Technische Universität München
Fakultät für Mathematik

A Combinatorial Optimization Approach to Constrained Clustering

Steffen Alexander Borgwardt

Technische Universität München
Fakultät für Mathematik

A Combinatorial Optimization Approach to Constrained Clustering

Steffen Alexander Borgwardt

Vollständiger Abdruck der von der Fakultät für Mathematik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. Michael Ulbrich
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. Peter Gritzmann
	2.	Prof. Dr. Uriel G. Rothblum, Technion - Israel Institute of Technology, Haifa / Israel (schriftliche Beurteilung)
		Univ.-Prof. Dr. Martin Brokate (mündliche Prüfung)

Die Dissertation wurde am 10.6.2010 bei der Technischen Universität München eingereicht und durch die Fakultät für Mathematik am 23.12.2010 angenommen.

Für meine Familie

ABSTRACT

Motivated by a real-world application in the consolidation of farmland, we approach the field of clustering by methods of combinatorial optimization: We study two polytopes tied to the clustering of a geometric point set into clusters of prescribed sizes. First, we characterize the vertices of the 'gravity polytope' as belonging to clusterings for which there is a 'full cell decomposition' of the underlying space such that each cluster lies in its own cell. Hereby we obtain an alternative characterization of power diagrams, a generalized class of Voronoi diagrams. We show that a vertex of the gravity polytope (as well as a corresponding full cell decomposition) can be computed by solving a linear program over the 'partition polytope'. This leads to intuitive and efficient algorithms for the classification of new data points and the prediction of data values. We then study the edge-structure of the two polytopes and derive a tight upper bound on the combinatorial diameter of the partition polytope. Finally, we use some ideas inspired by our polytopal studies for a combinatorial optimization model for our real-world problem.

ZUSAMMENFASSUNG

Ausgehend von einer Praxisanwendung in der Flurbereinigung betrachten wir Clustering mit Hilfe von Methoden der kombinatorischen Optimierung: Wir untersuchen zwei Polytope, die mit dem Partitionieren einer geometrischen Punktmenge in Cluster vorgeschriebener Größen zusammenhängen. Zunächst charakterisieren wir die Ecken des 'Schwerpunktpolytops' als assoziiert zu Clusterings, für die es eine 'volle Zellzerlegung' des zu Grunde liegenden Raums gibt, so dass jeder Cluster in seiner eigenen Zelle liegt. Hierdurch erhalten wir eine alternative Charakterisierung von Power-Diagrammen, einer verallgemeinerten Klasse von Voronoi-Diagrammen. Wir zeigen, dass eine Ecke des Schwerpunktpolytops (und eine zugehörige volle Zellzerlegung) durch das Lösen eines linearen Optimierungsproblems im 'Partitionspolytop' berechnet werden kann. Dies führt zu intuitiven und effizienten Algorithmen für die Klassifizierung neuer Datenpunkte und die Vorhersage von Datenwerten. Anschließend untersuchen wir die Kantenstruktur der beiden Polytope und leiten eine scharfe obere Schranke für den kombinatorischen Durchmesser des Partitionspolytops her. Abschließend benutzen wir einige aus unseren polytopalen Studien abgeleitete Ideen für ein kombinatorisches Optimierungsmodell für unser Problem aus der Praxis.

Contents

1 Introduction

1.1 Clustering

'We are living in a world full of data.'

unknown source

Every single day, we are surrounded by a lot of information. One of the primitive human instincts is to devise some means to represent this information as data, to store it and to categorize it [And73]. To understand a new piece of information that we obtain, we always try to identify some **features** describing it, and compare them to the features of things we already know. This is done by an intuitive sense of similarity. By this, we derive a categorization of the information we possess into classes of similar features [XW05].
It is not surprising that related questions about an efficient and useful classification of data independently arose in a vast amount of fields of research such as biology, zoology, genetics, medicine, psychiatry, sociology, criminology, anthropology, archaeology, geology, geography, remote sensing, machine learning, information retrieval, pattern recognition, artifical intelligence, market research, economics and more. See [Sco82; ELL01; Mir05] for short surveys of the early history of this field.
The different starting points and criteria of these fields of research have lead to different taxonomies for the objects and the algorithms involved in above process [JD88; HJ97; JMF99; ELL01; Ber02]. The information investigated is denoted as patterns, feature vectors, observations, instances, **data vectors, data points, or items** [JMF99].
A commonly agreed on list of the steps necessary for the classification of data looks as follows [JD88]:

1. Feature representation
2. Definition of a similarity measure
3. Clustering or grouping

The representation of the available data features involves a unification of the information we have and the construction of data structures able to capture this information. Additionally, we have to think about the number of classes, labels or **clusters** that we want in the underlying data [JMF99].

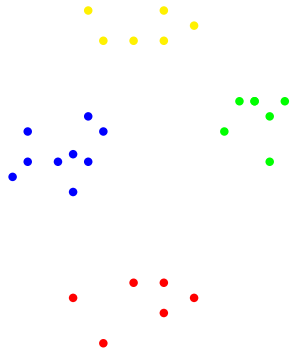


Figure 1: Humans intuitively group two- or three-dimensional data vectors according to their Euclidean distance.

Typically, information is represented as a **data set** of multidimensional data vectors (or points), where each dimension belongs to a single feature [DH73]. The components of such data vectors can be of two fundamentally different types: They can describe **quantitative features**, either being continuous values (e.g. when measuring the size of something) or discrete values (e.g. when counting a number of objects), or they describe **qualitative features** like the gender of people or color of objects.
The second step is the definition of a proximity, **distance or similarity measure**, an answer to the question of what makes two data vectors similar to each other. While this is usually a difficult task for qualitative features, quantitative measures can often be directly represented as (normed) rational or irrational numbers. The corresponding data vectors then are vectors in a high-dimensional **geometric space**. When working in such a space, the **Euclidean distance** has a high appeal as a similarity measure, being the intuitive way to measure distances in a two- or three-dimensional space in everyday life [JMF99]. Figure 1 depicts this effect.
The third and final step in the above list is the grouping of data. There are two general types of tasks that appear in this context:

1. **Clustering** (labeling /unsupervised classification)
2. **Data classification** (supervised classification)

A **clustering** process is the actual process of grouping the underlying data we have into clusters, depending on its representation and the chosen similarity measure. The term clustering is also used for the resulting assignment of data vectors to clusters. We call the task of creating a clustering for a data set a **clustering problem**.

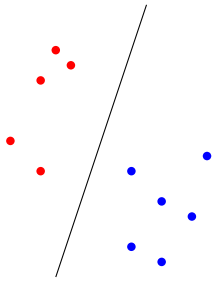


Figure 2: Linear separability of two clusters is an intuitive property of 'good' clusterings.

One way of doing this is to create a **hard clustering**, where the data set is partitioned such that each data vector belongs to exactly one cluster. In hard geometric clustering, it is often desired to obtain some kind of **separation** for the clusters, with the most common one being the **linear separability** of the clusters [BHR92; Mir05]. See Figure 2.
Even if the structure of the underlying data is not prone to yield linearly separable clusters, transformations of the geometric space containing the data vectors may amend the situation. This is an active field of research [SS02; Bor07].
Alternatively, one can create a **fuzzy** clustering. Here, each data vector has a variable degree of membership in each of the clusters [MTA08].
Coming from many different fields of research, there is a wide array of clustering algorithms, most tailored to work well for a special structure of data sets. In [XW05], there is an extensive list of references to corresponding literature. Generally speaking, there are two important types of such algorithms:
Hierarchical algorithms determine a hierarchical structure of clusterings based on the similarity of data vectors. See e.g. [War63; Kin67; SS73] for some classical algorithms. Typically, the results of such an algorithm are described using a binary tree or a dendogram. The root node of such a structure describes a top-level clustering, where the whole data set belongs to a single cluster. The leaf-nodes are clusterings of the data set so that each data vector forms its own cluster. The levels in between contain a hierarchy of nested clusterings, with their position representing the level of similarity of data vectors in the same cluster. See [JMF99; XW05] for a survey of the different concepts of such algorithms.
Partitioning algorithms (or partitional algorithms) usually determine only a single clustering being optimal with respect to some criterion. The input for a partitioning algorithm typically contains the number k of clusters to be created.

A classical example for this type of algorithms is the k-means algorithm [Mac67]. See [Els97; JMF99; Ber02; KP04; Mir05; XW05] for surveys of the concepts of partitioning algorithms.
The process of **data classification** or supervised learning starts with a given clustering of a data set. A new data vector then is associated with one of the existing clusters, depending on the similarity measure and given clustering. Closely related to this are **prediction or data imputation** algorithms which are used to complement a data vector which is 'missing' one (or more) coefficients. It is possible to perform these algorithms solely with the information of the data set, but in many cases it has proven advantageous to use the information provided by a given clustering of the data set as well [EE04; WM04; Mir05].

In this thesis, we are concerned with hard partitioning clustering of a data set. Unless stated otherwise, we consider a set of data vectors $X := \{x_1, \ldots, x_n\}$ which is represented as a set of vectors in the d-dimensional Euclidean space $(\mathbb{R}^d, \|\cdot\|_2)$, with the Euclidean distance being a viable similarity measure. Further, we interpret the linear separability of two clusters to be a desirable differentiation criterion of two clusters.

1.2 Constrained Clustering

Many clustering problems require the creation of clusters that satisfy some constraints. This is commonly referred to as **constrained clustering**, an active field of research which is not as well-explored as the general type of clustering. See [BDW08] for a survey of different types of constraints and related methods and [DB08] for a list of further references.
The two most important types of constraints applied to clustering are **instance-level constraints** and **balancing constraints**.
Instance-level constraints are used to integrate a-priori knowledge about whether two data vectors should belong to the same cluster or not into the clustering process: **Must-link constraints** specify that two data vectors have to be in the same cluster. **Cannot-link constraints** specify that two data vectors may not belong to the same cluster [WC00; WCRS01].
This knowledge may come from some insight into the structure of the data set, e.g. from partially clustered data or similar data sets that have been investigated before. Instance-level constraints then are used to 'guide' a clustering algorithm to create a clustering of favorable properties. More complex constraints like

cluster-level constraints, which ask for properties of whole clusters, can often be reduced to instance-level constraints [BDW08].
The second important type of clustering constraints are **balancing constraints**. These typically take the form of **size restrictions** on the clusters, which bound the number of data vectors in the clusters in various ways [GBH98; ZG03; HK08]. For example, one could ask for all clusters to contain the same amount of data vectors, or for clusters to satisfy certain lower and upper bounds on the number of data vectors they contain.

Throughout Chapter 2, 3 and 4, we will consider clustering with size restrictions. In our case, the constraints define the exact number of data vectors belonging to each cluster.
In Chapter 5, we consider a clustering problem motivated by a real-world application. As we will see, it contains a mathematically difficult type of size restrictions. All data vectors have associated weights, and the size restrictions are defined as lower and upper bounds on the sums of weights of the data vectors in the clusters. Additionally, we will use both types of instance-level constraints to obtain a clustering of the desired properties.

1.3 Polytopal Studies

We approach the constrained clustering problems in this thesis with methods of combinatorial optimization and discrete geometry. In this approach, two polytopes take a central position, the **'gravity polytope'** and the **'partition polytope'**. The gravity polytope is defined as the convex hull of **'gravity vectors'** consisting of the centers of gravity of each cluster. By this construction, it is closely tied to the **(bounded-shape) partition polytope** investigated in [BHR92; HOR98]. Instead of the centers of gravity, the vectors which define a bounded-shape partition polytope consist of the sums of vectors in each cluster, and clusters are restricted by lower and upper bounds on their sizes.
In [BHR92] it was shown that the vertices of such a polytope belong to clusterings for which each pair of clusters allows strict linear separation. On the other hand, there are clusterings such that each pair of clusters is strictly linearly separable, but which do not belong to a vertex of the bounded-shape partition polytope.
With this knowledge, we turn to a characterization of the vertices of the gravity polytope in Chapter 2. We start by noting that the same separability result holds for the gravity polytope. In fact, the clusterings belonging to vertices of the

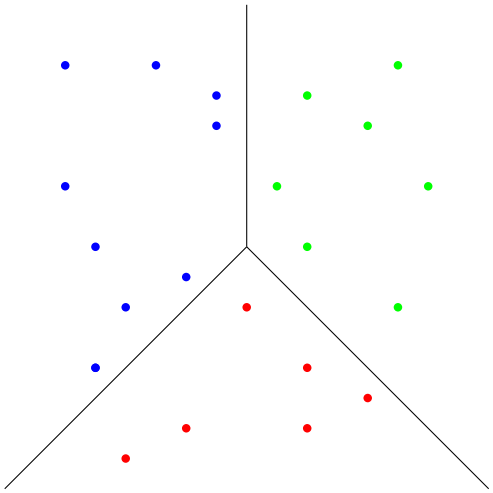


Figure 3: A (full) cell decomposition of $\mathbb{R}^2$ into 3 cells, with each cell containing the data vectors of exactly one cluster.

gravity polytope allow **'cell decompositions'** such that each cluster lies in its own cell [BG09]. A cell decomposition is defined by a special set of hyperplanes partitioning the underlying geometric space into polyhedral cells. But even this stronger property does not imply that the gravity vector of such a clustering necessarily is a vertex of the gravity polytope.

We then show that the cell decompositions constructed as described in [BG09] are of a special type, which we call **'full cell decompositions'**. Full cell decompositions are cell decompositions for which the hyperplanes associated with any subset of cells still define a cell decomposition. This property leads to the desired characterization of the vertices. Figure 3 shows a (full) cell decomposition for three clusters in $\mathbb{R}^2$.

Further, we see that the calculation of a vertex of the gravity polytope is equivalent to the calculation of a certain **least-squares assignment**. This allows us to identify the cell decompositions we construct for a vertex clustering of the gravity polytope as an alternative characterization of **power diagrams**, a class of generalized Voronoi diagrams [Aur87; AHA98].

In Chapter 3, the vertex characterization results of Chapter 2 are used for a combinatorial optimization approach to data classification and prediction. Having a cell decomposition such that each cluster lies in its own cell, we devise basic algorithms for data classification and prediction and investigate their running

time in the arithmetic model of computation [AHU85; Cha03]. We then turn to some further advantages of having a full cell decomposition for these algorithms. The calculation of a vertex of the gravity polytope (with the associated clustering then allowing such a full cell decomposition) can be done in a special **transportation polytope**, the partition polytope. With its matrix being totally unimodular [KW68], it suffices to solve a linear program. Similarly, the calculation of a full cell decomposition itself can be done by solving a linear program as well.
We then consider several ideas as to what makes a cell decomposition desirable from a data classification point of view. We close the chapter with some thoughts about the role of the underlying geometric information and the size restrictions.
In Chapter 4, we continue our polytopal studies by turning to the edge-structure of both the partition and gravity polytope.
In the partition polytope, there is a one-to-one correspondence between the vertices and the clusterings of a set of data vectors. The edges of the polytope are in one-to-one correspondence to **'cyclical exchanges'** of data vectors between the clusters.
We derive a new tight bound on the diameter of the partition polytope (in a worst case) as the sum of the sizes of the two largest clusters. This result is a direct generalization of the diameter bound for the Birkhoff polytope [BR74] and is related to a diameter bound for permutation polytopes [GP06]. It is achieved by a constructive graph-theoretical approach.
We start by showing that it is possible to cover the vertices of maximal degree in a graph that decomposes into cycles by a set of vertex-disjoint cycles. The 'difference' of two clusterings can be represented by a graph with this property. The cover of its vertices of maximal degree then allows us to construct (at most) two cyclical exchanges such that the maximal number of 'ill-assigned' data vectors in a cluster is reduced by at least one. We obtain a polynomial-time algorithm for the construction of a small set of cyclical exchanges to transform a clustering into another.
On the other hand, the edge-structure of the gravity polytope proves more difficult to analyze. Only some clusterings belong to vertices of the gravity polytope. We show that under some (mild) assumptions, its edges correspond to cyclical exchanges. These assumptions are shown to be necessary. Further, we explain how to add a geometric representation to a set of data vectors such that a single specific cyclical exchange corresponds to an edge of the associated gravity polytope.

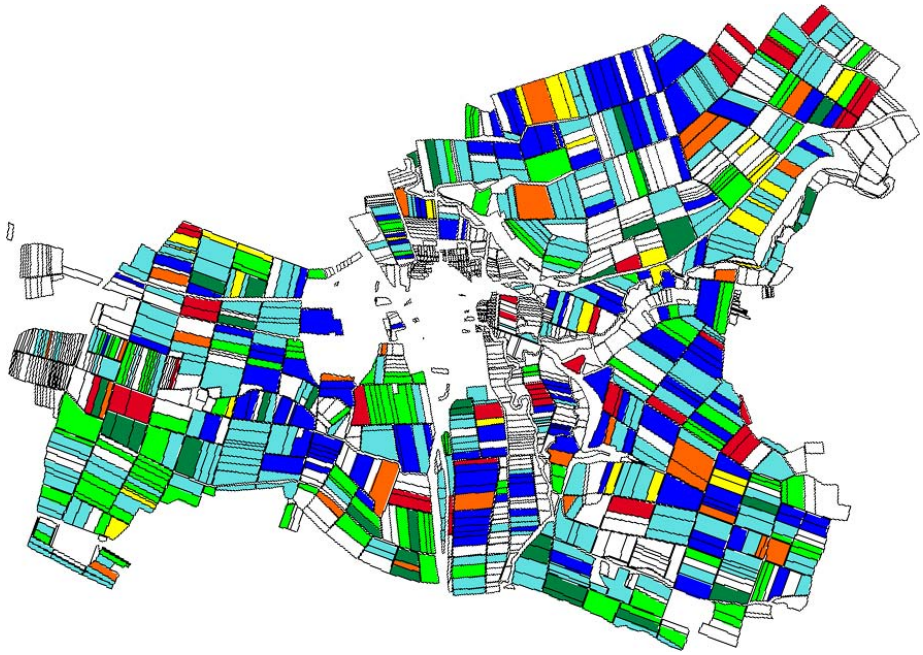


Figure 4: An agricultural region. Different colors represent different cultivating farmers. The lots of the farmers are small and scattered.

