
Algorithm 1: Pseudocode of GBCC

```
1  for i := 1 to n do
2      if v_i^c = true then
3          v_i^s := active;
4          v_i^ID := getUniqueClusterID();
5          v_i^a := 0;
6      else
7          v_i^s := free;
8          v_i^ID := 0;
9      end
10 end
11 while ∃v_i^s = active do
12     A := {v_i ∈ V | v_i^s = active};
13     foreach v_i ∈ A do
14         N := {v_j ∈ V | w_ij > 0};
15         foreach v_j ∈ N do
16             if updateCondition(v_i, v_j) ∨ (v_j^s = free ∧ gradientCondition(v_i, v_j))
                then
17                 v_j^ID := v_i^ID;
18                 v_j^a := w_ij;
19                 if v_j^s = free ∨ passive then
20                     v_j^s := pending;
21                 end
22             end
23         end
24         v_i^s := passive;
25     end
26     P := {v_i ∈ V | v_i^s = pending};
27     foreach v_i ∈ P do
28         v_i^s := active;
29     end
30 end
```

final outcome is that i conquers j again as $w_{hi} < w_{ij} < w_{fj}$. The last repetition of the while-loop finds j as the only `active` vertex but since there are no `free` vertices left and the update condition does not hold for any of its neighbors the while-loop finally ends.

Figure 18b is an example where three clusters meet in a triangle. On the first glance this scenario seems to be more complex but if one compares the neighborhood relations of vertices a, c and e with the vertex j of figure 18a it becomes clear that there is only little difference between the two examples.

Figures 18c and d highlight why (19) is formulated with a '$<$'-sign and (20) with a '$\leq$'-sign. If one applies algorithm 1 to figure 18c the green cluster of center a is the first to arrive at the junction c. From there it allocates vertex d. Whether h is first allocated by the green or the blue cluster is ambiguous and depends on the indexing of the vertices. Nonetheless h will belong to the blue cluster after the next while-loop. In this particular situation the green cluster holds vertices c and d. The next loop brings the conquering of c by h. Now the green cluster is split into two disconnected sets which obviously would be an undesirable result. As d was allocated by c its allocation weight is w_{cd}. Because of the '$\leq$'-sign in (20) the update condition is thus satisfied since $w_{hc} < w_{cd} \leq w_{cd}$ and c finally conquers d.



(a) Star connection

(b) Delta connection

(c) Override problem

(d) Update problem

Figure 18: Problems that proof the correct generalization for arbitrary graphs.

A similar problem occurs in figure 18d which takes a closer look on what happens within a cluster. Starting from center a the gradient condition allows the procedure to move forward until vertices h and c are reached. c cannot allocate d because $w_{bc} > w_{cd}$. h on the other hand can update c since $w_{gh} < w_{hc} < w_{bc}$ which sets the allocation weight of c to w_{hc}. In the next while-loop it can then allocate d. If d is the first to allocate e we have the same problem again and j has to update e so that f can be allocated.

To give an example how GBCC behaves on a kNN graph a clustering result of the Mixing Clusters dataset is given in figure 19. The underlying graph is an undirected 13-nearest neighbor graph. The red edges denote the allocation weights. Seen from the center of a cluster each vertex at the cluster border can thus be connected by a path along the allocation weights so that the distances increase monotonically. In between two border vertices of different clusters there might be an edge that connects both vertices with a weight that is greater than the weights of both border vertices. In that case we have found a path $\mathcal{P}$ as illustrated in the conceptual drawing in figure 16.

To conclude this section we still need to clarify what the role of the vertex state `pending` is in the given pseudocode example. One might wonder why newly allocated and updated vertices cannot be set directly to `active`. The introduction of the state `pending` serves as a means to ensure an implementation-independent formulation of the procedure. Without this intermediate state vertices would be set directly to `active` which would make line 12 ambiguous. In that case it would not be clear whether the set A would have to be supplemented by the new `active` vertices. For each new `active` vertex we would have to test its neighbors and then set it back to `passive`. This process could happen multiple times as there could be many vertices in the neighborhood that would update the vertex under consideration. Therefore the `pending` state introduces a period in which the smallest allocation weight can be determined without triggering the retesting of the neighbors. On the other hand we have the if-clause in line 19 that ensures that if an `active` vertex is among the neighbors its state is not set to `pending` since it is still in the queue for testing its own neighbors and will do so anyway.

3.5 Implementation and Testing

For the purpose of initial testing GBCC was implemented in MATLAB R2016b (The MathWorks, Inc.; mathworks.com). The pseudocode in algorithm 1 adheres to the original MATLAB code and should be applicable for other programming languages as well. The experiments which will be presented in the following section were conducted on an Apple Mac Mini Server (late 2012) running MATLAB on Arch Linux. The results of K-Means and HCA were computed using Epina ImageLab (Epina GmbH; imaglab.at) on Microsoft Windows using the same machine in dual-booting mode.

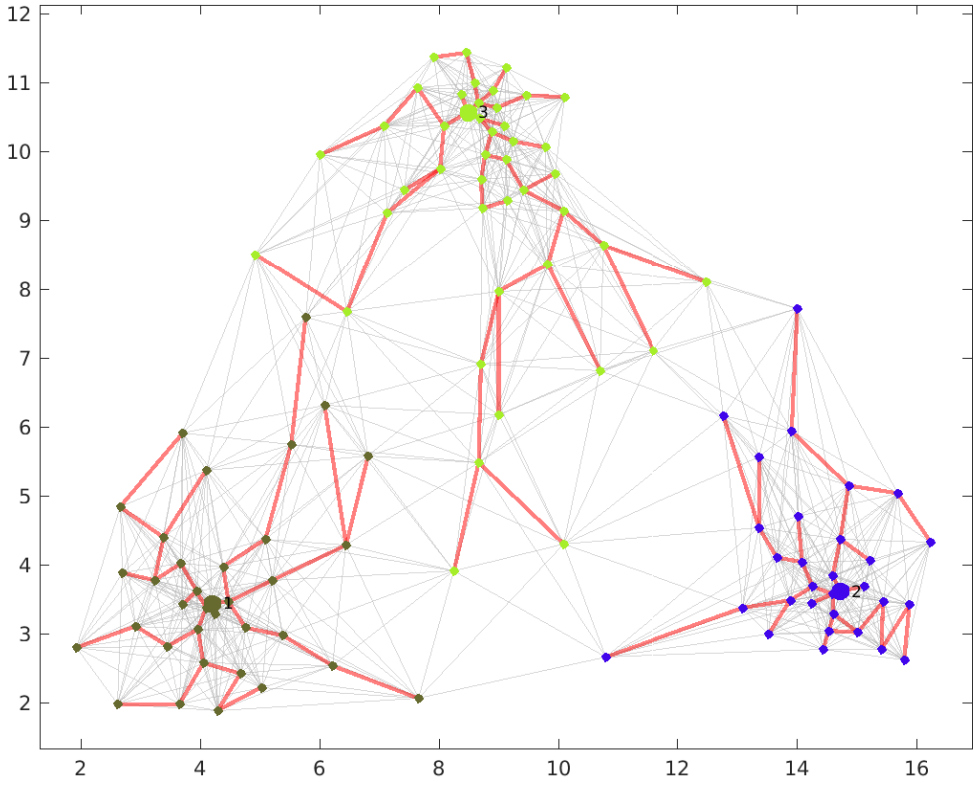


Figure 19: Clustering result using GBCC on the Mixing Clusters dataset.

4 Experimental

As we saw in section 2.6 the evaluation of clustering algorithms and clustering results is not a trivial task and sometimes 'evaluated' results tell little about the general applicability of the clustering algorithm. The biggest problem that we face here is that our perception is limited to three dimensions and as clustering algorithms usually operate in high-dimensional spaces the 'true' structure is hidden from our vision.

In their paper Von Luxburg et al. (2012) severely criticized the usual methods of cluster validation as insufficient and often completely misleading. The author shares the same opinion in that regard, though nonetheless some sort of validation, be it 'insufficient' or not, is absolutely necessary to get a notion about the characteristics of a clustering algorithm and how the results relate to other methods. The author argues that in the end clusters and classes are an invention of humankind which makes the human being the measure of all things in that context. We will therefore start our evaluation and comparisons on datasets which can be easily understood by humans.

4.1 2D Artificial Datasets

4.1.1 Introductory Examples

In the following we will compare clustering results of GBCC with other methods such as K-Means, DBSCAN, Spectral Clustering and HCA. The first two experiments have been conducted on the Mixing Clusters dataset which was initially introduced in figure 9 and the Introductory dataset of figure 2. These have the advantage that they contain only a few clusters which are also easy to perceive. In the view of the author $K = 3$ clusters constitute a reasonable assumption for both datasets and as a means to make the results comparable we will force the algorithms to produce the desired number of clusters. However one should keep in mind that this is not a fair comparison as algorithms such as DBSCAN and Single Linkage will have difficulties with this restriction. The results of the following experiments can be found on pages 67-77.

The experiment on the Mixing Clusters dataset is depicted in figure 21. The unclustered data is given in figure 21a. The results of GBCC are depicted in figures 21b and c, where the former also shows the underlying undirected 13-nearest neighbor graph as gray lines and the allocation weights as red lines. Next to the latter is the K-Means result in 21d. In order to force DBSCAN (figure 21e) to produce three clusters the parameters MinPts and Eps had to be adjusted manually. In this particular case the algorithm also produces an additional noise cluster which is colored in black. DBSCAN cannot be prevented from doing so since increasing the size of the data clusters causes them to merge before the noise cluster disappears. Spectral clustering in figure 21f was conducted using the normalized Laplacian $\mathbf{L}_{\text{rw}}$ of the same undirected 13-nearest neighbor graph. The experiment concludes with the results of HCA using Ward's Method in 21g and Single Linkage in 21h.

Despite some minor differences the results in figures 21c-g look quite similar. In fact K-Means and Spectral Clustering produce the same result. Contrary to that Single Linkage shows one large cluster that contains all objects except for two singleton clusters. This method tends to form many clusters of great size differences and therefore has difficulties dealing with problems which are not well-separated.

The clustering of the Introductory dataset is depicted in figure 22 which was conducted using the same course of action. The methods DBSCAN and Single Linkage provide the expected result since both work well on connected clusters. Now the methods K-Means and Ward's Method clearly show that their variance-oriented clustering approach has difficulties with the elongated clusters. At first glance the result of GBCC in figure 22c surprises as the green cluster is not fully separated from the pink cluster. This phenomenon can be explained by taking a closer look at the graph in figure 22b. As GBCC was forced to optimize the number of clusters to $K = 3$ the center of the green cluster, which is labeled with the number '2', was placed in the far left corner. Since the growth of the cluster can only happen along increasing edge weights the allocation weights become

larger and larger if one follows the red lines from the left to the right. At a certain point the allocation weights are so 'ill-conditioned' that the pink cluster can advance in this territory even though it has to cross the gap. Such behavior is definitely undesirable but cannot be prevented if the number of clusters is set too low. In fact connected clusters are rather problematic for GBCC as the distant tips are prone to conquering by a different cluster.

On the other hand this problem also depends on the underlying graph structure. Figure 23 depicts four clusterings of the Introductory dataset where different graphs have been used as input for the algorithm. The Delaunay Triangulation in figures 23a-b clearly shows that it is ill-suited for GBCC. Since this graph is constructed by forming triangles which do not contain any other point within their circumcircle the neighborhood of a vertex becomes very constrained. This makes the growth of the clusters difficult because there are far less vertices that satisfy the gradient condition. In this particular case the cluster originating form center '2' can only partially allocate the underlying connected cluster and the remaining vertices keep their 0-initialized cluster ID. Furthermore the other two clusters both have vertices right inside of them that could not be allocated.

The kNN graph is not as limited in that regard which can be seen in figures 23c-h. In these examples a directed kNN graph was used to show that making the graph undirected has little advantage and only increases memory load. The 15-NN graph in figures 23c-d still shows that the clusters have difficulties in allocating all vertices. With the 40-NN graph in 23e-f we arrive at the same result as in figure 22c, where only a small number of vertices is wrongly allocated. However as mentioned above this problem should not be tackled by increasing the number of neighbors. A better solution is to allow a greater number of clusters K as in figures 23g-h. GBCC is after all designed to move along a distance gradient and relies on the competition between neighboring clusters.

4.1.2 Gradient-Separable Problems

Since GBCC is designed to solely work on distance gradients it is a well suited algorithm to solve gradient-separable problems. An exemplary dataset of this kind is the Russian Dolls dataset which was introduced in figure 10. As the clusters vary greatly in size the clustering result of this dataset is separated into figures 24 and 25 where the former is a global view and the latter a zoomed-in view of the dataset. The rectangle in figure 24a indicates the zooming range of figure 25.

The underlying graph for this experiment is an undirected 15-nearest neighbor graph depicted in figures 24b and 25b. The global view of the GBCC result in 24c already shows the three clusters and the zoomed-in view in 25c reveals that the smallest cluster is correctly separated from its larger neighbor. As could be expected the results of K-Means, Ward's Method and Single Linkage could not produce a satisfying result. Again the variance-oriented approach of K-Means and Ward's Method simply overlook the small variations in density and do not find the third cluster. Single Linkage merges the smallest

cluster into its larger neighbor after the hierarchy level of $K = 31$ clusters. The result of DBSCAN in figures 24e and 25e was produced by manually setting the parameters MinPts and Eps. In this dataset it was not possible to force the number of clusters to $K = 3$ so that the smallest cluster remains separated. The black cluster is the unclustered data which DBSCAN defines as noise. So even though the result appears to show a satisfying result it is not the case here. Spectral Clustering on the other hand identified all three clusters with only minor errors with respect to the smallest cluster.

4.1.3 Combined Problems

Some strong points and some limits of the tested algorithms were already revealed by the above experiments though so far the problems have been kept quite simple. We will now expose the clustering algorithms to two multi-challenge datasets. Both problems implement clusters of great size differences with both large and small complex structures. By intention the datasets have been designed in a way that a 'true' clustering cannot be determined even by human beings as cluster borders are rather ambiguous.

The clustering of the Complex dataset is depicted in figures 26 and 27 with the rectangle in 26a indicating the data range of the zoomed-in view. In this dataset we find a small complex structure in the lower left corner which is embedded within a large noise-like structure. As mentioned above a fair comparison between clustering algorithms cannot be obtained by forcing an equal number of clusters. DBSCAN and especially Single Linkage have problems with these restriction as the experiments on the Mixing Clusters and the Russian Dolls dataset confirmed. Therefore the goal of this clustering was to produce a more or less equal amount of detail in the small complex structure without giving any attention to the resulting number of clusters. This approach makes sense if one considers the fact that in a real-world clustering the information that we want to find might be hidden within larger structures. In that case the results show the difficulty of selecting a number of clusters K so that the results are comparable on a certain level of scale.

