Ostfalia
Hochschule für angewandte
Wissenschaften

Fakultät Informatik

A Heuristic Approach based on Composition Operators for Finding Linear Projections of High-Dimensional Data that Show Class Separation

Kai Michael Blum

Matrikel-Nr. 70413986

Masterarbeit im Studiengang Informatik
zur Erlangung des akademischen Grades:
Master of Science

Ostfalia Hochschule für angewandte Wissenschaften

1. Prüfer: Prof. Dr.-Ing. habil. Dirk Joachim Lehmann
2. Prüfer: Prof. Dr. Frank Höppner

Salzgitter · Suderburg · **Wolfenbüttel** · Wolfsburg

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere, dass ich alle wörtlich oder sinngemäß aus anderen Werken übernommenen Aussagen als solche gekennzeichnet habe, und dass die eingereichte Arbeit weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen ist.

Wolfenbüttel, 25.07.2022	[illegible]
Ort, Datum	Unterschrift

Abstract

Class separation of n dimensional data is a challenging task. Thus, reducing dimensionality by projecting multivariate datasets to 2D projection space is a good approach in many cases. Star Coordinates are a very powerful tool to visualize n dimensional data as a linear projection in 2D projection space.

In order to find projections of interest in projection space, a Python version of Composition Operators has been implemented. This item-based algorithm allows an interactive user-based exploration through projection space. While restricting and forcing data points to move within projection space, different compositions of interest can be proposed.

Different heuristic extensions for Composition Operators are introduced to attempt optimally separating projections. These heuristics try to find projection parameters that visually separate existing classes in projection space automatically.

Qualitative and quantitative evaluation is conducted with different well-known datasets. Furthermore, results are compared to a related technique of manipulating Star Coordinates. Apart from different visualizations, metrics are used to measure the quality of separation for various linear projections.

It can be shown that optimally separated projections do exist. However, not for every dataset, not for every configuration and not with every provided method. Using heuristics to find better separations automatically is — at least in most cases — successful and improves the separation compared to an initial projection in almost every case. The different effects of the various heuristic approaches on different data sets show that the properties of data sets are an important component for separation.

Contents

List of abbreviations

OSC	Orthographic Star Coordinates
SC	Star Coordinates
CO	Composition Operators
LSS	Least Square Solution
DSC	Distance Consistency
CD	Centroid Density
PT	Penalty Threshold
CDC	Centroid Distance Change
VML	Visual Machine Learning
RSS	Random Selection Shift
OSS	Order Selection Shift
MSS	Minimum Selection Shift
PSS	Point Selection Shift

1. Introduction

Almost everyone produces data every day by using their phone, using their computer, checking in at a hotel, going to the grocery, buying a ticket and many more activities. Visualizing such data with charts, plots, graphics, or animations is an important task. It makes data-driven insights and complex data relationships way easier to understand [16]. Outliers, trends, or even patterns can often be found just by a look at visualized data. Furthermore, visualized data increases an audience's interest way more than just blank numbers and facts. In addition to that, data visualization is, at least in most cases, a very quick way to get a first impression of presented data. As more and more data is produced, it is an incredibly increasing market. Global visualization market has a forecast of increasing with a Compound Annual Growth Rate (CAGR) of 9.47% from 2017 to 2023 to 7.76 billion USD [25].

However, to find a proper data visualization for every use-case can be quite challenging, as not every visualization gives information on every kind of data. Especially high dimensional datasets confront an observer with the problem of not being capable of perceiving connections correctly. Thus, lower dimensional embeddings of data are needed. Star Coordinates (SC) [19] are a technique to embed n-dimensional data in 2-dimensional projection space. Having a single 2-dimensional projection is often not enough to find the right visualization for used data. Composition Operators (CO) [23], an item-based projection manipulation algorithm, can help to find such projections by manipulating existing embeddings in SC widget. A projection of interest is always linked to a certain purpose, such as class separation. CO are a very powerful tool for projection manipulation, but require user-interaction.

This work aims to extend the idea of CO by a heuristic for optimally separating classes embedded in 2-dimensional projection space by SC. The heuristic should be able to find a suitable projection in projection space to achieve this goal automatically. This would make user interaction obsolete and abstract a given problem from its domain.

1.1. Motivation

Visualizing data in the first place does not need any knowledge about data domain. However, in order to classify this data correctly, it is often essential to have someone with domain knowledge. While the knowledge domain of a data scientist is data visualization, the problem domain of underlying data is usually in the field of another knowledge domain such as medicine, agriculture or other fields. Fitting domain knowledge is typically very complicated and expensive, as multiple experts have to work on the same topic.

Thus, having an automatically class separating heuristic in projection space would also separate domain expertise. In other words, it would decouple informatics or data visualization from the problem domain. A domain expert could then use the heuristic independently to solve his classification problem without further help of a data scientist.

Nonetheless, this would suggest that there is an automatically heuristic that finds optimally separating classes in every case. Since non-separable and non-linearly separable datasets exist, this cannot be possible for all datasets. A challenge for the resulting heuristic will be to find suitable termination criteria for these non-separable datasets.

In terms of visual data exploration, this heuristic connects data analysis techniques with interactive visualization methods. It gives a good insight on the different classes, but also on the linear dependencies between the attributes of a dataset. Furthermore, the resulting projection gives information on the importance of a certain attribute for the classification problem.

1.2. Task

The aim of this work is to extend the idea of 'Composition Operators' which was introduced by Lehmann et al. [23]. Earlier work by Molchanov and Linsen [28] and Sips et al. [35] is used to provide a good overview and reference on the topic. Different heuristics will be tested with regard to performance and results in order to extend CO. However, before doing so, an in depth literature research has to be done. The following points are the central aspects of this thesis:

A) Systematic and comparative literature research

- Approaches to the design of projections
- Approaches to finding (mining) separating linear embeddings
- Approaches to Visual Machine Learning (VML) based on embeddings/linear projections

B) Implementation

- Implementation of Composition Operators in Python
- Implementation of Molchanov et al. [28] in Python as a comparison
- Extension of CO with a heuristic for classes to be optimally separated in the embeddings
- Evaluating experiments (visual classification of example datasets)

A visualization of the evaluated results is also performed. If possible, the initial implementation of CO should also be interactive. Therefore, visualization would have to take place in real-time. Heuristic results The following questions should be answered by this thesis:

- Are there any datasets which are linear separable?
- Can it be guaranteed that a separating embedding will always be found if the dataset allows linearly separating embedding?

1.3. Structure of the Thesis

This thesis is structured as follows:

Chapter 2 contains a systematic and comparative literature research on related work. After an introduction on Visual Analytics, an overview of non-linear and linear projections is given. In the course of this, Star Coordinates are introduced as one of the main elements of this thesis. An overview of different types of heuristics is followed by approaches of separation in linear projections. The chapter is then closed with a peak at Visual Machine Learning based on projections.

In order to understand main aspects of this work, chapter 3 gives an insight on background. Centroids are introduced as a cluster representation technique. Furthermore, a detailed overview on the idea of Composition Operators is provided.

Chapter 4 explains the approach to different heuristics for problem-solving. These heuristic are presented in three rather similar variants that differ in complexity and one heuristic that operates differently. In addition, convergence criteria are discussed and approaches are explained mathematically.

Chapter 5 describes the implemented software. It is divided into two different versions, both of which can be controlled with a user interface. In addition to the user interface, functional aspects of the implementation are also explained using pseudocode. The technique by Molchanov et al. [27] has been implemented as well and will be described briefly.

Chapter 6 contains a description of methods and metrics for evaluation. This is preceded by a brief explanation of the choice of datasets and the datasets from UCI Machine Learning Repository themselves. Furthermore, a reference for evaluation of the datasets with their key values is presented. In addition to that, results for the designed evaluation are visualized, analysed and compared to related work.

These results are discussed in chapter 7. Main achievements of this thesis are presented and critical conclusions are drawn under given limitations for class separation tasks.

Chapter 8 closes this thesis with an outlook towards future work. In particular, the focus is on the use of an approach with Visual Machine Learning and optimization of the existing heuristic towards separation.

2. Related Work

The following chapter gives an overview about related work on the topic.

2.1. Visual Analytics

'The goal of visual analytics is to turn the information overload into an opportunity' [20]. Therefore, it is extremely important to extract relevant information from large amounts of data. The basic aspect is to visually represent this information to allow human interaction. [20]

To do so, there is a wide range of visualization techniques available. Figure 1 illustrates some of these techniques. Different visualizations show different contexts and focus on different aspects within data. Thus, finding a suitable representation for data can be quite challenging.

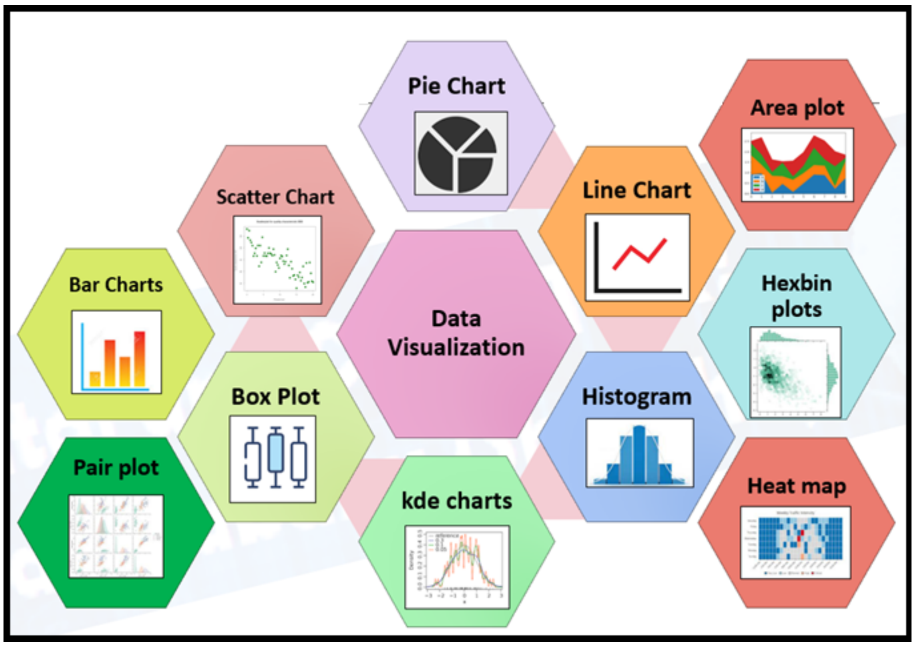


Figure 1.: Different data visualization techniques [29]

Figure 2 shows the scope of visual analytics. It becomes clear that visual analytics is more than just the visualization of data. Although visualization is a key component, its combination with human factors such as interaction, cognition, perception, or presentation and data analysis such as statistical analytics, information analytics or geospatial analytics makes it a considerably broad field of work. [20]

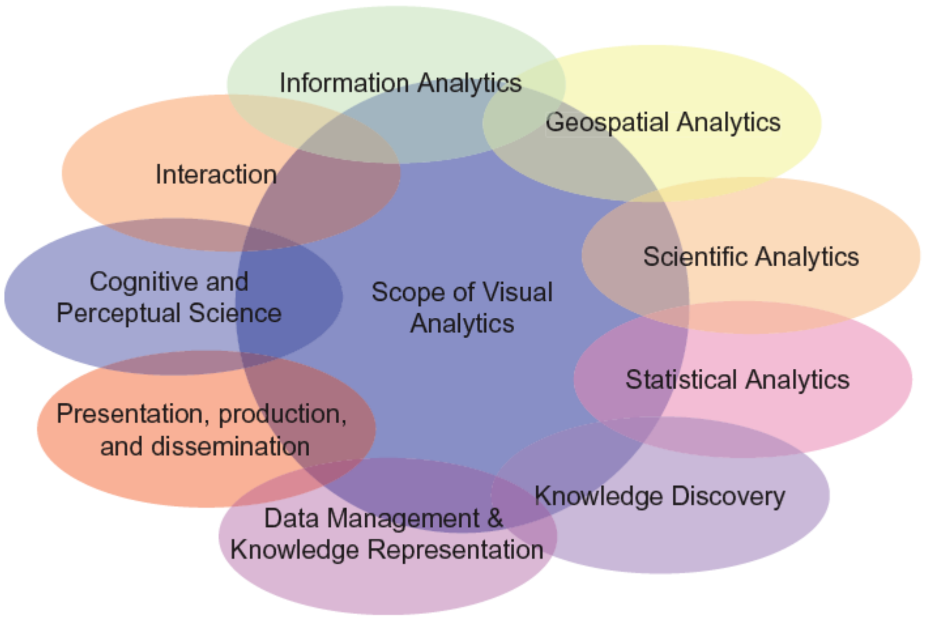

Information Analytics
Geospatial Analytics
Interaction
Scientific Analytics
Cognitive and Perceptual Science
Scope of Visual Analytics
Statistical Analytics
Presentation, production, and dissemination
Knowledge Discovery
Data Management & Knowledge Representation

Figure 2.: The Scope of Visual Analytics [20]

However, all those components do not just coexist in the scope of visual analytics but rather interact with each other as illustrated in Figure 3. Visual Analytics Process always starts with data. Since incomplete or distorted data cannot be visualized reasonably, the first step is pre-processing and transformation. [21]

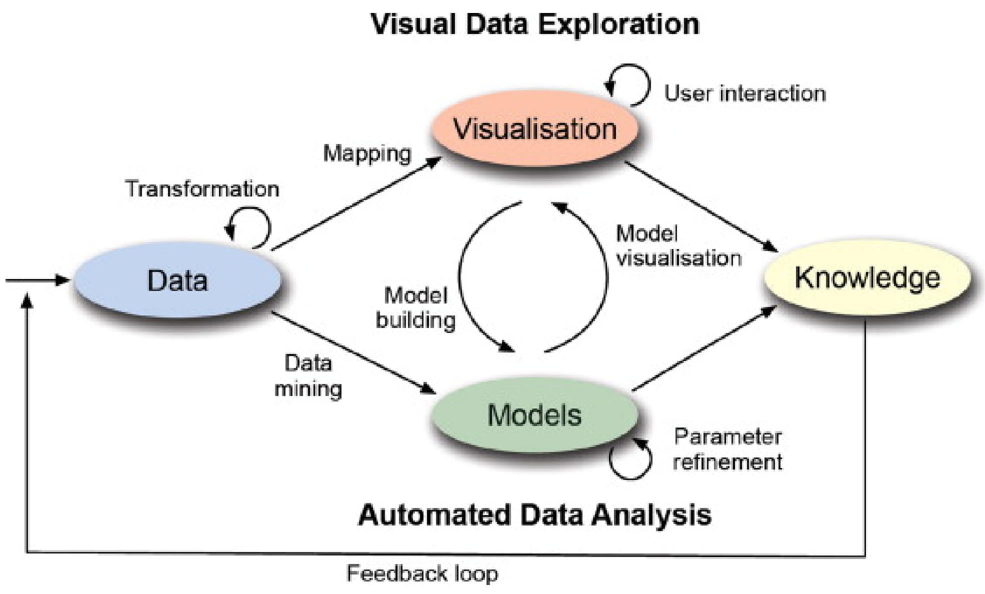


Figure 3.: The Visual Analytics Process [21]

It is then followed up by either visual or automatic analysis methods. In case of the latter, a model is constructed by applying data mining methods and parameters are refined by interaction with data. Findings can be evaluated by visualizing the model and interacting with it. This leads to an alternation between visual and automatic methods, which results in a continuous refinement of the model. Ultimately, all these steps lead to a gain in knowledge. [21]

2.2. Non-linear Projections

Approaches to the design of projections from nD to 2D or 3D space are widely used to visualize high dimensional data. According to the work of Sedlmair et al. [34] there are rarely any benefits and even downsides from using 3D instead of 2D projections especially for verification of clusters while being at higher cost. Thus, further remarks refer to 2D projections.

Projections are called non-linear if the projectors to produce the embedding are non-linear, such as manifolds. Non-linear projection schemes support record-to-record (local) based analysis. This means they give information on intrinsic structure as well as topological properties of data records. There are already several approaches to project data non-linearly such as Sammon Mapping [33], CCA [8], Isomap [37], SNE [14] or Laplacian Eigenmaps [3]. These well known techniques work quite well for artificial datasets and local structures [39]. Newer approaches like t-SNE [39] and UMAP [26] are also capable of dealing with real world data and preserve global structures, at least to a certain degree.

However, this work focuses on dimension-to-dimension (global) analysis. Global analysis allows finding patterns between different dimensions of datasets. Since these non-linear approaches mainly focus on analysing record-to-record relations, they are not suitable for this work. Thus, this work continues with linear projections. [23]

2.3. Linear Projections

Many techniques in the field of linear projections have been developed over the last decades. Linear projections embed multivariate data in a projection space with a linear function. The easiest of these schemes are bi-variate scatterplots which focus on two data dimensions at a time. Scatterplot matrices (SPLOM) are used to analyse multivariate datasets by embedding every possible combination of 2 dimensions in a matrix of scatterplots. However, this approach has two downsides: The matrix of scatterplots increases quadratic with the number of dimensions and relations can still only be observed between two dimensions at a time.

A well-known technique to embed multiple dimensions at once are Inselberg's Parallel Coordinates [17][18]. Parallel Coordinates represent multivariate data as a series of lines between parallel axes. While it helps to perceive relationships and trends, it is very hard to interact and identify single data points.

Other projection based multivariate visualization techniques arrange variables in radial layouts. Two of the most popular of these techniques are RadViz [15][7] and SC [19]. Both are quite similar in their approach, but differ by a non-linear normalization step in RadViz, which causes non-linear distortions [32]. This leads to SC being chosen as the linear projection tool for this work.

SC offer a way of representing n-dimensional data in 2D projection space, with each attribute n being shown as an axis. In Figure 4 the well-known wine dataset [9] is visualized with SC with 13 axis, as 13 different attributes were taken into account. Different colours in the figure indicate different classes of the data.

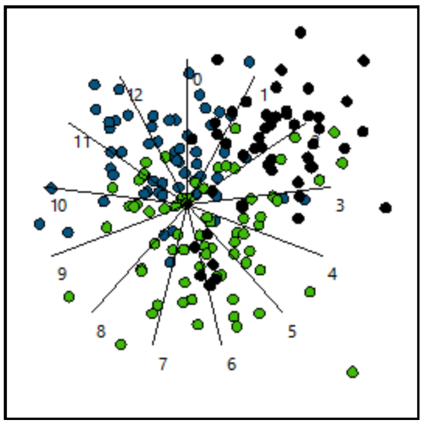


Figure 4.: Wine dataset visualized with SC

Mathematically, SC map a data record d with n dimensions onto a 2D projection space by multiplying with a projection matrix A:

$$p = A \cdot d \tag{2.1}$$

where A is a $2 \times n$ matrix of n anchor points $a_i = (x_i \; y_i)^T$ to represent the impact of each dimension on the projection:

$$A = \begin{pmatrix} x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \end{pmatrix} \tag{2.2}$$

Projecting a whole dataset D is equivalent to mapping a single data record d:

$$\underbrace{P}_{2\times m} = \underbrace{A}_{2\times n} \cdot \underbrace{D}_{n\times m} \tag{2.3}$$

The relation between the different domain spaces is illustrated in Figure 5: Data D in nD data domain is projected by a projection matrix in projection matrix domain $2n$D, to a projection P

in 2D projection domain.

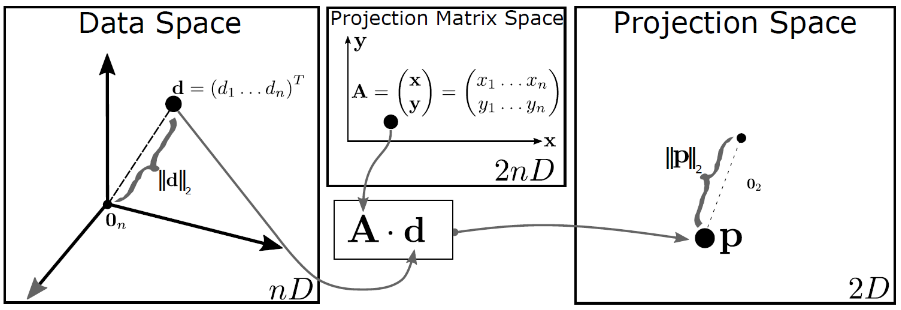

Data Space
$\mathbf{d} = (d_1 \ldots d_n)^T$
$\|\mathbf{d}\|_2$
$\mathbf{0}_n$
nD
Projection Matrix Space
$\mathbf{A} = \begin{pmatrix}\mathbf{x}\\\mathbf{y}\end{pmatrix} = \begin{pmatrix}x_1 \ldots x_n\\y_1 \ldots y_n\end{pmatrix}$
$2nD$
$\mathbf{A} \cdot \mathbf{d}$
Projection Space
$\|\mathbf{p}\|_2$
$\mathbf{0}_2$
$\mathbf{p}$
$2D$

Figure 5.: Considered Domains: (left) nD Data Domain, (middle) 2(nD) Projection Matrix Domain, (right) 2D Projection Domain [23]

While any initial projection matrix A is possible, scaling it into a circle with $360°/n$ degrees between the axis with equal length is very intuitive (see Figure 4). As dimensions are linearly connected in SC, interaction with anchor points a_i of the projection matrix A allows the user to get a dimension-wise data interpretation. The influence of an anchor point a_i towards the resulting projection is related to its norm. The larger a norm of an anchor point, the higher the influence on the projection. In theory, the influence of a single anchor point can become infinitely high. However, separating data by infinitely large scaled axes loses the expressiveness and the value of the statement, as there is no practical use for the result.

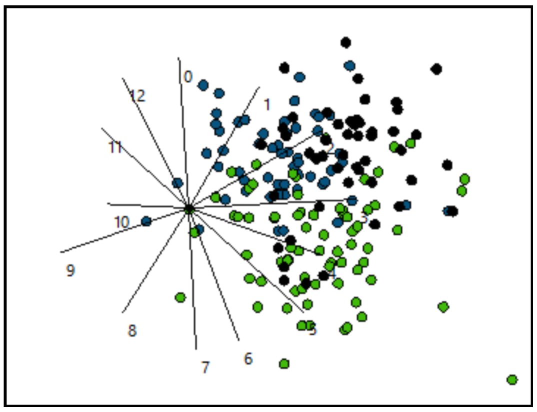


Figure 6.: Wine dataset visualized with SC (non circle)

Figure 6 shows wine dataset visualized with SC with a manipulated projection matrix A. It can be seen that some axes are extended while some are shortened. Thus, the impact on the projection has changed. The green and the black class have less overlap than before. It can be guessed that the attribute on axis 3 is a good attribute to classify between black and green as it increased in size.