1.4 Real-World Application

The methods described in Chapter 3 currently are applied in practice, in several fields such as the creation of 'risk classes' from the data of customers of an insurance company, the prediction of the probability of suffering from severe dementia depending on patient data over a time line or the derivation of a categorization scheme for sediment samples in aquatic ecology.

In Chapter 5, we turn to a real-world application in the consolidation of farmland which originally incited our interest in constrained clustering. Most of the results and algorithms in this chapter have been published [BBG09].

In many rural communities, a small number of farmers cultivate a big number of small lots which are scattered over an extended agricultural region. Due to this, they have high driving costs, and heavy machinery cannot be used profitably. Figure 4 shows such a region. This example will be used in Chapter 5 to depict our methods.

In such a situation, often a classical land consolidation process is initiated. It performs a complete restructuring of infrastructure and lot structure in the agricultural region, making it a long-term and expensive process.

Conversely to this, we devise a model for the consolidation of farmland as a clustering problem, where the existing lot structure is kept as is. Thus no extensive reassignment of legal property is necessary, so that our results are easy to realize in practice.
Some of the lots are 'fixed' by the farmers. The cultivation rights for the remaining lots then are reassigned combinatorially. Of course, to each farmer a set of lots has to be assigned that totals to about his original size and value. As the lots differ with respect to these two measures, we obtain a constrained clustering problem with non-uniform size restrictions, which makes it inherently difficult. The goal is the minimization of the total driving and cultivation cost of the farmers of the whole region. To understand how this can be achieved, we perform an economic analysis of these two cost factors and discuss what makes a distribution of farmland a 'good' one.
The results are intuitively clear: The lots have to be swapped such that large groups of connected lots are assigned to single farmers and such that the lots of each single farmer are not too far from each other. We enforce the creation of large connected lot groups belonging to single farmers by using instance-level constraints.
The clustering problem can be modeled as an integer linear program. In the case where all lots have identical size and bonity, the corresponding constraint matrix is totally unimodular which allows an efficient solution of the problem as linear program.
In the case of different lot sizes and bonities, an application of the Simplex algorithm yields a provably low number of lots that are not uniquely assigned to a single farmer. By this, it serves as an efficient approximation algorithm when combined with a post-heuristic for these lots. Figure 5 shows one of the lot redistributions calculated this way.
We also describe an alternative approach using an adaption of the k-means algorithm. Finally, we close the chapter with some empirical results for the practical performance of both our algorithms.

1.5 Acknowledgement

The results of Chapter 2, 3 and 5 were obtained in joint work with Andreas Brieden and Peter Gritzmann. Most of the results of Chapter 5 have been published in [BBG09]. Papers summarizing the results of the other chapters are in preparation.

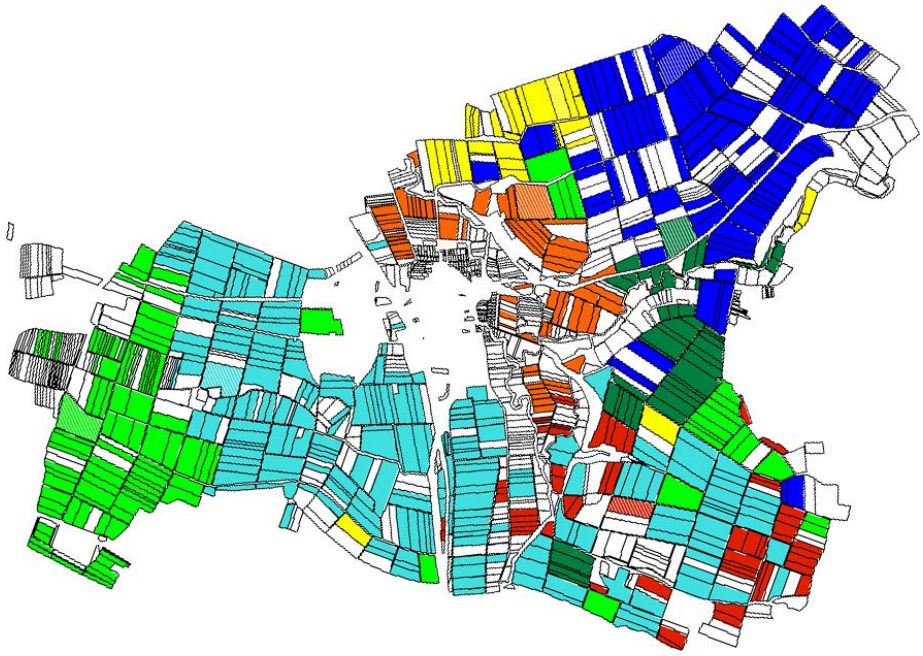


Figure 5: One of the lot redistributions calculated in Chapter 5.

.

First and foremost, I would like to thank my advisor Prof. Peter Gritzmann for his supervision, for sharing his visionary ideas and the insightful discussions about the topics of this work. I would also like to thank Prof. Andreas Brieden for the enthusiastic introduction into the real-world application of clustering in the consolidation of farmland and the helpful and motivating discussions.
Special thanks also go to Lucia Roth, Michael Ritter and Rene Brandenberg, to who I could always turn to discuss about mathematical problems and everything else. I am very grateful for the friendly working atmosphere created by all the current, former and associated members of the research group "Angewandte Geometrie & Diskrete Mathematik" at the Technische Universität München that I know, namely Oliver Bastert, Julia Böttcher, David Bremner, Markus Brill, Christoph Buchheim, Nico Düvelmeyer, Peter Heinig, Raymond Hemmecke, Markus Jörg, Stefan König, Michael Öllinger, Bernhard Reinbold, Felix Schmiedl, Anastasia Shakhshneyder, Matthias Silbernagl, Tanja Stadler, Anusch Taraz, Barbara Wilhelm and Andreas Würfl.
Finally, my deep and heartfelt thanks go to my family for their love and support.

2 Vertex Characterization of the Gravity Polytope

In this chapter, we consider the 'gravity polytope' defined by the centers of gravity of clusters of all possible clusterings of a given set of data points. We investigate the connection between the 'gravity vectors' of clusterings, the separability of clusters and partitions of the underlying space into cells. Our results will lead to some algorithmic ideas for data classification in the next chapter.
We refer the reader to an overview of our notational conventions and a list of the most important symbols used in the **Notation and Symbols** appendix.

2.1 The Gravity Polytope

We start by providing the necessary terminology.
In the following, let $d \in \mathbb{N}$ be the dimension of a geometric space, let $n \in \mathbb{N}$ be the number of data points we have, and let $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ with $x_i \neq x_j$ for all $i \neq j;\ i, j \in \{1, \ldots, n\}$ be our set of data points. Let further $k \in \mathbb{N}$ be the number of clusters and $\kappa_1, \ldots, \kappa_k \in \mathbb{N}$ with $\sum_{i=1}^{k} \kappa_i = n$ be the number of data points in the clusters. We start by defining our basic clustering terms formally.

Definition 2.1 (k-Clustering)
*A k-**clustering** $C := (C_1, \ldots, C_k)$ of X is a partition of X into k non-empty sets $C_1, \ldots, C_k$. C_i is the i-**th cluster** of C for $i \in \{1, \ldots, k\}$.*

Often, the size of the clusters created is important.

Definition 2.2 (Cluster Size and Clustering Shape)
Let $C := (C_1, \ldots, C_k)$ be a k-clustering of X. For $i \in \{1, \ldots, k\}$, the number of elements in C_i is the **size** *$|C_i|$ of C_i. The tuple $|C| := (|C_1|, \ldots, |C_k|)$ is the* **shape** *of C.*

The shape captures all information about the sizes of the clusters of a clustering. Next, we consider clusterings for which the shape is fixed by the choice of $\kappa_1, \ldots, \kappa_k$.

Definition 2.3 (k, $(\kappa_1, \ldots, \kappa_k)$-Clustering)
*A $k, (\kappa_1, \ldots, \kappa_k)$-**clustering** C of X is a k-clustering of X with $|C| = (\kappa_1, \ldots, \kappa_k)$.*

For a given set of n points and a fixed number k of clusters, there is a polynomial number (in n) of k-clusterings or $k, (\kappa_1, \ldots, \kappa_k)$-clusterings satisfying the definitions above. We denote these sets of clusterings as follows.

Definition 2.4 (Set of (Feasible) Clusterings)
$\mathcal{C}(X,k) := \{C : C \text{ is a } k\text{-clustering of } X\}$ *is the* **set of k-clusterings** *of X and*

$$\mathcal{C}(X,k,\kappa_1,\ldots,\kappa_k) := \{C : C \text{ is a } k,(\kappa_1,\ldots,\kappa_k)\text{-clustering of } X\}$$

is the **set of $k,(\kappa_1,\ldots,\kappa_k)$-clusterings** *of X.*

With k and $(\kappa_1,\ldots,\kappa_k)$ clear from the context and fixed, we use the informal term **prescribed-shape clustering (PSC)** for a clustering respecting these numbers. We also use the notation $PSC(X) = PSC(X,k;\kappa_1,\ldots,\kappa_k) := \mathcal{C}(X,k,\kappa_1,\ldots,\kappa_k)$. If the fact that we consider a PSC is clear as well, we also simply use the term **clustering**.
Now we turn to the definition of the polytope we want to investigate. To do so, we need a formal definition of the center of gravity of a cluster as the arithmetic mean of the points in the cluster.
Definition 2.5 (Center of Gravity)
The **center of gravity** *of a cluster C_i is $c_i := \frac{1}{\kappa_i}\sum_{x\in C_i} x$ for $i \in \{1,\ldots,k\}$.*

Next, we move to $\mathbb{R}^{d\cdot k}$ by combining all the centers of gravity of a clustering to a single vector.
Definition 2.6 (Gravity Vector)
The **gravity vector** *of a PSC C is $v(C) := (c_1^T,\ldots,c_k^T)^T \in \mathbb{R}^{d\cdot k}$.*

Each PSC of X defines a gravity vector. We denote the set of all these gravity vectors as follows.
Definition 2.7 (Set of Gravity Vectors)
$V = V(X,k;\kappa_1,\ldots,\kappa_k) := \{v(C) : C \in PSC(X,k;\kappa_1,\ldots,\kappa_k)\}$ *is the* **set of gravity vectors**.

Now we are ready to define the polytope that we will analyze in this chapter. We define it as the convex hull of the set of gravity vectors of PSCs for a point set X.
Definition 2.8 (Gravity Polytope (Q))
The **gravity polytope** *is $Q = Q(X,k;\kappa_1,\ldots,\kappa_k\} := conv\ V$.*

When the parameters of $Q(X,k;\kappa_1,\ldots,\kappa_k\}$ are clear from the context, we will simply use the term Q to refer to $Q(X,k;\kappa_1,\ldots,\kappa_k\}$. Additionally, we will use Q as an abbreviation for 'a gravity polytope, 'the gravity polytope', or 'the corresponding gravity polytope'.
We introduce the association of a clustering with a given gravity vector as follows.

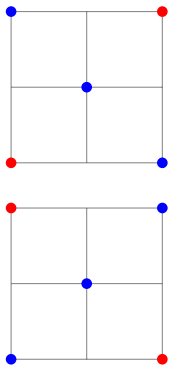


Figure 6: Two clusterings with the same associated gravity vector.

Definition 2.9 (Associated Clustering)
Let $v \in Q$. A PSC C with $v(C) = v$ is a **clustering associated with** v.

Two PSCs C, C' may have the same associated gravity vector $v(C) = v(C')$. Let

$$X := \left\{ \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\} \subset \mathbb{R}^2.$$

Clustering these points into two clusters, we may obtain $C = (C_1, C_2)$ with

$$C_1 = \left\{ \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}, C_2 = \left\{ \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\}$$

or $C' = (C'_1, C'_2)$ with

$$C'_1 = \left\{ \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ -1 \end{pmatrix} \right\}, C'_2 = \left\{ \begin{pmatrix} -1 \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right\}.$$

Both C and C' are associated with the gravity vector $v = (c_1^T, c_2^T)^T = (0, 0, 0, 0)^T$. Figure 6 shows the two clusterings.
In the following, we will need a definition that allows us to formally describe the process of exchanging points in between clusters.

Definition 2.10 (Application of a Cyclical Exchange)
Let $C := (C_1, \ldots, C_k)$ be a PSC, let $I = \{i_1, \ldots, i_t\} \subset \{1, \ldots, k\}$ be an index set, and let $x_i \in C_i$ for $i \in I$. Applying a **cyclical exchange** $CE := (x_{i_1}, \ldots, x_{i_t})$ *to C means deriving a PSC $C' = (C'_1, \ldots, C'_k)$ from C by setting $C'_{i_l} = (C_{i_l} \cup \{x_{i_{l-1}}\}) \backslash \{x_{i_l}\}$ for all $l \in \{1, \ldots, t\}$. The* **cyclical exchange** CE' **inverse to** CE *is defined as $CE' := (x_{i_t}, \ldots, x_{i_1})$, and can analogously be applied to C' to derive C.*

Recall (from our notational conventions in the 'Notation and Symbols' appendix) that we use index sets in a 'modulo way', implying that we also set $C'_{i_1} = (C_{i_1} \cup \{x_{i_t}\})\backslash\{x_{i_1}\}$ for the choice of $l = 1$, as $x_{i_0} = x_{i_t}$.
Informally, we say that a cyclical exchange $CE := (x_{i_1}, \ldots, x_{i_t})$ **moves** the points $x_{i_1}, \ldots, x_{i_t}$. Note that, by this definition, a single cyclical exchange moves at most one point of each cluster.
Let C and C' be two PSCs. Clearly, it then is possible to derive C' from C by applying a set of cyclical exchanges where each $x \in X$ is moved at most once. In this case, the order in which these cyclical exchanges are applied is arbitrary.

Lemma 2.11
Let $C := (C_1, \ldots, C_k)$ and $C' := (C'_1, \ldots, C'_k)$ be two PSCs of $X := \{x_1, \ldots, x_n\}$. Then there is a set of cyclical exchanges $\mathcal{CE} := \{CE_1, \ldots, CE_s\}$ where all $x \in X$ are moved by at most a single $CE_i \in \mathcal{CE}$, such that C' can be derived from C by applying the cyclical exchanges in $\mathcal{CE}$ to C.

Proof. We show that $\mathcal{CE}$ can be constructed greedily. If $C = C'$, $\mathcal{CE} = \emptyset$, and we are done. Otherwise, there is an $x_{i_1} \in X$ such that $x_{i_1} \in C_{i_1}$, but $x_{i_1} \in C'_{i_2}$ for some $i_2 \in \{1, \ldots, k\}\backslash\{i_1\}$. With the cluster sizes fixed, there is an $x_{i_2} \in X$ with $x_{i_2} \in C_{i_2}$, but $x_{i_2} \in C'_{i_3}$ for $i_3 \in \{1, \ldots, k\}\backslash\{i_2\}$. We continue like this until we move an item x_{i_t} to an i_r-th cluster from which we already 'took' an item. This yields a cyclical exchange $CE_1 := (x_{i_r}, \ldots, x_{i_t})$.
Applying CE_1 to C, we obtain a PSC C'' for which the number of points x with $x \in C''_i$, but $x \notin C'_i$ for some $i \in \{1, \ldots, k\}$ is strictly lower than for C. We then repeat our greedy construction of a cyclical exchange until the PSC derived is C'. By this construction, any item is only moved by at most one cyclical exchange. This proves the claim. □

Note that the set of cyclical exchanges to derive a PSC from another is not uniquely defined, not even the number of cyclical exchanges is fixed. On the other hand, the fact that such a set exists can be used to show that gravity vectors that are vertices of the gravity polytope have the special property of necessarily being associated with only a single PSC.