The underlying graph for GBCC is a directed 30-nearest neighbor graph. The desired number of clusters for the optimization $|\hat{K} - K|$ was set to $\hat{K} = 20$ using an interval of $[k_{\min}, k_{\max}] = [2, 30]$ and $[q_{\min}, q_{\max}] = [1, 7]$. The detected number of clusters amounts to $K = 13$. As can be seen by taking a close look at figure 26b only two centers are placed into the noise-like structure whereas the remaining centers accumulate in the complex smaller structure. By comparing the global view to the zoomed-in view of figure 27b one can see how GBCC reacts to noise. The gray cluster labeled with '1' bypasses its violet neighbor '2' and grows into the sparsely populated space below. Since there is no other center in that region which center '2' has to compete with, it can grow freely until most of the objects below the complex structure are allocated.

As expected the number of clusters has to be much higher for the variance-oriented algorithms K-Means and Ward's Method which was set for both to $K = 21$ and results

in almost the same number of clusters within the complex structure. In this dataset the noise-rejection capability of DBSCAN clearly comes in handy if one interprets the large structure as noise. Here its parameters were set to MinPts = 2 and Eps = 1.4. As Eps imposes a global constraint on the inter-object distances it is not possible to cluster the complex structure and the noise-like structure at the same time. In comparison to GBCC Spectral Clustering treats the noise differently. Where GBCC detects the two centers '12' and '13' Spectral Clustering combines the entire region into one large cluster. Single Linkage was set to a number of $K = 133$ clusters to produce a comparable result. As most of the clusters are singletons it will be rather difficult to distinguish relevant information from noise in a similar higher-dimensional clustering problem.

The next clustering was conducted on the Chemical dataset which is depicted in figures 28, 29, 30 and 31. This artificial structure was designed with the purpose to mimic certain characteristics of HSIs. As spectral information often shows high correlation between neighboring variables the individual chemical compounds tend to form lobe-like clusters. If the intrinsic dimensionality of a dataset is not too high these features can be made visible by projection methods such as principal component analysis (PCA). The structures seen by the author in various of these plots provided the inspiration for this dataset.

As the scale differences of the clusters are again very high the results are partitioned into three zooming ranges which are denoted by the rectangles in figure 28a. The numbers 1, 2 and 3 correspond to the figures 29, 30 and 31.

The clustering was conducted on a directed 15-nearest neighbor graph. The settings for the optimization $|\hat{K} - K|$ were set to $\hat{K} = 20$ using an interval of $[k_{\min}, k_{\max}] = [2, 15]$ and $[q_{\min}, q_{\max}] = [1, 7]$. The resulting number of clusters here amounts to $K = 16$. In this case the goal of the clustering was to achieve a more balanced result between large and small features since the size differences are more gradual. Similar to the results of the Complex dataset the number of clusters for the variance-oriented methods K-Means and Ward's Method are much higher. Single Linkage again suffers from the varying densities in data distribution and produces a large number of singleton clusters. The parameters of DBSCAN were set to MinPts = 3 and Eps = 2. In this case the global constraint imposed by Eps becomes a problem because the lobe-shaped clusters that extend to the left of the data structure could constitute relevant chemical information.

With respect to density-separability there are two interesting data ranges to be found in figures 29b and 30b. Clusters '13' and '12' are both dense clusters that are embedded in an otherwise sparse region. These two situations are similar to the density-separable problem as defined by Zahn (1971). If one remembers that GBCC allocates the vertices by moving along increasing edge weights this result should not be possible because a path that extends from the dense region into the sparse region will allow a growth in this direction and therefore parts of the sparse clusters would be allocated. The explanation for this phenomenon lies in the underlying 15-nearest neighbor graph. kNN graphs have the special property that if the number of neighbors k is sufficiently small the k-neighborhood

is limited to other members of the cluster and does not extend beyond its borders. In the case of density-separated problems the vertices of the sparse regions will therefore be connected to the dense cluster but not vice versa. This means that the dense cluster allocates its vertices without any knowledge of the sparse cluster whereas the vertices of the sparse cluster cannot conquer the dense regions as their allocation weights are too large. It is noteworthy that also Spectral Clustering partially recognized cluster '12'.

Another example where both GBCC and Spectral Clustering produced comparable results is figure 31. Here both algorithms identified the two microscopic clusters which are located inside rectangle 3. If one also considers figures 25c and 25f as well as 27c and 27f with their respective number of clusters K the results indicate that GBCC and Spectral Clustering behave similarly under problems where clusters have great size and density differences.

4.2 Hyperspectral Images

From the viewpoint of a human being HSIs are huge compared to other types of images. Even though their spatial resolution tends to be much smaller than common digital photos the hints and clues one is trying to find to confirm a hypothesis are scattered across a multitude of layers. Despite this difficulties HSIs offer a great advantage over other high-dimensional data when comparing multivariate algorithms. As each object also has a corresponding pixel coordinate the labels obtained by a clustering algorithm can be used to construct a segmented image where each color corresponds to a certain class or cluster. As the human perception has evolved to quickly recognize shapes by color contrast comparing different cluster images thus becomes a much easier task than comparing tabular results.

For the following experiments two different HSIs have been selected. The Coffee Beans dataset (Parrag et al., 2014) is an UV/VIS/NIR image of a mixture of roasted and green coffee beans as well as stones. From the viewpoint of process engineering similar data might be used to control a sorting machine in order to separate the three constituents through a classification engine. The Microplastic dataset (Löder, 2017) is an IR image of a mixture of plastic particles and a variety of organic materials. This dataset has been selected for its complex chemical composition and the occurrence of mixture effects between neighboring pixels. The data was obtained by measuring a filter cake that contains various particles that occur in rivers and lakes.

Both datasets represent challenging problems. However before we can go into the details of the individual experiments there are some issues to consider when dealing with HSIs which arise from the high dimensionality.

4.2.1 The Curse of Dimensionality

Consider an HSI that is made up of d layers and has a lateral resolution of $w \times h$ pixels. This corresponds to a d-dimensional data space which is filled with $w \times h$ objects. A typical property of such high-dimensional spaces is that they are literally empty of data. This phenomenon can be explained by the following example: Let $\mathcal{X}$ be a sample of n objects that are evenly distributed along a one-dimensional data space in the unit interval $[0, 1]$. In order to get an interval which contains only 10% of our data we simply have to choose $[0, a]$ where $a = 0.1$. If we move into a two-dimensional data space the unit interval becomes a unit rectangle. Assuming that we still have the same number of n evenly distributed objects we now have to choose $a = \sqrt{0.1}$ which in this case is the edge length of a smaller rectangle within the unit space. For a three-dimensional space the edge length of the corresponding cube would be $a = \sqrt[3]{0.1}$ and for a d-dimensional space $a = \sqrt[d]{0.1}$. Thus a converges towards 1 with increasing d.

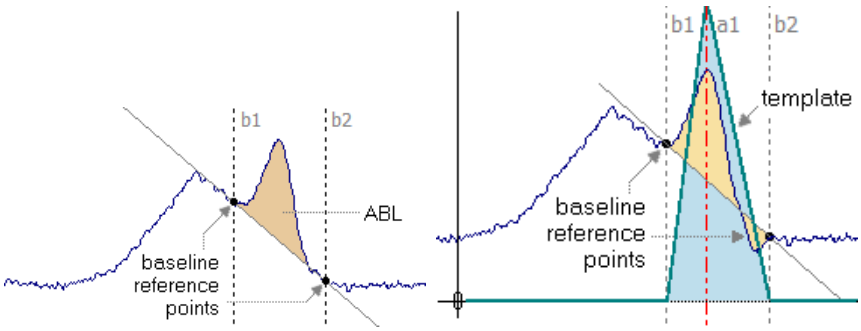
This phenomenon is called the *Curse of Dimensionality* (Bellman, 1961) which indeed can be very problematic for machine learning methods such as clustering and classification. The sparsity of the data also has effects on the kNN graph which we rely upon for GBCC and Spectral Clustering. Radovanović et al. (2010) found that with increasing dimensionality the *indegree* distribution becomes skewed to the right. The indegree of a vertex v_i is the number of edges e_{ji}. Respectively, the outdegree is defined as the number of edges e_{ij} which for kNN is obviously k. A skewness to the right therefore means that the graph tends to contain hubs where many vertices point towards a few which in turn only have k neighbors. This phenomenon is thus referred to as *hubness*.

As GBCC has undergone only little testing so far it is difficult to predict how hubness affects the clustering of the graph. However it can be assumed that certain vertices will thus play a strategic role for the growth of the clusters as the conqueror of such a hub will also allocate all other vertices that lie along paths of increasing distances similar to the example in figure 18c.

4.2.2 Spectral Descriptors

One way to cope with the effects of the Curse of Dimensionality is to reduce the dimensionality of the dataset $\mathcal{X}$. An obvious way to do this is to simply resample the spectra to a lower resolution. A drawback of this approach is that all variables are treated equally regardless of whether they contain relevant information or not. Furthermore the resampling factor would have to be very high because a reduction from 1000 to 500 variables still produces a high-dimensional space.

Another approach is to use mathematical functions that combine multiple spectral features into one descriptive variable. With respect to chemical data these functions are called spectral descriptors (Lohninger and Ofner, 2014). A selection of two descriptors is given in figures 20a and 20b. The former is the area between a peak and a baseline.



(a) ABL-Descriptor (b) TC-Descriptor

Figure 20: Spectral Descriptors are a means to reduce the dimensionality as well as the noise of the data. If chosen correctly the features of a complex chemical spectrum can be represented by only a handful of these descriptors. (Epina ImageLab Documentation, 2018)

By using this spectral descriptor all features between the baseline reference points b1 and b2 are represented by the resulting area. The latter is computed from the Pearson product-moment correlation coefficient of the template triangle which is defined by the points b1 and b2 as well as the peak position a1. The resulting value is then multiplied by the area between b1 and b2.

There are many other possibilities to construct spectral descriptors. Some examples are:

- the raw intensity of a certain feature
- the ratio of two intensities
- the ratio of two peak areas
- the centroid of the spectral line between two boundaries

Spectral descriptors can also be applied to the first and second derivative of the spectrum. By applying different types to key distinctive features one can thus construct a set of descriptors that can be used to replace the raw data. Despite their ability to greatly reduce the dimensionality they also tend to reduce noise and improve the structure of the data. This impacts the performance of many chemometric techniques in a positive way since the data now contains much more specific information. A drawback of spectral descriptors is that the design of a descriptive set for a complex dataset is time-intensive and there is also the danger that certain spectral features might be overlooked in that process.

4.2.3 Overview of the Experiments

The experiments on the Coffee Beans and Microplastic datasets were both conducted with different goals and objectives in mind. The original Coffee Beans dataset was spatially

downsampled by a factor of two to arrive at a resolution of 160×180 pixels. The spectral resolution was kept at the original 155 sampling points. The goal behind this clustering was to study the effects of high-dimensionality. Before these test were made it was not even clear whether GBCC would produce any usable result under such demanding circumstances. The optimization parameter $\hat{K}$ was set to 7 assuming 3 clusters for roasted beans, green beans and stones as well as 4 clusters for other compounds.

The goal behind the clustering of the Microplastic datasets was to detect polymer particles of polystyrene (PS), polyacrylonitrile (PAN), polypropylene (PP) and polymethylmethacrylate (PMMA) which are embedded in a complex matrix of bio-organic and inorganic particles. As these microplastic particles only occur in small abundances and are difficult to distinguish from other compounds in the 609-dimensional data space a set of 23 spectral descriptors was designed to specifically detect these four polymers. Table 4 lists the used types and their respective parameters. Because of the high spatial resolution of 276×295 pixels an additional smaller range of 150×150 pixels was extracted from the dataset for an independent test. For both ranges $\hat{K} = 12$ was assumed for the four polymers as well as 8 other organic and inorganic substances.

From the clusterings of the two-dimensional datasets which were presented in section 4.1 and other tests on smaller downsampled HSIs it was already known that the optimization procedure for determining the centers is the time-critical part. In order to reduce computational time the range of the order of neighborhood was set to $[q_{\min}, q_{\max}] = [1, 3]$. The range for computing the vertex weight was kept at $[k_{\min}, k_{\max}] = [2, k]$, where k denotes the number of neighbors of the underlying kNN graph.

With respect to the graph types it was assumed that k should be kept low so that the center optimization as well as the clustering step does not have to deal with too many neighbors. It was therefore set to $k = 15$ and $k = 30$ respectively. The full range Microplastic dataset only featured the 30-NN graph due to the increased computational cost. In addition both directed and undirected kNN graphs were computed for each dataset. Regarding the choice of k one should keep in mind that this results in very sparsely connected graphs. For images of a comparable spatial resolution this means that a vertex is connected to only 0.1% to 0.01% of the data. Contrary to that the 15-nearest neighbor graph of the Chemical dataset connected each vertex to 2.9% of the data. On the one hand this might seriously impact the center optimization and therefore the number of clusters. On the other hand it is not clear whether a 'good' result can be attributed to the clustering capabilities of GBCC or is mainly caused by the clustering tendencies of kNN graphs.

The experiments were conducted by first clustering the directed kNN graphs which took approximately 10 hours in total to finish. From that it was concluded that the clustering of the undirected kNN graphs should be feasible within an acceptable amount of time. The second run then took about 20 hours to complete. As expected the center optimization required about 80% of the total computational time and thus forms the

Table 4: Spectral Descriptors for the Microplastic dataset. A description of each type of descriptor can be found in the Epina ImageLab Documentaion (2018).

Name	Type	Range	Baseline	#Nb.	Derivative	Comment
DC0001	RBL	1450/1373	1489...1315	1	0	PAN/PMMA
DC0002	TC	1739	1824...1674	1	0	PMMA
DC0003	TC	2245	2287...2191	1	0	PAN
DC0004	TC	1732	1782...1658	1	0	PMMA
DC0005	TC	2947	3105...2804	1	0	PMMA/PP
DC0006	RBL	1450/1388	1527...1334	1	0	PMMA/PAN/PP
DC0007	TC	2923	2970...2869	1	0	PS
DC0008	TC	3028	3128...2981	1	0	PS
DC0009	TC	1601	1624...1562	1	0	PS
DC0010	RBL	2958/2877	3024...2754	1	0	PP/PMMA
DC0011	ABL	3012...2754		1	0	PP
DC0012	TC	1458	1493...1396	1	0	PP/PMMA
DC0013	TC	1377	1392...1350	1	0	PP/PAN
DC0014	TC	1751	1790...1728	1	1	PMMA/PAN
DC0015	TCI	1701	1655...1728	1	1	PMMA
DC0016	TC	1493	1473...1520	1	1	PMMA
DC0017	TC	2252	2276...2245	1	1	PAN
DC0018	TCI	2233	2245...2210	1	1	PAN
DC0019	TC	1462	1481...1450	1	1	PAN/PS
DC0020	TC	1612	1635...1597	1	1	PS
DC0021	TC	1462	1481...1439	1	1	PS/PAN
DC0022	TCI	2947	2958...2935	1	1	PP
DC0023	TC	1385	1400...1373	1	1	PP/PS/PAN

bottleneck of the clustering method.

Table 5 summarizes the results of the clusterings of all three datasets which highlights some interesting relations between the selected parameters. Microplastic A here refers to the extracted range and Microplastic B to the full image. If one compares the yielded number of clusters K to the corresponding graph type a larger degree of connectivity seems to positively impact the center optimization. The rightmost column gives the percentage of vertices that were not assigned to a cluster but remained in the `free` state. Apparently this amount also seems to decrease with increased connectivity.