SC are an affine projection, which means that a projective bias will appear in a way that a sphere in nD will become an ellipse in 2D [24]. This bias is created by the embedding function of SC. Assumptions based on the shape of the data in 2D projection space about the shape in nD are accordingly not possible. This problem is addressed by Orthographic Star Coordinates (OSC) [24]. OSC limit the size of our anchor points a_i by giving the following conditions:

$$\|x\|^2 = 1, \quad \|y\|^2 = 1, \quad < x, y >= 0 \tag{2.4}$$

with the latter being the scalar product. This technique for OSC is named Reconditioning. For practical usage of OSC, this condition forces the average length of all axis to stay on the same level regardless of the projection. This leads to the fact that if an axis is about to be extended, one or more other axis must be shortened in order to get an orthographic projection. [24]

An orthographic projection can be guaranteed by choosing angles of $\alpha = 2\pi/n$ and a radius of $r = \sqrt{2/n}$ in Formula (2.5) [24].

$$(x_i, y_i)^T = r \cdot (sin(i \cdot \alpha), cos(i \cdot \alpha))^T; i = 0, ..., n-1 \tag{2.5}$$

with n being the number of dimensions projected. While OSC work very well for low dimensionality, the representation method reaches its limits when applied in extremely high dimensionality. For $n \to \infty$ this leads to Formula (2.6):

$$\lim_{n\to\infty} \alpha = 2\pi/n \to 0 \quad ; \quad \lim_{n\to\infty} r = \sqrt{2/n} \to 0 \tag{2.6}$$

By increasing dimensionality, orthographic projections in SC lack degrees of freedom. A stretched area between X and Y becomes smaller and smaller until it reaches 0 [24]. Thus, the area for projections becomes infinitely small, as illustrated in Figure 7.

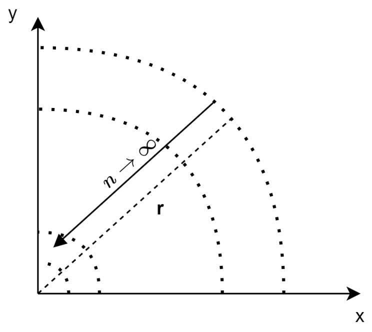


Figure 7.: Limitation of decreasing radius with increasing dimensionality

2.4. Heuristics

P.M. Todd defines heuristics as:

> '[...]approximate strategies or 'rules of thumb' for decision making and problem solving that do not guarantee a correct solution but that typically yield a reasonable solution or bring one closer to hand. As such, they stand in contrast to algorithms that will produce a correct solution given complete and correct inputs.[...]' [30]

This leads to the conclusion that heuristics do not take every given or missing information into account and thus, lead to a simplified decision-making of a given problem. Therefore, heuristics often make boundary assumptions and reduce complexity not only by the limitation of insufficient data but also for reasons of intentional simplification. However, transferring problems into real world scenarios with limited knowledge and time as well as uncertainty, heuristics become important and efficient tools which can outperform other strategies of higher complexity [13]. Especially, time is a very crucial factor in optimization problems such as flight scheduling or load balancing in telecommunication [4].

The biggest advantage of heuristics in comparison to complete search is their rather fast computation time. It makes them applicable in fields where complete search struggles to find a solution within an acceptable time. This is especially the case when a good approximation of the exact solution is sufficient for the use-case.

Table 1 shows a comparison between heuristics and complete search:

	heuristics	complete search
computation	fast	slow
solution	error prone	exact
mathematically provable	in most cases no	yes
based on	intuition, exploration, guesses	finite set of instructions

Table 1.: Comparison between heuristics and algorithms

While heuristics are usually designed to suit a certain problem, *metaheuristics* have been developed to suit a wide range of applications. They are independent of the underlying optimization problem. Metaheuristics are not to be understood as step-by-step manuals, but more as basic ideas and approaches such as tabu search or simulated annealing [36].

Research on the topic has come up with many 'rather standard metaheuristics' [4] which were restrictive on their own. In order to improve their behaviour in terms of efficiency and flexibility, combinations of metaheuristics and other optimization techniques have been developed [4]. These so-called *hybrid metaheuristics* are capable of dealing with real-world and large-scale problems [4]. Gallardo et al. [10] developed such a hybrid metaheuristic by combining an evolutionary algorithm with the branch and bound method.

2.5. Separation of Linear Projections

Linear projections can pursue different goals and be of different quality. A wide range of research has already been done on this topic.

In case of this work, projections are classified as good projections if they can be separated well. In order to be able to determine well separable projections, a metric is needed. Sips et al. [35] propose class consistency as a criterion to select good views. Class consistency describes the separation and density of data towards their cluster centroids (see section 3.1).

There are different approaches to find such projections. First of all, it is possible to manually explore through different parameter configurations. However, finding a suitable projection in a $2n$D search space is very unlikely, if not impossible. Another possibility is to investigate a complete or grand tour through the desired projection space [2][6]. These tours are very time-consuming, as they grow exponentially with the number of dimension. Another drawback is that it can not be guaranteed to overlook relevant projections, as only a cross-section of all possible projections is covered. The main idea of all these approaches is to define a certain input and analyse whether the resulting output suits a defined criterion.

Alternatively, it is possible to turn this principle around and define a desired result before proceeding the necessary input. This is possible with Composition Operators [23]. CO (see section 3.2) are a user-friendly and intuitive approach of manipulating a projection.

Another approach of this kind is presented by Molchanov and Linsen [27]. They create a 2D visualization of the different classes of a dataset and their respective centroid with SC. This initial projection can be modified by Drag & Drop of the class centroids. When dropping a centroid on a new position, a new projection matrix is calculated by the Least Square Solution (LSS) of a linear equation system to satisfy the user-defined shift. This interactive technique leads to better class separation and reduced clutter.

Teoh and Ma [38] present a method for interactive visual classification with decision trees by using SC. The technique is based on step-wise separation of classes by a user in SC widget and subplots of it.

All the latter techniques are based on user-interaction. According to Ankerst et al. [1] this leads to the following advantages:

1. pattern recognition capabilities of human brain can increase the effectiveness
2. deeper understanding of the results and thus more trust into the system
3. domain knowledge by the user can lead to better results and avoid overfitting

However, user-interaction always trades off with time and manpower. Although a user's domain knowledge may lead to better results, it is also possible that the results are negatively affected by the user's interaction. Furthermore, domain knowledge is not always existing. Nonetheless, advantages surpass disadvantages in most cases, which makes user-interaction a useful but expensive tool for visual analytics.

2.6. Approaches to Visual Machine Learning based on Projections

VML based on projections or linear embeddings is a rather new field of research. The following section will give a brief overview of published work on this topic.

Rauber et al. [31] propose the use of visual representations based on projections for predictive feedback on classification efficacy, as well as a projection-based visual analytics methodology to improve classification systems through feature selection.

Heidari et al. [12] applied a Random Projection Algorithm (RPA) to optimize a machine learning model for breast lesion classification. Their approach creates a more compact feature space that reduces correlation and redundancy. This results in an optimal feature vector with significantly higher classification performance.

Chu et al. [5] present a method for real-time electromyogram (EMG) pattern recognition. An extracted feature vector of the EMG signal is then linear-non-linearly projected by composing principal component analysis (PCA) and self-organizing feature map (SOFM). After that, multilayer perceptron (MLP) is used as classifier. As a result, recognition accuracy is more dependent on class separability of the projection than on the classifier itself.

With regard to this work, an extension through the use of VML could be useful. Therefore, feature vectors are extracted from projection space and validated with training data. Good separations in the projection space are essential for designing classifiers with that approach. However, having ideally separated classes in projection space is even better.

3. Background

In the following part of the work, basic terminology will be explained so that later explanations can be followed.

3.1. Cluster Representation

An alternative to displaying each individual data point of a class are the so-called *centroids*. These centroids are in the mathematical centre of a class. They can be calculated by averaging all points of a class. Centroids $\overline{C}$ in SC are points embedded in 2D which result in the sum of all x- or y-components of a class divided by the number of instances N belonging to the class as shown in Formula (3.1):

$$\overline{C} = \left[\frac{\sum X}{N}, \frac{\sum Y}{N}\right] \tag{3.1}$$

Taking a closer look at Figure 4 three different centroids can be calculated. These centroids are shown in Figure 8 marked in red colour. For a better overview, each section shows only the projection points of their respective class.

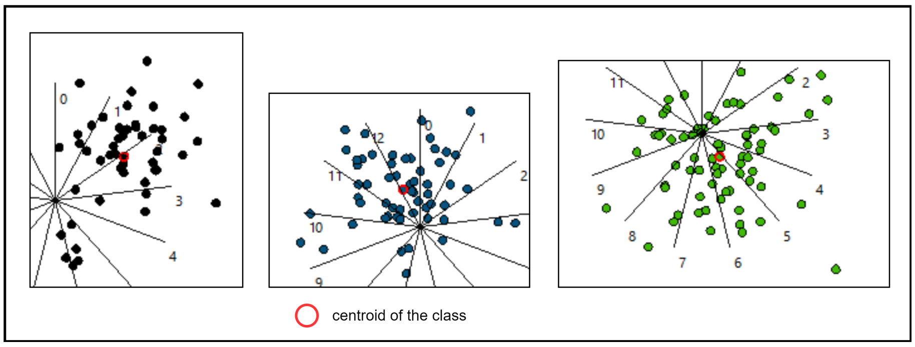


Figure 8.: Centroids for wine dataset visualized with SC

As centroids represent a whole class of points, it is easy to manipulate data points class-wise by manipulating their centroids.

3.2. Composition Operators

The following statements about Composition Operators are derived from notes by Prof. Dr.-Ing. habil. Dirk J. Lehmann [23]. CO are a direct and explicit projection point manipulation technique for linear projections. In this work, they are used to interactively transform an initial projection of a nD dataset linearly embedded with SC into a novel SC projection. Thereby, user constraints influence this projection.

Considering SC, a projection is created by the free choice of a projection matrix A. In contrast, CO reverse this process and generate projections using a freely definable projection P. This means that the degree of freedom shifts from A to P. Thus, CO calculate a new projection Matrix A (see Formula (2.2)) with a given projection P and their respective dataset D under certain user constraints.

The basic idea of CO is to interactively shift a point or a set of points into a freely chosen direction vector. The algorithm then tries to create a new projection by variation of projection parameters to realize this shift. While the user can define those *shift-able* points, it is also possible to set points as *sticky*. These points are supposed to remain in their position even with a variation of projection parameters. Projection points that are neither considered *shift-able* nor *sticky* are not taken into account when calculating a new projection matrix. Furthermore, CO can be used interactively to find a projection of interest. While doing so, it is also possible to change the status of a point from *sticky* to *shift-able* and the other way around.

However, not every manipulation can be realized in every situation which gives the assumption that it does not exist. CO then reflect the structure of data by a least-square solution.

There are three different shifting operations available for CO:

- one-point shift
- shift-able sets
- shift-able sets vs sticky sets

They are illustrated in the following Figure 9. The predicted movement of the projection points is indicated by arrows next to them. The blue *shift-able* points should move right towards their desired location $\overline{p_m}$ respectively $\overline{P_m}$. The red *sticky* points on the other hand should not move at all.

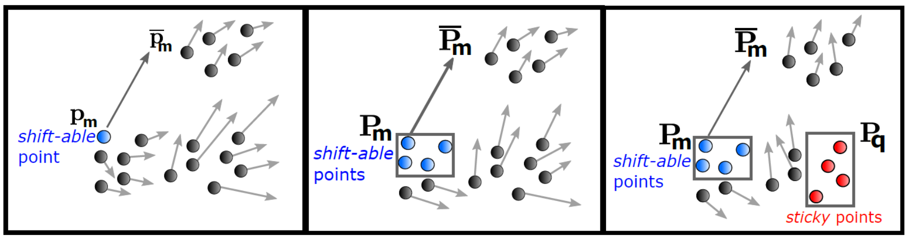


Figure 9.: CO: (left) one *shift-able* point p_m, (middle) a set P_m of *shift-able* points, (right) with an additional set P_q of *sticky* points [23]

The shift $\Delta p = (\Delta p_x \ \Delta p_y)^T$ that moves *shift-able* projection points p to their new position $\overline{p}$ are given by user input. It contains information about the corresponding shift in x- (Δp_x) and y-direction (Δp_y). To turn these shifts into a new visualization the projection is manipulated by calculating a new projection matrix $\overline{A} = A + \Delta A$ depending on the modified data. The resulting projection matrix $\overline{A}$ leads to new projection points $\overline{p} = p + \Delta p$ by multiplication with their respective data record d. CO are reversible. Thus, a shift in the opposite direction with the same length results in the original representation again.

Before getting into the calculation of projection manipulation, a few mathematical notations have to be clarified.

$1_n = (1\ 1\ ...\ 1)^T$ is defined the one column vector, I_n is an $n \times n$ identity matrix. $\|a\|_2$ is the euclidean norm of a column vector a calculated as:

$$\|a\|_2 = \sqrt{\sum_i^n a_i^2} \tag{3.2}$$

A dataset D contains m data records d of length n which is the amount of attributes in the dataset. It is defined as:

$$D = (d_1\ d_2 ... d_m)\ with\ d = (d_1\ d_2 ... d_n)^T \tag{3.3}$$

In order to define *sticky* and *shift-able* points, selected data records d are relabelled as m_i for *shift-able* and q_j for *sticky* points and saved as $n \times q$ matrix M respectively $n \times k$ matrix Q:

$$M = (m_1\ m_2 ... m_q) \tag{3.4}$$

$$Q = (q_1\ q_2 ... q_k)\ with\ M \cap Q = \emptyset \tag{3.5}$$

Their corresponding projections are P_m respectively P_q.

With D' being all data records that are neither sticky nor shift-able, our dataset D is composed of the following components:

$$\underbrace{\overbrace{D}^{Data}}_{n \times m} = \underbrace{D'}_{n \times (m-q-k)} \cup \underbrace{\overbrace{M}^{Shift\text{-}able\ Data}}_{n \times q} \cup \underbrace{\overbrace{Q}^{Sticky\ Data}}_{n \times k} \tag{3.6}$$

The control parameter ε addresses the problem of shift lack. Shift lack describes the phenomenon that points with a smaller norm can not be shifted as far as points with a larger norm could be shifted by the same shift vector Δp. By setting ε to a value between 0 and 1, norm equalization can be performed in order to map all points to the largest norm $\varepsilon = 0.5$, to the smallest norm $\varepsilon = 0$ or anything in between that. However, in this work, norm equalization is not performed. Thus, M_ε equals M.

Another user parameter δ controls the ratio between shift and stickiness. The weightings are defined by two polynomials ${f_1}^{\delta}$ and ${f_2}^{\delta}$ of degree 2 (see 3.7).

$${f_1}^{\delta} = 3 \cdot \delta - 2\delta^2, \ {f_2}^{\delta} = 1 - 4 \cdot \delta + 4\delta^2 \tag{3.7}$$

Setting this parameter to a value between 0 and 1 leads to different weightings. As they do not follow a linear distribution, three key values are presented:

- $\delta = 0 \quad \rightarrow$ maximum stickiness
- $\delta = 0.5 \rightarrow$ maximum shift
- $\delta = 1 \quad \rightarrow$ equally weighted

3.2.1. One-point Shift

A one-point shift consists of a single data record d selected as *shift-able* point m. At the same time, no other point is marked as *sticky*. Thus, the change ΔA of the resulting projection matrix $\overline{A}$ is only dependent on m and the user given shift vector Δp as shown in Formula (3.8).

$$\overline{A} = \left(\frac{1}{1 + \|m\|_2^2} \right) \Delta p \cdot m^T + A \tag{3.8}$$

3.2.2. Shift-able Sets

When shifting not only one point but a set of points, not every point m_i can be shifted exactly to $m_i + \Delta p$ due to the previously mentioned shift lack. Since there are no *sticky* points ($Q = \emptyset$) the main goal of this composition operator is to get a best possible projection of $m_i + \Delta p$ regardless of all other points D'.

The optimal projection composition operator for shift-able sets is calculated by Formula (3.9) and Formula (3.10).

$$\overline{A_\varepsilon} = \left[\Delta p \cdot (M_\varepsilon \cdot 1_q)^T \cdot H_\varepsilon^{-1} \right] + A \tag{3.9}$$

with

$$H_\varepsilon = (M_\varepsilon M_\varepsilon^T + I_n) \tag{3.10}$$

3.2.3. Shift-able Sets vs Sticky Sets

CO for a set of *shift-able* points M while having a set of *sticky* point Q try to achieve two goals:

(A) Keep a set Q as close as possible to their initial position.

(B) Shift a set M as close as possible to a desired location.

They do not necessarily contradict each other. However, if they do so, weighting can be controlled by the prior introduced parameter δ.

In case of multiple iterations of shifts, there are two different variants of calculating the respective Composition Operator. One without and one with a memory function for the *sticky* points Q. Hereby, memory divides the definition of initial position into the starting position of the prior shift (*without* memory) and the position before any shift has happened at all (*with* memory). That means that the Composition Operator *with* memory always keeps the starting configuration of *sticky* points as a reference.

Formula (3.11) with Formula (3.13) shows the Composition Operator without memory and with control ε and blending δ.

$$\overline{A}^{-} = \left[f_1^{\delta} \cdot \Delta p \cdot (M_\varepsilon \cdot 1_q)^T \cdot H_{\varepsilon\delta}^{-1} \right] + A \tag{3.11}$$

Formula (3.12) with Formula (3.13) shows the Composition Operator with memory, control ε and blending δ.

$$\overline{A}^{+} = \left[f_1^{\delta} \cdot \Delta p \cdot (M_\varepsilon \cdot 1_q)^T + f_2^{\delta} \cdot P_q Q^T) + A(I_N + f_1^{\delta} \cdot M_\varepsilon M_\varepsilon^T \right] H_{\varepsilon\delta}^{-1} \tag{3.12}$$

with

$$H_{\varepsilon\delta} = (f_1^{\delta} \cdot M_\varepsilon M_\varepsilon^T + f_2^{\delta} \cdot QQ^T + I_n) \tag{3.13}$$

For any additional information on CO as well as proofs of the formulas, please contact Prof. Dr.-Ing. habil. Dirk J. Lehmann.

4. Approach

The following chapter describes the different approaches to define a heuristic that manipulates a given projection into a projection of optimally separated classes. The central idea of all these heuristics is to maximize the visual distance between the involved classes. Therefore, all variants use the idea of centroids as a representation for classes (see chapter 3.1). Thus, the actual distance between two classes will be measured by the distance between their associated centroids. Furthermore, a central centroid C_C which is calculated by the mean of all other centroids is defined. This central centroid is an auxiliary quantity to calculate the direction of the shift vector Δp.

This shift vector Δp is influenced by three different components.

1. distance d_{ij} between a chosen centroid C_i and its nearest centroid C_j
2. distance d_{iC} between a chosen centroid C_i and the central centroid C_C
3. randomly generated noise factor, following a normal distribution between -1 and 1

While the first component will always be the defining element for the shift, the impact of the second and the third component are slowly fading with $\frac{1}{n}$ while iterating the heuristic. This reduces the impact of randomness towards convergence.

The shift vector Δp is calculated by the following Formula (4.1).

$$\Delta p = \underbrace{C_i - C_j}_{d_{ij}} + \frac{100}{100+n} \cdot \underbrace{C_i - C_C}_{d_{iC}} + \frac{100}{100+n} \cdot noise \tag{4.1}$$

The result is normalized, hence Δp has a length of 1 regardless of the calculated distances. Nonetheless, the direction of Δp is directly influenced by the magnitude of the distances. As the goal is to maximize distances between the centroids, short distances have to be weighted more than longer distances for the resulting shift. This aspect is not covered by the pure distance calculation. Thus, d_{ij} and d_{iC} are transformed to v_{ij} and v_{iC} by re-scaling them, as shown in Formula (4.2).

$$\Delta p = \underbrace{\frac{\|d_{iC}\|_2^2}{(\|d_{ij}\|_2^2 + \|d_{iC}\|_2^2)} \cdot d_{ij}}_{v_{ij}} + \frac{100}{100+n} \cdot \underbrace{\frac{\|d_{ij}\|_2^2}{(\|d_{ij}\|_2^2 + \|d_{iC}\|_2^2)} \cdot d_{iC}}_{v_{ij}} + \frac{100}{100+n} \cdot noise \tag{4.2}$$

Figure 10 shows an example with three different centroids. The centroid C_2 shall be shifted. Therefore, the distances d_{12} between C_2 and the nearest other class centroid C_1 and d_{2C} between C_2 and the central centroid C_C are calculated. The direction vectors v_{21} and v_{2C} imply the impact of the respective distances towards the resulting direction. The combination of these two vectors leads to the actual shift Δp for C_2.

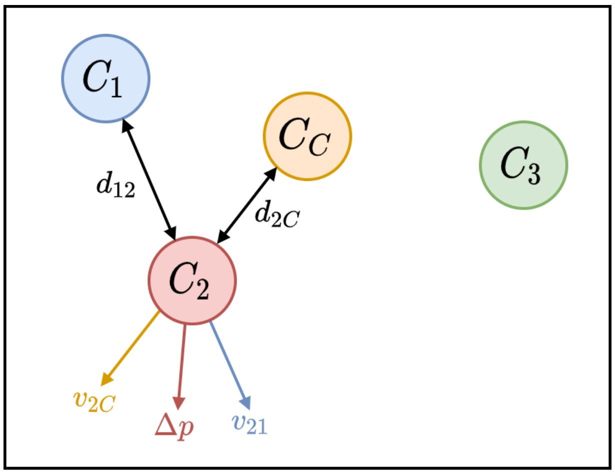


Figure 10.: Calculation of shift vector Δp in the heuristic

To use CO, the configuration of the points must be carried out in addition to the direction vector Δp. Hence, all projection points of a class that shall be shifted are marked as *shift-able*. Points of all other classes are marked as *sticky*. In case of a single point shift, only the point that shall be shifted is marked as *shift-able*. All other points, also those from the same class, are marked as *sticky*.

In the following sections, four different variations of this basic idea for a heuristic are described. Whilst some of them follow rather random instructions, others follow a more analytical approach.

4.1. Variant 1 - Random Selection Shift

The first and simplest approach randomly picks a centroid C_i and shifts it away from its closest other centroid C_j and the central centroid C_C. Thus, this variant is named Random Selection Shift (RSS). In order to find out which other centroid C_j is the closest to C_i all distances are computed, and the minimum is selected.

4.2. Variant 2 - Order Selection Shift

In difference to RSS, centroids are now moved in a given order. This leads to the name Order Selection Shift (OSS). Beginning with the centroid of the class with the lowest number (0 in most cases), centroids are shifted clockwise one after the other. When every class was shifted, it starts all over again. A centroid C_i is shifted away from its closest other centroid C_j and the central centroid C_C. In order to find out which other centroid C_j is the closest to C_i all distances are computed and the minimum is selected.