Lemma 2.12
Let v^* be a vertex of Q. Then there is exactly one PSC C with $v(C) = v^*$. We call this PSC the **clustering of** *v^*.*

Proof. We prove the claim by contradiction. By definition of Q there is a PSC with $v^* = v(C) = (c_1^T, \ldots, c_k^T)^T$. Let v^* be a vertex of Q and $C := (C_1, \ldots, C_k)$,

$C' := (C'_1, \ldots, C'_k)$ be two clusterings with $v(C) = v(C') = v^*$. Then there is a vector $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ (with $a_i \in \mathbb{R}^d$ for all $i \in \{1, ..., k\}$) with $a^T v^* > a^T v$ for any $v \in Q \backslash \{v^*\}$.
As the number of points in the clusters is fixed, we can find a set of cyclical exchanges $\mathcal{CE}$ to derive C' from C such that no $x \in X$ is moved twice (Lemma 2.11). Without loss of generality, we assume $x_1 \in C_1, \ldots, x_t \in C_t$ and $x_1 \in C'_2$, $\ldots, x_{t-1} \in C'_t, x_t \in C'_1$ to represent the points of a cyclical exchange CE in $\mathcal{CE}$. As v^* is maximal in Q with respect to $a^T v$ and

$$a^T v^* = \sum_{i=1}^{k} a_i^T c_i = \sum_{i=1}^{k} \frac{a_i}{\kappa_i}^T \sum_{x_l \in C_i} x_l = \sum_{i=1}^{k} \frac{a_i}{\kappa_i}^T \sum_{x_l \in C'_i} x_l,$$

we see that

$$\frac{a_1}{\kappa_1}^T x_1 + \cdots + \frac{a_t}{\kappa_t}^T x_t = \frac{a_2}{\kappa_2}^T x_1 + \cdots + \frac{a_1}{\kappa_1}^T x_t,$$

as otherwise an application of CE to C (or the inverse one to C') yields another PSC C'' with $a^T v(C'') > a^T v^*$. Equivalently, we can write above equality in the form

$$\frac{a_1}{\kappa_1}^T (x_t - x_1) + \cdots + \frac{a_t}{\kappa_t}^T (x_{t-1} - x_t) = 0.$$

By applying CE to C, the centers of gravity are moved, as $x_i \neq x_j$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$, and we get a new gravity vector $v'' \neq v^* \in Q$ with $a^T v'' < a^T v^*$ by our choice of a. This implies that

$$\frac{a_1}{\kappa_1}^T (x_t - x_1) + \cdots + \frac{a_t}{\kappa_t}^T (x_{t-1} - x_t) < 0,$$

contradicting our assumption that there are two clusterings with the same vertex as gravity vector. □

We will also use the shorter term **vertex clustering** for a clustering associated with a vertex in Q if the context is clear.
The gravity polytope is tied to polytopes that have been studied in the literature. Rothblum et al. studied polytopes in $\mathbb{R}^{d \cdot k}$ belonging to k-clusterings where, instead of considering the centers of gravity of the clusters, they added up the coordinates of the points contained in each cluster and considered a vector containing these sums [BHR92; HOR98].
They investigated the vertices of these polytopes for both the case that the number of points is arbitrary in each cluster, implying that they are not necessarily

non-empty, and the case that there are lower and upper bounds on the sizes of the clusters. In the latter case, they talked about **bounded-shape partition polytopes**. In [BHR92] it was shown that the vertices of these (latter) polytopes belong to clusterings for which each pair of clusters allows strict linear separation.
In later work, Rothblum et al. also considered **shaped partition polytopes** where the set of admissible shapes for the clusters is defined explicitly, see e.g. [HOR99; HOR00].
Our gravity polytope can be interpreted as a scaled variant of a special case of both these types of polytopes: It corresponds to the case where the lower bounds on the cluster sizes are equal to the upper bounds, or to a shaped partition polytope with a single feasible shape.
By this, the gravity polytope and these polytopes share some common properties. We start by transferring the separability result for clusters in [BHR92] to the gravity polytope, and then extend this known result by creating special 'cell decompositions' of the underlying space such that each cell of these decompositions contains exactly one of the clusters. As we will see, these cell decompositions are closely related to a special class of generalized Voronoi diagrams [Aur87; AHA98].

2.2 Separability

First, we recall some standard terminology for the separability of two sets.

Definition 2.13 (Linear Separability)
Let $A, B \subset \mathbb{R}^d$. A and B allow **(weak linear) separation** *(or are* **(weakly linearly) separable***) if there is a hyperplane $H_{a,\beta} \subset \mathbb{R}^d$ with $a \in \mathbb{R}^d \backslash \{0\}$ and $\beta \in \mathbb{R}$ such that $A \subset H^{\geq}_{a,\beta}$ and $B \subset H^{\leq}_{a,\beta}$.*
A and B allow **strict (linear) separation** *(or are* **strictly (linearly) separable***) if there is a hyperplane $H_{a,\beta} \subset \mathbb{R}^d$ with $a \in \mathbb{R}^d \backslash \{0\}$ and $\beta \in \mathbb{R}$ such that $A \subset H^{>}_{a,\beta}$ and $B \subset H^{<}_{a,\beta}$.*

We call a clustering separable if all pairs of its clusters are separable.

Definition 2.14 (Separability of Clusterings)
Let $C := (C_1, ..., C_k)$ be a k-clustering of X. C allows (weak, strict) **linear separation** *(or is (weakly, strictly)* **linearly separable***) if C_i and C_j are (weakly, strictly) separable for any $i \neq j;\ i, j \in \{1, ..., k\}$.*

Note that when considering the separability of clusters C_i and C_j of a finite point set X, strict separation already implies **strong separation**, implying that the hyperplane in between the two clusters can be moved slightly, and its direction can be perturbed slightly while still keeping the strong separation property.
With these terms, we are capable of explaining the connection between vertices of the gravity polytope and separability of the associated clustering. This result first was proven by Barnes, Hoffman and Rothblum (for a related polytope) [BHR92].

Theorem 2.15 (Barnes, Hoffman, Rothblum 92)
Let v^* be a vertex of Q. Then the PSC C^* associated with v^* by $v^* = v(C^*)$ allows strict linear separation.

Proof. Let v^* be a vertex of Q and let $C^* := (C_1^*, \ldots, C_k^*)$ be the unique clustering associated with $v^* = v(C^*) = ((c_1^*)^T, \ldots, (c_k^*)^T)^T$ (see Lemma 2.12).
Then there is a vector $a = (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ (with $a_i \in \mathbb{R}^d$ for all $i \in \{1, \ldots, k\}$) with

$$a^T v^* = \sum_{i=1}^{k} a_i^T c_i^* > \sum_{i=1}^{k} a_i^T c_i = a^T v$$

for any $v := v(C) = (c_1^T, \ldots, c_k^T)^T \in Q \backslash \{v^*\}$.
It suffices to prove the strict separability of C_1^* and C_2^*. Let $x_1 \in C_1^*$, and $x_2 \in C_2^*$. We consider the clustering $C := (C_1, ..., C_k)$ with $C_1 := (C_1^* \cup \{x_2\}) \backslash \{x_1\}$, $C_2 := (C_2^* \cup \{x_1\}) \backslash \{x_2\}$ and $C_i := C_i^*$ for all $i \in \{3, ..., k\}$. Then $c_1 = c_1^* + \frac{1}{\kappa_1}(x_2 - x_1)$, $c_2 = c_2^* + \frac{1}{\kappa_2}(x_1 - x_2)$ and $c_i = c_i^*$ for all $i \in \{3, ..., k\}$. We know that

$$\sum_{i=1}^{k} a_i^T c_i^* > \sum_{i=1}^{k} a_i^T c_i$$

and thus

$$a_1^T c_1^* + a_2^T c_2^* > a_1^T c_1 + a_2^T c_2.$$

We conclude

$$0 > \frac{a_1}{\kappa_1}^T (x_2 - x_1) + \frac{a_2}{\kappa_2}^T (x_1 - x_2),$$

and hence

$$0 > \left(\frac{a_1}{\kappa_1} - \frac{a_2}{\kappa_2} \right)^T (x_2 - x_1).$$

By choosing $a_{21} := \frac{a_1}{\kappa_1} - \frac{a_2}{\kappa_2} \in \mathbb{R}^d$, we get $a_{21}^T x_1 > a_{21}^T x_2$. As x_1 and x_2 were chosen arbitrarily from C_1^* respectively C_2^*, we get

$$\min_{x \in C_1^*} a_{21}^T x > \max_{x \in C_2^*} a_{21}^T x.$$

Thus C_1^* and C_2^* are strictly separable. Using the same construction for each pair of clusters C_i^*, C_j^*, we get vectors $a_{ji} \neq 0 \in \mathbb{R}^d$ with

$$\min_{x \in C_i^*} a_{ji}^T x > \max_{x \in C_j^*} a_{ji}^T x$$

for all $i \neq j;\ i, j \in \{1, ..., k\}$, implying the pairwise strict separability of all clusters. □

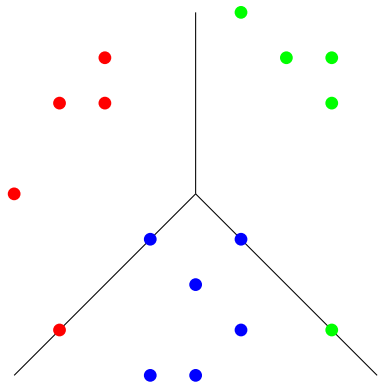
Note that the vector $a = (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with

$$a^T v^* = \sum_{i=1}^{k} a_i^T c_i^* > \sum_{i=1}^{k} a_i^T c_i = a^T v$$

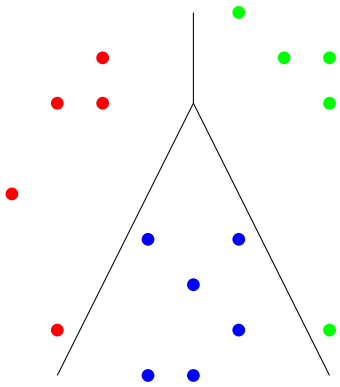
for any $v := v(C) = (c_1^T, \ldots, c_k^T)^T \in Q \backslash \{v^*\}$ has to be from int$(N(v^*))$, where $N(v^*)$ denotes the cone of outer normals of v^* with respect to Q. Note further that $a \in$ int$(N(v^*))$ already implies $a_{ji} \neq 0 \in \mathbb{R}^d$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$ due to the unique association of a clustering with v^*.
Using an $a \in$ bd$(N(v^*))$ with $\frac{a_i}{\kappa_i} \neq \frac{a_j}{\kappa_j}$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$, i.e. an $a \in \mathbb{R}^{d \cdot k}$ with only $a^T v^* \geq a^T v$ for any $v \in Q \backslash \{v^*\}$, yields a system of hyperplane directions allowing (at least) weak separation of the clusters of the clustering associated with the vertex v^*, easily seen by just replacing all strict inequalities in above proof by weak inequalities. The clustering may also be strictly separable with respect to such an a, but we cannot guarantee this property.

Corollary 2.16
Let v^* be a vertex of Q, let $C := (C_1, \ldots, C_k)$ be the clustering of v^*, and let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$. If $a \in$ int$(N(v^*))$, C is strictly separable with respect to separation directions $a_{ji} := \frac{a_i}{\kappa_i} - \frac{a_j}{\kappa_j}$ for clusters C_i and C_j. If $a \in$ bd$(N(v^*))$ and $\frac{a_i}{\kappa_i} \neq \frac{a_j}{\kappa_j}$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$, C is (at least) weakly separable with respect to separation directions $a_{ji} := \frac{a_i}{\kappa_i} - \frac{a_j}{\kappa_j}$ for clusters C_i and C_j.

Figure 7 shows a clustering that is a vertex of its respective gravity polytope, and hyperplane directions such that in a), we only achieve weak separation, but strict separation in b).



(a) Hyperplane directions allowing only weak separation.



(b) Hyperplane directions allowing strict separation.

Figure 7: A 3-clustering in $\mathbb{R}^2$ belonging to a vertex of the corresponding gravity polytope.

If $X \subset \mathbb{R}^d$ does not have a full-dimensional convex hull, any clustering of X clearly is weakly separable by using a 'separating' hyperplane H with $X \subset H$ for any pair of clusters. In the following, we assume X to have a full-dimensional convex hull, unless noted otherwise.
With X having a full-dimensional convex hull, we cannot even guarantee weak separability of a clustering C if $v(C) \in \text{relbd}(Q)$ (but not a vertex). Note that we talk about the relative boundary (and later interior) of Q, as Q is not full-dimensional:
The coordinates of $k-1$ cluster centers of gravity suffice to uniquely determine the k-th cluster's center of gravity.

Lemma 2.17
Q is not full-dimensional.

Proof. Let $C := (C_1, \dots, C_k)$ be any PSC with $v = v(C) \in Q \subset \mathbb{R}^{d \cdot k}$.
For $a = (\kappa_1, \dots, \kappa_1, \kappa_2, \dots, \kappa_2, \dots, \kappa_k, \dots, \kappa_k)^T \in \mathbb{R}^{d \cdot k}$, we have

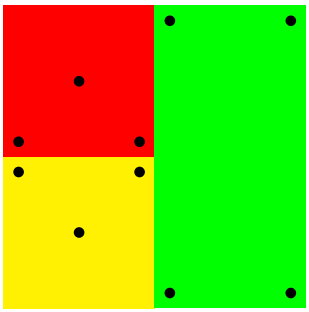
$$a^T v = \sum_{i=1}^{k} a_i^T c_i = \sum_{i=1}^{k} \frac{1}{\kappa_i} (\kappa_i, \dots, \kappa_i)^T \Big(\sum_{x_l \in C_i} x_l \Big) = \sum_{l=1}^{n} (1, \dots, 1)^T x_l,$$

independently of C. □

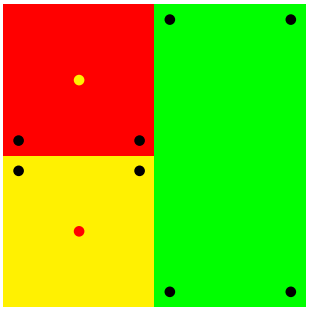
The problem case $\frac{a_i}{\kappa_i} = \frac{a_j}{\kappa_j}$ plays the central part in the following statement.

Lemma 2.18
Let C be a PSC with $v = v(C) \in$ relbd(Q). Then C is not necessarily weakly separable.

Proof. Let $v \in \text{relbd}(Q)$ and C be an associated clustering. Then there is a vector $a \in \mathbb{R}^{d \cdot k}$ such that $a^T v \geq a^T v'$ for any $v' \in Q$. We now construct a clustering with gravity vector v that is not separable, but satisfies this property.
Consider two clusters C_1 and C_2 belonging to clustering C. If $\frac{a_1}{\kappa_1} = \frac{a_2}{\kappa_2}$, we know that the value of $a^T v(C)$ does not change if we swap points internally between C_1 and C_2.
Figure 8 a) shows a 3-clustering C in $\mathbb{R}^2$. The different colors associate the points with a cluster (C_1 being the green, C_2 being the yellow and C_3 being the red cluster), yielding a clustering $C = (C_1, C_2, C_3)$ with gravity vector v. v is optimal with respect to $a^T v$ for $a = (1, 0, 0, 0, 0, 0)^T$.
We now exchange a pair of points between the red and yellow cluster to obtain the clustering C' of Figure 8 b).



(a) A strictly separable clustering.



(b) A clustering that is not separable, but lies on the relative boundary of Q.

Figure 8: Two 3-clusterings of $X \subset \mathbb{R}^2$ with associated gravity vectors on the same facet of Q.

The gravity vector $v' = v(C')$ still is optimal with respect to $a = (1,0,0,0,0,0)^T$, yet C_2 (yellow) and C_3 (red) are not separable. By this, v' cannot be a vertex of Q, but $v' \in \text{relbd}(Q)$, without being separable. This proves the claim. □

Let C be a clustering with associated gravity vector v. For any $a \in \mathbb{R}^{d\cdot k}$ with $\frac{a_i}{\kappa_i} = \frac{a_j}{\kappa_j}$ for some $i \neq j;\ i,j \in \{1,\dots,k\}$, the clusters C_i and C_j are treated as a single cluster $C_i \cup C_j$ with respect to $a^T v$. Due to this, it is a natural assumption to use $\frac{a_i}{\kappa_i} \neq \frac{a_j}{\kappa_j}$ for any $i \neq j;\ i,j \in \{1,\dots,k\}$. We will do so, unless stated otherwise.
If there is no pair of indices $i \neq j;\ i,j \in \{1,\dots,k\}$ with $\frac{a_i}{\kappa_i} = \frac{a_j}{\kappa_j}$, we immediately see that we can apply the construction from Theorem 2.15 to obtain a system of weakly separating hyperplanes for a clustering C with gravity vector $v(C) \in \text{relbd}(Q)$, just like in Corollary 2.16.

Corollary 2.19
Let $v^* \in \text{relbd}(Q)$, let $C := (C_1,\dots,C_k)$ be a clustering associated with v^*, and let $a := (a_1^T,\dots,a_k^T)^T \in \mathbb{R}^{d\cdot k}$ with $a^T v^* \geq a^T v$ for any $v \in Q\backslash\{v^*\}$, and $\frac{a_i}{\kappa_i} \neq \frac{a_j}{\kappa_j}$ for all $i \neq j;\ i,j \in \{1,\dots,k\}$. Then C is weakly separable with respect to separation directions $a_{ji} := \frac{a_i}{\kappa_i} - \frac{a_j}{\kappa_j}$ for clusters C_i and C_j.

On the other hand, having a strictly separable clustering does not imply that its gravity vector v is on the relative boundary of Q. In the following lemma, we construct a strictly separable clustering in the relative interior of its gravity polytope Q.

Lemma 2.20
Let C be a PSC with $v = v(C) \in \text{relint}(Q)$. Then C may be strictly separable.