Only the directed 30-NN graph of the Microplastic B dataset yielded the desired number of clusters. This might be due to the restricted range of the order of neighborhood q. Here it seems that a maximum of 3 is simply too low for images of that size. As the number of clusters of most of the experiments is much too high to be visualized as a segmented image only a selection of the results will be presented in the following section.

Table 5: Summary of the Experiments on Hyperspectral Images

Dataset	Resolution	n	d	Graph	$\hat{K}$	K	`free` [%]
Coffee Beans	(160×180)	28800	155	15-NN	7	79	13.2
				30-NN		33	8.3
				undir. 15-NN		52	0.2
				undir. 30-NN		26	0
Microplastic A	(150×150)	22500	23	15-NN	12	125	13.8
				30-NN		41	4.7
				undir. 15-NN		41	3.7
				undir. 30-NN		12	0.4
Microplastic B	(276×295)	81420	23	30-NN	12	259	5.6
				undir. 30-NN		38	4.1

4.2.4 Selected Findings

The figures that are referenced in this section can be found on pages 78-86. The clustering of the undirected 30-NN graph of the Coffee Beans dataset is summarized in figure 32. The segmented image obtained by GBCC is depicted in figure 32a which is compared to the K-Means algorithm in 32b. Regarding the comparability of the two images one should keep in mind that K-Means was set to yield 8 clusters which is less than a third of what GBCC yielded. Furthermore it is difficult for the human perception to distinguish between 26 different color shades. For example only three of the five lime green clusters actually represent roasted coffee beans.

Nonetheless the two cluster images reveal some notable aspects of GBCC in comparison with K-Means. A rather striking spatial feature is the vertical stripe that was caused by damaged pixels in the spectroscopic detector. GBCC segmented the stripe into two clusters whereas K-Means did not detect it at all. Spectral samples of the clusters at spots 1 and 3 are given in figure 32c and 32e were the spike in the affected layer is clearly visible.

Another important difference between the two algorithms is that GBCC separated the roasted and green coffee beans as well as the stones whereas K-Means combined the stones with the roasted beans into one cluster. Here spot 2 represents a spectrum of a stone, spot 4 a roasted and spot 6 a green coffee bean. The respective figures are 32d, 32f and 32h. This result can be interpreted in two different ways. On the one hand it could mean that K-Means has more difficulties operating in the 155-dimensional data space than GBCC. On the other hand it could mean that for K-Means $K = 8$ is simply not sufficiently high enough to also separate the stones from the roasted beans.

A more in-depth discussion will now follow on the results of the Microplastic datasets. Figure 33 summarizes the clustering of the undirected 30-NN graph of the Microplastic A dataset which yielded the desired 12 clusters. To get a better impression about what

structures the clustering algorithms have to deal with a PCA plot of the first two principal components is given in 33c. One should keep in mind that PCA tends to reduce noise in the data and that the situation in the 23-dimensional data space is much more complex than it appears in the plot. There are five regions that can be distinguished in that figure. Each lobe that protrudes from the dense cloud at the origin represents one of the four polymers PS, PAN, PP and PMMA. Because of the use of the aforementioned spectral descriptors all other bio-organic and inorganic particles form the dense mass of data points at the origin.

The clustering results of GBCC and K-Means are depicted in figures 33a and 33b. Spots 1 to 4 reference spectral samples of the polymers PP, PS, PMMA and PAN which can be found in figures 33g-j. If one disregards the exact cluster affiliation of each pixel an overall assessment of the two algorithms shows that both detected the same spatial features. However if one takes a closer look at the detected particles there are some differences to be noted. For one GBCC assigned clusters 3, 5 and 9 to pixels which mostly contain one of the polymers where K-Means assigned clusters 6, 7, 8, 9, 10, 11 and 12. All remaining clusters were assigned to structures of the background. This discrepancy can be explained if one compares this clustering problem to the Chemical dataset in figure 28. As the center optimization is designed to be invariant to size differences of clusters it gives less weight to the lobes than K-Means and thus also focuses on the denser structures closer to the origin.

From these circumstances there now arises the question whether GBCC actually identified the polymers by placing centers into the lobes. Figures 33l-w which depict spectral samples of the centers give a surprising but yet not unexpected result. Of the 12 centers only centers 3 and 9 represent the polymer PAN whereas all other polymers were not detected. The remaining centers are thus situated in the dense cloud close to the origin. Why only PAN was detected can be explained by comparing the data density within each of the four lobes in the PCA plot. Here PAN clearly forms the densest cluster followed by PMMA, PS and finally PP.

To get a better impression how the density of the polymer clusters relate to the cloud at the origin three of GBCC's internal parameters have been visualized in figures 33d-f. The vertex weight was computed from the average distance to the 30-nearest neighbors. This illustration confirms that the PAN spectra form a very dense cluster in relation to the other polymers if one compares spots 1 to 4. Especially PP which is indicated by spot 1 seems to be very sparse. Figure 33d also shows various other dark spots representing dense clusters at the origin which explains why GBCC did not detect the other polymers. Similar conclusions can be drawn from the visualizations in figures 33e and 33f. The former illustrates the final allocation weight of each vertex after GBCC finished its clustering step. Here the allocation weights of PAN and of many of the background pixels are much lower than those of the other polymers. The latter is the summation of all allocation weights if one follows an allocation path from the center to a specific vertex. As the spots

1 to 3 are barely recognizable this too indicates the sparsity of these clusters.

If only PAN was recognized by the center optimization one might ask why the other polymers were clustered at all. The answer can be found by closely comparing the spectral samples in figures 33l-w to the cluster image in figure 33a. Centers 3 and 9 are both PAN spectra but because of the lack of centers in the two neighboring lobes they were able to advance into the domain of the PS and PP spectra. Center 5 which mainly clustered the PMMA lobe also competed against center 3 in the PP lobe. This explains why the particle indicated by spot 1 is made up of clusters 3 and 5 and why the PAN fibers which are mainly covered by cluster 3 also show labels of cluster 9.

From a mere spectroscopic point of view one might argue that GBCC failed to produce a valid clustering result for the Microplastic A dataset. However in this case one makes the same wrong assumptions that have been discussed in section 2.6 regarding the evaluation using benchmark datasets. Even though all centers except 3 and 9 can be considered as noise or 'useless' information they could also constitute an invaluable discovery if one is following a different objective. Furthermore the center optimization is designed to detect dense clusters of varying sizes and using $\hat{K} = 12$ was simply too strict under these circumstances. This can be shown by assessing the centers of the other experiments. The clustering result of the Microplastic B dataset is given in figure 35a along with a selection of the 38 centers given in figures 35b-h. Here the center optimization detected PMMA in figures 35b and 35h, PAN in figures 35c, 35d, 35e and 35f, and PS in figure 35g.

Finally there remains cluster 11 in figure 33a which in the author's view shows the true potential of GBCC as a scale-invariant clustering method. This cluster is an artifact of the measurement process where a fluctuation in the flushing gas caused the appearance of a prominent negative CO_2 peak which can be seen in figure 33v. This feature was detected because of spectral descriptor DC0009 which is designed to detect a certain peak at 1601 cm^{-1} in the PS spectrum. As the pixels in the fluctuation zone also exhibit a peak in that range which seems to correlate with the fluctuation in the gas current the descriptor causes them to form a dense cluster inside of the cloud at the origin. Even though this cluster contains many pixels and thus covers a large portion of the clustered image it remains hidden if one tries to find it using PCA or K-Means. This cluster is also visible in figure 34 which shows the result of the directed 30-nearest neighbor graph and figure 35a.

4.3 Conclusions

Visual assessment of the experiments on the 2D artificial datasets revealed how GBCC relates to other established clustering algorithms. It seems that of the five other methods Spectral Clustering produces clusters of comparable shapes and like GBCC it can deal with great size and density differences. Furthermore the comparisons also revealed that both K-Means and Ward's Method as well as DBSCAN and Single Linkage seem to

behave in a similar way. In the latter case the similarity of Single Linkage to DBSCAN only becomes evident if one considers the singleton clusters as noise or unclustered data.

The experiments on HSIs showed that GBCC can also cope with structures in high-dimensional data spaces. Some of the conclusions drawn from these results indicate a behavior similar to the results obtained from the 2D artificial datasets. Regarding the underlying kNN graphs that were used to represent the neighborhood relations in the spectral data it could be shown that an increase in connectivity positively affects the center optimization as well as the clustering step of GBCC.

Even though the experiments show some characteristics of GBCC they should be considered as preliminary steps rather than a full evaluation. The leap from 2D artificial datasets to high-dimensional HSIs revealed some algorithmic as well as computational issues that will be discussed in the following:

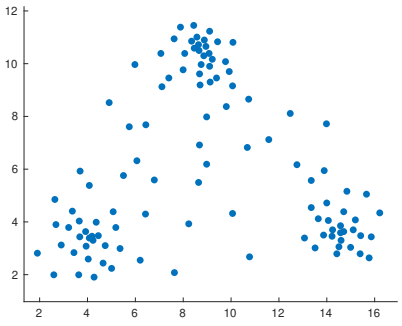
In the current implementation GBCC can be considered a computationally intensive algorithm. Especially the center optimization is a hindrance for its application as an exploratory data analysis tool. Here it should be noted that even though the optimization is parallelized it still uses a 'brute force' approach to optimize the number of centers to the desired number of clusters. A binary search for the optimal parameters might be an easier way to achieve the same result in far less time.

The sparse matrix used to represent the graph also plays a key role when it comes to speeding up the method. Even though this data structure can store large graphs using only little memory it is nonetheless expensive in computational terms. This is due to the way values at certain indices are stored. If one wants to access the value w_{ij} the address in memory is determined by a search for that indices which can be a time-consuming process. Contrary to that an array-based data structure enables us to calculate the memory address directly. The use of MATLAB's sparse matrix implementation which is based on the compressed sparse column format enabled GBCC to run on any graph that can be described through a weighted adjacency matrix. Now that it has become clear that GBCC performs well on kNN graphs the center optimization as well as the clustering step might speed up significantly by using a simpler and less general data structure that is optimized for these types of graphs. In the light of these considerations it might be interesting to study whether increasing the connectivity in high-dimensional data might not be easier achieved by increasing the number of neighbors k rather than by making the kNN graph undirected. This way the data structure needed to describe a kNN graph could be kept quite simple and memory addresses could be calculated directly.

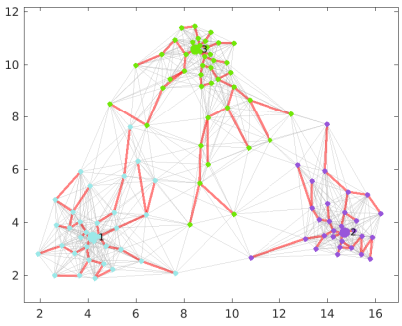
Regarding the center determination the experiments revealed that GBCC is heavily dependent on the positioning of these starting points. The clustering step needs the presence of other competing clusters to correctly determine boundaries and a lack of such can allow it to move against distance gradients. Therefore further energy should be directed toward the development of the optimization procedure and investigating different approaches how dense areas can be detected in high-dimensional data spaces. One idea

that was inspired by the findings of Radovanović et al. (2010) regarding hubness of kNN graphs is to use the indegree of a vertex instead of the vertex weight to determine suitable center candidates.

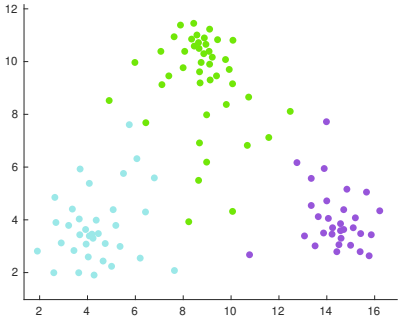
In the view of the author this thesis shows that conducting a clustering on an unknown dataset requires substantial knowledge about the characteristic of the applied method in order to correctly interpret the found clusters. If one remembers the different clustering problems that were discussed in section 2.4 a usable result might be obtained easier by comparing results of two or more antithetic methods rather than by selecting the algorithm which is deemed as the best for the assumed underlying data structure. In that regard GBCC proofs to be a valuable contribution because of its scale-invariant nature and its ability to solve gradient-separable problems.



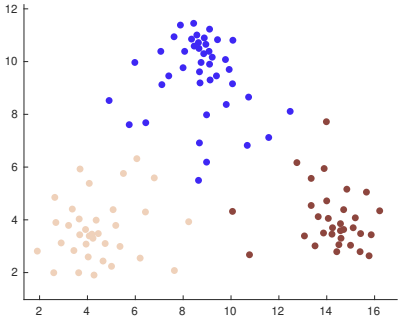
(a) unclustered



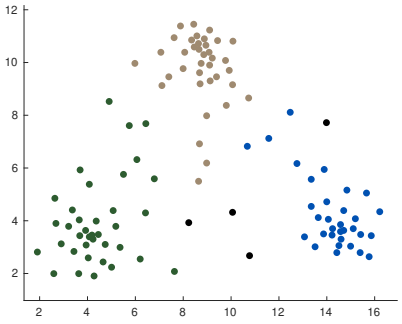
(b) GBCC with graph



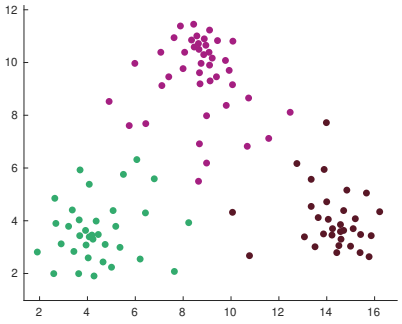
(c) GBCC



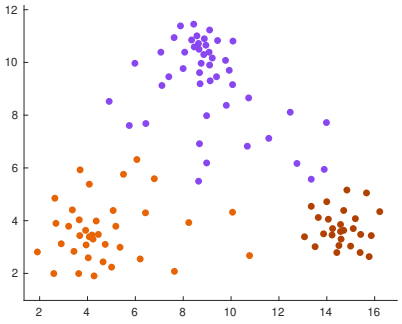
(d) K-Means



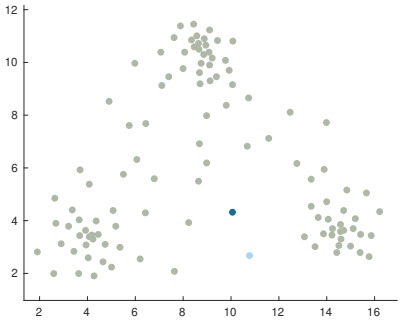
(e) DBSCAN



(f) Spectral Clustering

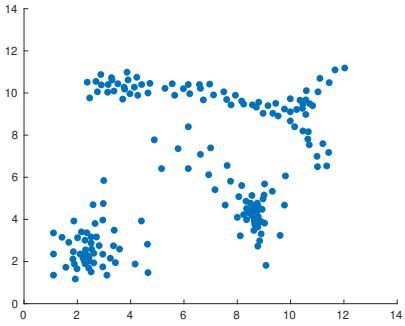


(g) Ward's method

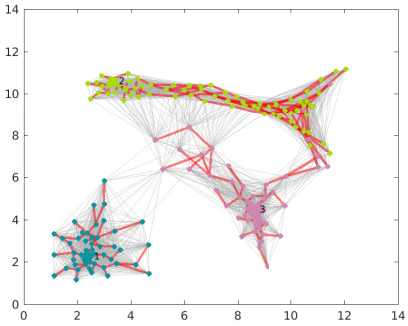


(h) Single Linkage

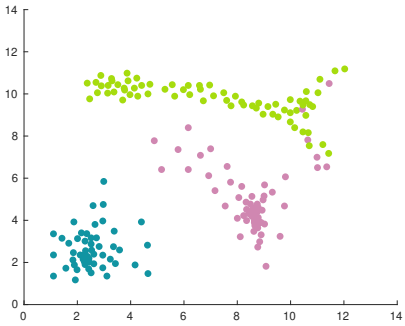
Figure 21: Clustering of the Mixing Clusters dataset forcing $K = 3$ clusters.