4.3. Variant 3 - Minimum Selection Shift

To determine which centroid should be moved, all distances d_{ij} between two different centroids are calculated. From these distances, the smallest single difference $min(d_{ij})$ between two centroids C_i and C_j is sought. This leaves open two possible candidates. Therefore, the total distances $d_{i,total}$ of C_i and C_j (see Formula (4.3)) to all other centroids are calculated as a second criterion. The one with the smaller $d_{i,total}$ is going to be the shifted centroid. Thus, this variant is called Minimum Selection Shift (MSS).

$$d_{i,total} = \sum_k \|C_{i/j} - C_k\| \; with \; k = 1, 2, ..., number \; of \; centroids \neq i/j \tag{4.3}$$

Taking a look back at Figure 10 the smallest distance $min(d_{ij})$ within this constellation is d_{12}. The third centroid C_3 is slightly nearer to C_2 than to C_1 which means that C_2 is the chosen centroid.

4.4. Variant 4 - Point Selection Shift

Instead of trying to maximize the distance between the cluster centroids, this variant tries to increase cluster density by moving single points towards their associated centroid. In order to select a point, all distances between projection points and the centroid of the class they belong to are computed. The maximum of all these distances $d_{iC,max}$ is the projection point p_i being shifted towards its centroid. Therefore, all other projection points are marked as *sticky.* By shifting a projection point towards its centroid, class density is being increased. As only a single point is shifted in this case, this variant is called Point Selection Shift (PSS).

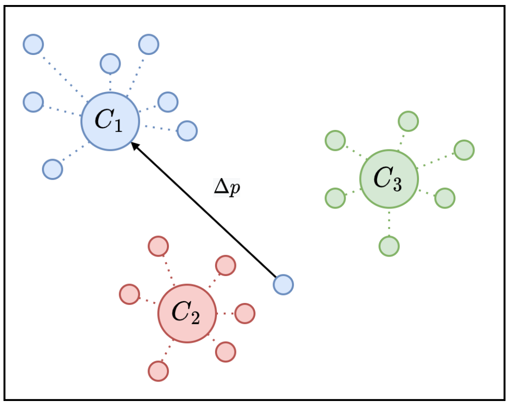


Figure 11.: Calculation of shift vector Δp in PSS

Taking the example from above, centroids C_{1-3} are visualized with their associated projection points in Figure 10. Measuring all the distances (dotted lines) leads to the result, that a point of the blue class has the maximum distance from its centroid C_1. Thus, this point is shifted towards C_1.

Besides shifting a selected projection point $p_{q,sel}$ straight towards its centroid C_q, Gaussian noise from -1 to 1 is added to vary the static shifts for new solutions. The actual shift can be calculated with Formula (4.4).

$$\Delta p = \|C_q - p_{q,sel}\| + noise \tag{4.4}$$

4.5. Termination Criteria of the Heuristic

Heuristics do not have a proof of convergence in general. Therefore, these procedures also lack a guarantee for finding any solution at all. From this insight, it can be deduced that termination criteria must be defined. [40]

The improvement of visualization by the heuristics can be measured by different approaches. One follows the idea of maximizing Distance Consistency (DSC) as presented in section 2.5. Another one is to maximize the distance of all class centroids. A last approach tries to minimize the distance between data points and their associated centroid, to increase the class density. Using these metrics leads to several termination criteria that can be defined for these heuristics. The simplest termination criterion for the heuristic is a fixed number of iterations n. This works for every presented variant. However, there are more complex termination criteria.

One of these is the distance between all centroids $d_{c,total}$. The total distance between all centroids $d_{c,total}$ is calculated by Formula (4.5).

$$d_{c,total} = \sum_{i=0}^{n-1} \sum_{j=i+1}^{n} \|C_i - C_j\| \; with \; i, j \in centroids \tag{4.5}$$

As long as the total distance between the centroids $d_{c,total}$ is increasing with every iteration, the heuristic should not terminate. A greater overall distance comes from greater individual spacing of the centroids, which allows for better separation. However, there are scenarios where for example only two classes are separated. A shift would always increase the distance between those two classes. Since the strategy of PSS counteracts this measure in its approach, it cannot be applied as a criterion for this heuristic. This leads to the conclusion that this criterion is not sufficient as the only one.

Another termination criterion is Distance Consistency [35]. Distance Consistency describes the ratio of projection points which are closer to the centroid of their class than to the centroid of any other class to the total number of projection points. A point that is closer to the centroid of its class than to the centroid of another class is considered distance consistent. A point that is closer to the centroid of another class is not distance consistent. With $p_{q,i}$ being the points i of centroid C_q, C_k being all other centroids and N being the number of all projection points and M the number of all centroids, DSC can be calculated by Formula (4.6).

$$DSC = \frac{|\{p_{q,i} : \ \|p_{q,i} - C_q\| \leq \|p_{q,i} - C_k\| \ \ \forall \, k \in |\{M\}|\}|}{N} \tag{4.6}$$

While other termination criteria are indirect metrics, increasing DSC leads to increasing separability. A value of DSC = 1 indicates an optimally separable projection and terminates the heuristic. However, for datasets with a lower amount of instances, DSC often does not change in every iteration.

Especially to measure and control the quality of PSS, a metric to control class density has to be used. The idea of Distribution Consistency as calculated in [35] does not quite fit the use-case. Thus, another metric to measure class density is used in this work. The distances between all points of a class towards their centroid is added up to a total distance $d_{i,c,total}$. The smaller this distance gets, the nearer points are towards their centroid, the higher class density is. For easier understanding, this measure is called Centroid Density (CD). Taking the notations from Formula (4.6), CD is calculated by Formula (4.7).

$$CD = \sum_q \sum_i \|C_q - p_q, i\| \tag{4.7}$$

It is not very helpful to use these criteria as hard factors for termination, since local minima could prevent the heuristic from finding a better solution. Even though a single criterion has not increased, the overall projection can lead to a better separability. Thus, a Penalty Threshold (PT) is introduced. In case not every termination criterion is fulfilled at once, a penalty counter (τ) is increasing until reaching the PT. However, an increase of DSC resets τ to zero, as the separability has improved.

In summary, the heuristic is to terminate when:

- a fixed number of iterations n has been processed
- $d_{c,total}$ and DSC decreased while CD increased from the last iteration
- penalty counter τ has reached the PT
- DSC reached 1

5. Implementation

This work is implemented in Python (Version 3.9), as it is a popular programming language with many useful packages, libraries, and frameworks especially for data science. The user interface was built with PySide6, which is the Python version of Qt. Qt is a well known user interface tool that is able to perform drag & drop actions. Furthermore, the principle of signals & slots is very intuitive to use. Datasets are handled with pandas. Pandas is a powerful and useful tool by providing many built-in data manipulation functions. Mathematical operations are done with numpy, because arrays are faster and more compact than lists. Additionally, numpy has a lot of built-in functions for matrix and vector operations. Basic mathematical operations are provided by math package.

5.1. Description of the Front End

Two different user interfaces have been implemented. One version to explore and interact with projections and get to know CO. The other one to set a framework to configure the application of heuristics to selected datasets.

5.1.1. Composition Operators - Basic application

In the sense of Visual Analytics Process (see section 2.1) it is important to get insights on data and the overall behaviour of CO. Thus, a basic application for user interaction was implemented as a QMainWindow (see Figure 12).

The user interface can be described as follows: On the left side there are different control elements, while on the right side the visualization of data is shown. Focusing on the left side of the interface, data can be selected by pressing the button "Choose .data file". By pressing "Choose .header file" the associated header can be chosen. Upon selection the nearby label updates the given path of the respective files. "Sticky" defines the weighting parameter δ (see section 3.2) by either moving the slider or entering a value into the line edit. Default value for δ is 1. Below, an option for orthographic scaling (see section 2.3) can be toggled by using the checkbox. By pressing the button "Start" prior configuration is taken to create a visualization on the right side of the interface.

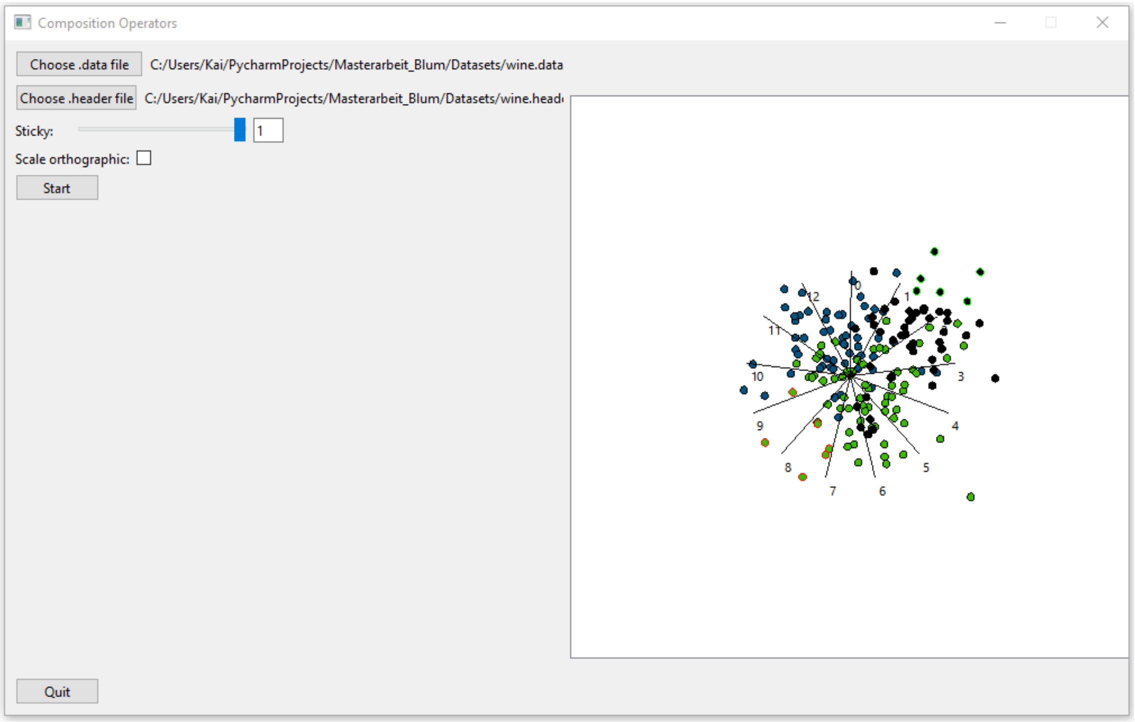

Composition Operators
Choose .data file
C:/Users/Kai/PycharmProjects/Masterarbeit_Blum/Datasets/wine.data
Choose .header file
C:/Users/Kai/PycharmProjects/Masterarbeit_Blum/Datasets/wine.head
Sticky:
1
Scale orthographic:
Start
Quit

Figure 12.: User interface for CO (interactive shifts)

The visualization is organized as a QGraphicsView in a QGraphicsScene. Points are drawn as QGraphicsEllipseItem while lines are drawn as QGraphicsLineItem. However, since a QGraphicsScene does not behave like a Cartesian coordinate system, projection points have to be transferred to the visualization plane. The connection between Cartesian coordinates and visualization plan is illustrated in Figure 13.

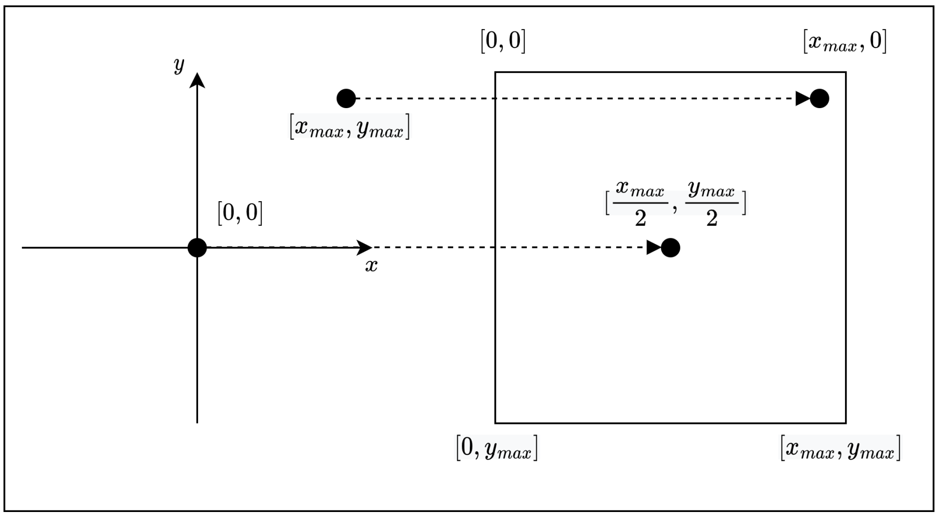


Figure 13.: Projection from Cartesian coordinates to visualization plane

Thus, the centre of visualization plane is defined as $[X_0, Y_0]$. These are derived from width ($X_0 = \frac{width}{2}$) and height ($Y_0 = \frac{height}{2}$) of QGraphicsScene. X and Y values of projection points can then be transferred by the following equations (5.1) and (5.2). The formula for Y differs because the y-axis is mirrored relative to the coordinate system.

$$X = X_0 + X_0 \cdot X_p \tag{5.1}$$

$$Y = Y_0 - Y_0 \cdot Y_p \tag{5.2}$$

A user can interact with the resulting visualization. Therefore, different actions can be performed.

- changing status of projection points between *sticky*, *shift-able* and *none* by double-clicking
- create a shift vector Δp by 'Drag & Drop' of one projection point
- reload the configuration by pressing the 'Start'-button again

While the class of a projection point is represented by its filling, the current status is indicated by the colour of the border (see Figure 12). Green borders indicate *shift-able* points (top-right corner, points of black class), red borders indicate *sticky* (bottom-left corner, points of green class) and black borders indicate that points are neither *shift-able* nor *sticky.* Drag & drop and thus creating a shift vector Δp is only available for non-sticky points.

As it is nearly impossible to always hit a data point perfectly on spot, detection of a click is realized by searching for data points within a certain radius r around the cursor position. In order to enable 'Drag & Drop' for elements a class GraphicsScene with the necessary signals and slots is inherited from QGraphicsScene, as Qt does not support this operation by default (Qt defines 'Drag & Drop' as a data transfer operation – as often used for uploads). Respective slots (mouseDoubleClickEvent, mousePressEvent and mouseReleaseEvent) have to be overridden for an inherited class GraphicsEllipseItem (from QGraphicsEllipseItem). The connected signals are defined in an inherited class MainWindow (from QMainWindow).

The drop of an item at a position that is not the starting position, emits the signal to proceed CO technique with the current configuration of the projection. The user input of Δp is calculated by the difference between the starting position on an emit of the drag signal and the ending position on an emit of the drop signal.

Applying Δp always takes place in visualization plane. Projection of data starts in Cartesian coordinates. This transition has to be reverted into Cartesian projection points by inverting the initial scaling of a new projection matrix $\overline{A}_{cartesian}$. Afterwards, the dataset is embedded into Cartesian SC with the new projection matrix $\overline{A}_{cartesian}$. Then it has to be transitioned back to visualization plane to update the visualization in every iteration. The process is illustrated in Figure 14.

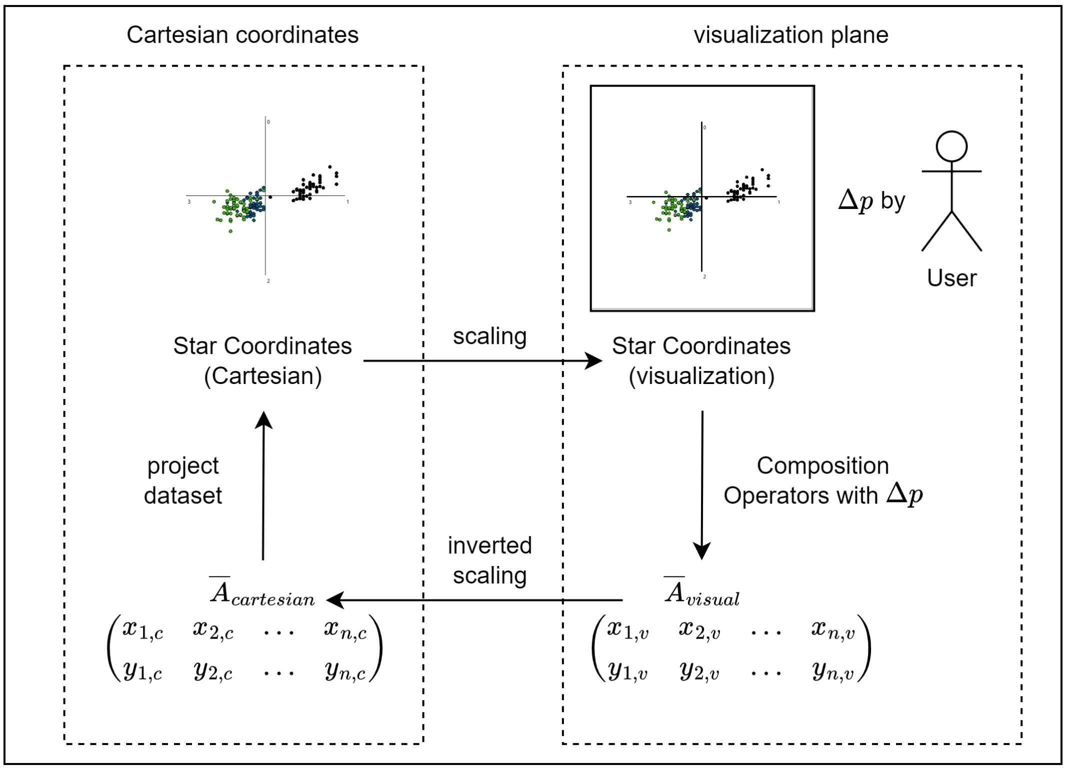

Cartesian coordinates
visualization plane
Δp by
User
Star Coordinates (Cartesian)
scaling
Star Coordinates (visualization)
project dataset
Composition Operators with Δp
inverted scaling
$\overline{A}_{cartesian}$
$\begin{pmatrix} x_{1,c} & x_{2,c} & \cdots & x_{n,c} \\ y_{1,c} & y_{2,c} & \cdots & y_{n,c} \end{pmatrix}$
$\overline{A}_{visual}$
$\begin{pmatrix} x_{1,v} & x_{2,v} & \cdots & x_{n,v} \\ y_{1,v} & y_{2,v} & \cdots & y_{n,v} \end{pmatrix}$

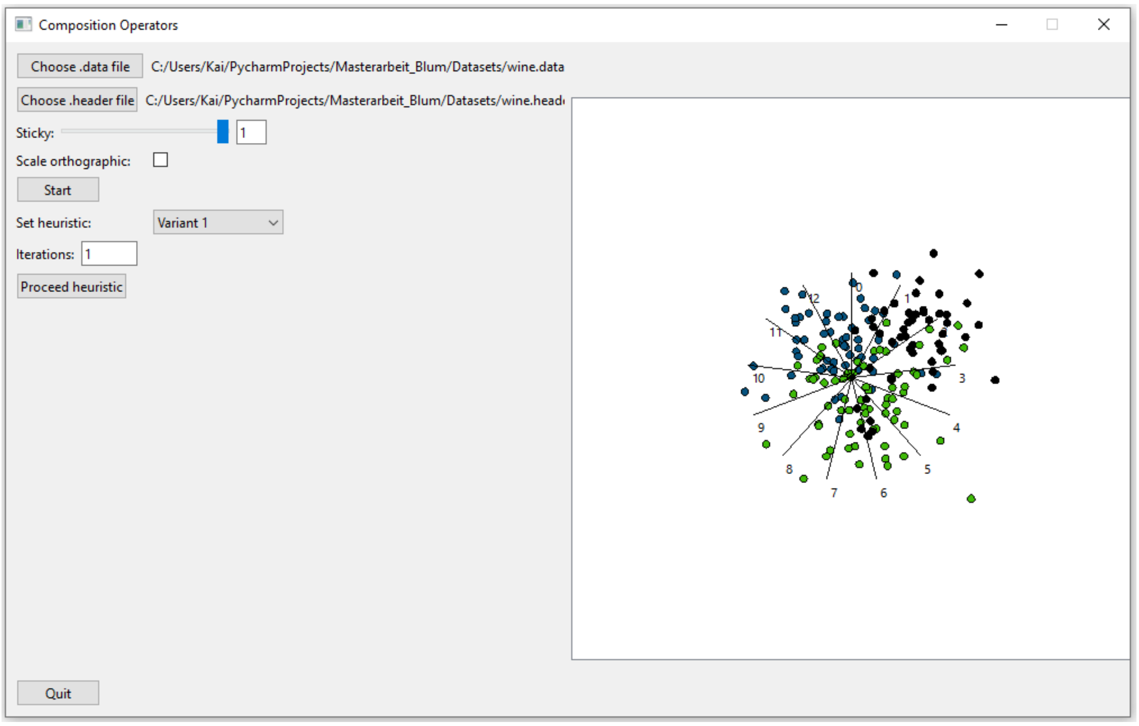
Figure 14.: Process of an iteration of visualization with a new projection matrix $\overline{A}$

5.1.2. Composition Operators - Heuristic Application

To apply the implemented heuristics and present the results accordingly, a second version of the user interface has been implemented. For this purpose, some features have been added and some have been deleted. The first part of the left side of the user interface is exactly the same as described in the section above. However, the second part extends the user input by relevant parameters for the heuristics.

A dropdown menu gives four different options to set a heuristic variant. These are equivalent to the variants presented in chapter 4. Furthermore, the number of iterations n for the heuristic can be defined by typing any number $n \in \mathbb{N}$. The default value is 1 (step). By pressing the button 'Proceed heuristic' n iterations of the chosen heuristic are proceeded and the result is shown in the visualization on the right. A continuous counter within the heuristic allows additional proceedings (even with a different amount of iterations n).

On the right side of the interface, interactive elements have been removed. Therefore, it is used exclusively for visualization. The resulting user interface is shown in Figure 15.


Composition Operators

Choose .data file C:/Users/Kai/PycharmProjects/Masterarbeit_Blum/Datasets/wine.data

Choose .header file C:/Users/Kai/PycharmProjects/Masterarbeit_Blum/Datasets/wine.head

Sticky: 1

Scale orthographic: ☐

Start

Set heuristic: Variant 1

Iterations: 1

Proceed heuristic

Quit

Figure 15.: User interface for CO (heuristic use)

5.2. Description of the Back End

The actual functionality of the software is triggered by the emit of the signal by the respective start button object. Functionality is handled within the connected slot. The following section will give an overview about the different heuristics and the functions that are used to realize them. Furthermore, the implemented penalty system will be explained. The implemented logic to handle user interface input will not be described in detail.