Proof. We construct a set of points $X \subset \mathbb{R}^2$ and a strictly separable PSC C of these points. The gravity vector $v = v(C)$ will lie strictly in the convex hull of the gravity vectors of three other PSCs of X, of which one is known to be in $\text{relint}(Q)$. This will prove the claim.
Consider the two-dimensional point set X in Figure 9. The lines represent a grid of distance 1 in both coordinates, the central intersection of the grid lines is the origin. The shown area consists of nine 4×4 squares. There are no points in $[-2,2]^2$. The other 4×4 squares contain the same pattern of points. Let c be the center of such a square. Then the points in this square are

$$c + \begin{pmatrix} 1.9 \\ 0 \end{pmatrix}, c - \begin{pmatrix} 1.9 \\ 0 \end{pmatrix}, c + \begin{pmatrix} 0 \\ 1.9 \end{pmatrix}, c + \begin{pmatrix} 0 \\ -1.9 \end{pmatrix},$$

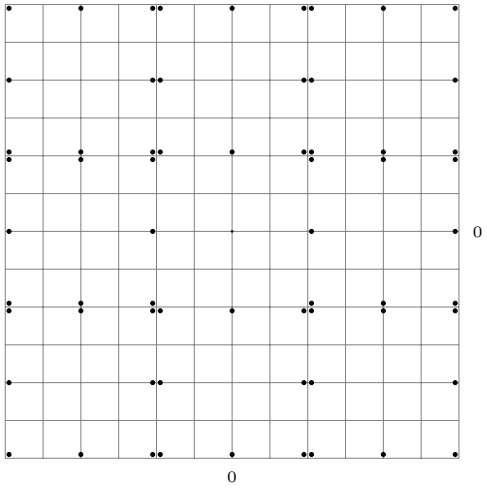


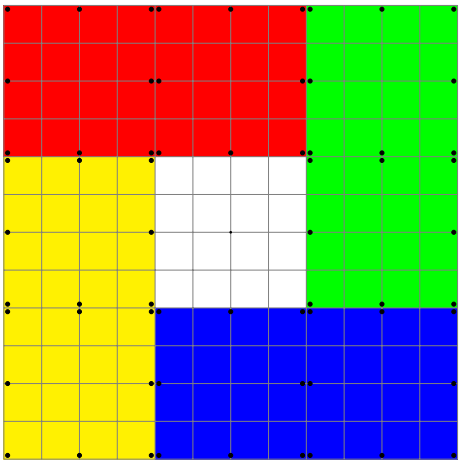
Figure 9: A point-symmetric point set $X \in \mathbb{R}^2$.

$$c + \begin{pmatrix} 1.9 \\ 1.9 \end{pmatrix}, c + \begin{pmatrix} -1.9 \\ 1.9 \end{pmatrix}, c + \begin{pmatrix} 1.9 \\ -1.9 \end{pmatrix} \text{ and } c + \begin{pmatrix} -1.9 \\ -1.9 \end{pmatrix}.$$

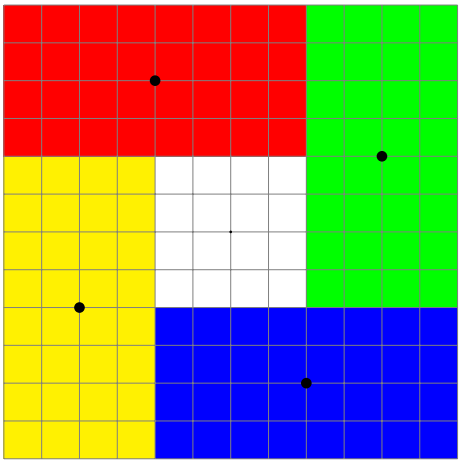
Note that X is point-symmetric with respect to the origin. The clustering C we consider contains four clusters which are symmetrically placed in X, see Figure 10 a). Figure 10 b) identifies their centers of gravity. C allows strict linear separation. We now prove that its gravity vector lies in relint(Q).
First of all, due to the point-symmetry of X, there are (many) clusterings with gravity vector 0. For a simple construction, one just always assigns the points $x \in X$ and $-x \in X$ to the same cluster. This gravity vector certainly is in relint(Q), as for any $a \neq 0 \in \mathbb{R}^{2\cdot 4}$ (that distinguishes the different clusters), there is a clustering C' with $a^T v(C') > 0$, and due to the point-symmetry of X also a clustering C'' with $a^T v(C'') = -a^T v(C') < 0$.
We now choose $b_1, b_2 \in \mathbb{R}^2$ such that $b_1 \perp b_2$ to construct 4-clusterings $C := (C_1, C_2, C_3, C_4)$ of X as follows:

$$\begin{aligned} C_1 &:= \{x \in X : x \in H^{\geq}_{b_1,0} \wedge x \in H^{\leq}_{b_2,0}\} \\ C_2 &:= \{x \in X : x \in H^{\geq}_{b_1,0} \wedge x \in H^{\geq}_{b_2,0}\} \\ C_3 &:= \{x \in X : x \in H^{\leq}_{b_1,0} \wedge x \in H^{\geq}_{b_2,0}\} \\ C_4 &:= \{x \in X : x \in H^{\leq}_{b_1,0} \wedge x \in H^{\leq}_{b_2,0}\} \end{aligned}$$

Figure 11 depicts two such 4-clusterings (for pairs of orthogonal vectors b_1 and b_2) and shows the centers of gravity of the clusters.

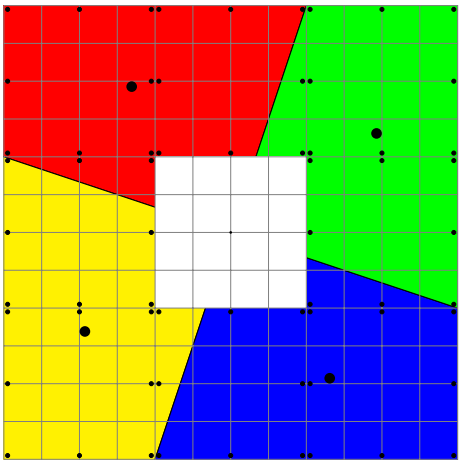


(a) The colors represent the 4 different clusters.

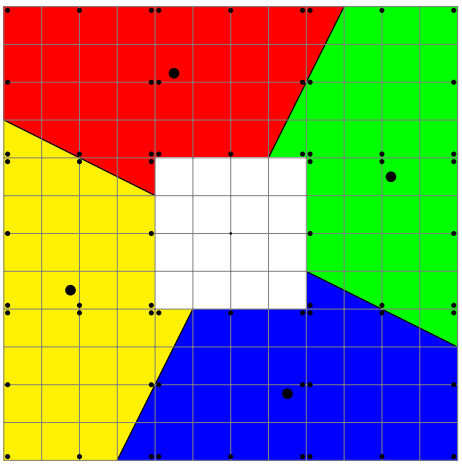


(b) The centers of gravity of the clustering.

Figure 10: The point set of Figure 9 partitioned into 4 strictly separable clusters.



(a) A 4-clustering for $b_1 = \binom{3}{-1}$ and $b_2 = \binom{1}{3}$.



(b) A 4-clustering for $b_1 = \binom{2}{-1}$ and $b_2 = \binom{1}{2}$.

Figure 11: Two different 4-clusterings of X. The big dots are the centers of gravity of the clusters.

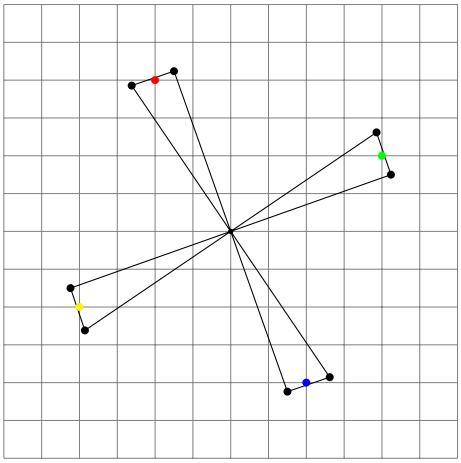


Figure 12: The centers of gravity of the clusters of Figure 10 lie strictly in the convex hull of the origin and the centers of gravity of the clusters of Figure 11 $a)$ and $b)$.

Finally, we look at the convex hull of $\binom{0}{0}$ and the centers of gravity of the clusters of the clusterings depicted in Figure 11. An easy calculation shows that the centers of gravity lie strictly in this convex hull. Figure 12 depicts the situation.□

Looking at this example, we see that the strict separability of a clustering does not imply that the associated gravity vector is a vertex of the polytope. Hence this property falls short of yielding a characterization of the vertices.
This is similar to the situation in a polytope without restrictions on the cluster sizes considered by Barnes, Hoffmann and Rothblum in [BHR92]. While the clusterings belonging to vertices of this polytope allow strict separation of their clusters by hyperplanes containing the origin, this property can also be achieved by clusterings that do not belong to such a vertex.
In the following, we discuss some extensions to the strict separability of a clustering to derive a characterization for the vertices of the gravity polytope. While we explained the above example numerically and pictographically, in the next sections we will develop analytical arguments that can be used to directly see that the clustering of Figure 10 is in the interior of the gravity polytope.

2.3 Cell Decompositions

Partitions of metric spaces and complexes in a metric space are a classical field in topology, see e.g. [Bre93]. Here, we choose a geometric approach and inter-

pretation for our results. We start by giving a fully geometric definition of cell decompositions. To do so, we need the following term.

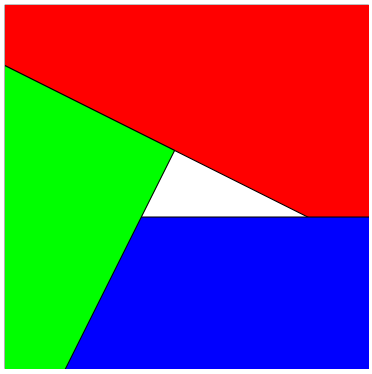
Definition 2.21 (k-Cell Arrangement)
Let $\mathcal{H}$ be a set of hyperplanes $\mathcal{H} := \{H_{a_{ij},\gamma_{ij}} : i \neq j;\ i,j \in \{1,\dots,k\}\}$ with $H_{a_{ij},\gamma_{ij}} := \{x \in \mathbb{R}^d : a_{ij}^T x = \gamma_{ij}\}$, where $a_{ij} \in \mathbb{R}^d, \gamma_{ij} \in \mathbb{R}$, $a_{ji} = -a_{ij}$ and $\gamma_{ji} = -\gamma_{ij}$ for all $i \neq j;\ i,j \in \{1,\dots,k\}$.
Let further $\mathcal{P} := (P_1,\dots,P_k)$ with $P_i := \{x \in \mathbb{R}^d : a_{ij}^T x \leq \gamma_{ij}$ for all $j \in \{1,\dots,k\}\backslash\{i\}\}$ for all $i \in \{1,\dots,k\}$ be the set of k **cells** *of $\mathcal{H}$, and let P_i be the i-**th cell**. $\mathcal{H}$ is a $k-$**cell arrangement** if $int(P_i) \neq \emptyset$ for all $i \in \{1,\dots,k\}$.*

With a_{ij} and γ_{ij} fixed and known, we will often identify the hyperplanes $H_{a_{ij},\gamma_{ij}}$ with H_{ij}. Above definition describes a special set of hyperplanes creating cells in a given geometric space $\mathbb{R}^d$. A cell decomposition is the set of cells created by such a system of hyperplanes if it satisfies some additional constraints.

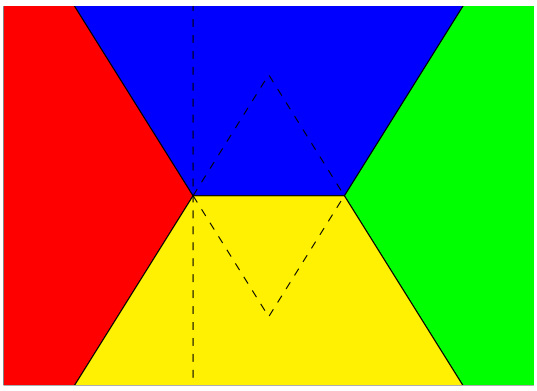
Definition 2.22 ($\mathcal{H}$-Cell Decomposition)
*Let $\mathcal{H}$ be a k-cell arrangement. The cells $\mathcal{P} := (P_1,\dots,P_k)$ of $\mathcal{H}$ are an $\mathcal{H}$-**cell decomposition** of $\mathbb{R}^d$ if*

1. $\bigcup_{i=1}^{k} P_i = \mathbb{R}^d$
2. $P_i \cap H_{ij} = P_j \cap H_{ij}$ *for all* $i \neq j;\ i,j \in \{1,\dots,k\}$

An $\mathcal{H}$-cell decomposition $\mathcal{P} := (P_1,\dots,P_k)$ partitions $\mathbb{R}^d$ into k convex and full-dimensional polyhedral cells such that each pair P_i, P_j of cells intersect their (weakly) separating hyperplane H_{ij} identically. Note that an $\mathcal{H}$-cell decomposition not only is a special kind of partition of $\mathbb{R}^d$, but that it also has a special geometric representation.
If $\mathcal{H}$ is known from the context, we will use the shorter wording **cell decomposition**. Also, if we want to emphasize the number k of cells created, we will use the term k**-cell decomposition**.
Figure 13 shows two k-cell arrangements that do not induce a cell decomposition due to a violation of the first condition, respectively the second condition. Figure 13 $a)$ shows cells that do not partition the Euclidean plane. Figure 13 $b)$ shows a k-cell arrangement that is no cell decomposition due to the (vertical, dashed) hyperplane separating the red and green cells having a non-empty intersection with the red cell, but an empty one with the green cell.
Note that the cells in Figure 13 $a)$ also violate the second condition. As we see in the following lemma, this is no coincidence.



(a) A violation of condition 1 of a cell decomposition



(b) A violation of condition 2 of a cell decomposition

Figure 13: Two k-cell arrangements in $\mathbb{R}^2$ whose cells do not form a cell decomposition.

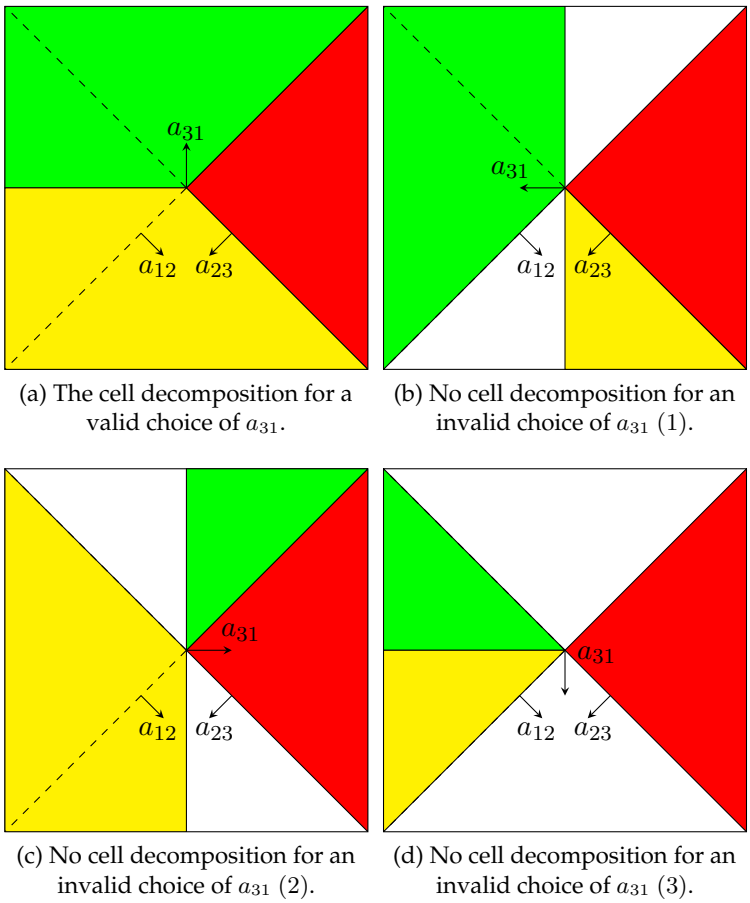
Lemma 2.23
Let $\mathcal{H}$ be a k-cell arrangement inducing cells $\mathcal{P} := (P_1, \dots, P_k)$ with $P_i \cap H_{ij} = P_j \cap H_{ij}$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$. Then $\bigcup_{i=1}^{k} P_i = \mathbb{R}^d$.

Proof. We prove the claim by showing that if $\bigcup_{i=1}^{k} P_i \neq \mathbb{R}^d$, there are indices $i \neq j;\ i, j \in \{1, \dots, k\}$ with $P_i \cap H_{ij} \neq P_j \cap H_{ij}$.
By $\bigcup_{i=1}^{k} P_i \neq \mathbb{R}^d$, there is an $x \in \mathbb{R}^d$ with $x \notin \bigcup_{i=1}^{k} P_i$. By our construction of the cells $\mathcal{P} := (P_1, \dots, P_k)$, there is an inclusion-maximal and full-dimensional convex polyhedron defined by some of the hyperplanes in $\mathcal{H}$ such that $x' \notin \bigcup_{i=1}^{k} P_i$ for all interior points x' of this polyhedron. Each facet of this polyhedron contains a point in its relative interior. Let x'' be such a point, then $x'' \in H_{ij}$ for some $i \neq j;\ i, j \in \{1, \dots, k\}$ and without loss of generality $x'' \in P_i$, and then $x'' \notin P_j$, as it is an interior point of the facet and as P_i and P_j are full-dimensional. Thus $x'' \in P_j \cap H_{ij}$, but $x'' \notin P_i \cap H_{ij}$. □

Informally, Lemma 2.23 shows that the second condition of the definition of cell decompositions implies the validity of the first one. For the sake of intuitivity of the definition, we stick with the original formulation denoting two conditions.
We now turn to some special properties of 3-cell decompositions, as these play an important role in the remainder of this chapter. Additionally, many of our figures show 3 clusters, respectively a cell decomposition into 3 cells. We start by identifying what the separation directions of such a 3-cell decomposition look like.

Lemma 2.24
Let $\mathcal{P} := (P_1, P_2, P_3)$ be an $\mathcal{H}$-cell decomposition of $\mathbb{R}^d$. Then $a_{31} \in -cone\{a_{12}, a_{23}\}$, $a_{12} \in -cone\{a_{23}, a_{31}\}$, $a_{23} \in -cone\{a_{31}, a_{12}\}$.