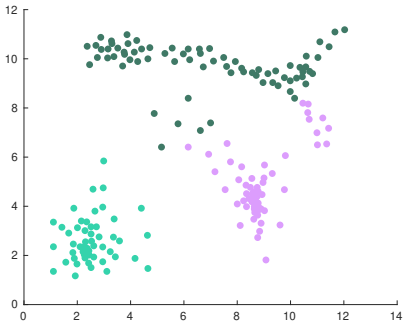
(a) unclustered



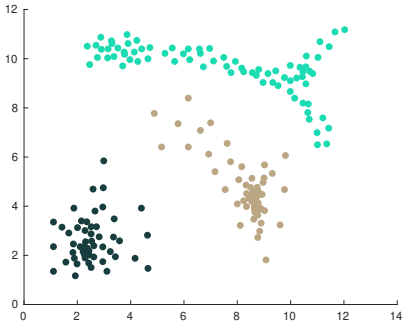
(b) GBCC with graph



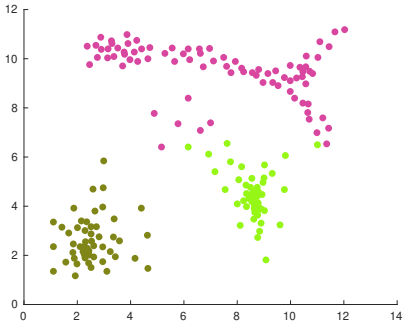
(c) GBCC



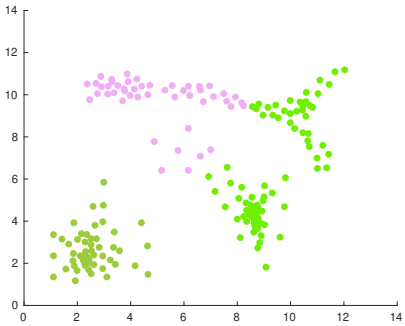
(d) K-Means



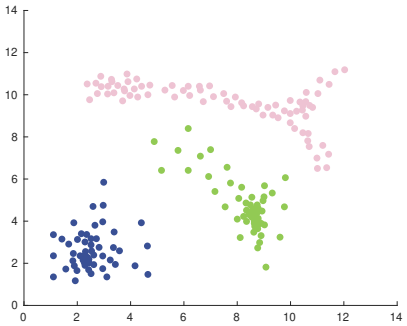
(e) DBSCAN



(f) Spectral Clustering

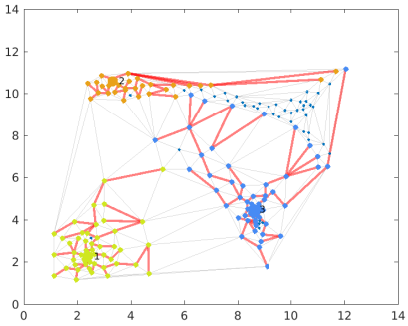


(g) Ward's method

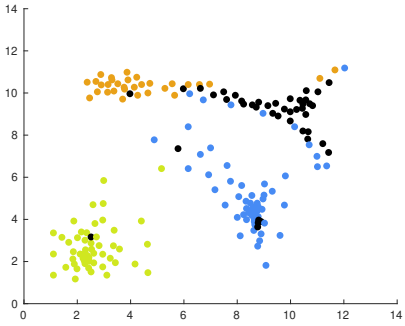


(h) Single Linkage

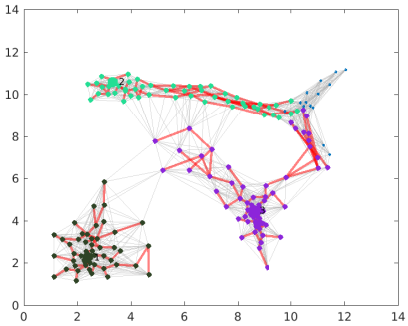
Figure 22: Clustering of the Introductory dataset forcing $K = 3$ clusters.



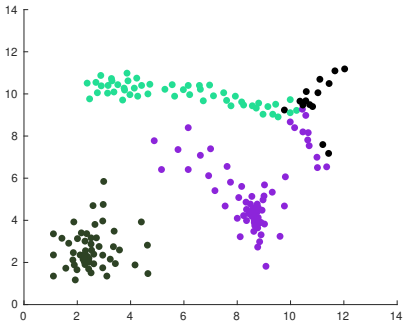
(a) DT($\mathcal{X}$)



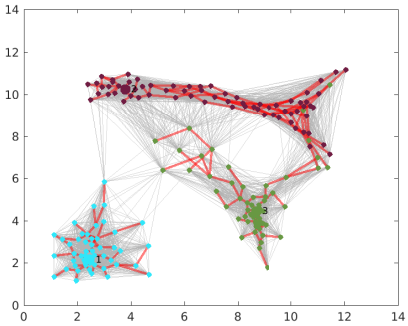
(b) GBCC of DT($\mathcal{X}$)



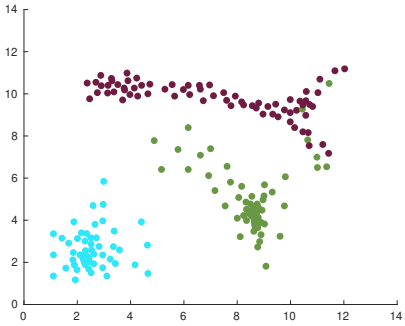
(c) 15-NN graph



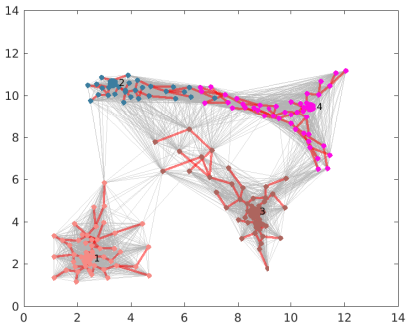
(d) GBCC of 15-NN



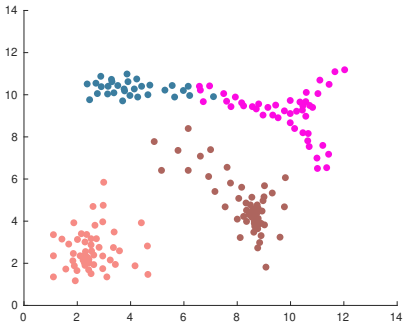
(e) 40-NN graph



(f) GBCC of 40-NN

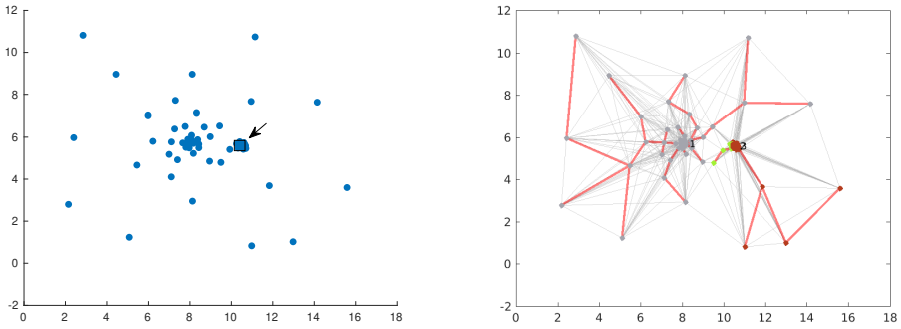


(g) 40-NN forcing $K = 4$



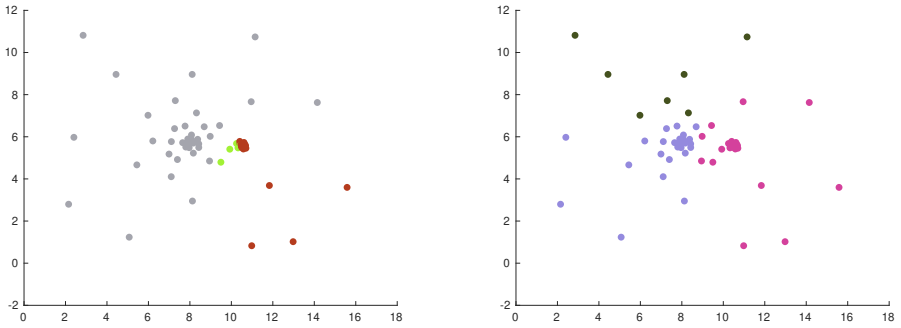
(h) GBCC of forcing $K = 4$

Figure 23: Graph dependency of GBCC.



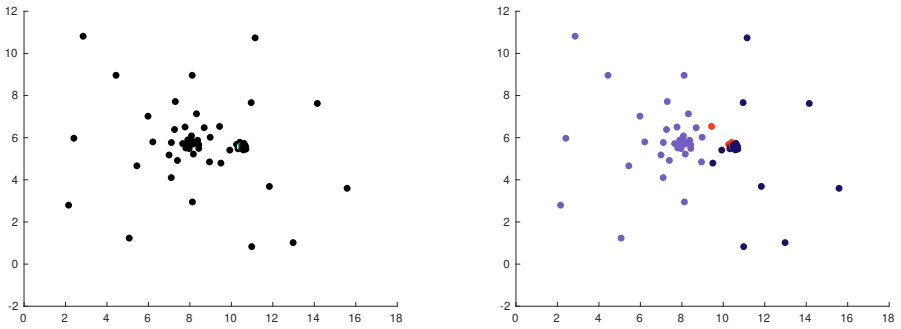
(a) unclustered

(b) GBCC with graph



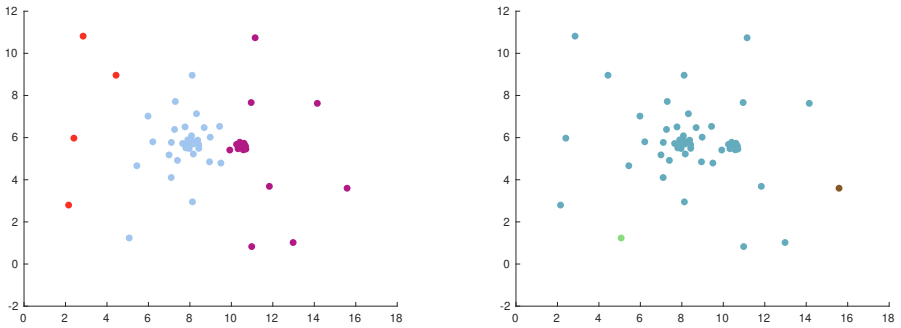
(c) GBCC

(d) K-Means



(e) DBSCAN

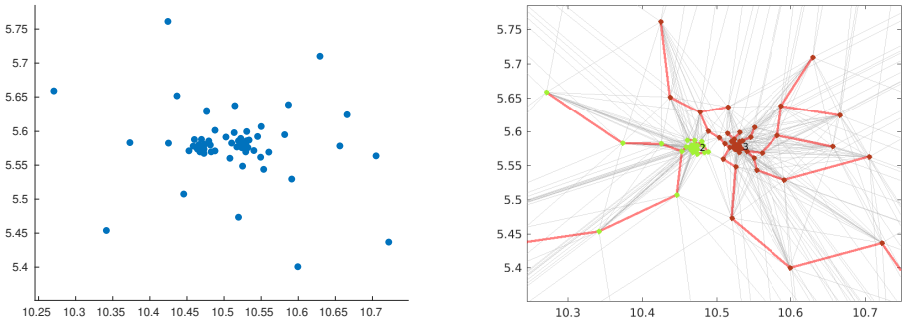
(f) Spectral Clustering



(g) Ward's method

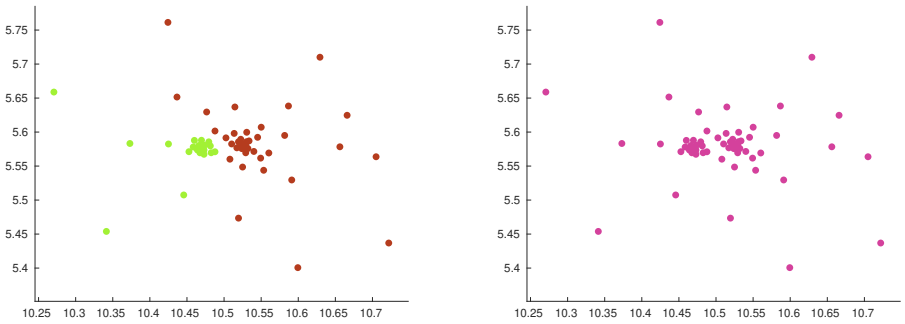
(h) Single Linkage

Figure 24: Clustering of the Russian Dolls dataset forcing $K = 3$ clusters. The range denoted by the rectangle in (a) corresponds to the zoomed-in view in figure 25.