5.2.1. Description of Heuristics

In the following, pseudocode for MSS (see Figure 16) will be discussed. Since the general structure of the heuristics is quite similar, pseudocode for the other implemented heuristics can be found in Appendix A. Differences are marked with green color for the corresponding lines.

The general structure and process of the heuristic can be understood through the corresponding comments in the pseudocode. By using functions, the process is linear and easy to follow. For this reason, the remainder of this section concentrates on the functions behind the heuristics in chronological order. These contain the essential mathematical operations for carrying out the shifts. These functions are *calc_centroid_distances()*, *calc_dist_p_to_assoc_centroid()*, *calc_dsc()*, *selecting_shifting_class()* and *calc_dp()*. Some of them operate in a similar way but for different subjects. Thus, not every pseudocode will be shown in the following description.

Heuristic 1: Minimum Selection Shift - MSS

```
1 function heuristic_order_selection_shift(df_p, iterator, num_classes):
      Data: df_p is the dataframe in projection space, iterator is the current iteration
            step, num_classes is the amount of different classes in the dataset
      Result: dp shift vector, selected_class to be shifted, calculated DSC_value,
              calculated CD_value, calculated total_dist
      /* extract star coords and class of data to a new dataframe              */
2     df_star = df_p[['X','Y','class']]
      /* calculate class centroids and save them in a new dataframe            */
3     df_centroids = df_star.groupby('class', sort=True).mean().reset_index()
      /* calculate coordinates of the central centroid                         */
4     central_centroid = df_centroids[['X', 'Y']].mean()
      /* call a function to calculate all distances between centroids          */
5     df_distances = calc_centroid_distances(df_centroids)
      /* calculate the distance from each point to its associated centroid     */
6     df_centroid_distances = calc_dist_p_to_assoc_centroid(df_centroids, df_star)
      /* calculate CD, DSC and total_dist for result                           */
7     CD_value = df_centroid_distances['distance'].sum()
8     DSC_value = calc_dsc(df_centroids, df_star)
9     total_dist = df_distances['distance'].sum()
      /* find the minimum distance between two class centroids                 */
10    min_dist_idx = df_distances['distance'].idxmin()
11    min_class1 = df_distances.loc[min_dist_idx, 'class1']
12    min_class2 = df_distances.loc[min_dist_idx, 'class2']
      /* select the class that is to be shifted                                */
13    selected_class, other_class = select_shifting_class(df_distances, min_class1,
        min_class2)
      /* calculate the new shifting vector dp                                  */
14    dp = calc_dp(df_centroids, selected_class, other_class, central_centroid, num_iter)
```

Figure 16.: Pseudocode for MSS

The function *calc_centroid_distances()* calculates the euclidean distance between every combination of two different classes and saves them into a new dataframe. The pseudocode for this function is shown in Figure 17.

```
function calc_centroid_distances(df_centroids):
    Data: dataframe df_centroids with all positions of the centroids in projection space
    Result: dataframe df_distances with all distances between class centroids
    /* create dataframe for results                                          */
    df_distances = pd.DataFrame(columns=['class1', 'class2', 'distance'], dtype=float)
    /* index for writing to df_distances                                     */
    idx = 0
    /* calculate euclidean distance for every combination of centroids       */
    for index in list(combinations(df_centroids.index, 2)) do
        p1 = [df_centroids.loc[index[0], 'X'], df_centroids.loc[index[0], 'Y']]
        p2 = [df_centroids.loc[index[1], 'X'], df_centroids.loc[index[1], 'Y']]
        df_distances.loc[idx, 'class1'] = index[0]
        df_distances.loc[idx, 'class2'] = index[1]
        df_distances.loc[idx, 'distance'] = math.dist(p1, p2)
        /* increase the index before going through the next iteration        */
        idx = idx + 1
    end
```

Figure 17.: Function calc_centroid_distances

The function *calc_dist_p_to_assoc_centroid()* proceeds according to the same scheme. The difference is that it does not calculate the distance between the individual centroids, but the distance between points and their associated centroids. This leads to a dataframe that consists of every single data point linked with its associated centroid and the distance between those two points.

The functionality of *calc_dsc()* can be derived from Formula (4.6). Distances between a point and all centroids of the dataset are calculated, and the minimum distance is chosen. Depending on which point is closest to which centroid, the function calculates a score between 0 and 1.

The function *selecting_shifting_class()* calculates the total distance of the two nearest centroids to all other centroids in the projection and, based on the result, sets one of the two classes as selected_class and the other as other_class.

Finally, the function *calc_dp()* is an exact representation of Formula (4.2). The result of this function is the shift vector Δp. Since operations of this function are simple linear steps, this function needs to further explanation.

5.2.2. Description of Penalty Counter Calculation

A pseudocode for the function *handle_penalty_counter()* is given below (see Figure 18). This function manages the calculation of the current penalty τ. The old and new values of DSC, CD and total_dist (td) are passed to the function. The implemented conditions of this function are derived from the explanations in section 4.5.

```
function handle_penalty_counter($cnt, DSC\_o, DSC\_n, td\_o, td\_n, CD\_o, CD\_n$):
    Data: counter $cnt$ for the current penalty $\tau$, old and new values for $DSC$, $TD$ and
          $CD$ to compare towards termination criteria
    Result: new value $cnt$ for $\tau$
    if $DSC\_n \leq DSC\_o$ /* DSC decreased */
     then
        $cnt = cnt + 1$
    end
    if $td\_n \leq td\_o$ /* or td decreased */
     then
        $cnt = cnt + 1$
    end
    if $CD\_n \geq CD\_o$ /* or CD decreased */
     then
        $cnt = cnt + 1$
    end
    if $DSC\_n > DSC\_o$ /* DSC increased */
     then
        $cnt = 0$
    end
    /* DSC did not change and CD decreased or td increased */
    if $(DSC\_n == DSC\_o)$ & $((CD\_n < CD\_o) \mid (td\_n > td\_o))$ then
        $cnt = 0$
    end
    if $(CD\_n < CD\_o)$ & $(td\_n > td\_o)$/* CD decreased and td increased */
     then
        $cnt = 0$
    end
```

Figure 18.: Function handle_penalty_counter

6. Evaluation

This chapter deals with the evaluation of the developed heuristics. Thus, metrics and methods are introduced and explained. Beforehand, the datasets used are presented in detail.

6.1. Datasets Used for Evaluation

Different datasets have been used to evaluate the implementation described above. Selection criteria have been different amounts of almost exclusively or exclusively numeric attributes, number of instances, different number of classes, no missing values, a focus on classification tasks and already classified data. Furthermore, comparability to related work such as [23] or [28] has been one of the most important criteria for a dataset to be chosen. All datasets have no missing values. They can all be accessed via UCI Machine Learning Repository [9].

In order to prepare datasets for evaluation, they have been split into a data file (*dataset.data*) and a header file (*dataset.header*). If not already within the dataset, every dataset has been extended by a column *'class'* that represents the classification of each instance. To simplify processing, the class names were manually renamed with ascending numbering.

Header files contain the name of the columns in the first row and their corresponding data type in the second row. Numeric data types are named as *'number'*, the column *'class'* is marked as *'classifier'*. All other data types, e.g. string, are named as *'other'*. The following Table 2 shows the allocation of data types for *ecoli.header*. Entries in the header file are divided by *comma*, as they are read as a *.csv* formatted file.

seq_name	mcg	gvh	lip	chg	aac	alm1	alm2	class
other	number	number	number	number	number	number	number	classifier

Table 2.: Allocation of data types for *ecoli.header*

Before delving into each individual dataset, Table 3 provides a brief overview of their key data.

dataset	instances	attributes	numeric attributes	classes
iris	150	4	4	3
ecoli	336	8	7	7
wdbc	569	32	30	2
wine	179	13	13	3
yeast	1484	8	7	9
statlog	2000	36	36	6(7)

Table 3.: Key data on used datasets

6.1.1. Iris

Iris dataset is one of the most popular datasets which was contributed by R.A. Fisher. It contains 150 instances with 4 attributes. All of them are numeric. They are split into 3 different classes with 50 instances per class.

6.1.2. Ecoli

Ecoli dataset contains data on protein localization sites. It contains 336 instances with 8 attributes. 7 of them are numeric. The instances are split into 7 classes with different amounts of instances per class.

6.1.3. Wdbc

Wisconsin diagnostic breast cancer (wdbc) dataset contains data about characteristics of cell nuclei that are taken from breast mass. The dataset contains 569 instances with 32 attributes. 30 of them are numeric. The instances are split into 2 classes with different amounts of instances per class.

6.1.4. Wine

Wine dataset is a well-known dataset containing chemical analysis of different wines in order to determine their origin. It contains 179 instances with 13 attributes. All of them are numeric. The instances are split into 3 classes with different amounts of instances.

6.1.5. Yeast

Like ecoli, yeast contains data on protein localization sites. It contains 1484 instances with 8 attributes. 7 of them are numeric. The instances are split into 9 classes with different amounts of instances per class.

6.1.6. Statlog

Statlog dataset contains data on '[..]multi-spectral values of pixels in 3x3 neighbourhoods in a satellite image, and the classification associated with the central pixel in each neighbourhood' [9]. It contains 2000 instances with 36 attributes. All of them are numeric. The instances are split into 7 classes with different amounts of instances. However, since class 6 does not have any instances included, statlog is considered having 6 classes.

6.2. Design of Evaluation

Evaluation of the heuristics will consist of three different parts:

1. Comparison of the four presented heuristics with a default setup
2. Analysis of behaviour of the same heuristics by changing different parameters
3. Comparison of the results with other results e.g. [27]

These aspects will be discussed on the different datasets with different metrics and visualizations.

6.2.1. Metrics of Evaluation

Apart from visualization, a few aspects can help to measure the quality of the used heuristics. While some of these aspects are metrics as a direct measure of the outcome of the heuristics, other aspects deal with the reliability and reproducibility of the results.

Distance Consistency

DSC is a metric to measure how many projection points of a class are closer to its own centroid than to the centroid of another class (see section 4.5). A projection that has a DSC value of 1 is fully separated, as all projections points lie closest to their own centroid. However, even for a DSC below 1, fully separation can be achieved. Thus, DSC is especially meaningful for compact classes. Figure 19 illustrates an example for optimal separation of classes while DSC is smaller than 1.

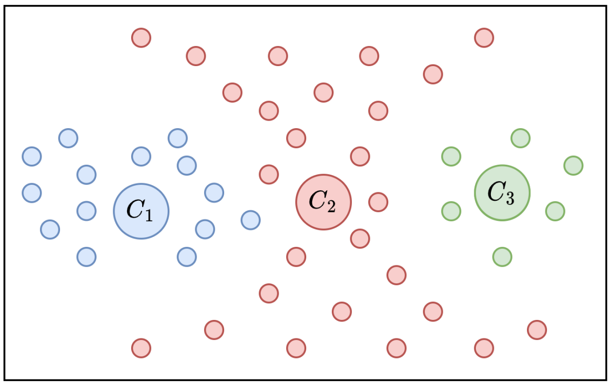


Figure 19.: Example for optimally separated data with a DSC below 1

In the following evaluation, the value is given as a percentage for better readability. Thus, 1 is equivalent to 100%.

Centroid Density

CD is an introduced metric to measure the distance of projection points towards their associated centroid (see section 4.5). The distance is the sum of all euclidean distances between projection points and their associated centroid. A lower distance CD leads to higher density.

$$\Delta CD = \frac{CD_{start}}{CD_{end}} \cdot 100\% \tag{6.1}$$

For evaluation, the change of density ΔCD will be measured by forming the dividend between the starting CD_{start} of the initial projection and the resulting CD_{end} of the final projection (see Formula (6.1)). A value of more than 100% means that density has increased, a value under 100% indicates a decrease of overall density.

Centroid Distance Change

The total distance between the centroids of the classes $d_{c,total}$ can be measured by the sum of the euclidean distances between every combination of two centroids in projection space. To compare the result of the heuristics, the change of this distance Centroid Distance Change (CDC) is measured as:

$$CDC = \frac{d_{c,total,end}}{d_{c,total,start}} \cdot 100\% \tag{6.2}$$

Despite a greater distance (CDC $>$ 100%) between the centroids, better separability is not guaranteed. It can also not have increased or even decreased depending on the distribution of points. Therefore, CDC is more of an indicator than a direct measure for separability.

A combination of CDC and ΔCD would operate like Fisher's method of scoring as described in [11]. Since it is unclear in what way the two different metrics interact or contradict for information on quality of an outcome in this work, they are measured separately.

Convergence and Reliability

Since heuristics do not guarantee a solution, termination criteria have been defined in section 4.5. To test these, a sample of runs of a heuristic is performed. This sample can then be analysed by its estimation of the expected value and the sample variance. The sample variance of the number of iterations gives an indication of the quality of the termination criteria. Furthermore, the sample variance of DSC can be used as a measure for reliability of the presented solution. The estimation of the expected value $\overline{X}$, sample variance s^2 and the variance coefficient v, a resulting ratio, are mathematically defined as:

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i \quad ; \quad s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X}) \quad ; \quad v = \frac{s}{\overline{X}} \tag{6.3}$$

A low sample variance indicates high reliability of presented results. Accordingly, a result is a good representative for the set of all results. Variance coefficient v can be used to get information on the relative variance, as it does not depend on the actual size of $\overline{X}$. Since different datasets are expected to result in different magnitudes of variance, the use of this measure is beneficial.

6.2.2. Initial Configuration for Evaluation

For comparison, an initial configuration of a dataset with equally distributed axes of the same length is created. Figure 20 illustrates the initial projection in SC for all used datasets.

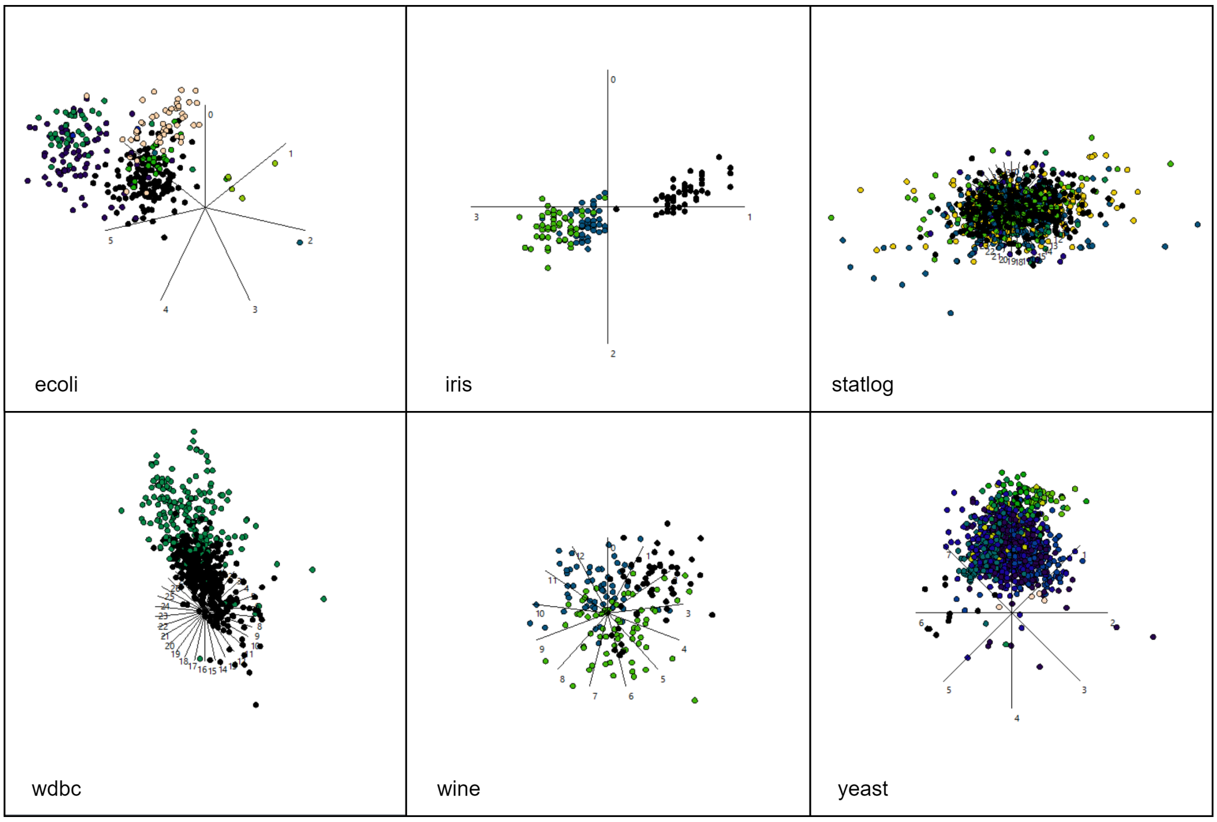


Figure 20.: Datasets visualized in their initial SC projection

The following Table 4 summarizes key values of the initial projection for each dataset.

dataset	DSC_{start}	CD_{start}	$d_{c,total,start}$
ecoli	63.88%	11636	3073
iris	89.93%	3585	340
statlog	21.06%	65743	112
wdbc	86.27%	22820	79
wine	72.32%	7703	201
yeast	27.44%	44547	2613

Table 4.: Key values of the initial projection for each dataset

Based on the values shown in Table 4 and the visualizations illustrated in Figure 20 it can be said that the used datasets differ greatly from each other in every measured aspect. These key values of the initial projection depend on the properties of the data sets. The values for DSC range from 21.06% to 89.93%. CD for statlog is almost 19 times the value for iris dataset. The distance between the centroids ranges from 79 to 3073. The dimension of individual key values does not allow any conclusions to be drawn about the dimension of other key values. This shows that the values in absolute numbers are not decisive, but rather their relative change must be considered as a metric for separation.

6.2.3. Comparison of Heuristics

In order to get a first comparison between the heuristics, a test configuration has been set. All datasets have been tested with the same configurations throughout this example. Table 5 shows the settings.

Max. Number of Iterations	Penalty Threshold	Orthographic
1000	100	Yes

Table 5.: Standard configuration for heuristic comparison

6.2.4. Behaviour with Different Parameters

After comparing the presented heuristics with a standard configuration, they will be tested with modified configurations in the following section. In addition to modifying termination criteria, orthography will be used or not used in different scenarios. Since overall results for MSS are superior to those of RSS, OSS and PSS, it is taken as the representative for centroid shifting heuristics in further investigation.

Modification of Termination Criteria

A strongly influencing degree of freedom are the termination criteria. They can be modified by leaving some of them out, changing the PT, changing their composition or their weighting in PT calculation. Table 6 to 8 show configurations for the related tests.

A reduction of the PT should lead to earlier convergence, as the heuristics are less fault tolerant.

Max. Number of Iterations	Penalty Threshold	Orthographic
1000	25 | 50	Yes

Table 6.: Configuration for heuristic comparison with reduced PT

Instead of decreasing PT it is increasing to a higher value. A higher fault tolerance should lead to later convergence.

Max. Number of Iterations	Penalty Threshold	Orthographic
1000	150 | 200	Yes

Table 7.: Configuration for heuristic comparison with increased PT

As heuristics do not terminate by their design, removing all termination criteria leads to a lack of convergence at all. Thus, the heuristics will always stop after a given number of iterations. There is no guarantee that a result for this configuration corresponds to a local or global maximum, as the heuristic stops regardless of prior results.

Max. Number of Iterations	Penalty Threshold	Orthographic
500 | 1000 | 2000	None	Yes

Table 8.: Configuration for heuristic comparison without PT and different iterations

In the basic version of these heuristics, each penalty leads to an increase of the penalty counter τ by one. This calculation stays the same over the entire running time. Changing this weighting with increasing iterations leads us to the principle idea of Simulated Annealing [22]. In this case, the penalty counter τ is increased by 1 for every 10% of the total maximum iterations.

$$\tau_{new} = \tau_{old} + 1 + \left(10 \cdot \left\lfloor \frac{i}{i_{max}} \right\rfloor\right) \tag{6.4}$$

This alternative calculation rule is tested with the standard configuration.

Change from Orthographic to Non-Orthographic Representation

Instead of creating an OSC representation for the result, it will be shown as Non-orthographic Star Coordinates. This leads to an increase of size of the axes with great influence on the separation. Therefore, scaling must be adapted for this visualization. A more detailed discussion of the advantages and disadvantages of the two types of representation has already been carried out in section 2.5 on a theoretic basis. These assumptions should be proven by tests. Table 9 shows the configuration for non-orthographic representation.

Max. Number of Iterations	Penalty Threshold	Orthographic
1000	100	No

Table 9.: Configuration for heuristic comparison without orthography

6.3. Results of Evaluation

The following section shows the results of the evaluation. First, the different heuristics are compared with each other. Then, different configurations are compared using MSS. The comparisons are made qualitatively (visually) and quantitatively (using the metrics listed). As it is one of the key components for this thesis, the following results will show many data visualizations. Evaluation is done on a 64-bit system with AMD Ryzen 5 3600 6-core CPU with 16 GB DDR4 RAM and AMD Radeon RX 5700 GPU with 8 GB VRAM. The operating system is Windows 10.

6.3.1. Results for Comparison of Heuristics

Comparison of the different heuristics is organized in two parts. First, qualitative results for each dataset used are shown by direct comparison of the heuristics. Then quantitative results are presented for each heuristic by direct comparison of their performance for every dataset used.

Qualitative Comparison of Heuristics

In the following, datasets will be discussed in alphabetical order. The visualizations are ordered TLBR from variant 1 – RSS to variant 4 – PSS. Different colours of data points in projection space signalize different classes. All visualizations have the same level of zoom and the exact same section of the visualization was taken.

The first dataset is ecoli. Looking at Figure 21, it can be seen that there are only slight improvements in separation. Especially for MSS, classes of same colour are a bit denser than for the other heuristics. Overall, however, there are no significant visual differences between the heuristics for this dataset.