Proof. It suffices to show $-a_{13} = a_{31} \in -\text{cone}\{a_{12}, a_{23}\}$, the other statements hold analogously. Suppose $H_{12} \| H_{23}$ and without loss of generality $H_{12}^{\leq} \subset H_{23}^{\leq}$. Then $a_{12} = l \cdot a_{23}$ for some $l > 0$. Now suppose $a_{31} \neq l' \cdot a_{12}$, then $H_{12}^{\leq} \cap H_{13}^{\geq} \neq \emptyset$ and thus there is a $x \in H_{12}^{\leq} \cap H_{23}^{\leq} \cap H_{31}^{\leq} \Rightarrow x \notin P_1 \cup P_2 \cup P_3$, in contradiction to $\mathcal{P}$ being a cell decomposition. Thus $a_{31} = l' \cdot a_{12} = l \cdot l' \cdot a_{23}$ for some $l' \in \mathbb{R}$. With $H_{12}^{\leq} \subset H_{23}^{\leq}$, we get $l' < 0$, and thus $a_{31} \in -\text{cone}\{a_{12}, a_{23}\}$.
Now suppose a_{12}, a_{23}, a_{31} are not collinear and consider $E := \text{span}\{a_{13}, a_{12}\} \subset \mathbb{R}^d$. There is an $x \in E$ with $a_{31}^T x < \gamma_{31}$ and $a_{12}^T x < \gamma_{12}$, and thus $x \notin P_1 \cup P_2$. By this, we necessarily have $x \in P_3$ due to our cell decomposition.



(a) The cell decomposition for a valid choice of a_{31}.

(b) No cell decomposition for an invalid choice of a_{31} (1).

(c) No cell decomposition for an invalid choice of a_{31} (2).

(d) No cell decomposition for an invalid choice of a_{31} (3).

Figure 14: Only a choice of $a_{31} \in -\text{cone}\{a_{12}, a_{23}\}$ can lead to a cell decomposition.

Suppose $a_{23} \notin E$. Note that this implies $d \geq 3$ and that there then is a y with $y \perp E$, $a_{23}^T y \neq 0$ and $x' = x + y \in H_{23}^{\leq}$. With $y \perp E$ we know that $a_{12}^T x' = a_{12}^T(x+y) = a_{12}^T x + a_{12}^T y = a_{12}^T x + 0 = a_{12}^T x < \gamma_{12}$ and analogously $a_{31}^T x' = a_{31}^T x < \gamma_{31}$. Thus $x' \notin P_1 \cup P_2 \cup P_3$, in contradiction to $\mathcal{P}$ being a cell decomposition.
Let now $a_{23} \in E$. $P_2 = H_{21}^{\leq} \cap H_{23}^{\leq}$. Suppose $a_{31} \notin -\text{cone}\{a_{12}, a_{23}\}$, then $a_{31} = l_{12}a_{12} + l_{23}a_{23}$ where at least one of l_{12}, l_{23} is > 0. If $l_{12} > 0$, there is an $x \in H_{12}^{\leq} \cap H_{23}^{\leq} \cap H_{31}^{\leq}$. If $l_{23} > 0$ (and $l_{12} \leq 0$), there again is an $x \in H_{12}^{\leq} \cap H_{23}^{\leq} \cap H_{31}^{\leq}$. Figure 14 depicts these different situations. This implies $x \notin P_1 \cup P_2 \cup P_3$, proving the claim. □

We get the following immediate corollary.

Corollary 2.25
Let $\mathcal{P} := (P_1, P_2, P_3)$ be an $\mathcal{H}$-cell decomposition of $\mathbb{R}^d$. If $H_{12} \| H_{23}$ then $H_{12} \| H_{31}$.

Informally this implies that a cell decomposition of 3 cells is either induced by 3 parallel hyperplanes, or that no two of the inducing hyperplanes are parallel, but the span of their a_{ij} is two-dimensional. If we consider this statement in an only two-dimensional space, we get the following interpretation.

Lemma 2.26
Let $\mathcal{P} := (P_1, P_2, P_3)$ be an $\mathcal{H}$-cell decomposition of $\mathbb{R}^2$. The hyperplanes of $\mathcal{H}$ satisfy one of the two following conditions:

- $|H_{12} \cap H_{23} \cap H_{31}| = 1$
- $H_{12} \| H_{23} \| H_{31}$

Proof. The second condition is satisfied if a_{12}, a_{23} and a_{31} are pairwise collinear, and by Corollary 2.25, we know that this is the case if any pair out of these vectors is collinear. So let now no pair of a_{12}, a_{23} and a_{31} be collinear.
With hyperplanes being lines in $\mathbb{R}^2$, there are unique $y_1 \in H_{31} \cap H_{12}$, $y_2 \in H_{12} \cap H_{23}$, $y_3 \in H_{23} \cap H_{31}$. If $y_1 = y_2$, then $y_1 = y_2 = y_3$, and we are done. So, suppose that $y_1 \neq y_2$, implying that y_1, y_2, y_3 are pairwise different from each other. This implies that there is no $y \in P_1 \cap P_2 \cap P_3$.
We now show that $y = \frac{1}{3}(y_1 + y_2 + y_3) \notin P_1 \cup P_2 \cup P_3$. By $y_1 \in H_{31} \cap H_{12}$, we know that $y_1 \in P_1$. Additionally, $y_1 \notin H_{23}$, implying that either $y_1 \notin P_2$ or $y_1 \notin P_3$. Without loss of generality, assume $y_1 \notin P_3$. Then $a_{23}^T y_1 < \gamma_{23}$, and analogously $a_{31}^T y_2 < \gamma_{31}$ and $a_{12}^T y_3 < \gamma_{12}$. By $y_2, y_3 \in H_{23}$ we have $a_{12}^T y_2 = a_{12}^T y_3 = \gamma_{23}$, and thus $a_{12}^T y = \frac{1}{3} a_{12}^T (y_1 + y_2 + y_3) < \gamma_{12}$. Analogously, we obtain $a_{23}^T y < \gamma_{23}$ and $a_{31}^T y < \gamma_{31}$, proving the claim. □

Lemma 2.24, Corollary 2.25 and Lemma 2.26 describe what 3-cell decompositions look like. We either have three cells with separating hyperplanes parallel to each other, or we obtain a partition of $\mathbb{R}^d$ by separating hyperplane directions being in a two-dimensional subspace. In $\mathbb{R}^2$, the latter case results in a situation as depicted in Figure 14 a).
We add another statement about 3-cell decompositions revealing an important connection to general k-cell decompositions.

Lemma 2.27
Let $\mathcal{H}$ be a k-cell arrangement in $\mathbb{R}^d$. If for all $I = \{i_1, i_2, i_3\} \subset \{1, \ldots, k\}$, $\mathcal{P}^I := (P_{i_1}^I, P_{i_2}^I, P_{i_3}^I)$ with $P_i^I := \{x \in \mathbb{R}^d : a_{ij}^T x \leq \gamma_{ij}$ for all $j \in I \backslash \{i\}\}$ is a cell decompo-

sition of $\mathbb{R}^d$*, then* $\mathcal{P}^{I'} := (P_{i_1}^{I'}, \dots, P_{i_t}^{I'})$ *with* $P_i^{I'} = \{x \in \mathbb{R}^d : a_{ij}^T x \leq \gamma_{ij}$ *for all* $j \in I' \backslash \{i\}\}$ *is a cell decomposition of* $\mathbb{R}^d$ *for any* $I' := \{i_1, \dots, i_t\} \subset \{1, \dots, k\}$.

Proof. We prove the claim by induction on k. If $\mathcal{H}$ induces a 3-cell decomposition, the claim is trivial. So consider $P^{K'}$ for index set $K' := \{1, \dots, k+1\}$ and suppose that P^K is a cell decomposition for all $K \subset K'$ with $|K| = k$. $\mathcal{H}$ is a k-cell arrangement, and with $\text{int}(P_i^{K'}) \neq \emptyset$ for all $i \in K'$ and $P_i^{K'} \subset P_i^I$ for all $I \subset K'$, $P_i^I \neq \emptyset$ for all $I \subset K'$ and $i \in I$.
With P^K being a cell decomposition for $K \subset K'$ with $|K| = k$, $\text{int}(P_i^{K'}) \cap \text{int}(P_j^{K'}) \subset \text{int}(P_i^K) \cap \text{int}(P_j^K) = \emptyset$ for all $i \neq j;\ i, j \in K'$.
By Lemma 2.23, we only have to show that $P_i^{K'} \cap H_{ij} = P_j^{K'} \cap H_{ij}$ for all $i \neq j;\ i, j \in K'$. We know that $P_i^K \cap H_{ij} = P_j^K \cap H_{ij}$ for all $i \neq j;\ i, j \in K$ with $K \subset K'$, $|K| = k$. $P_i^{K'} = \bigcap_{K:|K|=k} P_i^K$, thus

$$P_i^{K'} \cap H_{ij} = (\bigcap_{K:|K|=k} P_i^K) \cap H_{ij} = \bigcap_{K:|K|=k} (P_i^K \cap H_{ij}) =$$
$$\bigcap_{K:|K|=k} (P_j^K \cap H_{ij}) = (\bigcap_{K:|K|=k} P_j^K) \cap H_{ij} = P_j^{K'} \cap H_{ij}$$

for all $i \neq j;\ i, j \in K'$. This proves the claim. □

Note that Lemma 2.27 includes the case $I' = \{1, \dots, k\}$. Informally, Lemma 2.27 states that a k-cell arrangement for which all subset of 3 indices induce a cell decomposition induces a k-cell decomposition. Equivalently, we know that if a k-cell arrangement does not induce a cell decomposition, there is a system of 3 indices and corresponding hyperplanes that do not yield a 3-cell decomposition. We informally say that a clustering $C := (C_1, \dots, C_k)$ **allows a cell decomposition** if there is a cell decomposition $\mathcal{P} := (P_1, \dots, P_k)$ with $C_i \subset \text{int}(P_i)$ for $i \in \{1, \dots, k\}$. With our knowledge about the structure and importance of 3-cell decompositions (see Lemmata 2.24, 2.26, 2.27), we get several hints as to how to construct an example for a clustering that does not allow a cell decomposition, and which is minimal with respect to all relevant measures.

Lemma 2.28
There is a strictly separable 3*-clustering of a set of* 6 *points in* $\mathbb{R}^2$ *that does not allow a cell decomposition. All of the given parameters are minimal for this example.*

Proof. We prove the claim by constructing a point set and clustering with the given properties, and then showing that there is no strictly separable clustering

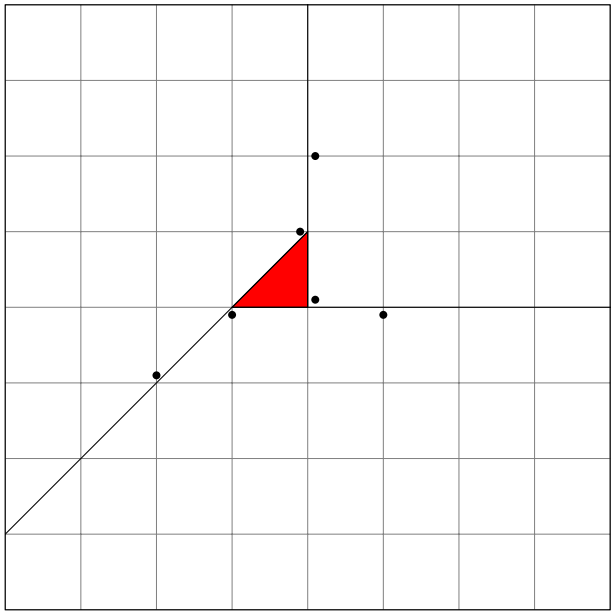


Figure 15: A minimal example for a strictly separable clustering that allows no cell decomposition.

that does not allow a cell decomposition if we lower the dimension, the number of clusters or the number of points.
Let $0 < \epsilon \in \mathbb{R}$ be arbitrarily small. Let

$$X := \left\{ \begin{pmatrix} -2 \\ \epsilon - 1 \end{pmatrix}, \begin{pmatrix} -1 \\ -\epsilon \end{pmatrix}, \begin{pmatrix} -\epsilon \\ 1 \end{pmatrix}, \begin{pmatrix} \epsilon \\ \epsilon \end{pmatrix}, \begin{pmatrix} \epsilon \\ 2 \end{pmatrix}, \begin{pmatrix} 1 \\ -\epsilon \end{pmatrix} \right\} \subset \mathbb{R}^2,$$

and let X be partitioned into 3 clusters

$$C_1 = \left\{ \begin{pmatrix} -2 \\ \epsilon - 1 \end{pmatrix}, \begin{pmatrix} -\epsilon \\ 1 \end{pmatrix} \right\}, C_2 = \left\{ \begin{pmatrix} -1 \\ -\epsilon \end{pmatrix}, \begin{pmatrix} 1 \\ -\epsilon \end{pmatrix} \right\}, C_3 = \left\{ \begin{pmatrix} \epsilon \\ \epsilon \end{pmatrix}, \begin{pmatrix} \epsilon \\ 2 \end{pmatrix} \right\}.$$

Let finally P_1, P_2, P_3 be the cells of C_1, C_2, C_3 for any choice of separating hyperplanes. Figure 15 depicts an example. The center of gravity of the 'marked triangle' is not in $P_1 \cup P_2 \cup P_3$ so that $\mathcal{P} := (P_1, P_2, P_3)$ is no cell decomposition. Clearly, such a triangle always exists.
In $\mathbb{R}$, a strictly separable clustering directly induces a partition of $\mathbb{R}$ into intervals, yielding a cell decomposition. If we only have two clusters C_1, C_2 in $\mathbb{R}^d$, then the two halfspaces created by the separating hyperplane between the two clusters form a cell decomposition of $\mathbb{R}^d$.

By Lemma 2.27, if the cells $P_1, \ldots, P_k$ do not form a cell decomposition of $\mathbb{R}^d$, then there is a subset of 3 indices in $\{1, \ldots, k\}$ for which the respective cells do not form a 3-cell decomposition of $\mathbb{R}^d$. Thus it suffices to only look at systems of 3 clusters C_1, C_2, C_3 with cells P_1, P_2, P_3.
Now suppose we have less than 6 points, without loss of generality $C_3 = \{x_3\}$, and thus $|C_3| = 1$. Let $H_{12} = \{x \in \mathbb{R}^d : a_{12}^T x = \gamma_{12}\}$ be the strictly separating hyperplane between C_1 and C_2. If $x_3 \in H_{12}$, for any point $y \in H_{12}$ with $y \neq x_3$ there is a hyperplane H_{13} and a hyperplane H_{23} separating C_3 from C_1, respectively C_2, strictly and intersecting H_{12} in y. Choosing two hyperplanes intersecting H_{12} in the same $y \in H_{12}$ yields the desired cell decomposition.
Let now, without loss of generality, $a_{12}^T x_3 < \gamma_{12}$. Thus H_{12} separates $C_1 \cup C_3$ from C_2. As C_1 and C_3 are strictly separable, there is an H_{13} separating the clusters strictly with $H_{13} \cap H_{12} = \{y\} \neq \emptyset$. Further, H_{23} can be chosen as a slight perturbation of H_{12} intersecting H_{12} in y and such that $a_{23} \in -\text{cone}\{a_{31}, a_{12}\}$ (as in Lemma 2.24). This yields the desired cell decomposition. □

In the following, we are not only interested in any cell decompositions, but in a special type of cell decompositions called 'a-induced cell decompositions'. These have a representation by an underlying arrangement where the separation directions are constructed using a vector $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ as follows.

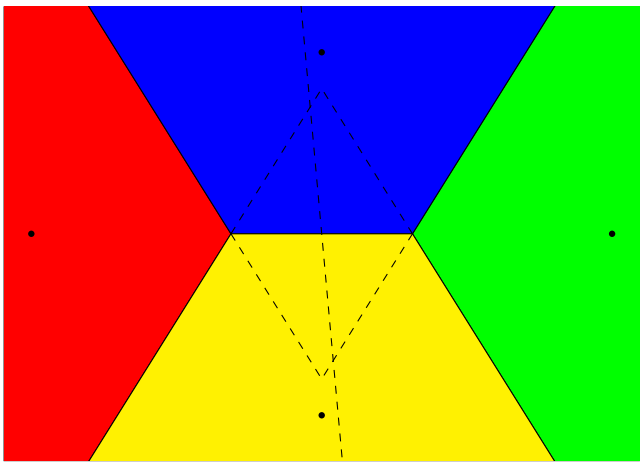
Definition 2.29 (a-induced Cell Decomposition)
Let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $a_i \neq a_j$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$. An **$a$-induced cell decomposition** *of $\mathbb{R}^d$ is an $\mathcal{H}$-cell decomposition where $a_{ij} := a_j - a_i$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$.*

With the a-induction of a cell decomposition only being a restriction to the directions of the hyperplanes of the inducing k-cell arrangement, we will also talk about **a-induced k-cell arrangements**.
The vector $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ can be interpreted as a number of k **sites** in $\mathbb{R}^d$, with one site for each cell. The separation directions of the cells then are given by the vector differences between their respective sites. Using this interpretation, and adding some restrictions on the positioning of the separating hyperplanes, will lead us to a class of generalized Voronoi diagrams, called **power diagrams** [Aur87; AHA98].
We complement our investigation of 3-cell decompositions by noting that any 3-cell decomposition of $\mathbb{R}^d$ is a-induced for some $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$.