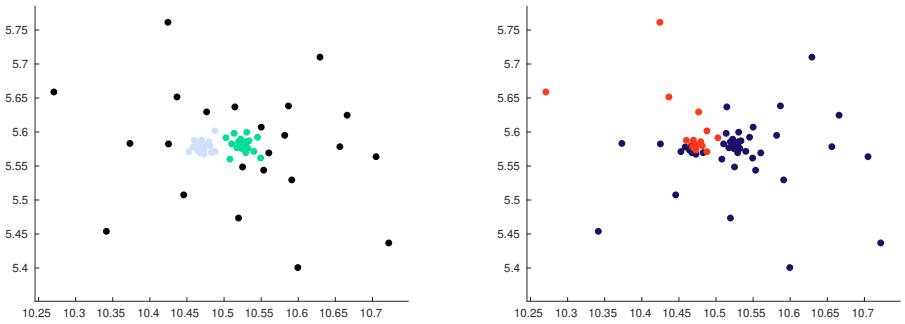
(a) unclustered

(b) GBCC with graph



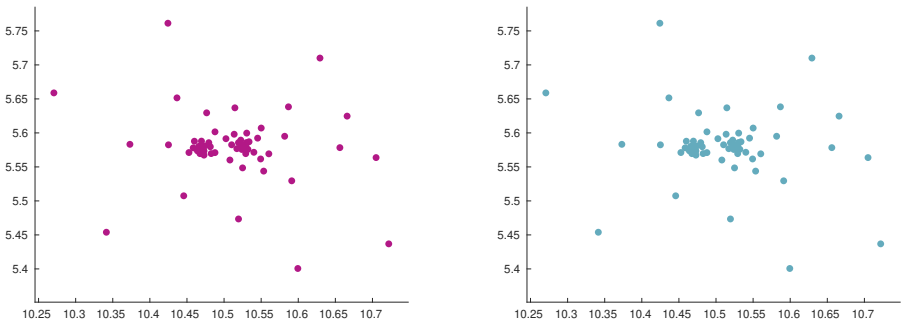
(c) GBCC

(d) K-Means



(e) DBSCAN

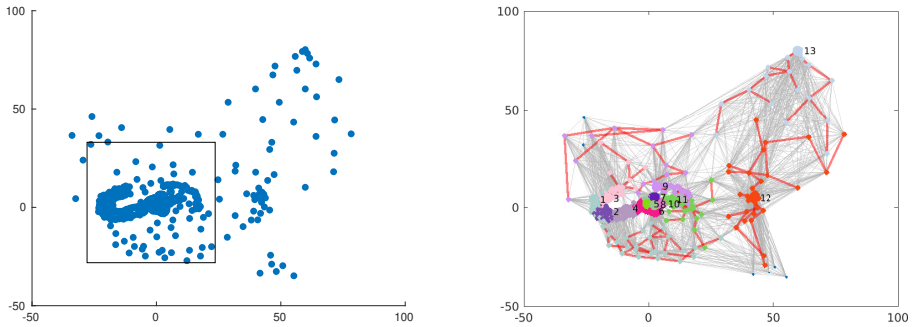
(f) Spectral Clustering



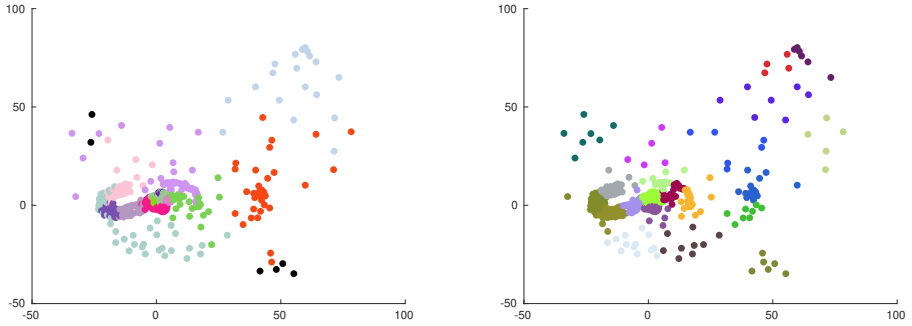
(g) Ward's method

(h) Single Linkage

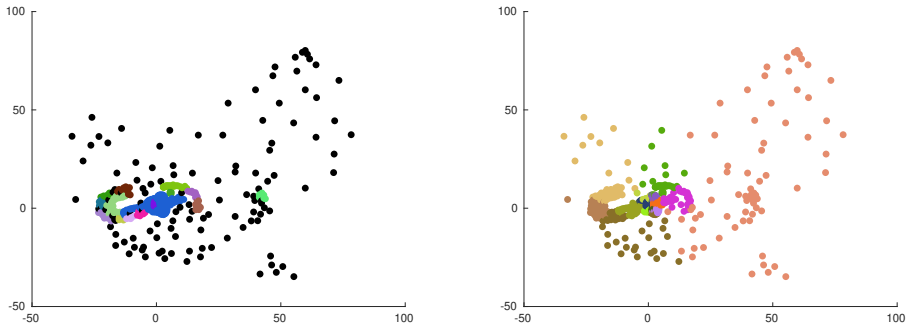
Figure 25: Zoomed-in Clustering of the Russian Dolls dataset forcing $K = 3$ clusters. The depicted range corresponds to the zooming rectangle in figure 24a.



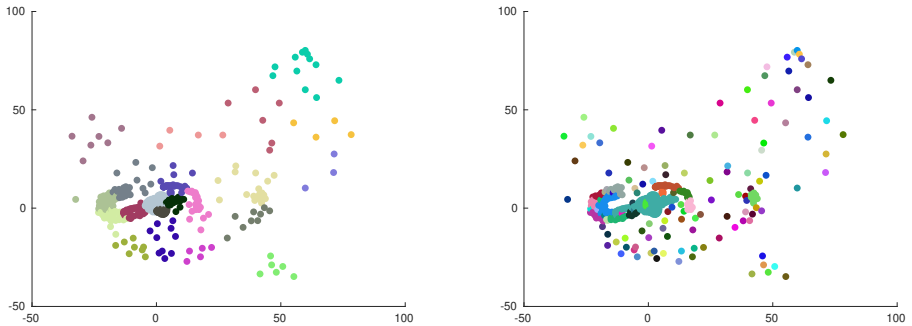
(a) unclustered (b) GBCC with graph



(c) GBCC yielding $K = 13$ (d) K-Means yielding $K = 21$

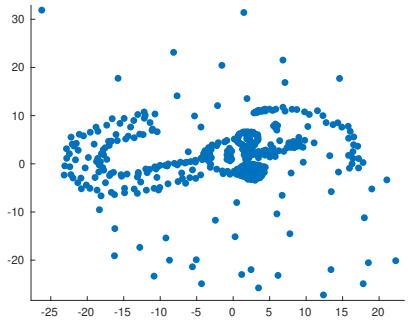


(e) DBSCAN yielding $K = 16$ (f) Spectral Clustering yielding $K = 14$

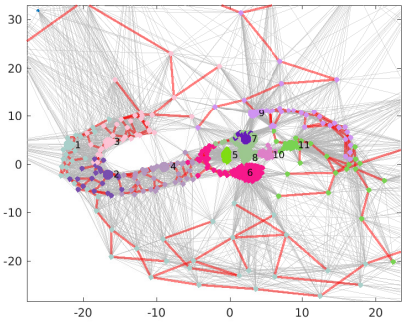


(g) Ward's Method yielding $K = 21$ (h) Single Linkage yielding $K = 133$

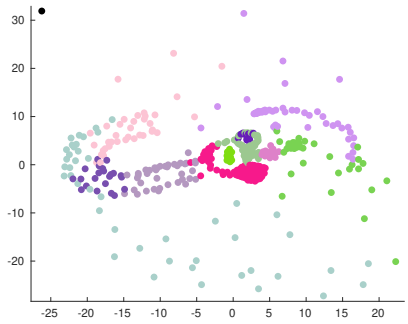
Figure 26: Clustering of the Complex dataset. The range denoted by the rectangle in (a) corresponds to the zoomed-in view in figure 27. In this clustering example the goal was to achieve an approximately equal amount of clusters in the complex structure in the lower left corner of the dataset.



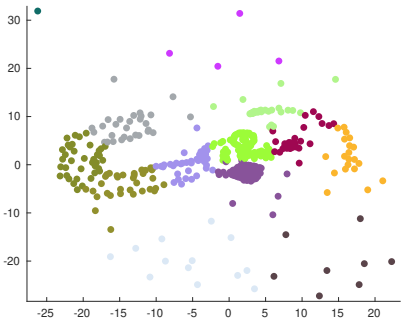
(a) unclustered



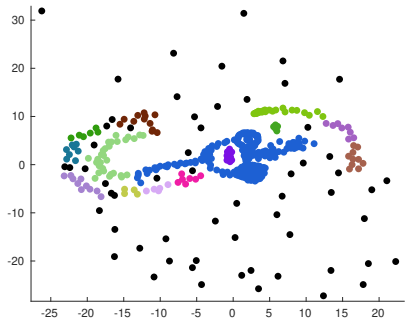
(b) GBCC with graph



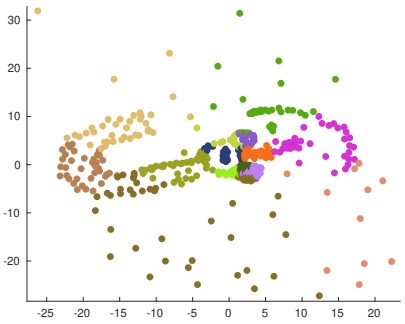
(c) GBCC yielding $K = 13$



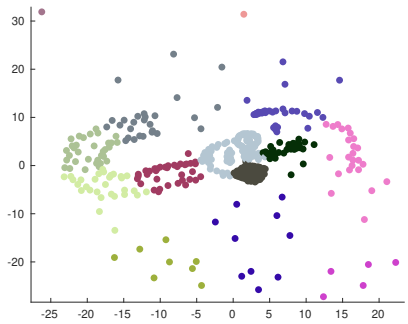
(d) K-Means yielding $K = 21$



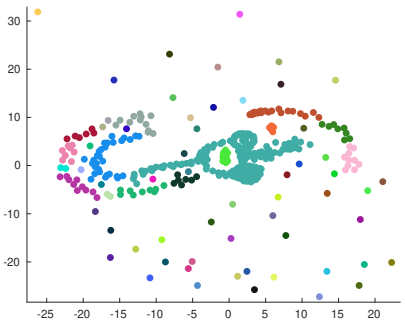
(e) DBSCAN yielding $K = 16$



(f) Spectral Clustering yielding $K = 14$

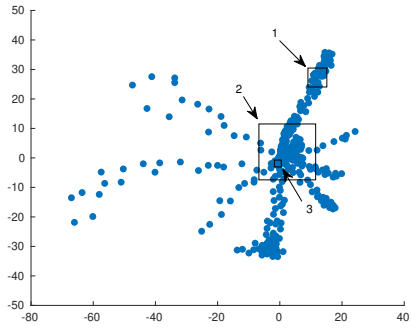


(g) Ward's Method yielding $K = 21$

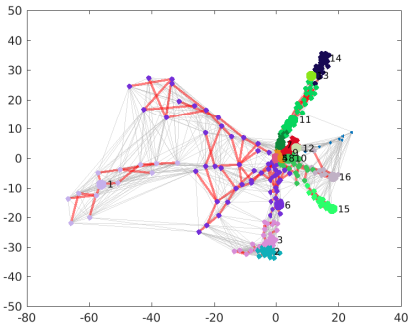


(h) Single Linkage yielding $K = 133$

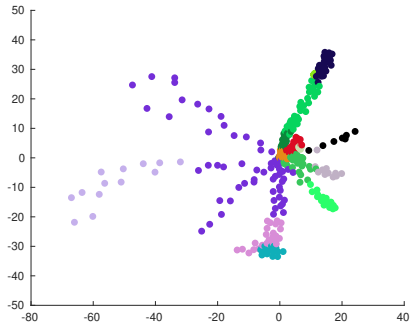
Figure 27: Zoomed-in Clustering of the Complex dataset. The depicted range corresponds to the zooming rectangle in figure 26a.



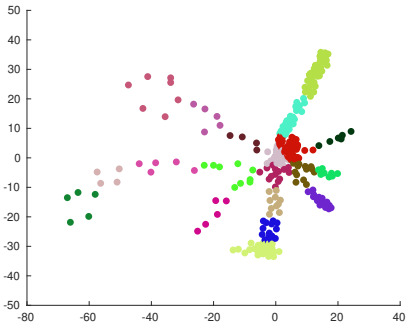
(a) unclustered



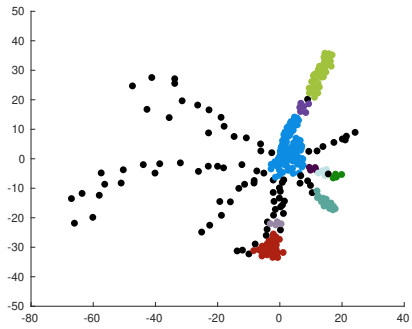
(b) GBCC with graph



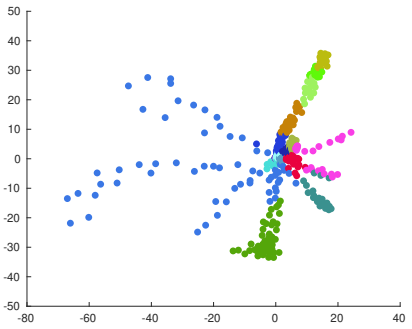
(c) GBCC yielding $K = 16$



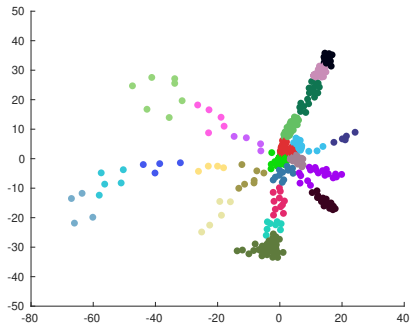
(d) K-Means yielding $K = 20$



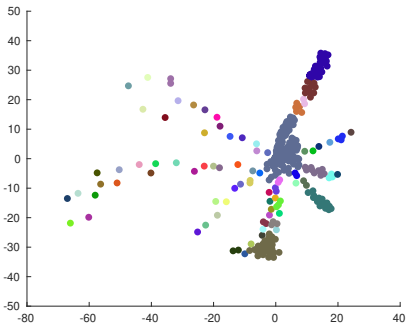
(e) DBSCAN yielding $K = 10$



(f) Spectral Clustering yielding $K = 16$

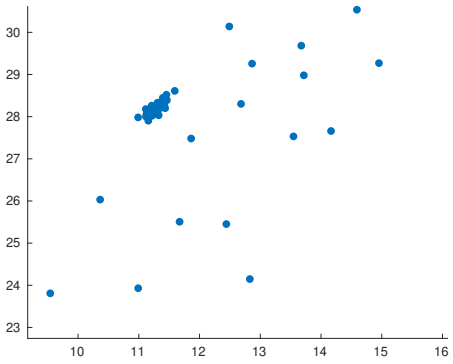


(g) Ward's Method yielding $K = 24$

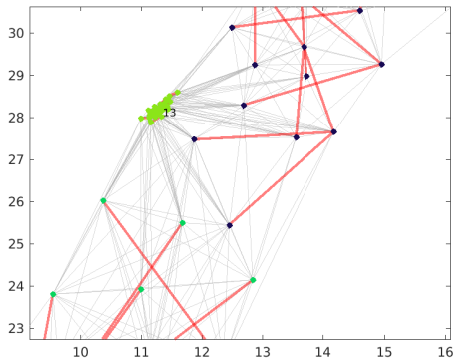


(h) Single Linkage yielding $K = 80$

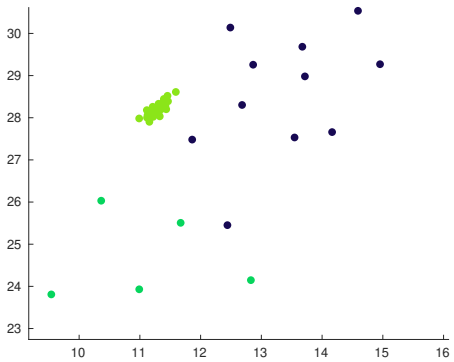
Figure 28: Clustering of the Chemical dataset. The ranges denoted by the rectangles 1, 2 and 3 in (a) correspond to the zoomed-in views in figures 29, 30 and 31.