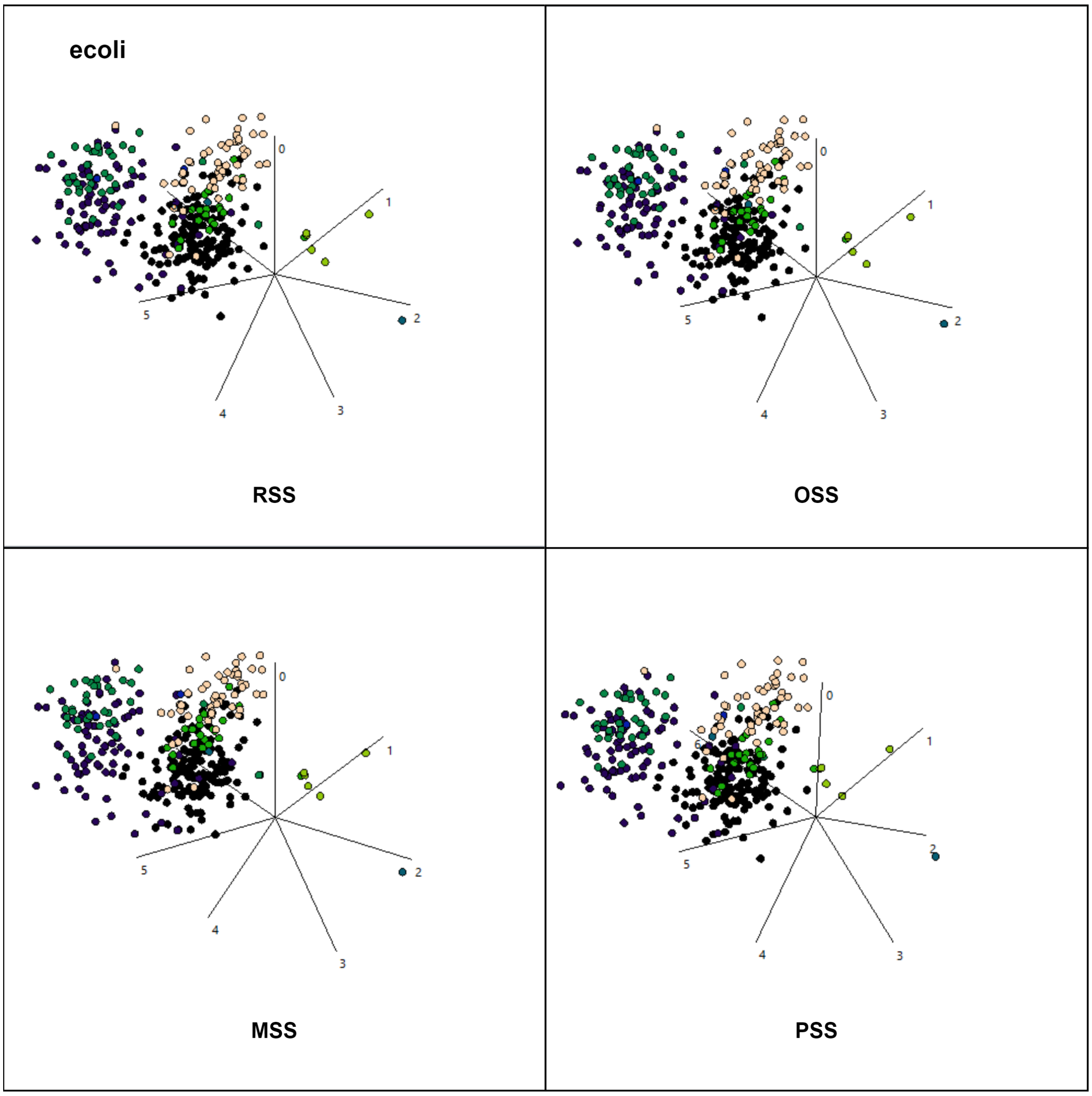


Figure 21.: Results for ecoli dataset with standard configuration

For iris dataset, illustrated in Figure 22, RSS, OSS and PSS seem to have low impact. This can be seen in the very small change in the axes in the projection. For MSS, the black class shifts with axis 1 towards the bottom of the projection. The overall distance between the classes has increased. Furthermore, the green and the blue class are more separated than before. Axis 0 has decreased significantly in size. From this, it can be deduced that this attribute has a minor influence on separation of the given classes.

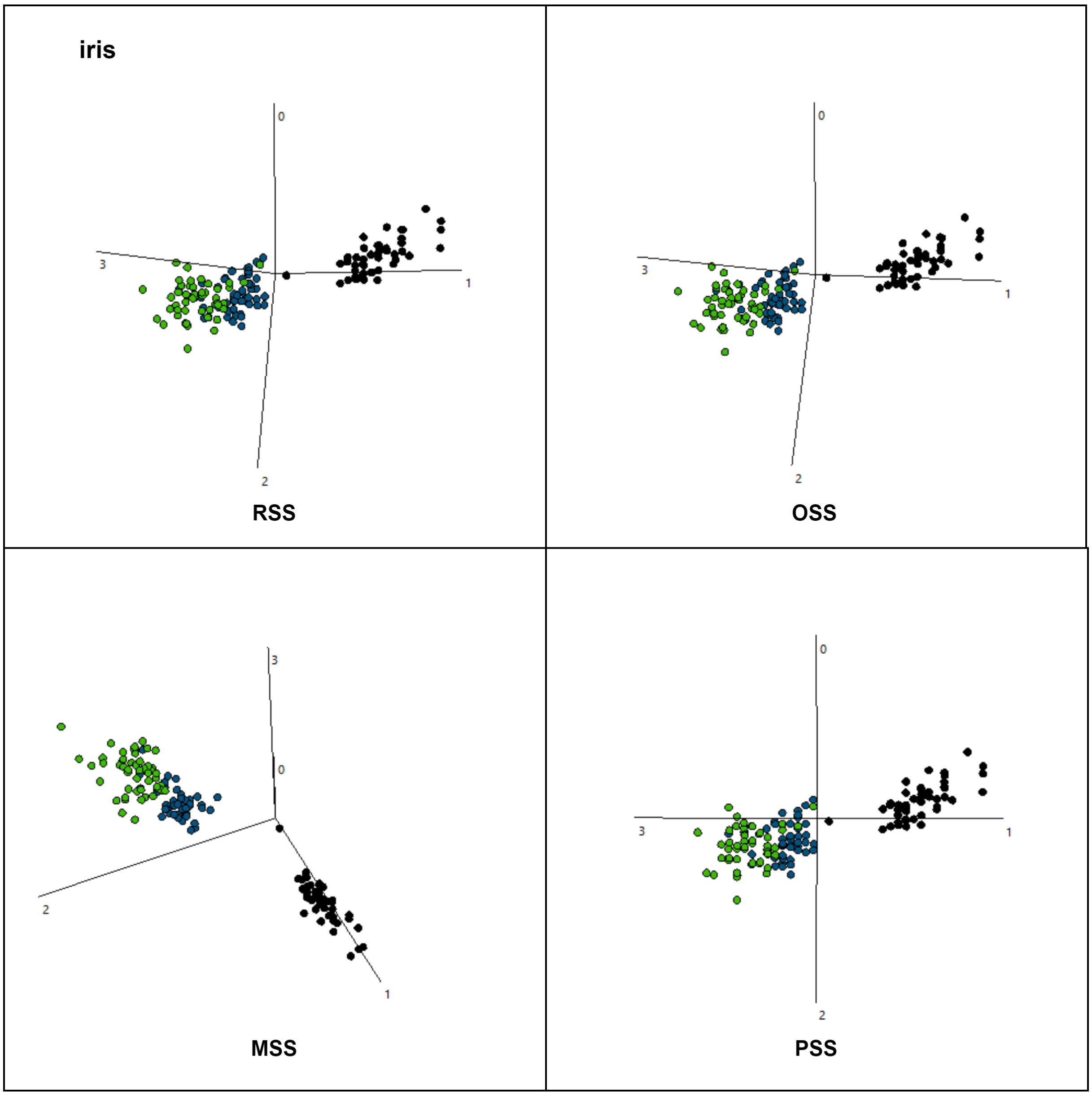


Figure 22.: Results for iris dataset with standard configuration

A similar observation can be made for statlog dataset. Results for statlog are illustrated in Figure 23. MSS leads to a reasonable increase of separation, while all other heuristics do not perceivably affect the projection in terms of separation. PSS causes a higher density of the overall projection. Because of the high density of data points in projection space, any statements about the impact of individual axes — especially for those with small impact — can hardly be made.

This is particularly difficult because the coordinate origin is completely hidden by data points. Therefore, it is visually difficult to relate axis lengths.

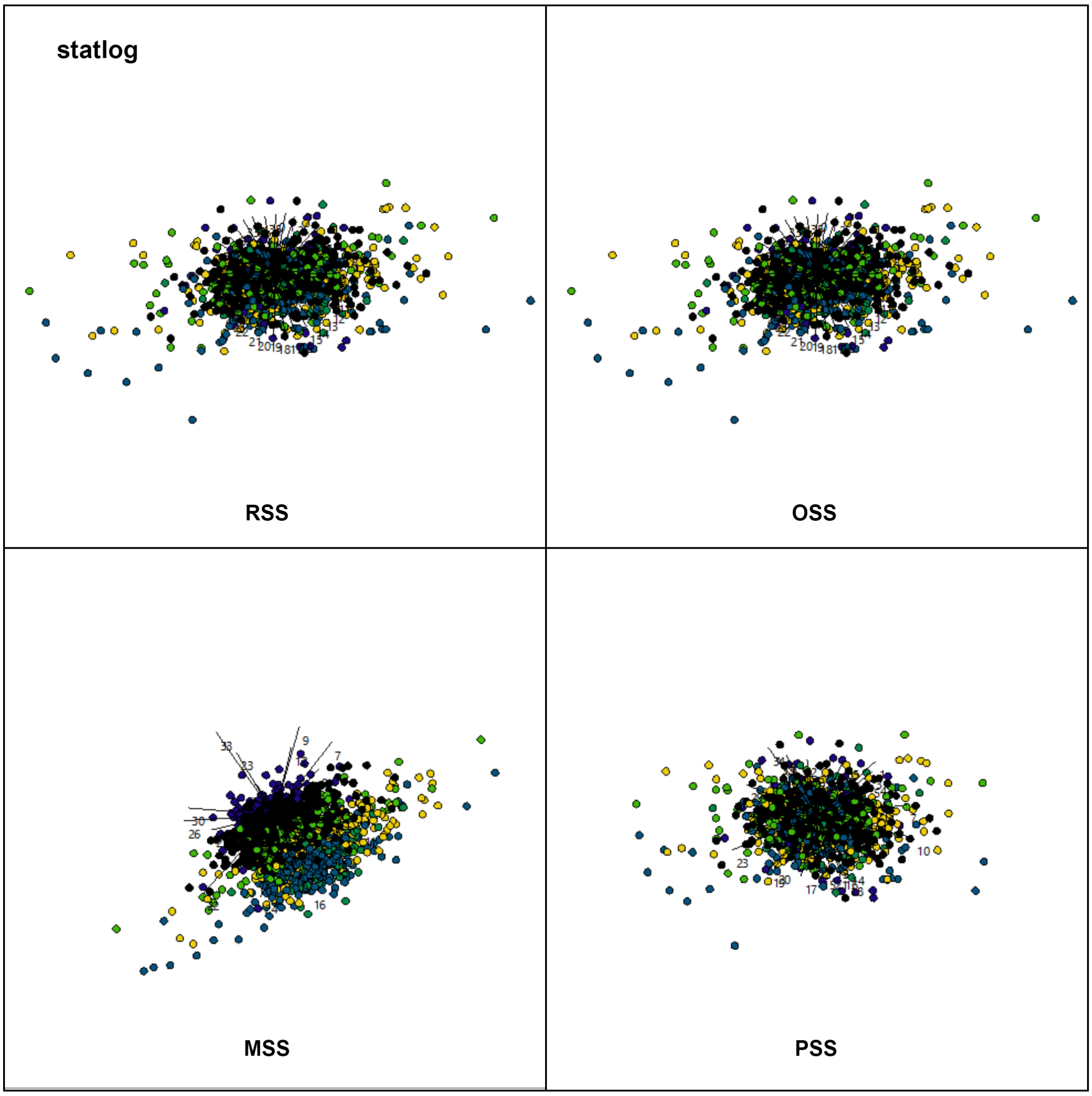


Figure 23.: Results for statlog dataset with standard configuration

Results for wdbc are shown in Figure 24. As expected for a dataset with two classes, the impact of the implemented heuristics on wdbc is quite high. The results are very different from the initial projection for every proposed heuristic. Results for RSS and OSS are quite similar. The projection for MSS seems to be separated better than the previous two, as the overlap is a lot smaller. For PSS the classes are very compact, but the overlap is quite high. Thus, separation is worse than for MSS.

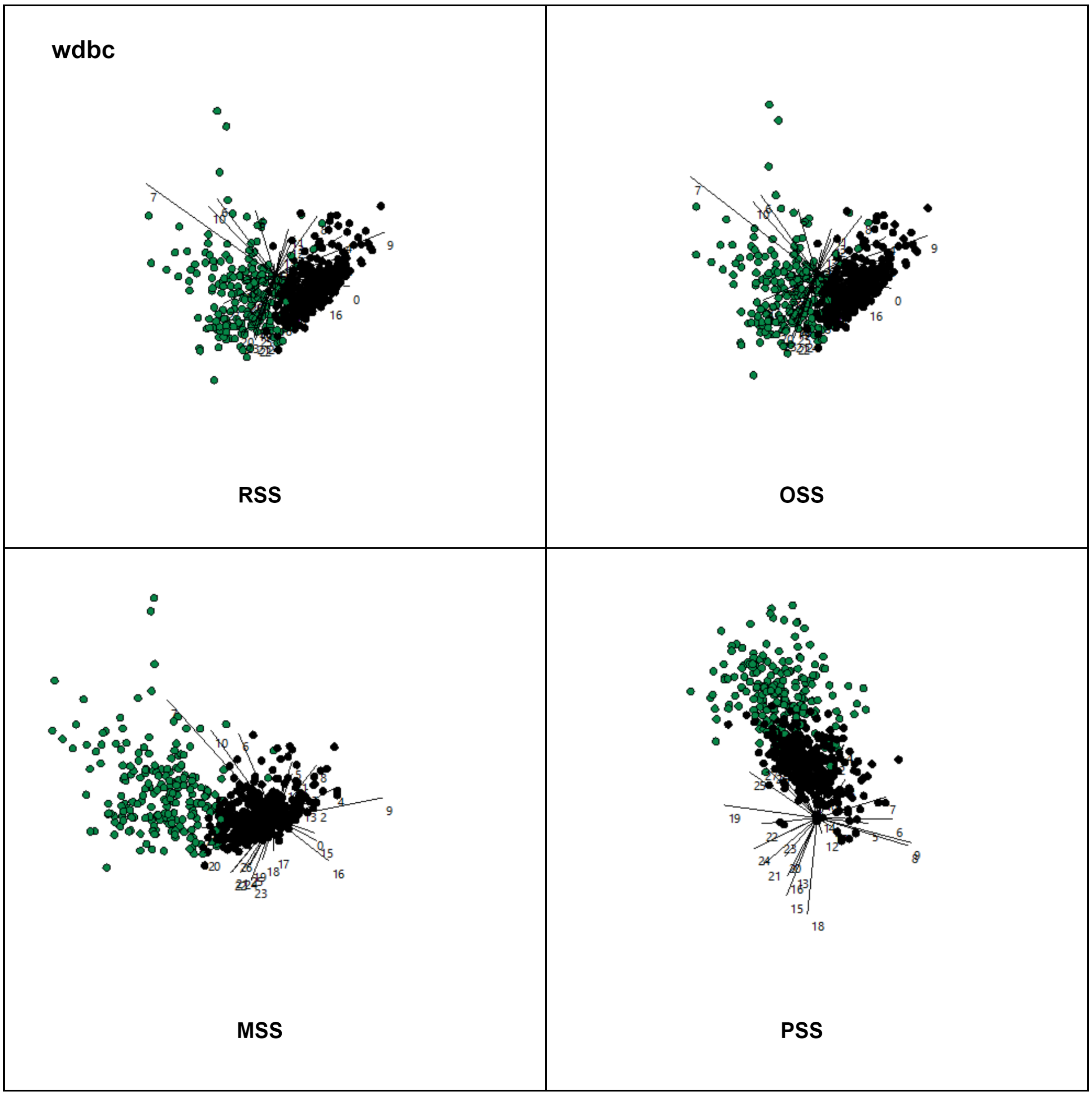


Figure 24.: Results for wdbc dataset with standard configuration

For wine dataset (see Figure 25), results for RSS, OSS and MSS seem to be quite similar on the first view. All three heuristics provide a good separation. However, there is still overlap between the classes for these methods. MSS creates an overall denser visualization for the dataset than RSS and OSS. For PSS, on the other hand, the resulting projection seems to be optimally separated. There is no visible overlap between the three different classes in projection space.

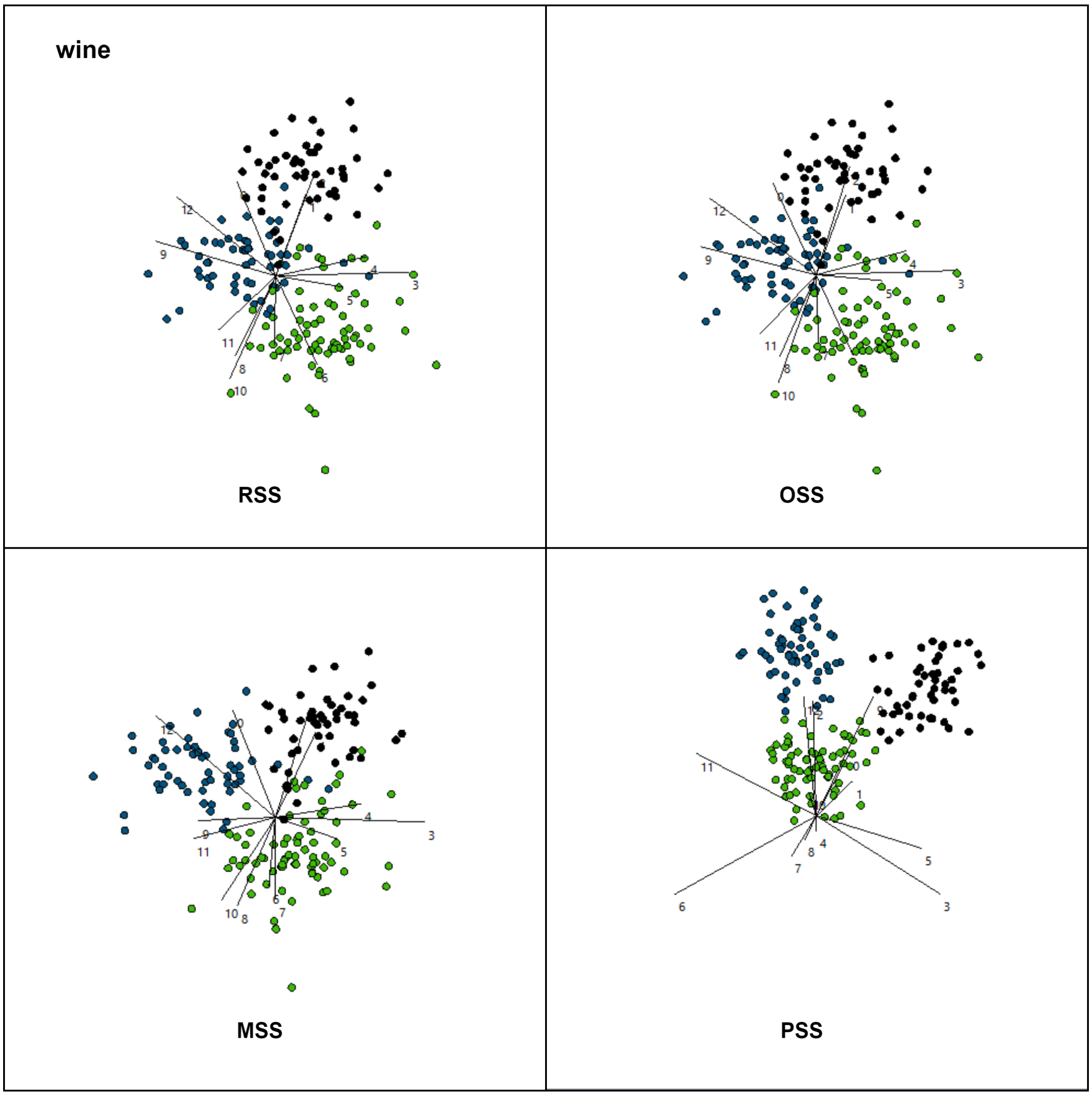


Figure 25.: Results for wine dataset with standard configuration

For RSS and OSS, it seems like they had no impact on the visualization for yeast dataset, as there is no visible change of the axes in projection space (see Figure 26). MSS and PSS show a slightly better separation than the initial projection. While both heuristics manipulated the projection, the ratios and angles of the axes are very different in both cases.

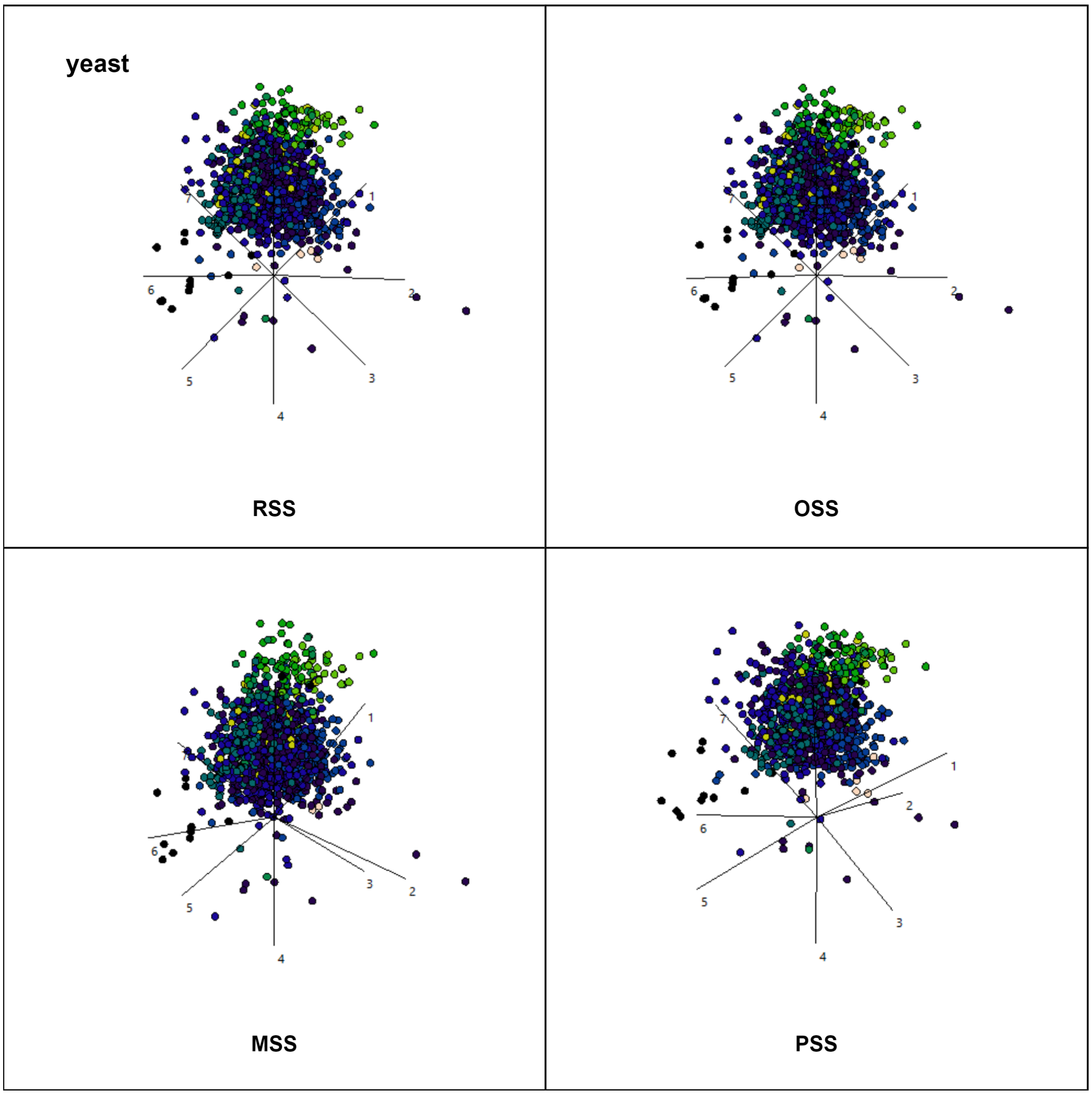


Figure 26.: Results for yeast dataset with standard configuration

Quantitative Comparison of Heuristics

The next step is the quantitative comparison of the four heuristics. Therefore, two tables have been prepared for every dataset. One consists of the key values for each dataset, proposed for every heuristic. The second one shows the actual impact of the heuristics on the key value for each dataset. As all important information for the evaluation is already contained in the second table, the first is not considered further. For the sake of completeness, the first tables can be found in Appendix B. Quantitative results will be related to the qualitative findings.