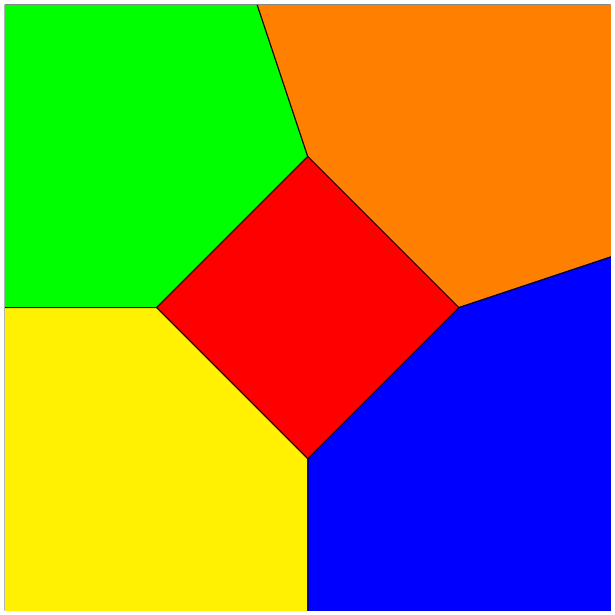
Lemma 2.30
Let $\mathcal{P} := (P_1, P_2, P_3)$ be an $\mathcal{H}$-cell decomposition of $\mathbb{R}^d$. Then $\mathcal{P}$ is an a-induced cell decomposition for some $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$.

Proof. We know that $a_{31} \in -\text{cone}\{a_{12}, a_{23}\}$ by Lemma 2.24. Thus there are scalar factors $l_{12}, l_{23}, l_{31} > 0$ with $l_{31}a_{31} = -l_{12}a_{12} - l_{23}a_{23} \Leftrightarrow l_{31}a_{31} + l_{12}a_{12} + l_{23}a_{23} = 0$. By this, we can choose $a_1 = 0 \in \mathbb{R}^d$, $a_2 = l_{12}a_{12}$ and $a_3 = l_{13}a_{13}$ such that $\mathcal{H}$ is a-induced for $a := (a_1^T, a_2^T, a_3^T)^T$. □

Of course, in the above construction it is not necessary to set $a_1 = 0$. If a different a_1 is chosen, all other a_i are translated by a_1. Note further that fixing l_{12}, we can still choose l_{31}, l_{23} appropriately. In this case, they are uniquely determined.
On the other hand, a k-cell decomposition with $k > 3$ is not necessarily a-induced. Figure 16 shows two cell decompositions in $\mathbb{R}^2$. Due to the symmetry of the example in Figure 16 a), the dashed hyperplane separating the red and the green cluster has to be vertical in the figure for the cell decomposition to be a-induced, but it is not. The dots indicate the relative positions that any possible sites of the cells would have to have with respect to each other to induce the 'visible' hyperplanes.
Figure 16 b) shows that even the 'visible' parts of a cell decomposition do not have to be a-induced. Suppose $C := (C_1, C_2, C_3, C_4, C_5)$, where C_1 is the red cell, C_2 the green one, C_3 the yellow one, C_4 the blue one and C_5 the orange one. We now try to construct corresponding sites a_1, a_2, a_3, a_4, a_5.
W.l.o.g. we assume $a_1 = 0 \in \mathbb{R}^2$. Due to the separating hyperplanes of the red cell to the other ones, their respective sites have to satisfy $a_2 = \tau_2 \cdot \binom{-1}{1}$, $a_3 = \tau_3 \cdot \binom{-1}{-1}$, $a_4 = \tau_4 \cdot \binom{1}{-1}$ and $a_5 = \tau_5 \cdot \binom{1}{1}$ for some $\tau_2, \tau_3, \tau_4, \tau_5 > 0$. Further, $\tau_2 = \tau_3 = \tau_4$ due to the directions of the separating hyperplanes of C_2 and C_3 (green and yellow) and C_3 and C_4 (yellow and blue). The hyperplane between C_2 and C_5 implies that $\tau_5 > \tau_2$, but the hyperplane between C_4 and C_5 implies that $\tau_5 < \tau_4 = \tau_2$. This yields a contradiction, and implies that we cannot choose sites inducing the hyperplanes shown.
Note further that choosing the separating hyperplane directions of clusters according to some sites does not even necessarily yield a partition of $\mathbb{R}^d$. Figure 17 shows the 'cells' of the clustering of Figure 10 with corresponding sites.
It is easy to see that any Voronoi diagram is an example for an a-induced cell decomposition, with the sites a_i being the centers used for the construction of the Voronoi cells. Looking at these examples, we see that the notion of a-induced cell decompositions is well-defined as a special case of general cell decompositions.



(a) Not every $\mathcal{H}$-induced cell decomposition is an a-induced cell decomposition.



(b) Even the 'visible' parts of a cell decomposition are not always a-induced.

Figure 16: Two examples for cell decompositions that are not a-induced.

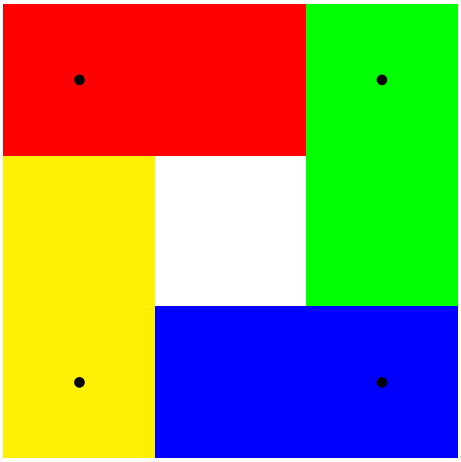


Figure 17: The separation directions of the clusters are induced by the sites shown, yet the cells do not form a partition of $\mathbb{R}^2$.

In the following, we identify a connection between a-induced cell decompositions and the vertices of the gravity polytope. We start by following the arguments in [BG09], as they are a fundamental basis to the remainder of this section and the following chapter. First, we need some more terminology.

Definition 2.31 (Clustering Graph)
Let C be a PSC. A **clustering graph** *$G = G(C) = (V, E)$ is a complete digraph with $V := \{v_1, \dots, v_k\}$. Let $\mathcal{H} := \{H_{a_{ij},\gamma_{ij}} : i \neq j;\ i, j \in \{1, \dots, k\}\}$, and let $a_{ji} = -a_{ij}$ and $\gamma_{ji} = -\gamma_{ij}$ such that $\max_{x \in C_i} a_{ij}^T x \leq \gamma_{ij} \leq \min_{x \in C_j} a_{ij}^T x$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$.*
A **γ-clustering graph** *is a weighted clustering graph $G = G(C) = (V, E, w)$ with $w : E \to \mathbb{R}$, $w((v_i, v_j)) = \gamma_{ij}$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$.*
Let further $\max_{x \in C_i} a_{ij}^T x = \gamma_{ij}^*$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$. A **$\gamma^*$-clustering graph** *is a weighted clustering graph $G = G(C) = (V, E, w)$ with $w : E \to \mathbb{R}$, $w((v_i, v_j)) = \gamma_{ij}^*$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$.*

$v_i \in V$ is associated with C_i for all $i \in \{1, \dots, k\}$. The cycles of a clustering graph correspond to cyclical exchanges. The cycles of γ- and γ^*-clustering graphs correspond to 'optimal' cyclical exchanges, depending on either the actual positions of the hyperplanes, or bounds on the positions of the points in the clusters.
Next, we require a simple graph theoretical statement [BG09].

Lemma 2.32
Let $G = (V, E, w)$ be a complete, weighted digraph without cycles of strictly positive weight. Then there is a weight-function $w' : E \to \mathbb{R}_0^+$ such that all cycles of $G' = (V, E, w + w')$ have length 0.

Proof. We consider the set W of weight-functions $w' : E \to \mathbb{R}_0^+$ such that $w + w'$ does not create a positive-weight cycle in G'. As w' with $w'(e) = 0$ for all $e \in E$ satisfies this property, $W \neq \emptyset$. We now choose $w' \in W$ such that the number of strictly negative cycles in G' is minimized and such that no single edge-weight can be increased without creating a strictly positive cycle. Increasing a single edge-weight does not increase the number of strictly negative cycles in G', implying that this assumption is no restriction.
If there are no cycles of strictly negative weight, we are done. Otherwise, let $CY = \{e_1, \dots, e_t\}$ be a cycle in G' with $e_1, \dots, e_t \in E$ and $(w + w')(CY) < 0$. None of its edges may be increased without creating a cycle of strictly positive weight in G', which means that e_i is on a cycle CY_i with $(w + w')(CY_i) = 0$ for all $i \in \{1, \dots, t\}$. Then $CY' = \bigcup_{i=1}^{t} (CY_i \backslash \{e_i\})$ is a closed walk in G', i.e. a sequence of cycles. Further

$$(w+w')(CY_i\backslash\{e_i\}) = (w+w')(CY_i) - (w+w')(e_i) = 0 - (w+w')(e_i) = -(w+w')(e_i).$$

As $0 > (w + w')(CY) = \sum_{i=1}^{t} (w + w')(e_i)$, we have

$$0 < -(w+w')(CY) = \sum_{i=1}^{t} -(w+w')(e_i) = \sum_{i=1}^{l} (w+w')(CY_i\backslash\{e_i\}) = (w+w')(CY'),$$

implying that there is a cycle in G' of strictly positive weight. This yields a contradiction, proving the claim. □

The γ^*-clustering graph $G = (V, E, w)$ of a vertex v^* of Q and a vector $a \in \mathbb{R}^{d \cdot k}$ with $a^T v^* > a^T v$ for any other $v \in Q$ only contains cycles of strictly negative weight, as each cyclical exchange applied to the clustering associated with v^* leads to a clustering with gravity vector v of lower value $a^T v$. Thus, we can apply Lemma 2.32 and assume that $w'((v_i, v_j)) > 0$ for each edge $(v_i, v_j) \in E$. We then only have cycles of length zero with respect to $(w + w')$. Doing so allows us to prove a direct generalization of Theorem 2.15 [BG09]. As a service to the reader, we adapt the proof to our notation.

Theorem 2.33 (Brieden, Gritzmann 09)
Let v^* be a vertex of Q, let $C := (C_1, \dots, C_k)$ be the PSC associated with v^* and let $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $a^T v^* > a^T v$ for any $v \in Q \backslash \{v^*\}$. Then there is an a_κ-induced cell decomposition $\mathcal{P} := (P_1, \dots, P_k)$ with $C_i \subset int(P_i)$ for $i \in \{1, \dots, k\}$, where $a_\kappa := (\frac{a_1}{\kappa_1}^T, \dots, \frac{a_k}{\kappa_k}^T)^T$.

Proof. Recalling 2.15, we only have to consider the positioning of the hyperplanes.
Let v^* be a vertex of Q and let $a := (a_1, \dots, a_k)^T \in \mathbb{R}^{d \cdot k}$ be any vector in int$(N(v^*))$, where $N(v^*)$ is the cone of outer normals of v^* with respect to Q. We show that there is a viable choice of the γ_{ij}, $i \neq j;\ i, j \in \{1, \dots, k\}$ such that $\mathcal{H} := \{H_{a_{ij}, \gamma_{ij}} : i \neq j;\ i, j \in \{1, \dots, k\}\}$ using $a_{ij} := \frac{a_j}{\kappa_j} - \frac{a_i}{\kappa_i}$ and γ_{ij} for all $i \neq j;\ i, j \in \{1, \dots, k\}$ is a cell decomposition $\mathcal{P} := (P_1, \dots, P_k)$ of $\mathbb{R}^d$ with $C_i \subset$ int(P_i) for $i \in \{1, \dots, k\}$.
Let G be the γ^*-clustering graph of C, then we choose $\gamma_{ij} := w + w'((v_i, v_j))$ according to Lemma 2.32 and the remark following it, implying that $\gamma_{ij}^* < \gamma_{ij}$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$. Thus $C_i \subset$ int(P_i) for all $i \in \{1, \dots, k\}$, which implies that all P_i are full-dimensional. With $a_{ij}^T x \leq \gamma_{ij}$ for any $x \in P_i$, and $a_{ij}^T x \geq \gamma_{ij}$ for any $x \in P_j$, int$(P_i) \cap$ int$(P_j) = \emptyset$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$.
By Lemma 2.23, we only have to show that $P_i \cap H_{ij} = P_j \cap H_{ij}$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$. We assume $P_i \cap H_{ij} \neq P_j \cap H_{ij}$ for some $i \neq j;\ i, j \in \{1, \dots, k\}$. Let, without loss of generality, $y \in H_{ij}$ with $y \in P_i$, but $y \notin P_j$. Then there is a $t \in \{1, \dots, k\} \backslash \{i, j\}$ with $a_{jt}^T y > \gamma_{jt}$. Since $y \in P_i$, $a_{it}^T y \leq \gamma_{it}$. Together, we have

$$0 = 0^T y = (a_{ij} + a_{jt} + a_{ti})^T y > \gamma_{ij} + \gamma_{jt} + \gamma_{ti} = 0,$$

yielding a contradiction, as, in the γ-clustering graph, any cycle (of γ_{ij}-values) sums up to 0. This proves the claim. □

Informally, Theorem 2.33 shows that clusterings of vertices of the gravity polytope allow a_κ-induced cell decompositions. With the clustering in Figure 15 not allowing any cell decomposition, we know that it is not a vertex of the respective gravity polytope.
Additionally, we saw examples of cell decompositions that cannot be a-induced (Figure 16). Thus, having a clustering that allows only a general cell decomposition, but not an a-induced cell decomposition, implies that the gravity vector of the clustering is not a vertex of the gravity polytope.

A slight refinement of the notion of a-induced cell decompositions will yield the desired vertex characterization, in the next section.

2.4 Full Cell Decompositions

We begin by showing that Theorem 2.33 actually proves the existence of a special type of a-induced cell decompositions for clusterings of vertices of the gravity polytope. We need some formal terminology.
The construction in the proof is based on Lemma 2.32, and creates a system of hyperplanes where, for any index set $I := \{i_1, \ldots, i_t\} \subset \{1, \ldots, k\}$, $\gamma_{i1} + \cdots + \gamma_{i_t} = 0$. The following lemma shows that this property is induced by all triples summing up to zero.

Lemma 2.34
Let $G = (V, E, w)$ be a complete weighted digraph with $w((v_i, v_j)) = -w((v_j, v_i))$ for all $v_i \neq v_j \in V$.
Let G contain a cycle with positive edge-weight and at least 3 edges. Then there is a cycle of positive edge-weight with exactly 3 edges in G.

Proof. Without loss of generality, let $CY = (v_1, v_2, v_3, \ldots, v_k, v_1)$ be a cycle of $k > 3$ edges and $w(CY) > 0$. Then $CY' := (v_1, v_2, v_3, v_1)$ is a cycle of 3 edges such that $(v_1, v_2), (v_2, v_3) \in CY$ and $(v_3, v_1) \notin CY$.
If $w((v_1, v_2)) + w((v_2, v_3)) + w((v_3, v_1)) > 0$, we are done, as CY' satisfies the claim. So suppose $w((v_1, v_2)) + w((v_2, v_3)) + w((v_3, v_1)) \leq 0$. We derive $CY'' = (v_1, v_3, \ldots, v_k, v_1)$ by replacing edges (v_1, v_2) and (v_2, v_3) by edge (v_1, v_3). It is a cycle in G consisting of $k - 1$ edges. As

$$w((v_1, v_2))+w((v_2, v_3))+w((v_3, v_1)) \leq 0 \Leftrightarrow w((v_1, v_2))+w((v_2, v_3)) \leq w((v_1, v_3)),$$

we see that $w(CY'') \geq w(CY) > 0$. Thus, CY'' is a cycle with an edge less than C and with higher total edge-weight than CY.
Repeating this contraction until the resulting cycle only has 3 edges, or until the cycle used for the contraction has positive edge-weight, proves the claim. □

Lemma 2.34 again underlines the importance of the triples of cells in the context of cell decompositions. If there is no cycle of 2 or 3 edges with strictly positive edge-weight, there is no such cycle in the whole graph. This statement is directly related to Lemma 2.27, which showed that if all 3-subsets of indices of a k-cell

arrangement induce a 3-cell decomposition, the arrangement must induce a k-cell decomposition.
Looking at Lemma 2.34, we know that we can represent the special type of cell decompositions constructed in Theorem 2.33 by only considering triples of indices in $\{1, \dots, k\}$.

Definition 2.35 (Full a-induced Cell Decomposition)
Let $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $a_i \neq a_j$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$. A **full a-induced cell decomposition** *of $\mathbb{R}^d$ is a cell decomposition where $a_{ij} := a_j - a_i$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$ and $\gamma_{i_1 i_2} + \gamma_{i_2 i_3} + \gamma_{i_3 i_1} = 0$ for all $\{i_1, i_2, i_3\} \subset \{1, \dots, k\}$.*

We obtain the following direct corollary to Theorem 2.33.

Corollary 2.36
Let v^* be a vertex of Q, let $C := (C_1, \dots, C_k)$ be the PSC associated with v^* and let $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $a^T v^* > a^T v$ for any $v \in Q \backslash \{v^*\}$. Then there is a full a_κ-induced cell decomposition $\mathcal{P} := (P_1, \dots, P_k)$ with $C_i \subset int(P_i)$ for $i \in \{1, \dots, k\}$, where $a_\kappa := (\frac{a_1}{\kappa_1}^T, \dots, \frac{a_k}{\kappa_k}^T)^T$.