(a) unclustered

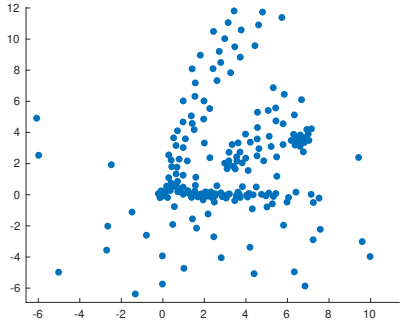


(b) GBCC with graph

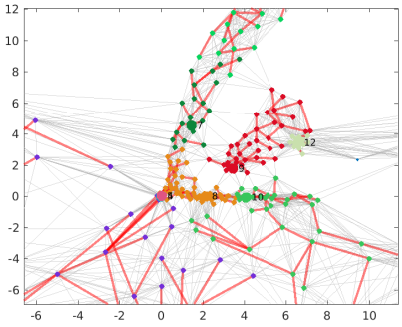


(c) GBCC

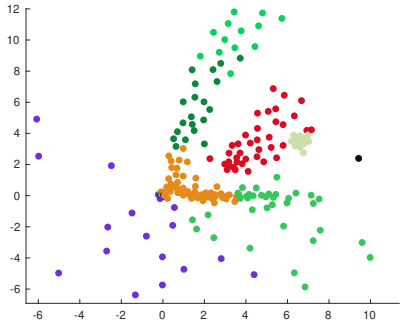
Figure 29: Zoom 1 of the Chemical dataset. This data range corresponds to rectangle 1 in figure 28a. GBCC was the only algorithm that could detect this cluster.



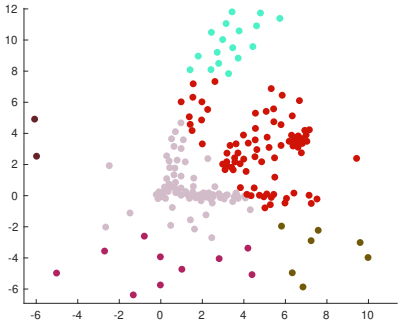
(a) unclustered



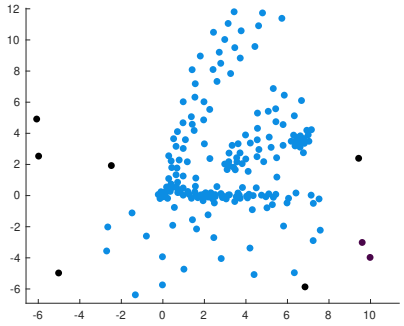
(b) GBCC with graph



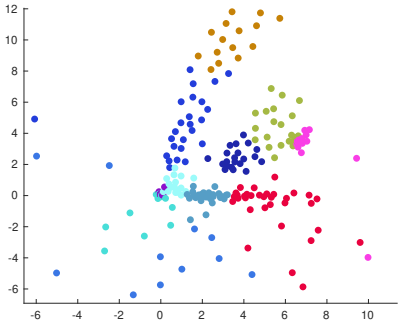
(c) GBCC yielding $K = 16$



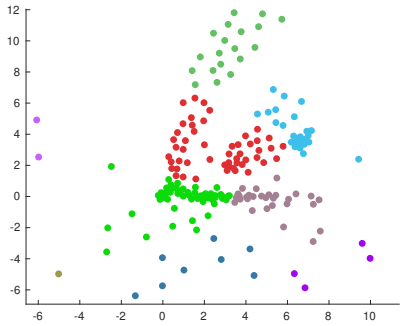
(d) K-Means yielding $K = 20$



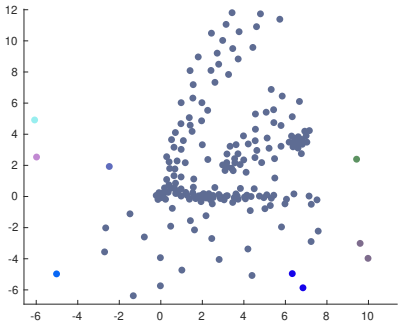
(e) DBSCAN yielding $K = 10$



(f) Spectral Clustering yielding $K = 16$

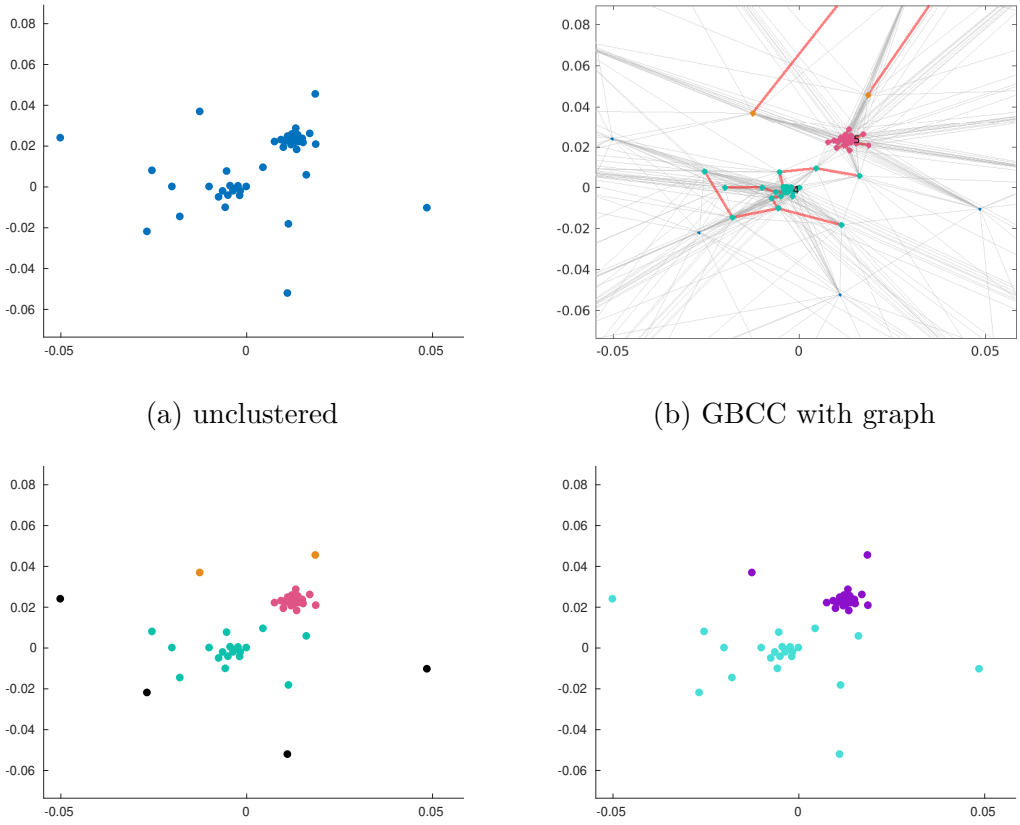


(g) Ward's Method yielding $K = 24$



(h) Single Linkage yielding $K = 80$

Figure 30: Zoom 2 the Chemical dataset. This data range corresponds to rectangle 2 in figure 28a.



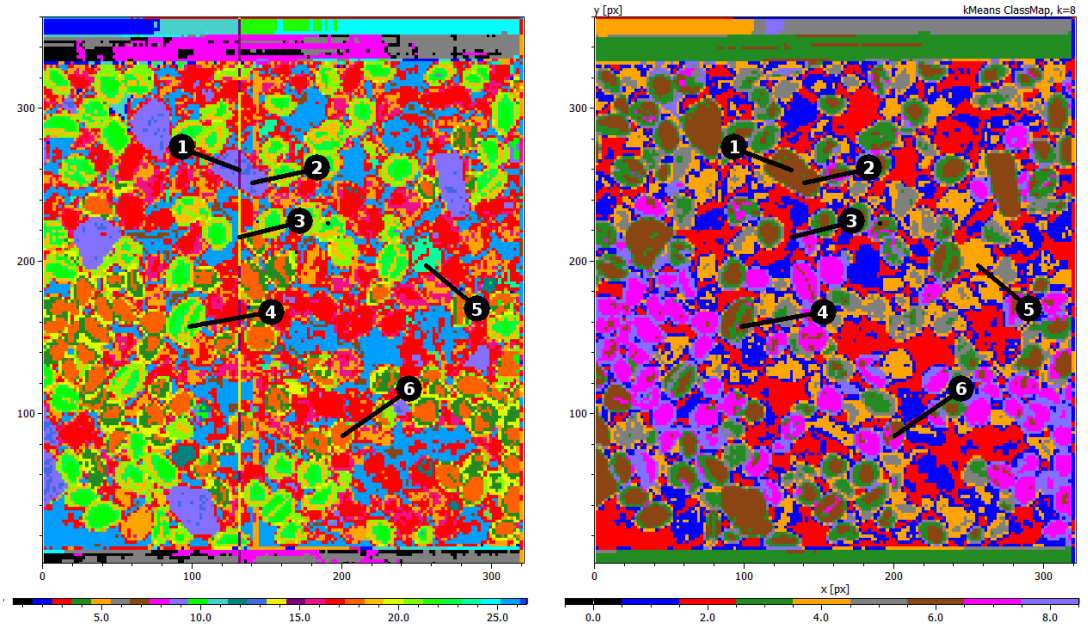
(a) unclustered

(b) GBCC with graph

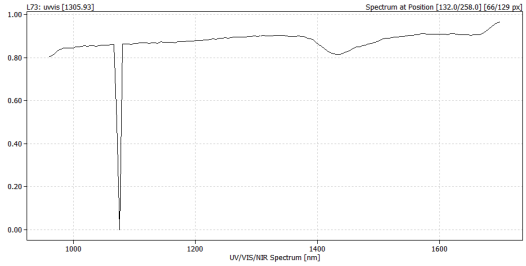
(c) GBCC

(d) Spectral Clustering

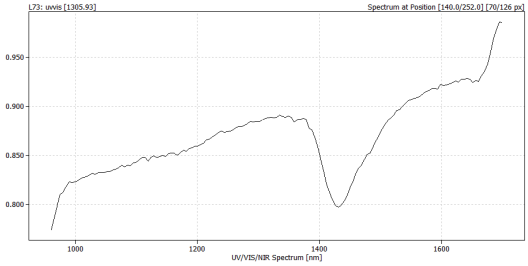
Figure 31: Zoom 3 of the Chemical dataset. This data range corresponds to rectangle 3 in figure 28a. GBCC and Spectral Clustering were the only algorithms that could detect these two clusters.