Table 10 shows the resulting differences for the defined metrics for RSS. RSS has rather small increases for ΔDSC in most cases and even a decrease for yeast dataset. However, for some datasets, ΔDSC has been increased significantly. The impact of ΔDSC correlates with a high number of iterations in most cases.

For statlog CDC has almost doubled, while it has even been reduced for wdbc. For both ΔDSC has been improved by an approximately same percentage. The overall number of iterations for RSS is very low.

With a Penalty Threshold of 100, the heuristic must have been terminated by producing worse results in every measured aspect before possibly reaching the threshold. However, since the shifted centroid is chosen completely at random for every iteration, sample runs can vary widely. Thus, the overall performance of RSS is not reliable.

dataset	ΔDSC	ΔCD	CDC	iterations
ecoli	1.19%	99.87%	100.42%	12
iris	0%	102.49%	105.29%	75
statlog	7.7%	99.67%	198.21%	64
wdbc	8.45%	138.83%	83.54%	852
wine	20.34%	102.17%	175.62%	423
yeast	-0.06%	99.19%	100.84%	33

Table 10.: Impact on key values in comparison to initial projection for each dataset by RSS

According to Table 11 the results for OSS do not negatively affect the separation for any dataset. However, OSS does not improve the results significantly more than RSS. Except for a much worse result for statlog, the overall results for OSS relate to those from RSS. Since a displacement of centroids is not carried out on the basis of the relative position in the projection space, but by a fixed rule, many unfortunate shifts occur in the process.

dataset	ΔDSC	ΔCD	CDC	iterations
ecoli	0%	99.86%	100.16%	8
iris	0.67%	102.92%	107.06%	79
statlog	1.7%	99.93%	117.86%	13
wdbc	8.45%	139.15%	82.28%	685
wine	20.9%	102.04%	177.61%	456
yeast	0%	99.63%	100.11%	18

Table 11.: Impact on key values in comparison to initial projection for each dataset by OSS

In contrast to RSS and OSS, Minimum Selection Shift takes the relative position in the projection space into consideration. The process of separation for wdbc dataset by MSS is illustrated in

Figure 27.

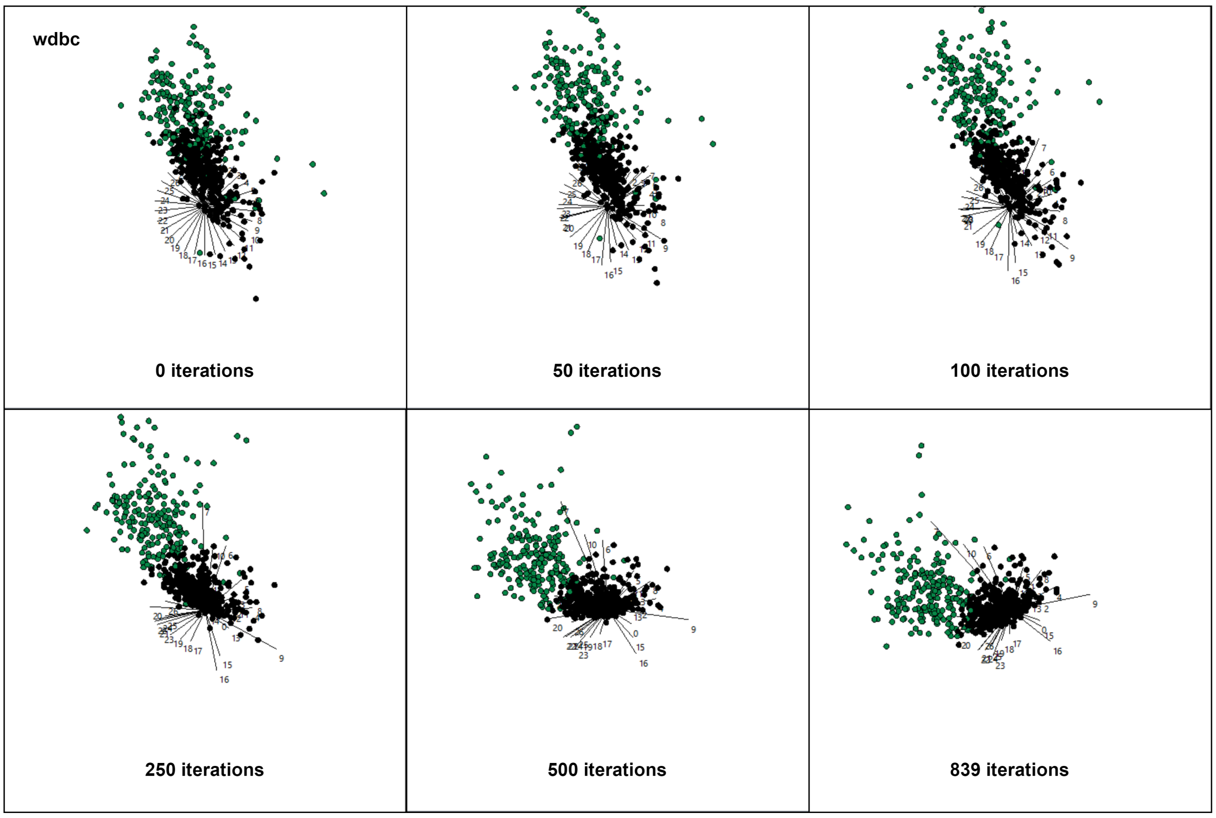


Figure 27.: Process of separation for wdbc dataset by MSS

According to Table 12, MSS leads to the best overall results. Apart from ΔDSC for yeast dataset, every single key value has increased compared to the initial projection. Therefore, according to the metrics, MSS leads to a better separation for each dataset. The overall trend of the impact on a dataset follows the pattern of RSS and OSS except for statlog, which showed a disproportionately large increase of separability.

dataset	ΔDSC	ΔCD	CDC	iterations
ecoli	3.88%	100.79%	102.05%	214
iris	3.36%	115.24%	130.59%	551
statlog	20.21%	104.97%	370.54%	396
wdbc	10.03%	121.54%	124.05%	858
wine	20.9%	104.25%	191.04%	499
yeast	0.88%	96.54%	105.82%	68

Table 12.: Impact on key values in comparison to initial projection for each dataset by MSS

PSS does not shift centroids away from the centr<e, but data points towards their associated

centroid. This is a contrasting approach to the other previously presented heuristics. The process of separation for wine dataset by PSS is illustrated in Figure 29.

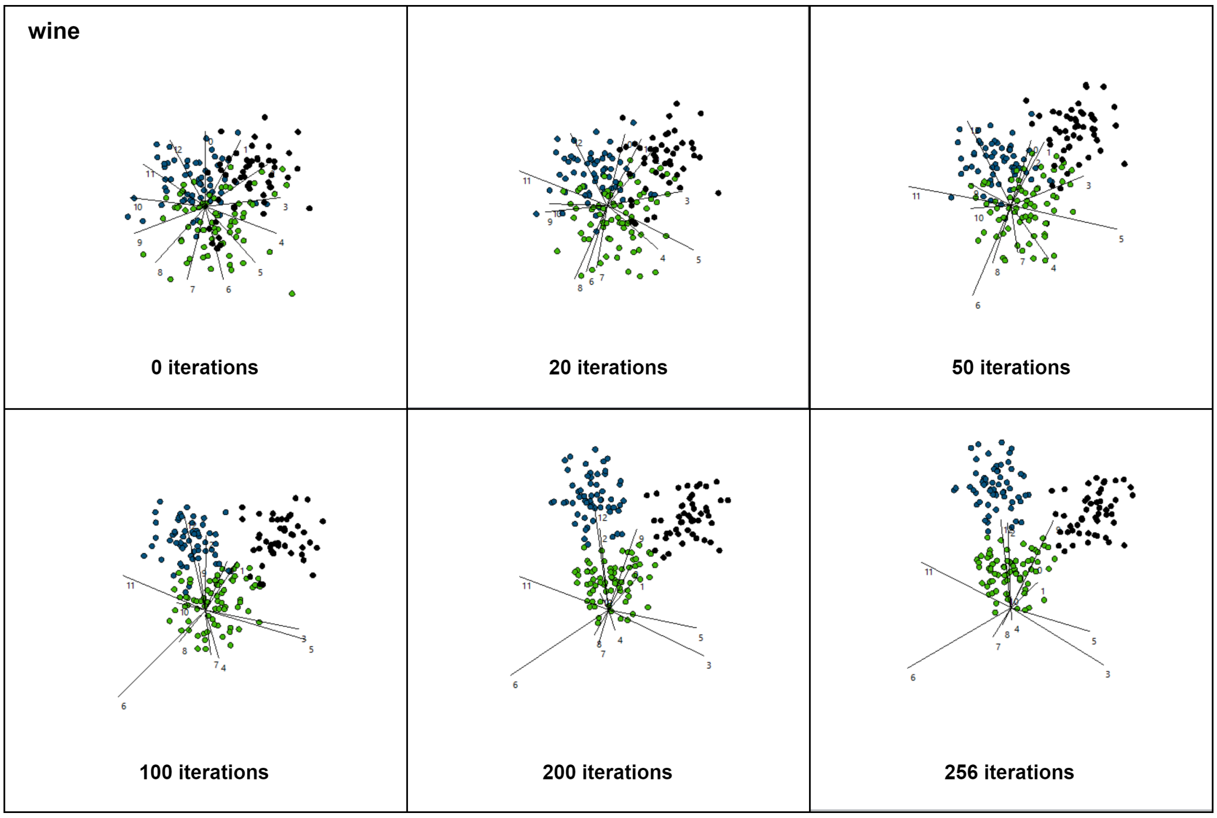


Figure 28.: Process of separation of wine dataset by PSS

Quantitative results for PSS show, that it is a double-edged sword. While it results in an optimal separation for wine dataset, the overall performance for PSS is below the other proposed methods. Looking at the projection of the wine dataset in comparison to the other datasets, the centroids of the classes lie along different axes. This fact is extremely helpful to PSS in its way of working.

dataset	ΔDSC	ΔCD	iterations
ecoli	-0.49%	99.21%	3
iris	0%	100%	3
statlog	1.45%	113.94%	20
wdbc	5.1%	129.91%	53
wine	27.68%	136.55%	255
yeast	-2.63%	101.41%	15

Table 13.: Impact on key values in comparison to initial projection for each dataset by PSS

Taking a closer look at the result for wine dataset, the classes can be visually separated by lines between them. Figure 29 illustrates the separation with red lines. It was thus shown, that an optimally separating linear projection exists. After achieving this results, another 150 runs have proven that this result can be achieved reliably in every run. However, proof is only given empirical, not mathematical.

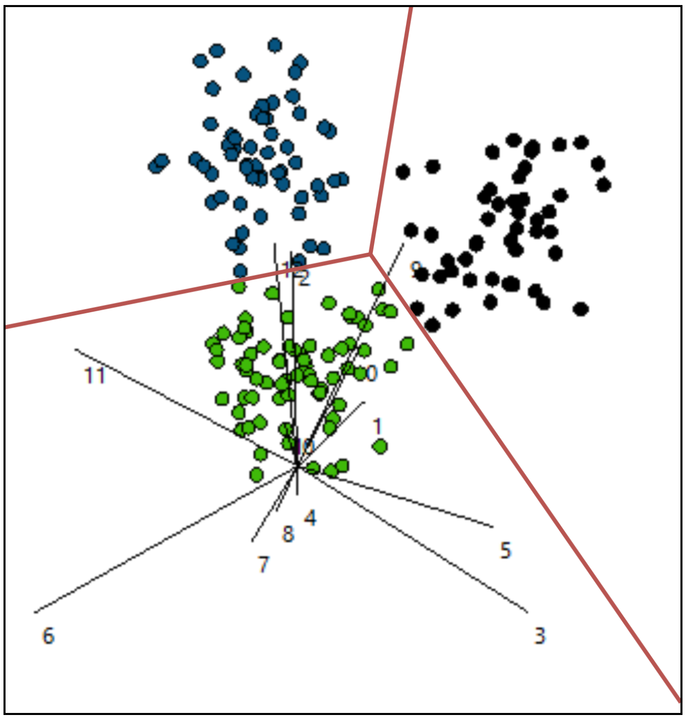


Figure 29.: Optimally separated projection of wine dataset by PSS

6.3.2. Results for Behaviour with Different Parameters

As MSS has shown the overall best performance for the tested datasets, behaviour with different parameters will be tested with this heuristic only. For better readability, discussion of visualization and measures will be done at the end of the section. This has the advantage that measured values and visualisations can be displayed together at the same time.

The results include 54 illustrations of 6 different datasets in 9 different parametrisations. In order to make sure a measure is no complete outlier, 5 runs for each dataset in each parametrisation have been carried out. However, an analysis for convergence and reliability will be done in the next section.

Results for Ecoli Dataset

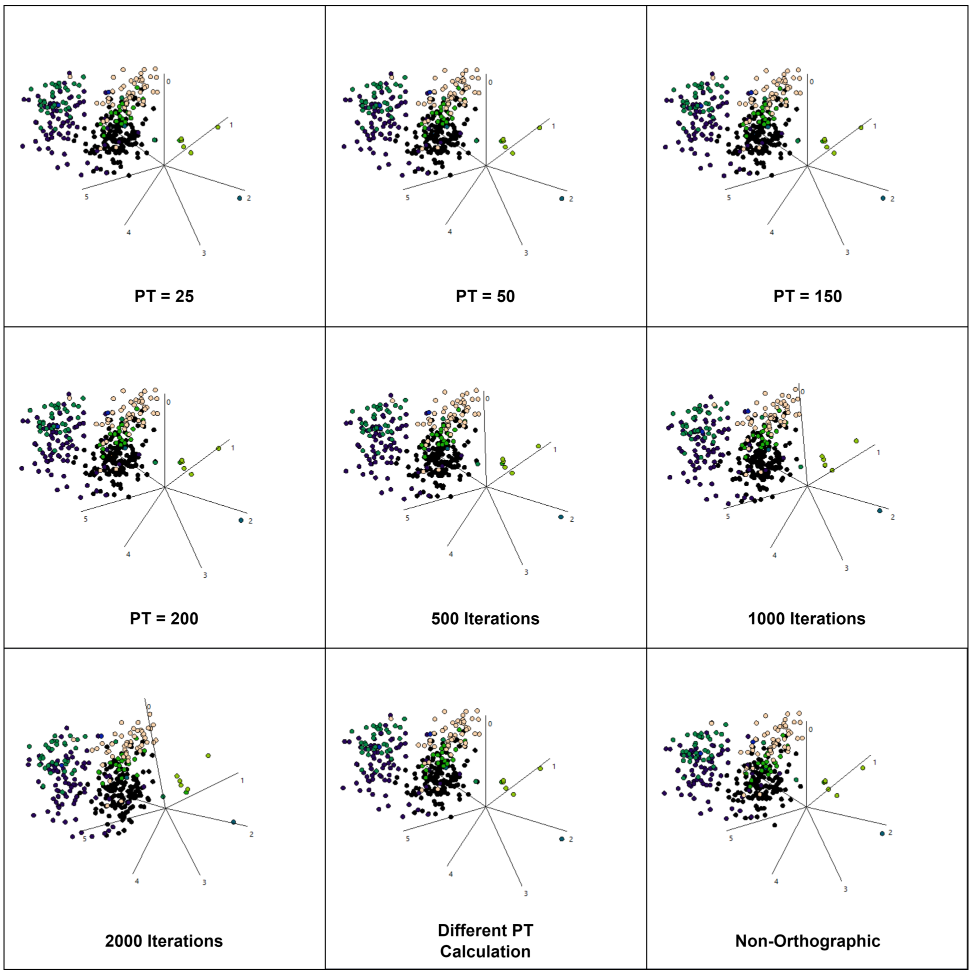


Figure 30.: Visualization for ecoli dataset with different modifications

parameter	ΔDSC	ΔCD	CDC	iterations
Standard	3.88%	100.79%	102.05%	214
PT=25	3.88%	101.01%	102.12%	210
PT=50	3.88%	100.99%	102.12%	216
PT=150	3.88%	100.88%	102.02%	235
PT=200	3.88%	100.88%	102.05%	254
500 Iterations	4.48%	100.32%	101.5%	500
1000 Iterations	5.08%	99.26%	99.97%	1000
2000 Iterations	5.08%	97.69%	96.29%	2000
PT Calculation	4.18%	100.71%	102.18%	163
Non-Orthographic	2.99%	71.55%	140.81%	48

Table 14.: Impact on key values by parameter change for ecoli dataset by MSS

Results for Iris Dataset

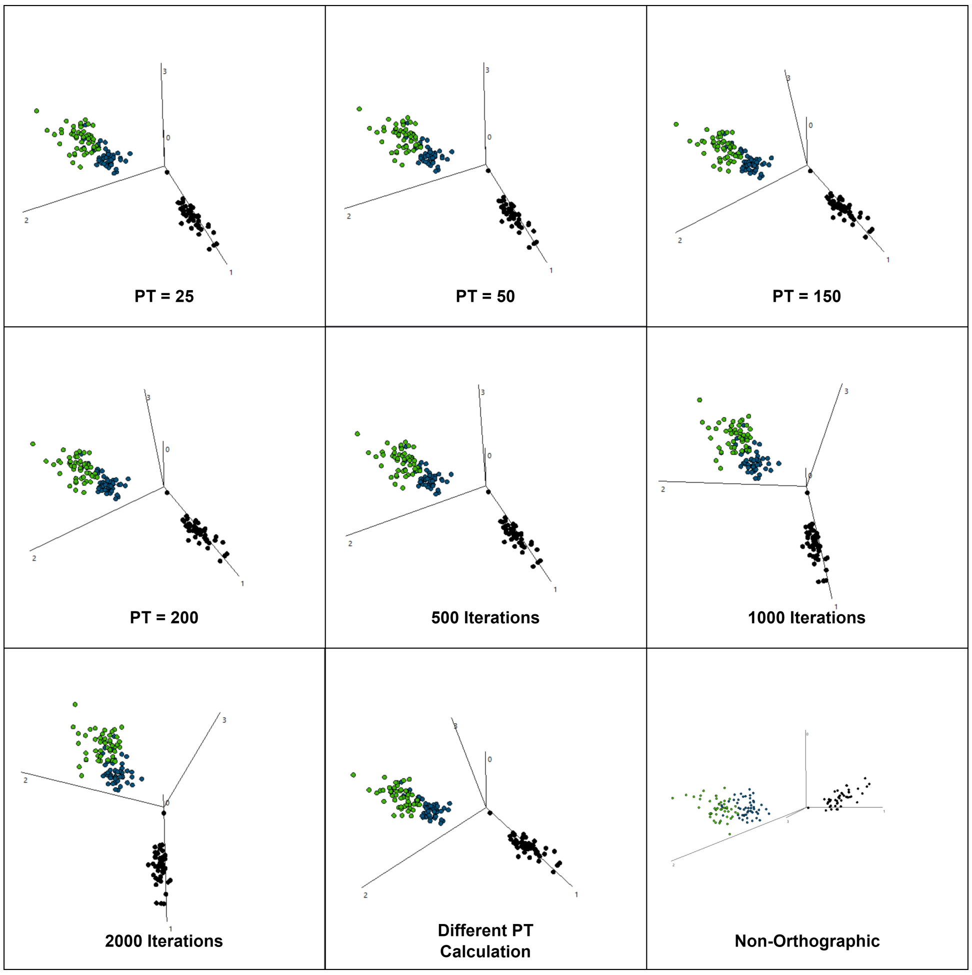


Figure 31.: Visualization for iris dataset with different modifications

parameter	ΔDSC	ΔCD	CDC	iterations
Standard	3.36%	115.24%	130.59%	551
PT=25	3.36%	115.72%	130.29%	528
PT=50	3.36%	115.5%	130.29%	538
PT=150	3.36%	118.67%	130.59%	401
PT=200	3.36%	118.2%	130.59%	421
500 Iterations	3.36%	116.32%	130.59%	500
1000 Iterations	4.03%	110.07%	129.71%	1000
2000 Iterations	4.03%	108.24%	128.82%	2000
PT Calculation	3.36%	119.58%	130.29%	338
Non-Orthographic	2.69%	56.7%	263.24%	677