Note that $a^T v^* > a^T v$ for any $v \in Q \backslash \{v^*\}$ implies $\frac{a_i}{\kappa_i} \neq \frac{a_j}{\kappa_j}$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$.
If the gravity vector $v^* = v(C)$ of a PSC C we consider only satisfies $a^T v^* \geq a^T v$ for each $v \in Q \backslash \{v^*\}$, and if $\frac{a_i}{\kappa_i} \neq \frac{a_j}{\kappa_j}$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$, we obtain a slightly weaker result.

Corollary 2.37
Let $v^* \in Q$, let $C := (C_1, \dots, C_k)$ be a PSC associated with v^* and let $a := (a_1^T, \dots, a_k^T)^T$
$\in \mathbb{R}^{d \cdot k}$ with $a^T v^* \geq a^T v$ for any $v \in Q \backslash \{v^*\}$. Then there is a full a_κ-induced cell decomposition $\mathcal{P} := (P_1, \dots, P_k)$ with $C_i \subset P_i$ for $i \in \{1, \dots, k\}$, where $a_\kappa := (\frac{a_1}{\kappa_1}^T, \dots, \frac{a_k}{\kappa_k}^T)^T$.

Proof. The claim follows directly from the proof of Theorem 2.33, by replacing the strict inequality $a^T v^* > a^T v$ with $a^T v^* \geq a^T v$. The only difference lies in the fact that we cannot guarantee that we can choose $\gamma_{ij} > \gamma_{ij}^*$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$. □

We note an important property of full a-induced cell decompositions.

Lemma 2.38
Let $\mathcal{P} := (P_1, \dots, P_k)$ be a full a_κ-induced cell decomposition of $\mathbb{R}^d$ and let, for any index set $I := \{i_1, \dots, i_t\} \subset \{1, \dots, k\}$, $\mathcal{P}^I := (P_{i_1}^I, \dots, P_{i_t}^I)$ with $P_i^I := \{x \in \mathbb{R}^d :$

*$a_{ij}^T x \leq \gamma_{ij}$ for all $j \in I \backslash \{i\}\}$. Then P^I is a full a_κ^I-induced cell decomposition of $\mathbb{R}^d$, where $a_\kappa^I := (\frac{a_{i_1}}{\kappa_{i_1}}^T, \ldots, \frac{a_{i_t}}{\kappa_{i_t}}^T)^T$. We call P_i^I the i-**th** I-**cell**.*

Proof. Let $\mathcal{P} := (P_1, \ldots, P_k)$ be a full a_κ-induced cell decomposition of $\mathbb{R}^d$ and let $I \subset \{1, \ldots, k\}$ with $|I| = t$, and $P^I := (P_{i_1}^I, \ldots, P_{i_t}^I)$. We have $C_i \subset \text{int}(P_i) \subset P_i^I$ for all $i \in I$ and $a_{i_1,i_2}^T x \leq \gamma_{i_1,i_2}$ for any $x \in P_{i_1}^I$ and $a_{i_1,i_2}^T x \geq \gamma_{i_1,i_2}$ for any $x \in P_{i_2}^I$, so that $\text{int}(P_{i_1}^I) \cap \text{int}(P_{i_2}^I) = \emptyset$ for all $i_1 \neq i_2;\ i_1, i_2 \in I$.
With $a_{ij} := \frac{a_j}{\kappa_j} - \frac{a_i}{\kappa_i}$ for $i \neq j;\ i, j \in \{1, \ldots, k\}$ and the γ_{ij} summing up to 0 for any cycle of indices in $\{1, \ldots, k\}$, the same holds for $i \neq j;\ i, j \in I$, respectively cycles of indices in I. Thus, the proof of $P_i^I \cap H_{ij} = P_j^I \cap H_{ij}$ for all $i \neq j;\ i, j \in I$ in Theorem 2.33 works analogously again, and we see that P^I is a full a_κ^I-induced cell decomposition. □

Informally, having a full a-induced cell decomposition means that the underlying arrangement still yields an a^I-induced cell decomposition if we only use the hyperplanes of any subset of indices I, respectively clusters. We will use this fact in the following chapter to stabilize the data classification approach presented there.
The notion of full a-induced cell decompositions is well-defined. Not every a-induced cell decomposition is a full a-induced cell decomposition. Figure 18 shows a cell decomposition which is a-induced by the sites shown. The central, vertical hyperplane separating the red and the green cluster is 'ill-positioned'. We note this fact by recalling Lemma 2.38, and seeing that e.g. the subsystem of hyperplanes of the red, green and blue clusters does not yield a cell decomposition.
On the other hand, there are examples for full a-induced cell decompositions. Figure 19 shows a Voronoi diagram. The cells of a Voronoi diagram always are a full a-induced cell decomposition. In the next section we turn to this intrinsic connection in more detail.
Having an a-induced cell decomposition is not the only way to create a cell decomposition with the property in Lemma 2.38, i.e. such that each subset of indices still leads to a cell decomposition. We define this class of cell decompositions.

Definition 2.39 (Full $\mathcal{H}$-Cell Decomposition)
An $\mathcal{H}$-cell decomposition $\mathcal{P} := (P_1, \ldots, P_k)$ is a **full $\mathcal{H}$-cell decomposition** *if for all $I = \{i_1, i_2, i_3\} \subset \{1, \ldots, k\}$, $\mathcal{P}^I := (P_{i_1}^I, P_{i_2}^I, P_{i_3}^I)$ with $P_i^I := \{x \in \mathbb{R}^d : a_{ij}^T x \leq \gamma_{ij}$ for all $j \in I \backslash \{i\}\}$ is an $\mathcal{H}^I$-cell decomposition of $\mathbb{R}^d$, where $\mathcal{H}^I := \{H_{ij} \in \mathcal{H} : i \neq j;\ i, j \in I\}$.*

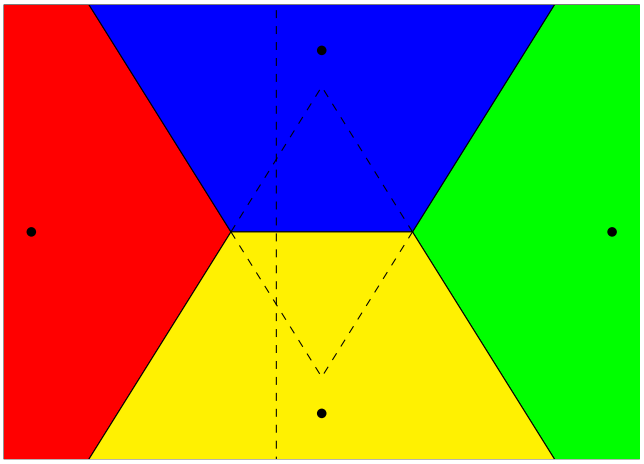


Figure 18: Not every a-induced cell decomposition is a full a-induced cell decomposition.

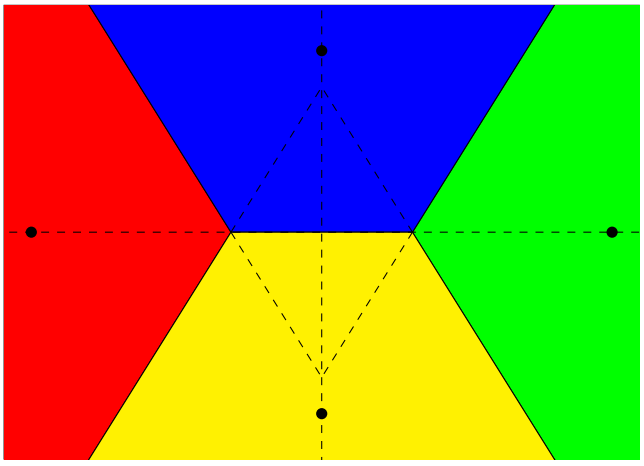


Figure 19: A Voronoi diagram is a full a-induced cell decomposition.

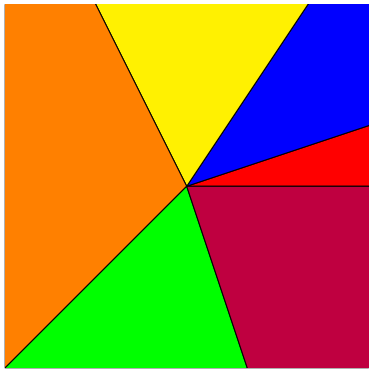


Figure 20: A 6-cell decomposition of $\mathbb{R}^2$. The visible system of hyperplanes can both be complemented to obtain a full a-induced cell decomposition, as well as a full cell decomposition which is not a-induced by any vector of sites a.

Again, we will use the simpler wording **full cell decomposition** if $\mathcal{H}$ is clear from the context. Note that any 2- or 3-cell decomposition trivially is a full cell decomposition, by this definition. Note further that the definition precisely satisfies the prerequisites needed for an application of Lemma 2.27. By this, we know that any subset of indices $I \subset \{1, \dots, k\}$ yields a full cell decomposition P^I, not only the ones with $|I| = 3$.

Lemma 2.38 shows that full a-induced cell decompositions are full cell decompositions as well. After all, this was the intention of our definition. Figure 16 showed two examples for cell decompositions that are not a-induced for any vector of sites a.

There also are full cell decompositions which are not a-induced by any vector of sites a. Consider the cell decomposition in Figure 20. The 'invisible' hyperplanes H_{ij} only have to satisfy $|H_{ij} \cap P_i| = 1$ for all $i \in \{1, \dots, k\}$ to yield a full cell decomposition. Of course, we can choose them in an a-induced way, but we do not necessarily have to do so. Note that this example considers only the representation of a full cell decomposition, with the 'top-level cells' induced remaining fixed.

It is possible to show that full cell decompositions can be defined equivalently as full a-induced cell decomposition for some vector of sites $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ if we place mild restrictions on the hyperplane directions and positions. We sum up these restrictions.

Definition 2.40 (Full Cell Decomposition in General Position)
A full $\mathcal{H}$-cell decomposition $\mathcal{P} := (P_1, \dots, P_k)$ satisfying

- *a_{ij} and a_{il} are not collinear for all $\{i, j, l\} \subset \{1, \dots, k\}$*
- *$H_{jl} \cap H_{li} \neq H_{jt} \cap H_{ti}$ for all $\{i, j, l, t\} \subset \{1, \dots, k\}$*

is in **general position**.

The first condition implies that a single cell is not separated from two different cells by the same vector direction. As any subset of indices still yields a cell decomposition, $H_{ij} \cap H_{jl} \cap H_{li} \neq \emptyset$. The second condition implies that $H_{ij} \cap H_{jl} \cap H_{li} \neq H_{ij} \cap H_{jt} \cap H_{ti}$ for index sets $\{i, j, l\}, \{i, j, t\}$. This means that the intersections of two triples of hyperplanes are not identical. Note that the equality case is only possible if $\dim(\text{span}\{a_{jl}, a_{li}, a_{ti}\}) = 2$.
We are now ready to prove the above claim.

Lemma 2.41
Any full $\mathcal{H}$-cell decomposition $\mathcal{P} := (P_1, \dots, P_k)$ in general position is a full a-induced cell decomposition for some $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$.

Proof. Let $\mathcal{H}$ induce a full cell decomposition, i.e. a cell decomposition such that for any $I := \{i_1, i_2, i_3\} \subset \{1, \dots, k\}$, $\mathcal{P}^I := (P^I_{i_1}, P^I_{i_2}, P^I_{i_3})$ with $P^I_i := \{x \in \mathbb{R}^d : a_{ij}^T x \leq \gamma_{ij} \forall j \in I \backslash \{i\}\}$ is a cell decomposition of $\mathbb{R}^d$. With $\mathcal{P}$ being in general position, so is $\mathcal{P}^I$ for all I.
Due to the first condition to $\mathcal{H}$, this implies that each triple of respective hyperplanes has a common intersection, recall Corollary 2.25 and Lemma 2.26. Additionally, by Lemma 2.30, for each triple of indices $I := \{i_1, i_2, i_3\}$, there are $a_{i_1}, a_{i_2}, a_{i_3}$ that satisfy the claim. It remains to show that the a_i can be chosen like that for the whole system of indices $i \in \{1, \dots, k\}$ at once. We do so by induction.
For $k = 3$, as stated above, we know that we can choose a_1, a_2, a_3. Thus, suppose the claim holds for $k - 1$, i.e. there are vectors $a_1, \dots, a_{k-1}$ such that $a_{ij} \| a_j - a_i$ for all $i \neq j;\ i, j \in \{1, \dots, k-1\}$. We now prove the claim for k cells.
For any index set $I_{ijk} = \{i, j, k\}$, an a_k^{ij} can be chosen such that $a_{ik} \| a_k^{ij} - a_i$ and $a_{jk} \| a_k^{ij} - a_j$. It is determined uniquely by the fixed a_i and a_j, as a_{ik} and a_{jk} are not collinear and as $\dim(\text{span}\{a_{ij}, a_{ik}, a_{jk}\}) = 2$, by Lemma 2.24. We now prove the claim by showing that $a_k^{ij} = a_k^{12}$ for all $i \neq j;\ i, j \in \{1, \dots, k-1\}$, by contradiction. To do so, we assume without loss of generality that $a_k^{13} \neq a_k^{12}$.

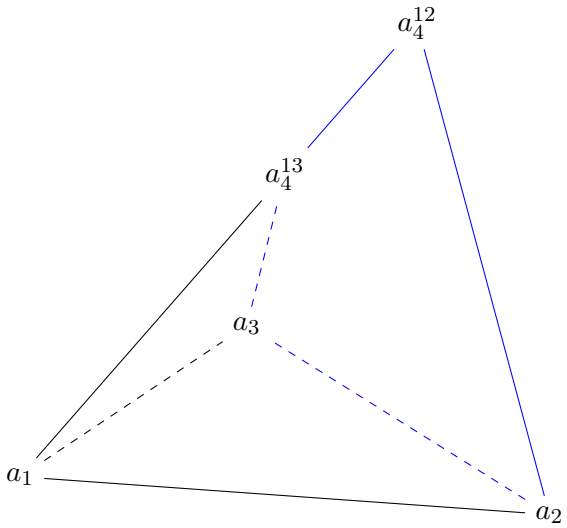


Figure 21: A 3-dimensional span of a_{14}, a_{24}, a_{34} implies a 3-dimensional E_{234} (blue) if $a_4^{12} \neq a_4^{13}$.

We consider the cell decompositions of index sets $I_{12k} = \{1, 2, k\}$, $I_{13k} = \{1, 3, k\}$, $I_{23k} = \{2, 3, k\}$. For ease of notation, we use $k = 4$ in the following.
Suppose $\dim(\text{span}\{a_{14}, a_{24}, a_{34}\}) = 3$. With our claim satisfied for I_{123}, I_{124} and I_{134}, $a_4^{12} \in \{a_1 + \mathbb{R}a_{14}\} \cap \{a_2 + \mathbb{R}a_{24}\}$ and $a_4^{13} \in \{a_1 + \mathbb{R}a_{14}\} \cap \{a_3 + \mathbb{R}a_{34}\}$. Let

$$E_{234} := \{a_2 + \text{span}\{a_{23}, a_{24}, a_{34}\}\}.$$

We know that $\dim(E_{234}) = 2$, as I_{234} yields a cell decomposition, implying that $a_{23} \in -\text{cone}\{a_{34}, a_{42}\}$. As $\{a_2 + \mathbb{R}a_{24}\} \cup \{a_3 + \mathbb{R}a_{34}\} \subset E_{234}$, we know that $a_4^{12}, a_4^{13} \in E_{234}$. As $a_4^{12}, a_4^{13} \in \{a_1 + \mathbb{R}a_{14}\}$ and as $a_4^{13} \neq a_4^{12}$, we conclude that $\{a_1 + \mathbb{R}a_{14}\} \subset E_{234}$ and obtain a contradiction of $\dim(\text{span}\{a_{14}, a_{24}, a_{34}\}) = 3$ and $\dim(E_{234}) = 2$. Figure 21 depicts the situation for $a_4^{13} \neq a_4^{12}$.
Let now $\dim(\text{span}\{a_{14}, a_{24}, a_{34}\}) \neq 3$. We only have to consider the case $\dim(\text{span}\{a_{14}, a_{24}, a_{34}\}) = 2$, as $\dim(\text{span}\{a_{14}, a_{24}, a_{34}\}) > 1$ due to a_{14}, a_{24} and a_{34} not being pairwise collinear. Let

$$E_{123} := \{a_1 + \text{span}\{a_{12}, a_{13}\}\}$$

be the affine plane containing a_1, a_2 and a_3. If $\{a_1 + \mathbb{R}a_{14}\} \not\subset E_{123}$, then $a_4^{12} \notin E_{123}$ and $a_4^{13} \notin E_{123}$ due to a_{12} and a_{14} not being collinear, and by this $\{a_2 + \mathbb{R}a_{24}\} \not\subset E_{123}$ and $\{a_3 + \mathbb{R}a_{34}\} \not\subset E_{123}$. As a_1, a_2, a_3 are distinct and a_{12} and a_{23} are not

collinear, then $\dim(\text{span}\{a_{14}, a_{24}, a_{34}\}) = 3$, in contradiction to our assumption. Thus $\{a_1 + \mathbb{R}a_{14}\} \subset E_{123}$, $\{a_2 + \mathbb{R}a_{24}\} \subset E_{123}$ and $\{a_3 + \mathbb{R}a_{34}\} \subset E_{123}$, and both a_4^{12} and a_4^{13} are in E_{123}.