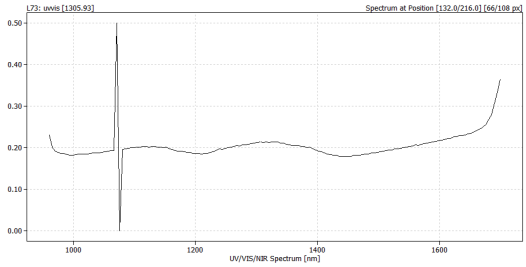
(a) GBCC (b) K-Means



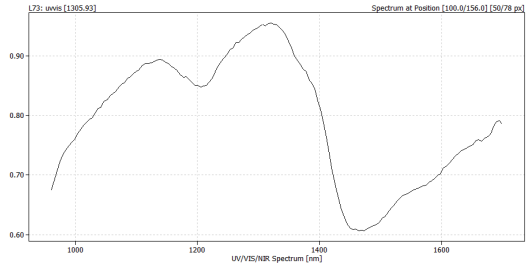
(c) Spot 1



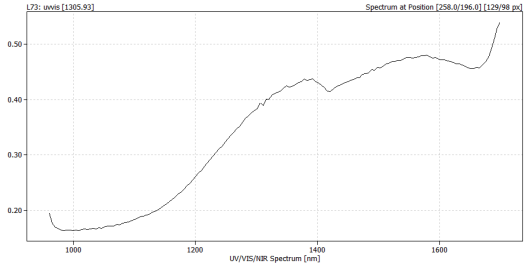
(d) Spot 2



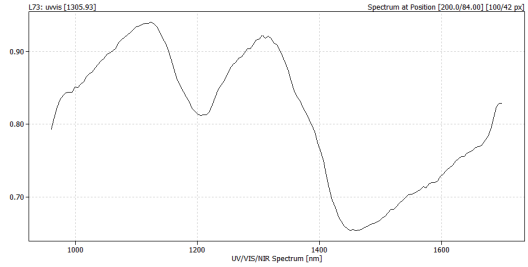
(e) Spot 3



(f) Spot 4

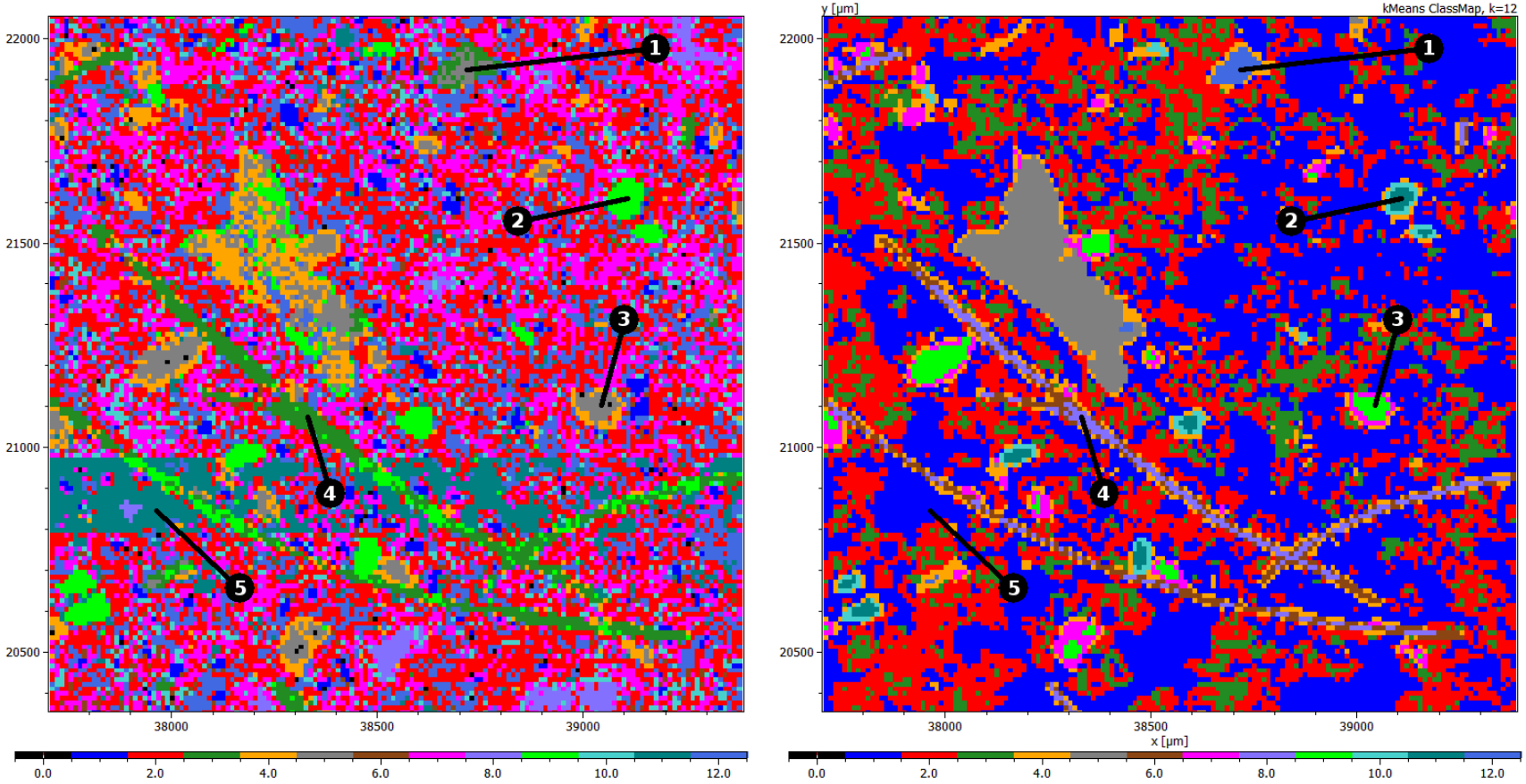


(g) Spot 5



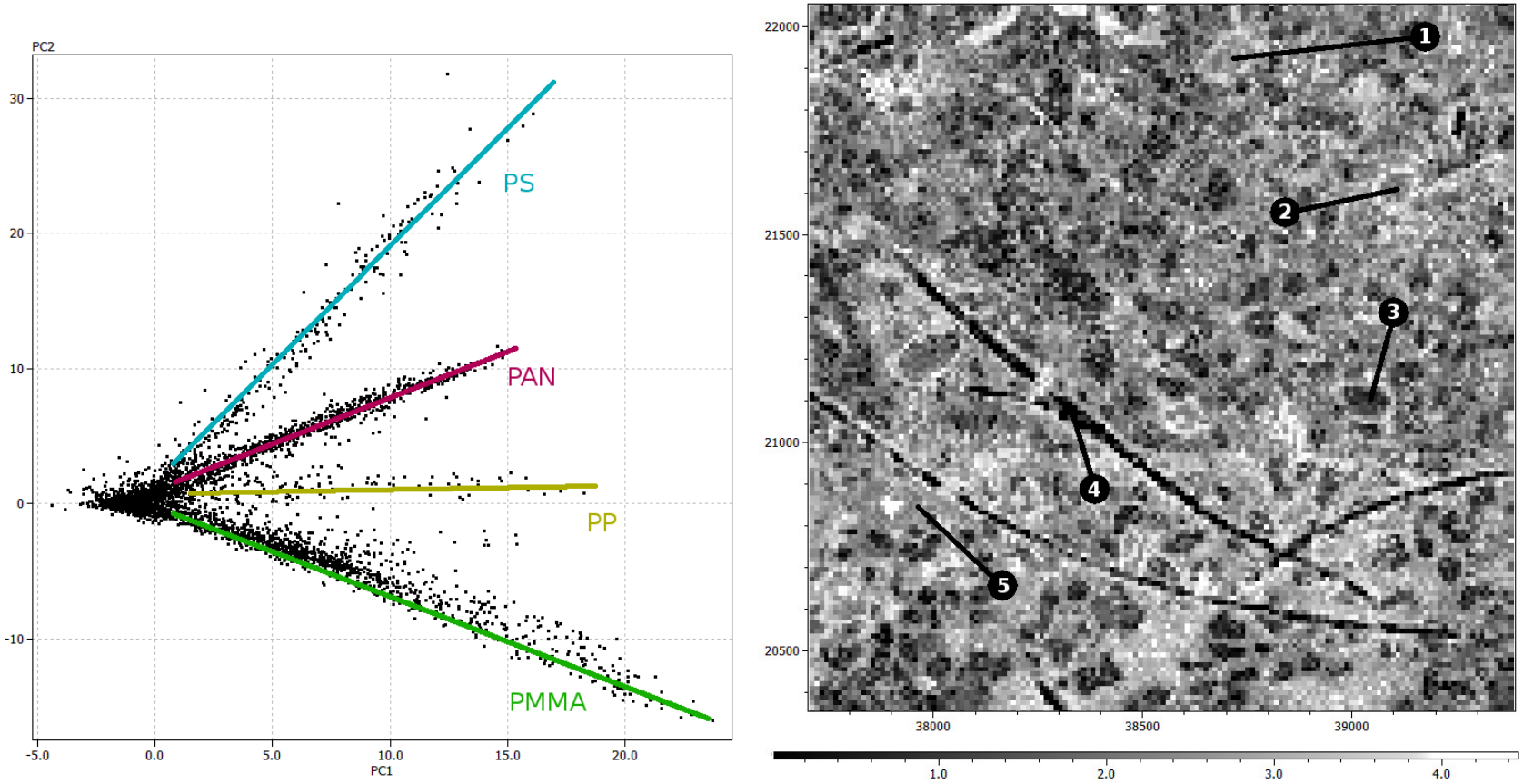
(h) Spot 6

Figure 32: Clustering of the Coffee Beans dataset.



(a) GBCC (b) K-Means

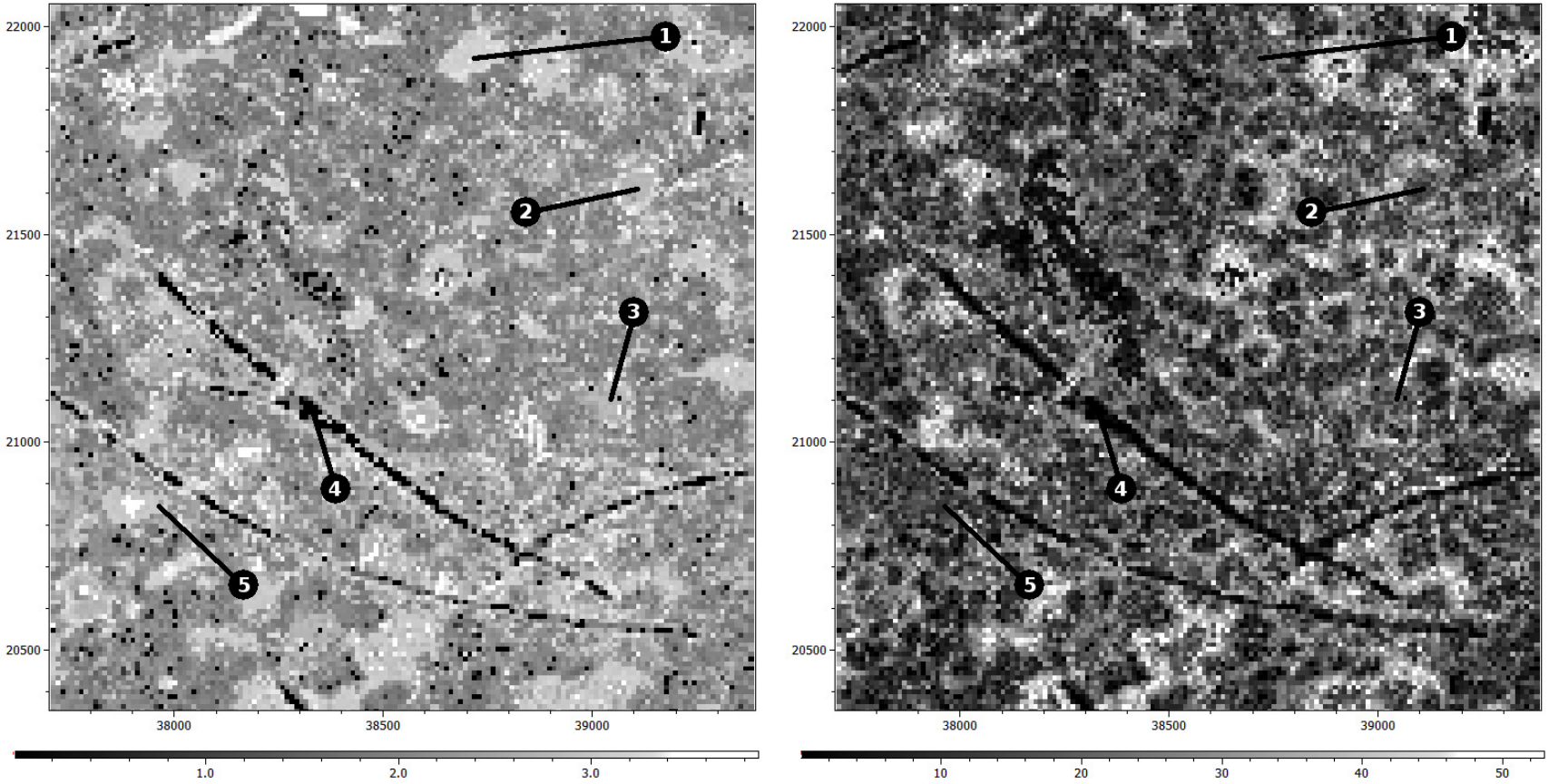
Figure 33: Clustering of the Microplastic A dataset.



(c) PCA

(d) Vertex weight

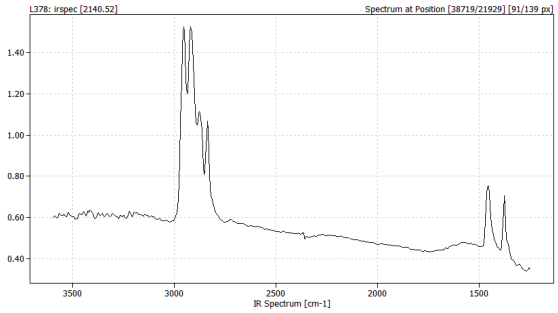
Figure 33: PCA and vertex weight of the Microplastic A dataset.



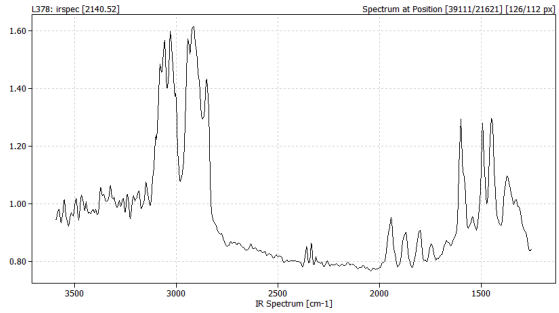
(e) Allocation weight

(f) Allocation path length

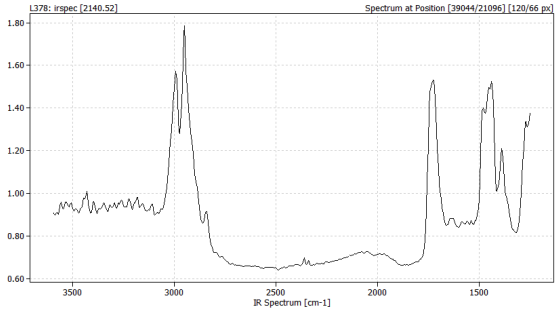
Figure 33: Allocation weight and allocation path length of the Microplastic A dataset.



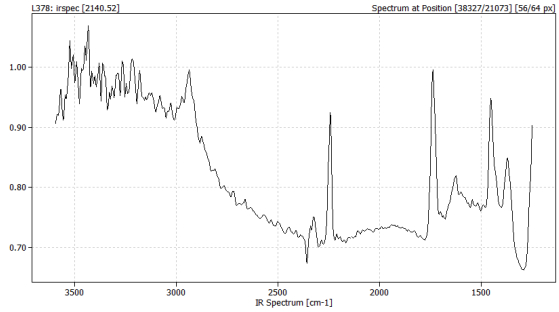
(g) Spot 1



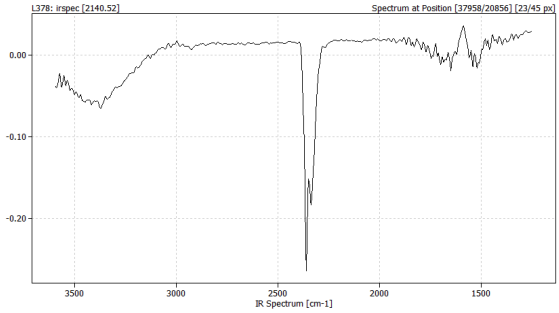
(h) Spot 2



(i) Spot 3

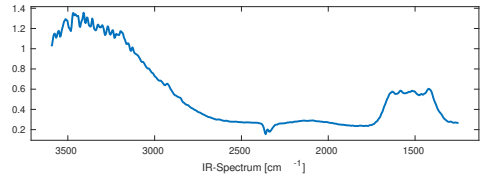


(j) Spot 4

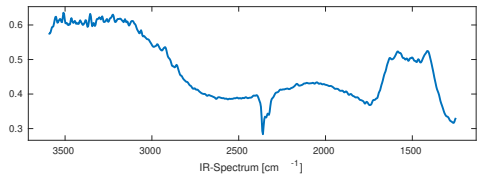


(k) Spot 5

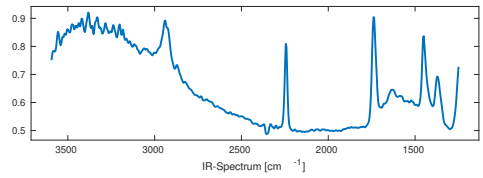
Figure 33: Spots of the Microplastic A dataset.



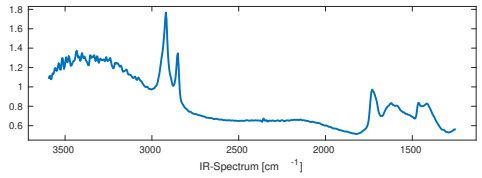
(l) Center 1



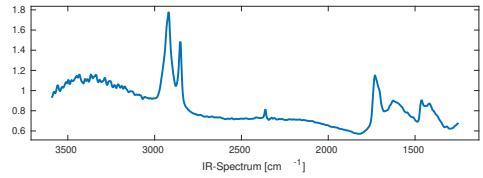
(m) Center 2



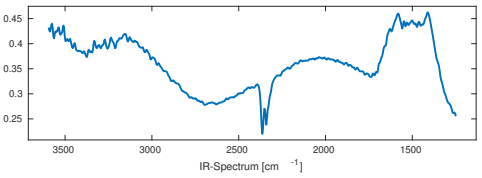
(n) Center 3



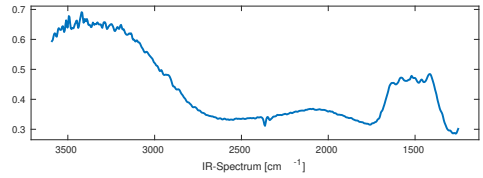
(o) Center 4



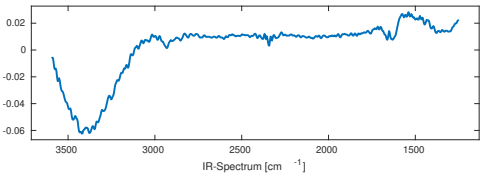
(p) Center 5



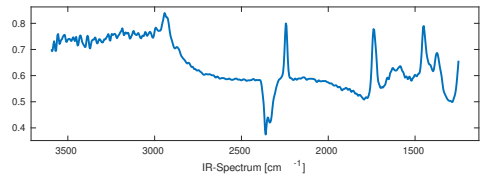
(q) Center 6



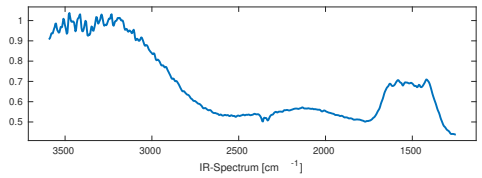
(r) Center 7



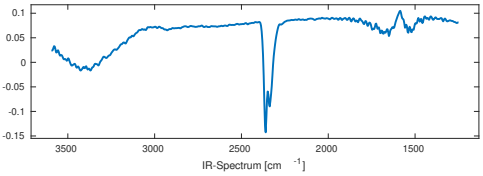
(s) Center 8



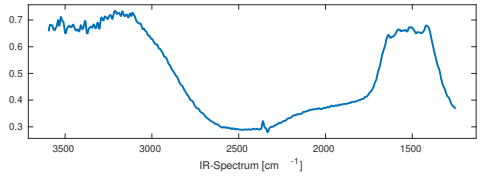
(t) Center 9



(u) Center 10



(v) Center 11



(w) Center 12

Figure 33: Centers of the Microplastic A dataset.