Table 15.: Impact on key values by parameter change for iris dataset by MSS

Results for Statlog Dataset

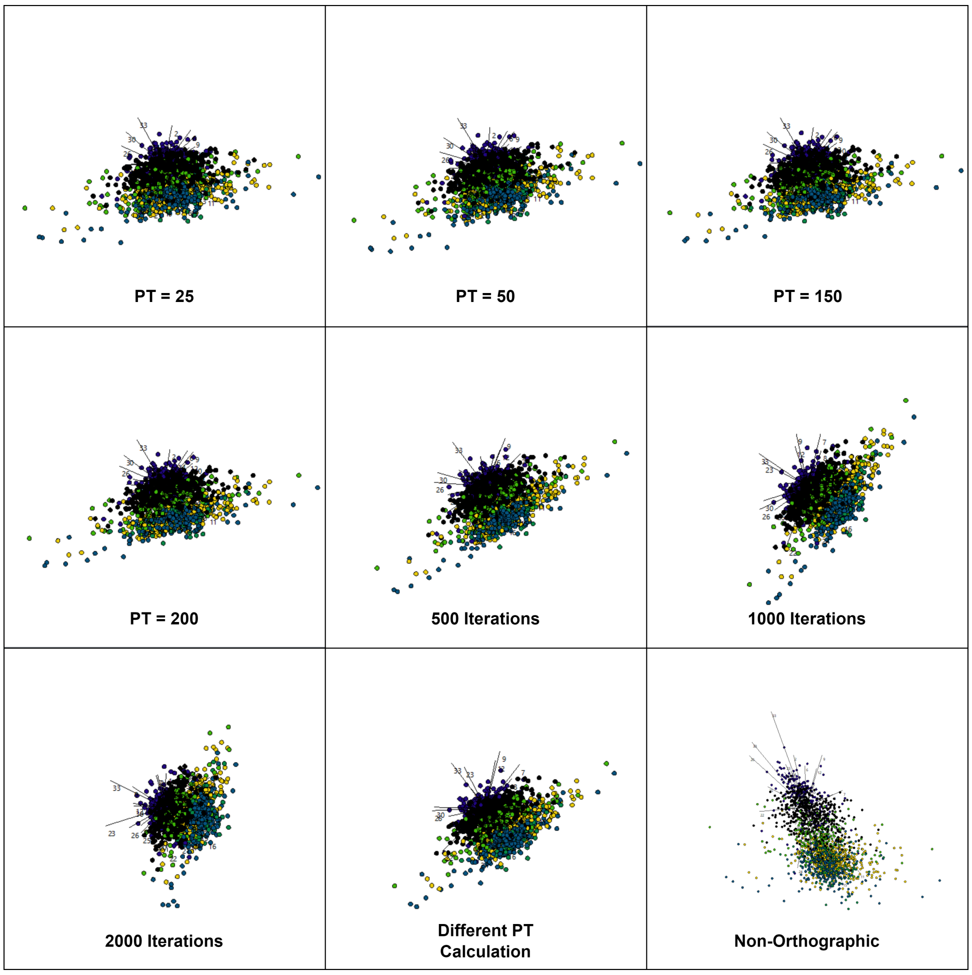


Figure 32.: Visualization for statlog dataset with different modifications

parameter	ΔDSC	ΔCD	CDC	iterations
Standard	20.21%	104.97%	370.54%	396
PT=25	18.31%	104.08%	384.82%	330
PT=50	20.21%	104.82%	392.86%	341
PT=150	20.31%	104.91%	405.36%	392
PT=200	20.76%	105.24%	412.5%	418
500 Iterations	24.61%	109.28%	434.82%	500
1000 Iterations	28.26%	120.25%	442.86%	1000
2000 Iterations	25.81%	140.41%	416.96%	2000
PT Calculation	28.66%	119.18%	451.79%	717
Non-Orthographic	26.06%	53.12%	1508.93%	1000

Table 16.: Impact on key values by parameter change for statlog dataset by MSS

Results for Wdbc Dataset

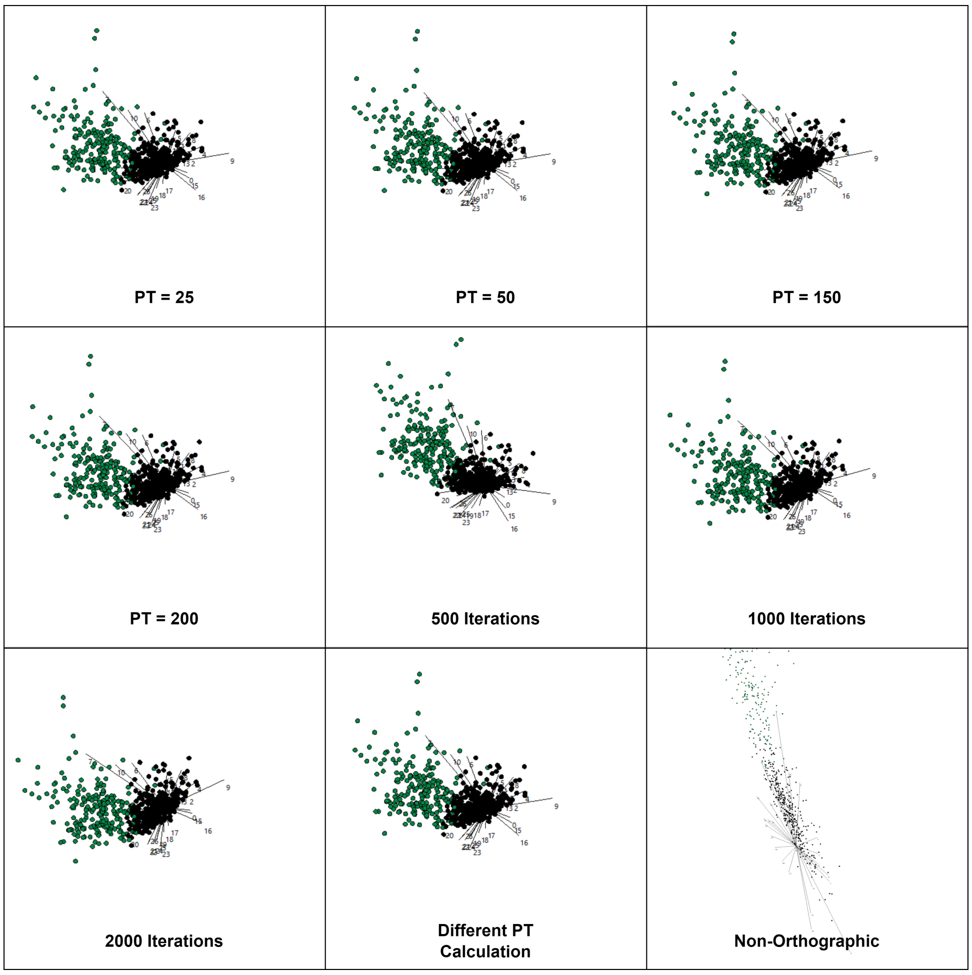


Figure 33.: Visualization for wdbc dataset with different modifications

parameter	ΔDSC	ΔCD	CDC	iterations
Standard	10.03%	121.54%	124.05%	858
PT=25	10.03%	120.91%	124.1%	811
PT=50	10.03%	121.56%	124.05%	813
PT=150	10.03%	121.54%	124.05%	871
PT=200	10.03%	121.53%	124.05%	893
500 Iterations	10.03%	120.9%	127.85%	500
1000 Iterations	10.03%	121.5%	122.78%	1000
2000 Iterations	10.03%	121.46%	122.78%	2000
PT Calculation	10.03%	121.56%	124.05%	799
Non-Orthographic	10.03%	18.9%	1076%	1000

Table 17.: Impact on key values by parameter change for wdbc dataset by MSS

Results for Wine Dataset

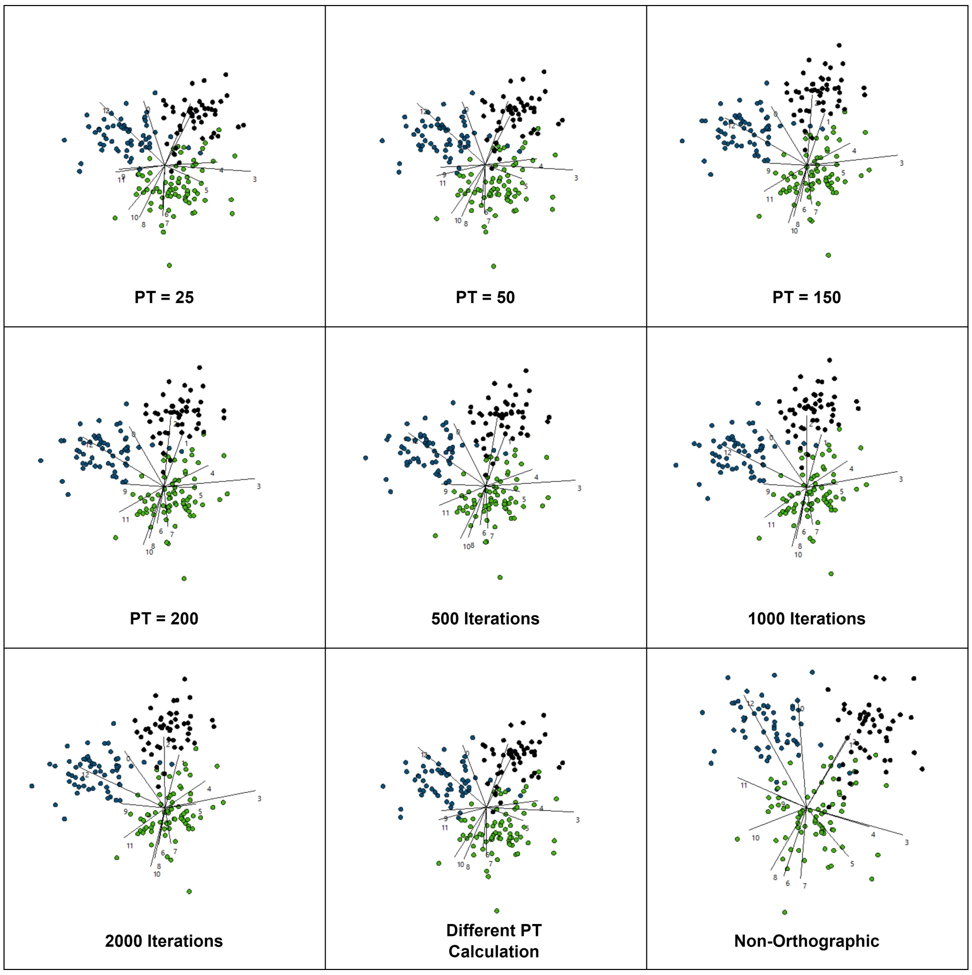


Figure 34.: Visualization for wine dataset with different modifications

parameter	ΔDSC	ΔCD	CDC	iterations
Standard	20.09%	104.25%	191.04%	499
PT=25	18.08%	104.12%	173.13%	269
PT=50	19.21%	104.22%	177.11%	306
PT=150	22.04%	104.04%	197.51%	677
PT=200	22.03%	104.12%	196.02%	613
500 Iterations	22.03%	104.14%	191.54%	500
1000 Iterations	22.03%	103.84%	201.49%	1000
2000 Iterations	22.03%	103.73%	202.49%	2000
PT Calculation	19.21%	104.21%	177.61%	313
Non-Orthographic	17.51%	79.58%	240.3%	264

Table 18.: Impact on key values by parameter change for wine dataset by MSS

Results for Yeast Dataset

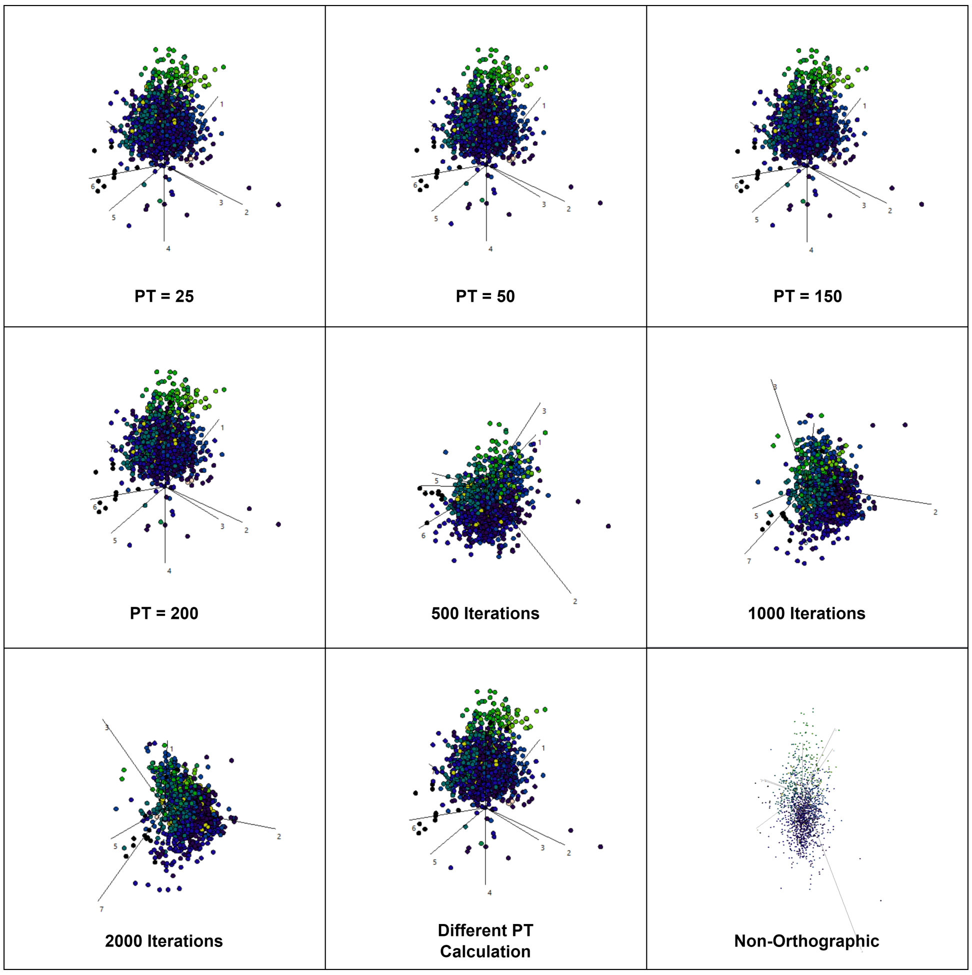


Figure 35.: Visualization for yeast dataset with different modifications

parameter	ΔDSC	ΔCD	CDC	iterations
Standard	0.88%	96.54%	105.82%	68
PT=25	1.02%	96.34%	106.05%	74
PT=50	0.88%	96.51%	105.86%	68
PT=150	1.02%	96.55%	105.93%	69
PT=200	1.02%	96.57%	105.66%	67
500 Iterations	2.56%	101.25%	77.23%	500
1000 Iterations	4.59%	101.66%	67.32%	1000
2000 Iterations	4.59%	101.71%	67.2%	2000
PT Calculation	1.02%	96.56%	105.93%	69
Non-Orthographic	8.5%	26.46%	345.73%	1000

Table 19.: Impact on key values by parameter change for yeast dataset by MSS

Analysis of Results for Different Parameters

Figure 30 shows the projections for different configuration parameters applied to ecoli dataset. There is no projection that looks completely different from all the other projections. A change of the PT works accordingly. However, for a low PT no local maximum has led to worse results and for an extremely large PT and no local maximum has been overcome towards a better solution.

The best separation, according to DSC, is reached in later iterations. These are not reached with the designed PT. For these results at 1000 and 2000 iterations, ΔDSC has a value of 1.2% higher than for standard parameters. This leads to the conclusion, that the local maximum that was found, is worse than an actual global maximum for this dataset. For both results, ΔCD and CDC have slightly decreased. Thus, they seem to be no good indicators for the separation of this dataset.

In Figure 31 the projections for different configuration parameters applied to iris dataset are shown. Visualization and values for iris dataset lead to the same results as for ecoli dataset. Thus, they underline the findings.

Figure 32 shows the projections for different configuration parameters applied to statlog dataset. In contrast to ecoli and iris, another calculation formula for PT has led to the best results. Nonetheless, the main trends of the other two data sets can also be observed here.

For wdbc dataset (shown in Figure 33) all different parametrisations lead to the exact same value of ΔDSC. The different visualizations appear to be rotated in projection space with more or less dense classes. However, the overlap between the black and the green class has to be the same in every case. This could be interpreted as a limit of separation given by the structure of the dataset itself. However, this hypothesis cannot be conclusively proven.

While results for wine dataset underline the overall trend of better solutions towards more iterations, a few other assumptions can be made. For non-orthographic visualization in Figure 34 it can be observed, that even on a small scale absolute distance between the class centroids does not inevitably lead to better separation. This can be observed in a much larger scale for many other non-orthographic visualizations e.g. for statlog dataset. Thus, especially for non-orthographic projections, CDC cannot be taken as a reliable metric for the grade of separation.

As the last dataset examined, yeast dataset is shown in Figure 35. For this dataset, the results are worst in terms of separation. Nevertheless, results underline the trend of an early terminating PT on a local maximum. Termination is very early after approximately 70 iterations for every PT. This leads to the conclusion that the heuristic was terminated by an overall worse projection.

In summary, different values for PT do not influence the results significantly. Looking at the results for forced numbers of iterations without termination criteria, it becomes clear that calculation of PT needs to be reconsidered. Another factor could be the immediate termination if

DSC, CD and CDC decrease on the same iteration. This condition leaves no space for a worse projection in every given aspect. However, while testing this criterion has only rarely terminated the heuristic.

6.3.3. Results for Convergence and Reliability

Convergence with regard to the number of iterations has been tested with a sample of 100 runs for all used datasets with MSS. The histogram for these 100 runs is illustrated in Figure 36. Statistical parameters for the sample are listed in Table 20.

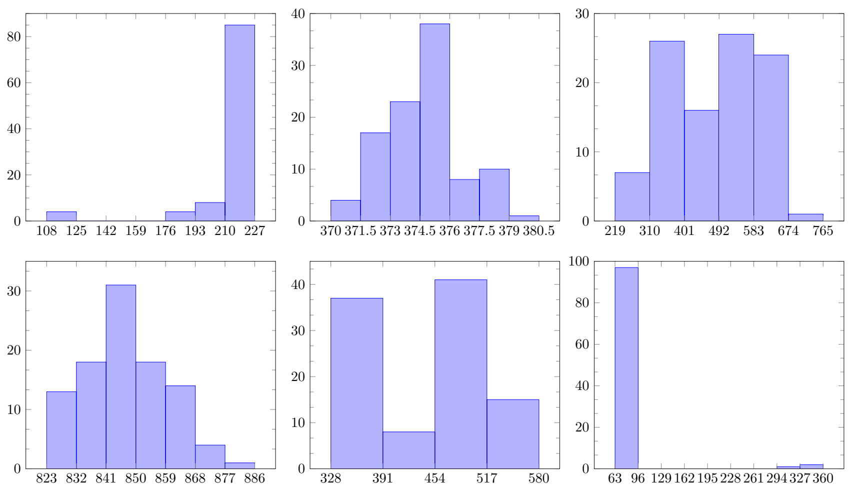


Figure 36.: Histogram for number of iterations in 100 runs with MSS in standard configuration per dataset (TLBR: ecoli, iris, statlog, wdbc, wine, yeast)

dataset	min	max	$\overline{X}$	s	s^2	v
ecoli	108	226	210.8515	22.1551	490.8477	0.1051
iris	370	380	374.8713	1.9883	3.9533	0.0053
statlog	219	689	479.802	120.4807	14515.6004	0.2511
wdbc	823	878	847.3636	12.0411	144.9889	0.0142
wine	328	574	447.0495	83.8255	7026.7075	0.1875
yeast	63	334	78.7822	43.7524	1914.2721	0.5554

Table 20.: Convergence with regard to the number of iterations for MSS

For iris and wdbc, convergence has a very low sample variance coefficient v. This means, that multiple runs converge after approximately the same amount of iterations. However, for all the other datasets, v is rather high. Especially, the extremely large difference between minimum and maximum iterations for yeast and ecoli is an indicator for high variation for the number of iterations. This is a first indicator, but does not necessarily lead to bad results. The design of the heuristic allows to find a solution of same quality at a later stage. Nevertheless, it would be desirable finding the best possible result at the fastest possible time. Therefore, further investigations are necessary regarding the reliability.

Reliability with regard to DSC for MSS has been tested with a sample of 100 runs for all used datasets with MSS. The histogram for these 100 runs is illustrated in Figure 37. Statistical parameters for the sample are listed in Table 21.

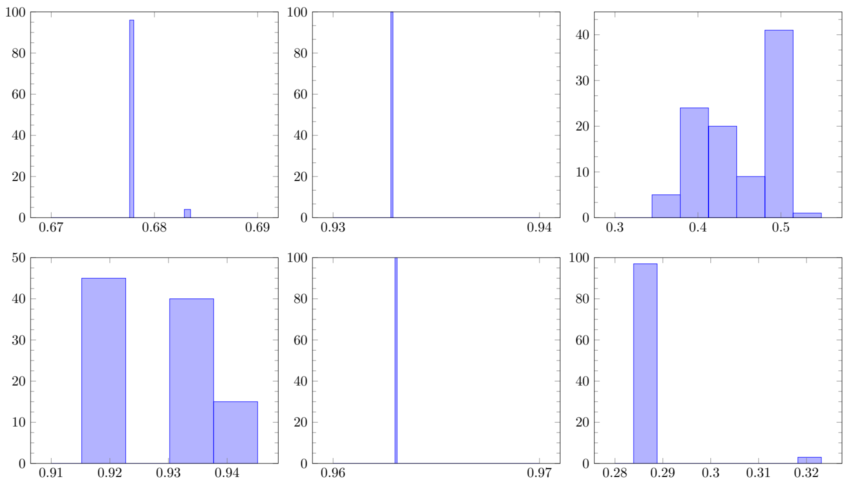


Figure 37.: Histogram for DSC in 100 runs with MSS in standard configuration per dataset (TLBR: ecoli, iris, statlog, wdbc, wine, yeast)

dataset	min	max	$\overline{X}$	s	s^2	v
ecoli	0.6776	0.6835	0.6779	0.0012	0	0.0017
iris	0.9328	0.9328	0.9328	0	0	0
statlog	0.3447	0.5406	0.4462	0.0451	0.0020	0.1011
wdbc	0.963	0.963	0.963	0	0	0
wine	0.9152	0.9435	0.9274	0.0099	0	0.0107
yeast	0.2839	0.3223	0.286	0.0065	0	0.0226

Table 21.: Reliability with regard to DSC for MSS

For iris and wdbc dataset, reliability has zero variance. Thus, the exact same result for DSC has been found throughout the entire sample size. Considering the convergence, termination criteria have been very good for iris and wdbc dataset.

In contrast to the high variance on convergence, ecoli dataset has a very high reliability for the resulting DSC. This means, that the same result is found after strongly fluctuating numbers of iterations. Thus, earlier convergence would result in a better performance.

Results for wine dataset and yeast do have some variance. The range between minimum and maximum is approximately 3-4%. However, the sample variance coefficient v is at about 1% for wine dataset and about 2% for yeast dataset. These values are not critical, but need could need improvement. In the histogram for the yeast data set, it can be seen that almost all example runs end at a value of approximately 0.286. However, since few runs end at a higher value, it can be assumed that the termination criteria stop the heuristic at a local maximum and that the larger maximum can thus only rarely be reached.

Apart from all the rather good results, statlog dataset has a very high sample variance coefficient v. In difference to yeast dataset, this is not because of single values above a local minimum, but because of variance between the results of every single run.