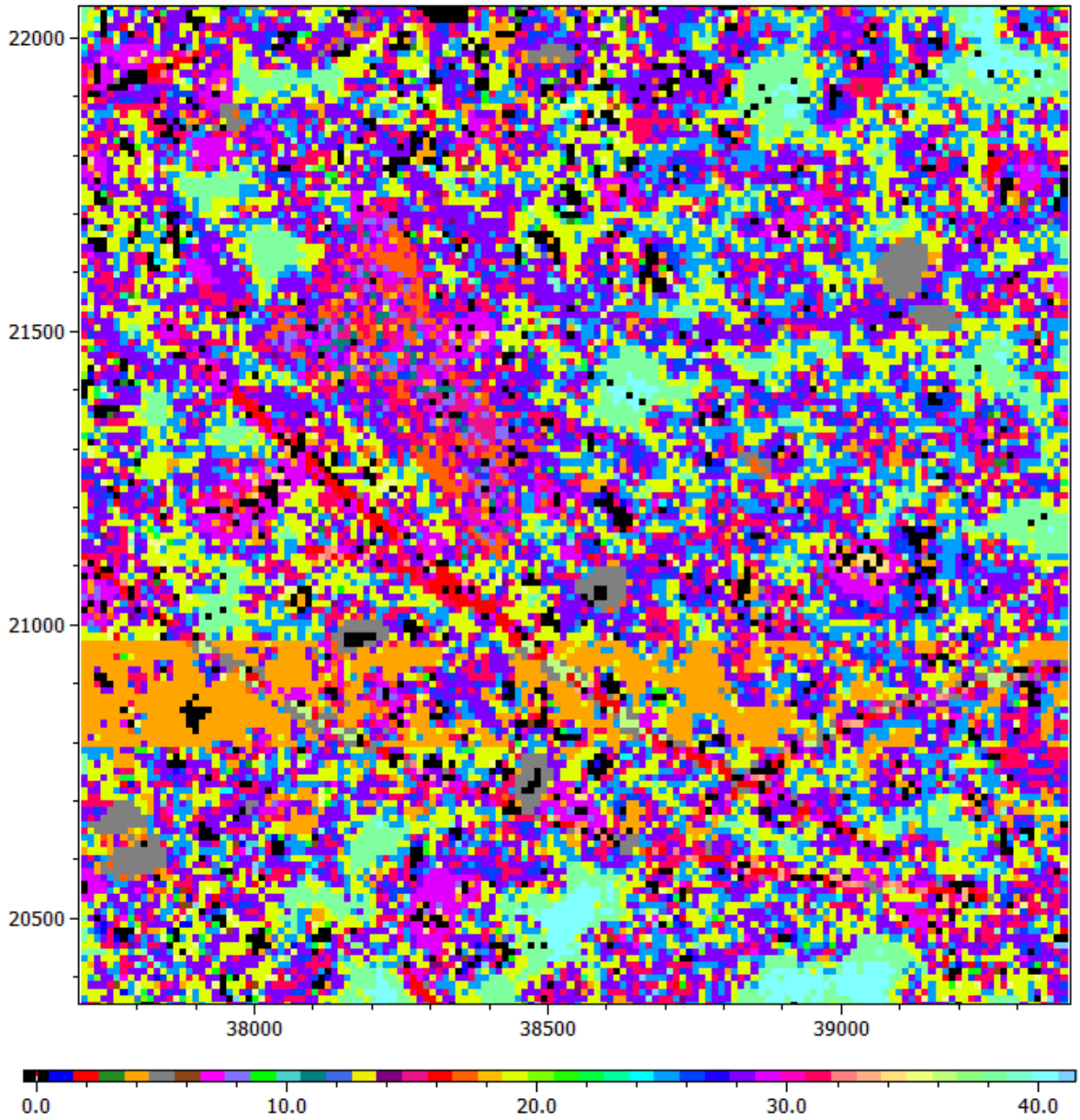
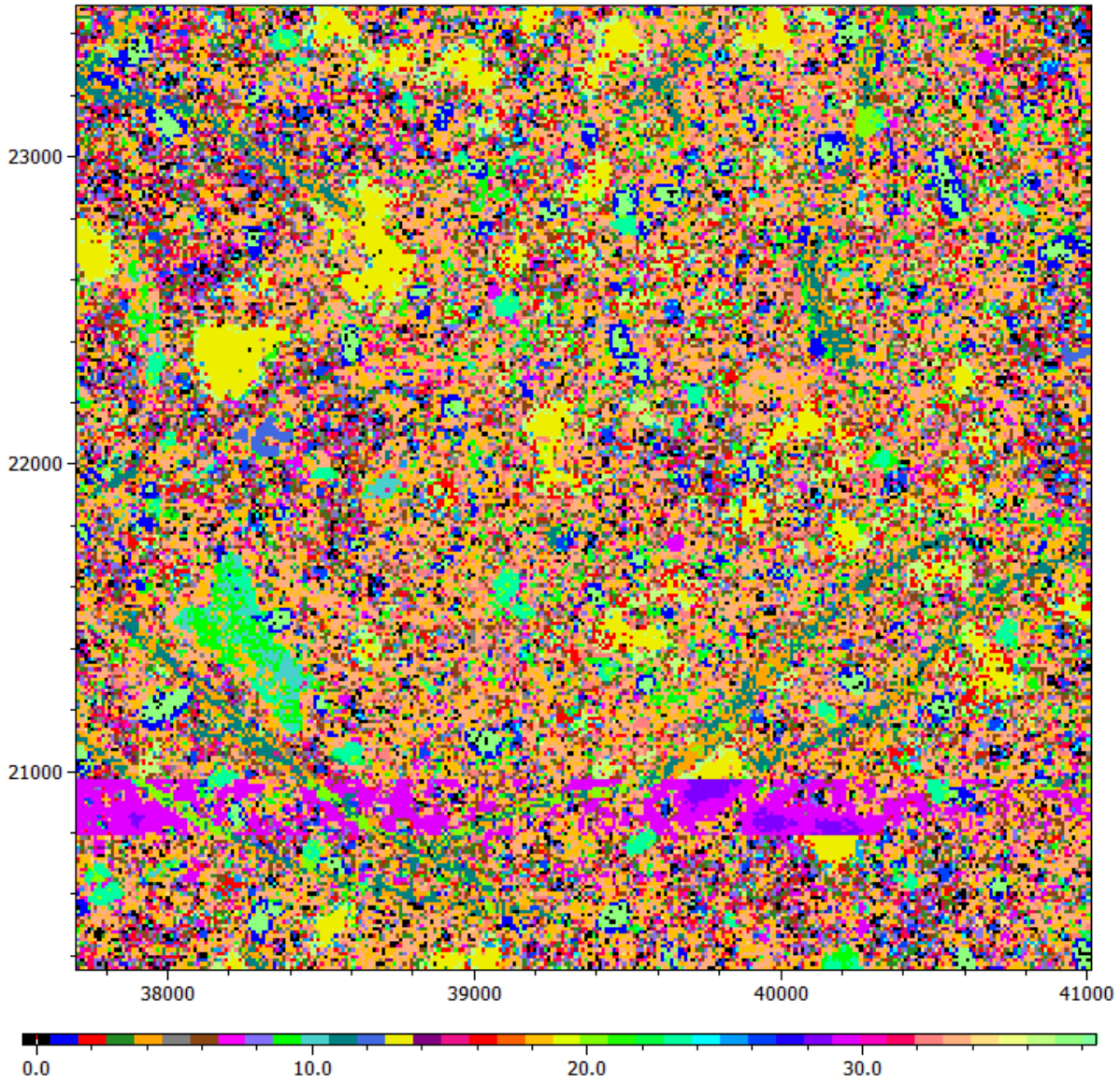
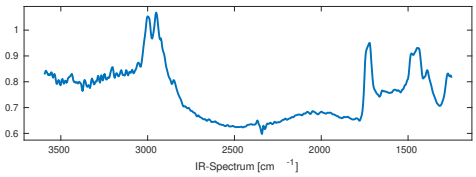


Figure 34: Result of the directed 30-nearest neighbor graph of the Microplastic A dataset. Due to the large number of clusters the boundaries are less well-pronounced though the PAN fibres and PS particles can still be distinguished from the background. Because the connectivity is less than in an undirected kNN graph 4.7% of the data remain unclustered as indicated by the black cluster 0.

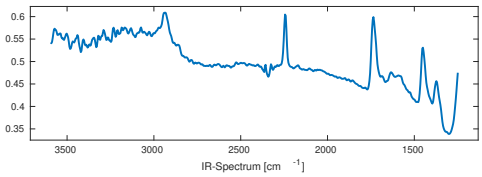


(a) GBCC

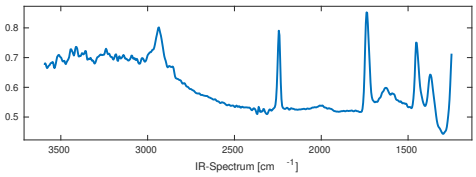
Figure 35: Result of the full-range clustering of the Microplastic B dataset. Despite the large number of clusters particle boundaries can still be distinguished from the background.



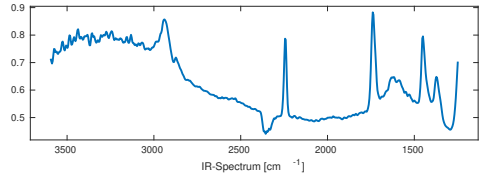
(b) Center 1



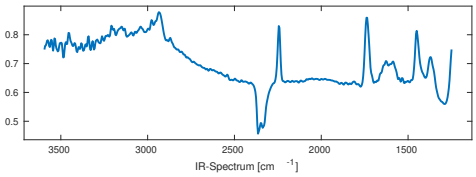
(c) Center 4



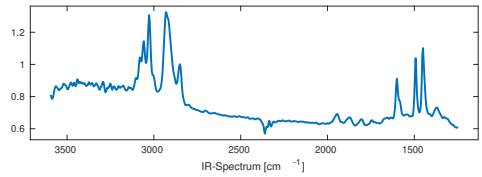
(d) Center 11



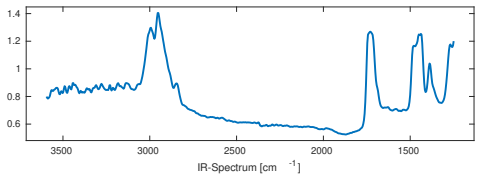
(e) Center 15



(f) Center 20



(g) Center 23



(h) Center 37

Figure 35: Centers of the Microplastic B dataset.

Bibliography

C. T. Zahn, "Graph-theoretical methods for detecting and describing gestalt clusters," *IEEE Transactions on computers*, vol. 100, no. 1, pp. 68–86, 1971.

J. Handl and J. Knowles, "An evolutionary approach to multiobjective clustering," *IEEE transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 56–76, 2007.

K. Bakeev, *Process Analytical Technology: Spectroscopic Tools and Implementation Strategies for the Chemical and Pharmaceutical Industries*. John Wiley & Sons, 2010.

A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

R. Xu and D. Wunsch, "Survey of clustering algorithms," *Neural Networks, IEEE Transactions on*, vol. 16, no. 3, pp. 645–678, 2005.

"Merrian-webster online dictionary," 2017. [Online]. Available: https://www.merriam-webster.com/dictionary/cluster%20analysis

U. Von Luxburg, R. C. Williamson, and I. Guyon, "Clustering: Science or art?" in *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012, pp. 65–79.

O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, no. 1, pp. 243–256, 2013.

R. Bellman, "Adaptative control processes," 1961.

H. Lohninger and J. Ofner, "Multisensor hyperspectral imaging as a versatile tool for image-based chemical structure determination," *Spectroscopy Europe*, vol. 26, no. 5, pp. 6–10, 2014.

D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatial–temporal data," *Data & Knowledge Engineering*, vol. 60, no. 1, pp. 208–221, 2007.

Q. Liu, M. Deng, Y. Shi, and J. Wang, "A density-based spatial clustering algorithm considering both spatial proximity and attribute similarity," *Computers & Geosciences*, vol. 46, pp. 296–309, 2012.

U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

G. N. Lance and W. T. Williams, "A general theory of classificatory sorting strategies: 1. hierarchical systems," *The computer journal*, vol. 9, no. 4, pp. 373–380, 1967.

M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

C. Zhong, D. Miao, and R. Wang, "A graph-theoretical clustering method based on two rounds of minimum spanning trees," *Pattern Recognition*, vol. 43, no. 3, pp. 752–766, 2010.

M. C. Nascimento and A. C. De Carvalho, "Spectral methods for graph clustering–a survey," *European Journal of Operational Research*, vol. 211, no. 2, pp. 221–231, 2011.

J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in neural information processing systems*, 2002, pp. 849–856.

R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

M. Forina, C. Armanino, M. Castino, and M. Ubigli, "Multivariate data analysis as a discriminating method of the origin of wines," *Vitis*, vol. 25, no. 3, pp. 189–201, 1986.

D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE transactions on pattern analysis and machine intelligence*, no. 2, pp. 224–227, 1979.

V. Parrag, J. Felföldi, and F. Firtha, "Coffee beans mixed with stones: a hyperspectral training data set for multivariate image analysis," 2014. [Online]. Available: http://imagelab.at/en_data_repository.html

M. Löder, "Microplastic," 2017, Private Communications.

M. Radovanović, A. Nanopoulos, and M. Ivanović, "Hubs in space: Popular nearest neighbors in high-dimensional data," *Journal of Machine Learning Research*, vol. 11, no. Sep, pp. 2487–2531, 2010.

"Epina imagelab documentation: Spectral descriptors," 2018. [Online]. Available: http://www.imagelab.at/help/spectral_descriptors.htm