All these findings lead to the conclusion, that either the designed heuristic is not suitable to find a better solution or the dimension of termination criteria does not fit the use-case for atleast some of the used datasets.

6.3.4. Comparison with Other Results

For comparison the method proposed by Molchanov and Linsen [27] is taken into account. A brief reminder: Projections represented as SC are modified by shifting a chosen class by Drag & Drop of their associated centroid (see 2.3). This method has numerous disadvantages compared to CO:

1. The method lacks of existence of *sticky* points. Thus, fixing points at a certain position while moving others is not intended. As a result, the method has many fewer degrees of freedom.

2. Only centroids of classes are considered *shift-able*. Therefore, only whole classes can be shifted by moving their centroid. CO offer an item-based shift configuration by setting status for every single point independently. This leads to more flexibility when manipulating projections.

3. For all used datasets, the equation $L' \cdot M = P$ is ill-defined, as there are more dimensions than classes. In order to compensate for the under-determined equation system, random samples of projection points are added. This leads to a concept shift.

4. The approach presented is an interactive user-based implementation. Thus, finding a good separation depends on the user. In order to test the method, an own implementation was made according to the description in the paper. In several tries and shifts the author of this thesis was not able to reach an optimally separating projection for wine dataset, although some of the axes have already been extremely large.

Therefore, presented results and the theoretical advantages lead to the conclusion that CO are the more promising technique for manipulation of linear projections in SC.

7. Discussion

In this work, the idea of Composition Operators has been extended. Different heuristic approaches using CO for finding linear projections of high-dimensional data that show class separation have been introduced.

At the outset, two question were defined that were to be answered in the course of this work.

1. Are there any datasets which are linear separable?
2. Can it be guaranteed that a separating embedding will always be found if the dataset allows linearly separating embedding?

The first one is answered very straight forward: Yes, there are linear separable datasets. It has been shown that optimal class separating projections do exist.

For the second question the answer is a bit more complicated. Results for multiple runs of the heuristics are very stable and accurate in the empirical test environment presented in this work. However, there is no mathematical proof given, that an existing separating embedding will always be found. Chosen termination criteria lead to reliable convergence in most cases, but do not always result in a best possible separation.

Presented heuristics improve separability in most cases. Minimum Selection Shift achieves the best overall results. Point Selection Shift leads to an optimal separation of wine dataset. Especially when the centroids of the classes lie along different axes on the projection, a good separation is possible.

The dimensioning of the termination criteria has led to very good results for some data sets. For others, however, too strict termination conditions meant that global maxima were not or only rarely reached. Thus, structure and calculation of Penalty Threshold are not optimized yet.

Even if the results presented in this thesis have partial potential for improvement, CO have been convincing as a manipulation technique for linear projections. It has been presented as a user-friendly item-based approach with high flexibility. The concept of manipulating a projection to then achieve the needed input seems to be beneficial for class separation tasks. Compared to other techniques such as [27], CO are superior in some respects.

Overall, the influence of the implemented heuristics on the separability is strongly dependent on the respective data set. For datasets with very high dimensionality n, the orthographic representation reaches its limits, as degrees of freedom vanish.

A final limitation to the approach taken in this thesis is the overall separability of a dataset. If a dataset is not linearly separable by definition — such as iris dataset — there is no possible way to separate classes within their projection. Therefore, the quality of a separation also depends on the dataset itself.

8. Future work

There are many options to potentially improve this implementation in the future. One of these options could be a dynamic combination of the presented variants MSS and PSS in order to improve density as well as distance between classes for a better separability at the same time. Another option could be to set other termination criteria for the heuristics or modify existent. This could be done in a way that they are influenced by characteristics of the underlying datasets. Also, starting with a different initial projection could suit the heuristics method of operation better and thus lead to better results. However, for all these assumptions, implementation and testing have to follow in order to verify them.

A different technique with a completely different principle for automatic manipulation of projections with Composition Operators could also lead to better results, especially for dense class centroids. A more complex system that takes more variables into consideration might lead to more accurate manipulation operations, which then leads to better separation in projection space.

Another field for future applications is Visual Machine Learning based on the class separation embedding. When creating well separated projections in 2D projection space, this result can be used to create feature vectors for VML. Thus, it can help in the development of classifiers as proposed in [31].

Bibliography

[1] Ankerst, Mihael ; Ester, Martin ; Kriegel, Hans-Peter: Towards an effective cooperation of the user and the computer for classification. In: Ramakrishnan, Raghu (Hrsg.) ; Stolfo, Sal (Hrsg.) ; Bayardo, Roberto (Hrsg.) ; Parsa, Ismail (Hrsg.): *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '00*. New York, New York, USA : ACM Press, 2000, S. 179–188. – ISBN 1581132336

[2] Asimov, Daniel: The Grand Tour: A Tool for Viewing Multidimensional Data. In: *SIAM Journal on Scientific and Statistical Computing* 6 (1985), Nr. 1, S. 128–143. – ISSN 0196-5204

[3] Belkin, Mikhail ; Niyogi, Partha: Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. In: *Neural Computation* 15 (2003), Nr. 6, S. 1373–1396. – ISSN 0899-7667

[4] Blum, Christian ; Roli, Andrea: Hybrid Metaheuristics: An Introduction. In: Kacprzyk, Janusz (Hrsg.) ; Blum, Christian (Hrsg.) ; Aguilera, Maria José B. (Hrsg.) ; Roli, Andrea (Hrsg.) ; Sampels, Michael (Hrsg.): *Hybrid Metaheuristics* Bd. 114. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008, S. 1–30. – ISBN 978-3-540-78294-0

[5] Chu, Jun-Uk ; Moon, Inhyuk ; Mun, Mu-Seong: A real-time EMG pattern recognition system based on linear-nonlinear feature projection for a multifunction myoelectric hand. In: *IEEE transactions on bio-medical engineering* 53 (2006), Nr. 11, S. 2232–2239

[6] Cook, Dianne ; Buja, Andreas ; Cabrera, Javier ; Hurley, Catherine: Grand Tour and Projection Pursuit. In: *Journal of Computational and Graphical Statistics* 4 (1995), Nr. 3, S. 155. – ISSN 10618600

[7] Daniels, Karen ; Grinstein, Georges ; Russell, Adam ; Glidden, Mason: Properties of normalized radial visualizations. In: *Information visualization* 11 (2012), Nr. 4, S. 273–300. – ISSN 1473-8716

[8] Demartines, P. ; Herault, J.: Curvilinear component analysis: a self-organizing neural network for nonlinear mapping of data sets. In: *IEEE transactions on neural networks* 8 (1997), Nr. 1, S. 148–154. – ISSN 1045-9227

[9] Dua, Dheeru ; Graff, Casey: *UCI Machine Learning Repository*. 2017. – URL `http://archive.ics.uci.edu/ml`

[10] Gallardo, José E. ; Cotta, Carlos ; Fernández, Antonio J.: Solving the Multidimensional Knapsack Problem Using an Evolutionary Algorithm Hybridized with Branch and Bound. In: Hutchison, David (Hrsg.) ; Kanade, Takeo (Hrsg.) ; Kittler, Josef (Hrsg.) ; Kleinberg, Jon M. (Hrsg.) ; Mattern, Friedemann (Hrsg.) ; Mitchell, John C. (Hrsg.) ; Naor, Moni (Hrsg.) ; Nierstrasz, Oscar (Hrsg.) ; Pandu Rangan, C. (Hrsg.) ;

Steffen, Bernhard (Hrsg.) ; Sudan, Madhu (Hrsg.) ; Terzopoulos, Demetri (Hrsg.) ; Tygar, Dough (Hrsg.) ; Vardi, Moshe Y. (Hrsg.) ; Weikum, Gerhard (Hrsg.) ; Mira, José (Hrsg.) ; Álvarez, José R. (Hrsg.): *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach* Bd. 3562. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, S. 21–30. – ISBN 978-3-540-26319-7

[11] Gu, Quanquan ; Li, Zhenhui ; Han, Jiawei: *Generalized Fisher Score for Feature Selection*

[12] Heidari, Morteza ; Lakshmivarahan, Sivaramakrishnan ; Mirniaharikandehei, Seyedehnafiseh ; Danala, Gopichandh ; Maryada, Sai Kiran R. ; Liu, Hong ; Zheng, Bin: Applying a Random Projection Algorithm to Optimize Machine Learning Model for Breast Lesion Classification. In: *IEEE transactions on bio-medical engineering* 68 (2021), Nr. 9, S. 2764–2775

[13] Hertwig, Ralph ; Pachur, Thorsten: Heuristics, History of. In: *International Encyclopedia of the Social & Behavioral Sciences*. Elsevier, 2015, S. 829–835. – ISBN 9780080970875

[14] Hinton, Geoffrey E. ; Roweis, Sam: Stochastic Neighbor Embedding. In: S. Becker (Hrsg.) ; S. Thrun (Hrsg.) ; K. Obermayer (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 15, MIT Press, 2002. – URL https://proceedings.neurips.cc/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf

[15] Hoffman, P. ; Grinstein, G. ; Marx, K. ; Grosse, I. ; Stanley, E.: DNA visual and analytic data mining. In: *Proceedings. Visualization '97 (Cat. No. 97CB36155)*, IEEE, 19-24 Oct. 1997, S. 437–441. – ISBN 0-8186-8262-0

[16] IBM: Data Visualization. (10/2/2021). – URL https://www.ibm.com/cloud/learn/data-visualization. – Zugriffsdatum: 7/5/2022

[17] Inselberg, Alfred: The plane with parallel coordinates. In: *The Visual Computer* 1 (1985), Nr. 2, S. 69–91. – ISSN 0178-2789

[18] Inselberg, Alfred: *Parallel Coordinates*. New York, NY : Springer New York, 2009. – ISBN 978-0-387-21507-5

[19] Kandogan, Eser: Star Coordinates: A Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions. In: *In Proceedings of the IEEE Information Visualization Symposium, Late Breaking Hot Topics*, 2000, S. 9–12

[20] Keim, D. A. ; Mansmann, F. ; Schneidewind, J. ; Ziegler, H.: Challenges in Visual Data Analysis. In: *Tenth International Conference on Information Visualisation (IV'06)*, IEEE, 05-07 July 2006, S. 9–16. – ISBN 0-7695-2602-0

[21] Keim, Daniel (Hrsg.): *Mastering the information age: Solving problems with visual analytics*. Goslar : Eurographics Association, 2010. – ISBN 978-3-905673-77-7

[22] KIRKPATRICK, S. ; GELATT, C. D. ; VECCHI, M. P.: Optimization by simulated annealing. In: *Science (New York, N.Y.)* 220 (1983), Nr. 4598, S. 671–680. – ISSN 0036-8075

[23] LEHMANN, D. J.: *Notes on Composition Operators*

[24] LEHMANN, D. J. ; THEISEL, H.: Orthographic Star Coordinates. In: *IEEETransactionson-VisualizationandComputerGraphics(ProceedingsIEEEInformationVisualization)* (2013)

[25] MARKET REPORTS WORLD: *Global Data Visualisation Market - Segmented by organizational derpartment, delivery mode, industry vertical (BFSI, IT&telecommunication, retail/e-commerce, education, manufacturing, government) and region - growth, trends and forecast (2018-2023)*. – URL `https://www.marketreportsworld.com/global-data-visualization-market-12343651`

[26] MCINNES, Leland ; HEALY, John ; MELVILLE, James: *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*. – URL `http://arxiv.org/pdf/1802.03426v3`

[27] MOLCHANOV, Vladimir ; FOFONOV, Alexey ; LINSEN, Lars: Continuous Representation of Projected Attribute Spaces of Multifields over Any Spatial Sampling. In: *Computer Graphics Forum* 32 (2013), Nr. 3pt3, S. 301–310. – ISSN 01677055

[28] MOLCHANOV, Vladimir ; LINSEN, Lars: *Interactive Design of Multidimensional Data Projection Layout*. 2014

[29] PANDIAN, Shanthababu: Effective Data Visualization Techniques in Data Science Using Python. In: *Analytics Vidhya* (1.8.2021). – URL `https://www.analyticsvidhya.com/blog/2021/08/effective-data-visualization-techniques-in-data-science-using-python/`. – Zugriffsdatum: 18.07.2022

[30] P.M., Todd: Heuristics for Decision and Choice. In: NEIL, J. S. (Hrsg.) ; PAUL, B. B. (Hrsg.): *International Encyclopedia of the Social & Behavioral Sciences*. Oxford : Pergamon, 2001, S. 6676–6679. – URL `https://www.sciencedirect.com/science/article/pii/B008043076700629X`. – ISBN 978-0-08-043076-8

[31] RAUBER, Paulo E. ; FALCÃO, Alexandre X. ; TELEA, Alexandru C.: Projections as visual aids for classification system design. In: *Information visualization* 17 (2018), Nr. 4, S. 282–305. – ISSN 1473-8716

[32] RUBIO-SÁNCHEZ, Manuel ; RAYA, Laura ; DÍAZ, Francisco ; SANCHEZ, Alberto: A comparative study between RadViz and Star Coordinates. In: *IEEE transactions on visualization and computer graphics* 22 (2016), Nr. 1, S. 619–628

[33] SAMMON, J. W.: A nonlinear mapping for data structure analysis. In: *IEEE Transactions on Computers* 18 (1969), Nr. 5, S. 401–409

[34] Sedlmair, Michael ; Munzner, Tamara ; Tory, Melanie: Empirical guidance on scatterplot and dimension reduction technique choices. In: *IEEE transactions on visualization and computer graphics* 19 (2013), Nr. 12, S. 2634–2643

[35] Sips, Mike ; Neubert, Boris ; Lewis, John P. ; Hanrahan, Pat: Selecting good views of high-dimensional data using class consistency. In: *Computer Graphics Forum* 28 (2009), Nr. 3, S. 831–838. – ISSN 01677055

[36] Sörensen, Kenneth: Metaheuristics-the metaphor exposed. In: *International Transactions in Operational Research* 22 (2015), Nr. 1, S. 3–18. – ISSN 09696016

[37] Tenenbaum, J. B. ; Silva, V. de ; Langford, J. C.: A global geometric framework for nonlinear dimensionality reduction. In: *Science (New York, N.Y.)* 290 (2000), Nr. 5500, S. 2319–2323. – ISSN 0036-8075

[38] Teoh, Soon T. ; Ma, Kwan-Liu: StarClass: Interactive Visual Classification Using Star Coordinates. In: Barbara, Daniel (Hrsg.) ; Kamath, Chandrika (Hrsg.): *Proceedings of the 2003 SIAM International Conference on Data Mining*. Philadelphia, PA : Society for Industrial and Applied Mathematics, 2003, S. 178–185. – ISBN 978-0-89871-545-3

[39] van der Maaten, Laurens ; Hinton, Geoffrey: Viualizing data using t-SNE. In: *Journal of Machine Learning Research* 9 (2008), S. 2579–2605

[40] Zimmermann, Hans-Jürgen: *Operations Research: Methoden und Modelle. Für Wirtschaftsingenieure, Betriebswirte, Informatiker*. Wiesbaden : Vieweg+Teubner Verlag, 2005 (Springer eBook Collection). – ISBN 978-3-322-93906-7

Appendix

Appendix A:

Pseudocode for Heuristics

```
Heuristic 2: Random Selection Shift RSS
1 function heuristic_random_selection_shift(df_p, iterator, num_classes):
      Data: df_p is the dataframe in projection space, iterator is the current iteration
            step, num_classes is the amount of different classes in the dataset
      Result: dp shift vector, selected_class to be shifted, calculated DSC_value,
              calculated CD_value, calculated total_dist
      /* extract star coords and class of data to a new dataframe                    */
2     df_star = df_p[['X','Y','class']]
      /* calculate class centroids and save them in a new dataframe                  */
3     df_centroids = df_star.groupby('class', sort=True).mean().reset_index()
      /* calculate coordinates of the central centroid                               */
4     central_centroid = df_centroids[['X', 'Y']].mean()
      /* call a function to calculate all distances between centroids                */
5     df_distances = calc_centroid_distances(df_centroids)
      /* calculate the distance from each point to its associated centroid           */
6     df_centroid_distances = calc_dist_p_to_assoc_centroid(df_centroids, df_star)
      /* calculate CD, DSC and total_dist for result                                 */
7     CD_value = df_centroid_distances['distance'].sum()
8     DSC_value = calc_dsc(df_centroids, df_star)
9     total_dist = df_distances['distance'].sum()
      /* randomly choose a class to be shifted                                       */
10    selected_class = np.random.randint(num_classes)
      /* select all distances for selected class                                     */
11    selected_class_distances = df_distances.(df_distances[selected_class])
      /* other_class is the nearest class to selected_class                          */
12    min_selected_class_distance = selected_class_distances['distance'].idxmin()
      /* calculate the new shifting vector dp                                        */
13    dp = calc_dp(df_centroids, selected_class, other_class, central_centroid, num_iter)
```

Figure 38:: Pseudocode for RSS

Heuristic 3: Order Selection Shift - OSS

```
1 function heuristic_order_selection_shift(df_p, iterator, num_classes):
      Data: df_p is the dataframe in projection space, iterator is the current iteration
            step, num_classes is the amount of different classes in the dataset
      Result: dp shift vector, selected_class to be shifted, calculated DSC_value,
              calculated CD_value, calculated total_dist
      /* extract star coords and class of data to a new dataframe                */
2     df_star = df_p[['X','Y','class']]
      /* calculate class centroids and save them in a new dataframe              */
3     df_centroids = df_star.groupby('class', sort=True).mean().reset_index()
      /* calculate coordinates of the central centroid                           */
4     central_centroid = df_centroids[['X', 'Y']].mean()
      /* call a function to calculate all distances between centroids            */
5     df_distances = calc_centroid_distances(df_centroids)
      /* calculate the distance from each point to its associated centroid       */
6     df_centroid_distances = calc_dist_p_to_assoc_centroid(df_centroids, df_star)
      /* calculate CD, DSC and total_dist for result                             */
7     CD_value = df_centroid_distances['distance'].sum()
8     DSC_value = calc_dsc(df_centroids, df_star)
9     total_dist = df_distances['distance'].sum()
      /* choose a class to be shifted by order                                   */
10    selected_class = num_iter % num_classes)
      /* select all distances for selected class                                 */
11    selected_class_distances = df_distances.(df_distances[selected_class])
      /* other_class is the nearest class to selected_class                      */
12    min_selected_class_distance = selected_class_distances['distance'].idxmin()
      /* calculate the new shifting vector dp                                    */
13    dp = calc_dp(df_centroids, selected_class, other_class, central_centroid, num_iter)
```

Figure 39:: Pseudocode for OSS

```
Heuristic 4: Point Selection Shift - PSS
1 function heuristic_order_selection_shift(df_p, iterator, num_classes):
      Data: df_p is the dataframe in projection space, iterator is the current iteration
            step, num_classes is the amount of different classes in the dataset
      Result: dp shift vector, selected_class to be shifted, calculated DSC_value,
              calculated CD_value, calculated total_dist
      /* extract star coords and class of data to a new dataframe              */
2     df_star = df_p[['X','Y','class']]
      /* calculate class centroids and save them in a new dataframe            */
3     df_centroids = df_star.groupby('class', sort=True).mean().reset_index()
      /* calculate coordinates of the central centroid                         */
4     central_centroid = df_centroids[['X', 'Y']].mean()
      /* call a function to calculate all distances between centroids          */
5     df_distances = calc_centroid_distances(df_centroids)
      /* calculate the distance from each point to its associated centroid     */
6     df_centroid_distances = calc_dist_p_to_assoc_centroid(df_centroids, df_star)
      /* calculate CD, DSC and total_dist for result                           */
7     CD_value = df_centroid_distances['distance'].sum()
8     DSC_value = calc_dsc(df_centroids, df_star)
9     total_dist = df_distances['distance'].sum()
      /* find the maxmimum distance                                            */
10    max_dist_idx = df_centroid_distances['distance'].idxmax()
      /* select the point that is to be shifted                                */
11    centroid_id = df_centroid_distances.loc[max_dist_idx, 'class']
12    centroid_coords = [df_centroids.loc[centroid_id, 'X'], df_centroids.loc[centroid_id,
        'Y']]
13    point_coord = [df_centroid_distances.loc[max_dist_idx, 'X'],
        df_centroid_distances.loc[max_dist_idx, 'Y']]
      /* create noise                                                          */
14    noise = np.random.normal(0, 1, 1) * 100 / (1000 + num_iter)
      /* calculate the new shifting vector dp                                  */
15    dp = [centroid_coords[0] - point_coord[0] + noise, centroid_coords[1] -
        point_coord[1] + noise]
```

Figure 40:: Pseudocode for PSS

Appendix B:

Key values for all heuristics for each dataset

dataset	DSC_{start}	DSC_{end}	CD_{start}	CD_{end}	$d_{c,total,start}$	$d_{c,total,end}$
ecoli	63.88%	65.07%	11636	11651	3073	3086
iris	89.93%	89.93%	3585	3498	340	358
statlog	21.06%	28.76%	65743	65963	112	222
wdbc	86.27%	94.72%	22820	16437	79	66
wine	72.32%	92.66%	7703	7539	201	353
yeast	27.44%	27.38%	44547	44911	2613	2635

Table 22:: Key values for RSS for each dataset

dataset	DSC_{start}	DSC_{end}	CD_{start}	CD_{end}	$d_{c,total,start}$	$d_{c,total,end}$
ecoli	63.88%	63.88%	11636	11652	3073	3078
iris	89.93%	90.6%	3585	3483	340	364
statlog	21.06%	22.76%	65743	65784	112	132
wdbc	86.27%	94.72%	22820	16399	79	65
wine	72.32%	93.22%	7703	7549	201	357
yeast	27.44%	27.44%	44547	44713	2613	2616

Table 23:: Key values for OSS for each dataset

dataset	DSC_{start}	DSC_{end}	CD_{start}	CD_{end}	$d_{c,total,start}$	$d_{c,total,end}$
ecoli	63.88%	67.76%	11636	11545	3073	3136
iris	89.93%	93.29%	3585	3111	340	444
statlog	21.06%	41.27%	65743	62631	112	415
wdbc	86.27%	96.3%	22820	18776	79	98
wine	72.32%	93.22%	7703	7389	201	384
yeast	27.44%	28.32%	44547	46156	2613	2765

Table 24:: Key values for MSS for each dataset

dataset	DSC_{start}	DSC_{end}	CD_{start}	CD_{end}
ecoli	63.88%	63.39%	11636	11729
iris	89.93%	89.93%	3585	3585
statlog	21.06%	22.51%	65743	57698
wdbc	86.27%	91.37%	22820	17566
wine	72.32%	100%	7703	5641
yeast	27.44%	24.81%	44547	43926

Table 25:: Key values for PSS for each dataset