We have $E_{123} \cap H_{12} \cap H_{23} \cap H_{31} = \{x_{123}\}$ for a single, uniquely determined point x_{123}, as the corresponding separating hyperplanes are not parallel (recall Lemma 2.26). Analogously, there are vectors $x_{124}, x_{134}, x_{234} \in E_{123}$ for the other hyperplane intersections. Due to the second condition to $\mathcal{H}$, $x_{123}, x_{124}, x_{134}, x_{234}$ are distinct. Figure 22 shows these points. The line segments correspond to the intersections of the hyperplanes with E_{123}. A, A', B, B' denote the Euclidean lengths of the line segments they are next to.

We now keep the points x_{123} and x_{134} of this construction fixed and change the right-hand side values γ_{12} and γ_{34} of hyperplanes H_{12} and H_{34} such that $x_{123} \in H_{34}$ and $x_{134} \in H_{12}$. It then is possible to change the right-hand side value of the hyperplane H_{24} such that all triples of hyperplanes again intersect in single points. This can be seen by some elementary geometric arguments, Figure 23 depicts the construction: We only have to check that $\frac{X}{Y} = \frac{A}{B'}$. This follows directly from the fact that $\frac{A}{A'} = \frac{B}{Y}$ and $\frac{B}{B'} = \frac{X}{A'}$.

By our induction hypothesis, we have a_1, a_2, a_3 satisfying the claim. Due to the orthogonality of a_{ij} to its hyperplane H_{ij}, it is easy to see that the triangle formed by a_1, a_2, a_3 is congruent to the triangle formed by x_{123}, x_{134} and the intersection of H_{23} and the translated H_{12} and H_{24} in the construction of Figure 23.

a_4^{12} is the unique intersection of $a_1 + \mathbb{R}a_{14}$ and $a_2 + \mathbb{R}a_{24}$, and a_4^{13} is the unique intersection of $a_1 + \mathbb{R}a_{14}$ and $a_3 + \mathbb{R}a_{34}$. With $a_{14} \perp H_{14}$, $a_{24} \perp H_{24}$ and $a_{34} \perp H_{34}$, and the fact that H_{14} and the translated H_{24} and H_{34} intersect in our construction, we know that $a_4^{12} = a_4^{13}$. Figure 24 shows the system of the a_i, which is congruent to the construction in Figure 23, just rotated by 90 degrees and scaled by a factor. Our arguments hold for arbitrary indices, and thus $a_k = a_k^{ij} = a_k^{12}$ for any $i \neq j;\ i, j \in \{1, \ldots, k\}$. This proves the claim. □

The restrictions we imposed on $\mathcal{H}$ in the above lemma are unproblematic for a cell decomposition with $C_i \subset \text{int}(P_i)$ for all $i \in \{1, \ldots, k\}$. Due to the pairwise strict separability of clusters it is easy to perturb the directions and positioning of the hyperplanes slightly to satisfy the prerequisites while still keeping the property $C_i \subset \text{int}(P_i)$ for all $i \in \{1, \ldots, k\}$.

We close this section by explaining what happens if we try the construction of Theorem 2.33 when we are not at a vertex of the gravity polytope. We then use our result for a first characterization of the vertices of the gravity polytope.

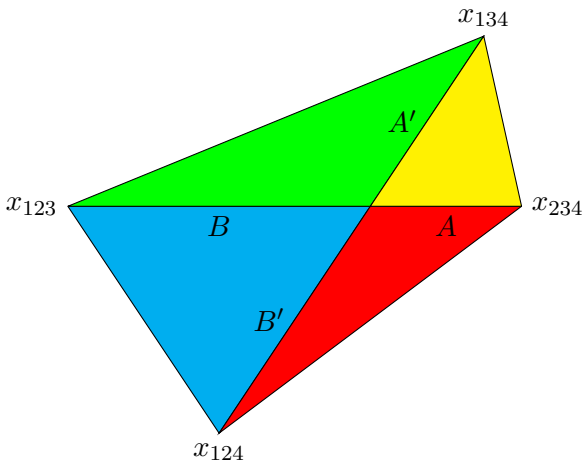


Figure 22: Hyperplane system of indices $\{1, 2, 3, 4\}$ with respective intersections $x_{123}, x_{124}, x_{134}, x_{234}$.

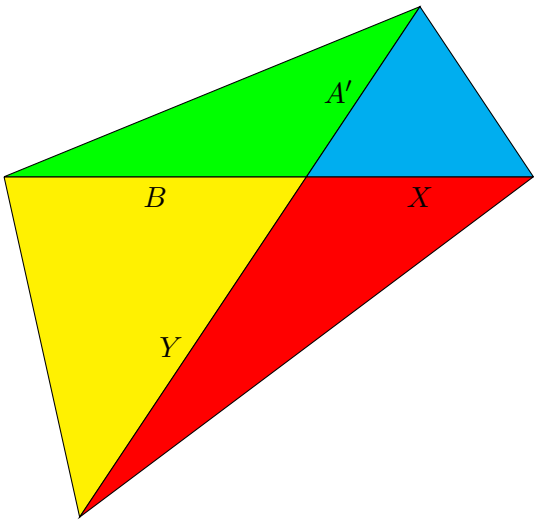


Figure 23: 'Swapping' the positions of H_{12} and H_{34} yields another system of intersecting hyperplanes.

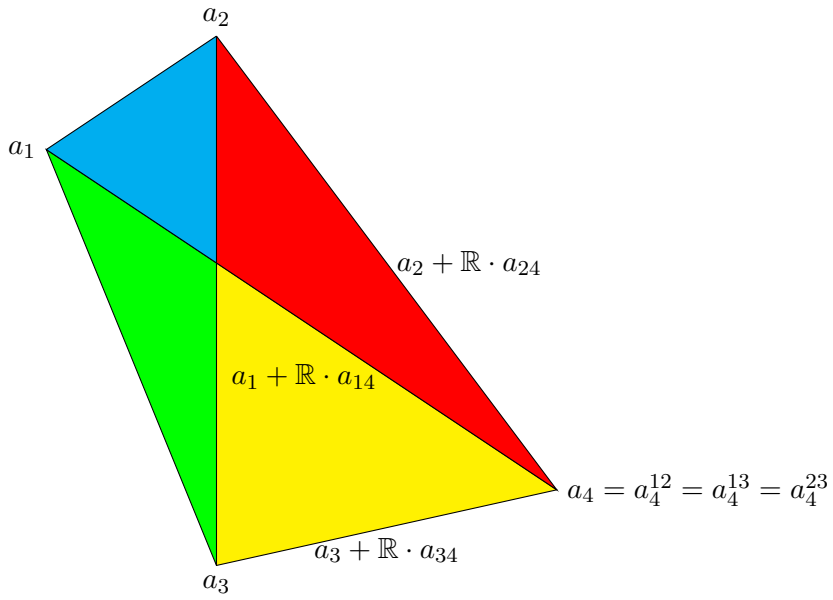


Figure 24: In the orthogonal system of the hyperplane vectors, $a_1 + \mathbb{R} \cdot a_{14}$, $a_2 + \mathbb{R} \cdot a_{24}$ and $a_3 + \mathbb{R} \cdot a_{34}$ intersect.

We show that applying the construction of a full a-induced cell decomposition using an a for which a gravity vector v is not uniquely optimal with respect to $a^T v$ in Q necessarily fails.

Lemma 2.42
Let C be a PSC and let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ such that there is a PSC C' with $a^T v(C') \geq a^T v(C)$, and let $\mathcal{H}$ be an a_κ-induced k-cell arrangement of $\mathbb{R}^d$.
Then $\mathcal{H}$ does not induce a full a_κ-induced k-cell decomposition with $C_i \subset int(P_i)$ for all $i \in \{1, \ldots, k\}$.

Proof. By Lemma 2.11, C and C' differ by a set of cyclical exchanges. A cyclical exchange corresponds to a cycle in the γ-clustering graph G. As C is not uniquely maximal in Q with respect to $a^T v(C)$, there is at least one cycle in G not lowering the objective function value. Among these, consider a cycle with a minimal number of edges, and the corresponding cyclical exchange CE.
Suppose CE is a cyclical exchange between only two clusters, e.g. of $x_i \in C_i$ and $x_j \in C_j$ to $x_i \in C'_j$ and $x_j \in C'_i$ for some $i \neq j;\ i, j \in \{1, \ldots, k\}$. Thus

$$\begin{aligned} \frac{a_i}{\kappa_i}^T x_i + \frac{a_j}{\kappa_j}^T x_j \leq \frac{a_i}{\kappa_i}^T x_j + \frac{a_j}{\kappa_j}^T x_i \quad &\Leftrightarrow \\ (\frac{a_j}{\kappa_j}^T - \frac{a_i}{\kappa_i}^T)x_i + (\frac{a_i}{\kappa_i}^T - \frac{a_j}{\kappa_j}^T)x_j \geq 0 \quad &\Leftrightarrow \\ (\frac{a_j}{\kappa_j}^T - \frac{a_i}{\kappa_i}^T)(x_i - x_j) \geq 0. \end{aligned}$$

This implies $C_i \notin \text{int}(P_i)$ and $C_j \notin \text{int}(P_j)$ for the cells P_i and P_j induced by $\mathcal{H}$. Now suppose $C_i \in \text{int}(P_i)$ for all $i \in \{1, \ldots, k\}$, so any pair of clusters is strictly separated by its hyperplane, thus $\max_{x \in C_i} a_{ij}^T x = \gamma_{ij}^* < \gamma_{ij} < -\gamma_{ji}^* = \min_{x \in C_j} a_{ij}^T x$ and CE has to involve at least 3 clusters. Without loss of generality, let CE be a cyclical exchange $CE := (x_1, \ldots, x_t)$, with $x_i \in C_i$ for all $i \in \{1, \ldots, t\}$ applied to C. We have $\gamma_{12}^* + \cdots + \gamma_{t1}^* \geq 0$, and by $\gamma_{ij} > \gamma_{ij}^*$, $\gamma_{12} + \cdots + \gamma_{t1} > 0$.
In the γ-clustering graph G, CE corresponds to a cycle $CY_1 := (v_1, \ldots, v_t, v_1)$ of at least 3 edges with positive total edge-weight. We satisfy all prerequisites of Lemma 2.34, thus there is a cycle CY_2 of only 3 edges in G with $w(CY_2) > 0$. Without loss of generality, let $CY_2 := (v_1, v_2, v_3, v_1)$, and then $w(CY_2) = \gamma_{12} + \gamma_{23} + \gamma_{31} > 0$. This proves the claim. □

Note that above lemma shows that an a_κ-induced k-cell arrangement for a clustering C not being uniquely optimal with respect to $a^T v(C)$ in Q either does not induce a full a_κ-induced cell decomposition, or that the cells of the induced full a_κ-induced cell decomposition do not strictly contain the clusters. Compare this to the example in Figure 18, where the positioning of the hyperplane between the red and the green cluster could be changed to have a full a_κ-induced cell decomposition.
Recall further that a PSC C not being (uniquely) optimal in Q with respect to $a^T v(C)$ does not imply that the clusters of C are not strictly separable. In contrast to this, we observe that we then get at most weakly separating hyperplanes if we construct a cell decomposition. Figure 25 shows an example.

Corollary 2.43
Let $v^* \in$ relbd(Q), let $C := (C_1, \ldots, C_k)$ be a PSC associated with v^* and let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $\frac{a_i}{\kappa_i} \neq \frac{a_j}{\kappa_j}$ for all $i \neq j$; $i, j \in \{1, \ldots, k\}$ and $a^T v^* \geq a^T v$ for any $v \in Q \backslash \{v^*\}$. Let further $a^T v^* = a^T v$ for some $v \in Q \backslash \{v^*\}$.
Then there is a full a_κ-induced cell decomposition $\mathcal{P} := (P_1, \ldots, P_k)$ with $C_i \subset P_i$ for $i \in \{1, \ldots, k\}$. Additionally, there is a cluster C_i such that $C_i \cap$ bd$(P_i) \neq \emptyset$.

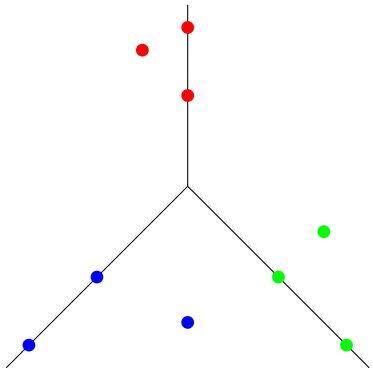


Figure 25: A clustering where the clusters are strictly separable with respect to the given hyperplane directions $a_j - a_i$, but for which there is no a-induced cell decomposition with the clusters being in the interior of the cells.

Proof. The first part is valid due to Corollary 2.37. Due to the existence of a $v \in Q \backslash \{v^*\}$ with $a^T v^* = a^T v$, there is a cyclical exchange $CE := (x_1, \ldots, x_t)$ with $x_i \in C_i$ for all $i \in \{1, \ldots, t\}$ such that $a^T v^* = a^T v(C')$, where C' is the clustering derived by the application of CE to C.
In the γ^*-clustering graph G, CE corresponds to a cycle of total edge-length 0. Thus $\gamma^*_{i(i+1)} = \gamma_{i(i+1)}$ for all $i \in \{1, \ldots, t\}$, and then $C_i \cap H_{i(i+1)} \neq \emptyset$ for all $i \in \{1, \ldots, t\}$, proving the claim. □

For any vertex v of Q, a vector a for which v is uniquely optimal in Q with respect to $a^T v$ yields a full a_κ-induced cell decomposition of $\mathbb{R}^d$ with $C_i \subset \text{int}(P_i)$ for all $i \in \{1, \ldots, k\}$. Full a_κ-induced cell decompositions with $C_i \subset \text{int}(P_i)$ can only be constructed at such a vertex v. This leads to our first characterization of the vertices of Q.

Theorem 2.44
Let $C := (C_1, \ldots, C_k)$ be a PSC. Then $v(C)$ is a vertex of Q if and only if there is a full a_κ-induced cell decomposition $\mathcal{P} := (P_1, \ldots, P_k)$ of $\mathbb{R}^d$ with $C_i \subset$ int(P_i) for all $i \in \{1, \ldots, k\}$.

Recalling Lemma 2.41 and the remark about perturbation following it, as well as Lemma 2.42, we obtain another characterization of the vertices of Q as a direct corollary of Theorem 2.44.

Theorem 2.45
Let $C := (C_1, \ldots, C_k)$ be a PSC. Then $v(C)$ is a vertex of Q if and only if there is a full cell decomposition $\mathcal{P} := (P_1, \ldots, P_k)$ of $\mathbb{R}^d$ with $C_i \subset$ int(P_i) for all $i \in \{1, \ldots, k\}$.

In the next section, we rephrase and interpret our ideas about full a-induced cell decompositions in the terminology of power diagrams and hereby exhibit a close relationship of the two fields.

2.5 Power Diagrams

In the following, we consider a special type of PSCs.

Definition 2.46 (Least-Squares Assignment (LSA))
Let $C := (C_1, \ldots, C_k)$ be a PSC of $X \subset \mathbb{R}^d$ and let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$. C is a **(strict) least-squares assignment (LSA) of X to a with respect to $\kappa_1, \ldots, \kappa_k$** *if and only if*

$$\sum_{i=1}^{k} \sum_{x \in C_i} \delta^2(x, a_i)$$

is (strictly) minimal for all PSCs, where $\delta(x, a_i)$ is the Euclidean distance of x and a_i in $\mathbb{R}^d$.

Informally, a LSA of X to a with respect to $\kappa_1, \ldots, \kappa_k$ associates a site $a_i \in \mathbb{R}^d$ with each cluster C_i and then assigns the $x \in X$ to the clusters such that the sum of the (squared) Euclidean distances of these points to the sites of their respective clusters is minimized. Additionally, each cluster C_i is assigned exactly κ_i points. When talking about LSAs, we only consider site vectors $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $a_i \neq a_j$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$. Using the same sites for two different clusters implies that an association of vectors with sites is not unique.
Aurenhammer, Hoffmann and Aronov established a strong connection of LSAs to a class of generalized Voronoi diagrams called power diagrams [AHA98]. As it will turn out, power diagrams are intimately related to full a-induced cell decompositions, allowing us to interpret Aurenhammer, Hoffmann and Aronov's results for LSAs in the context of the gravity polytope.
We start with a formal definition of power diagrams.

Definition 2.47 ((a, W)-Power Diagram)
*Let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $a_i \neq a_j$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$ and let $W := \{w_i : i \in \{1, \ldots, k\}\}$ be a set of k weights. Let further $\mathcal{P} := (P_1, \ldots, P_k)$ with $P_i := \{x \in \mathbb{R}^d : \delta^2(x, a_i) - w_i \leq \delta^2(x, a_j) - w_j$ for all $j \in \{1, \ldots, k\} \backslash \{i\}\}$ be the set of k cells for (a, W). The cells $\mathcal{P} := (P_1, \ldots, P_k)$ are an (a, W)-**power diagram** if int$(P_i) \neq \emptyset$ for all $i \in \{1, \ldots, k\}$.*

Trivially, an (a, W)-power diagram is a partition of $\mathbb{R}^d$ into convex, polyhedral cells. Each pair of cells is separated by a hyperplane consisting of all points

equally far away from both sites with respect to the weighted distance. If the context of a and W is clear, we use the shorter wording **power diagram** instead of (a, W)-power diagram.
Power diagrams can be used to induce clusterings.

Definition 2.48 (Clustering induced by an (a, W)-Power Diagram)
Let $\mathcal{P} := (P_1, \ldots, P_k)$ be an (a, W)-power diagram in $\mathbb{R}^d$ with $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ and let $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$. A clustering $C := (C_1, \ldots, C_k)$ satisfying $x_l \in C_i \Rightarrow x_l \in P_i$ is a **clustering induced by** (a, W)*, respectively* **by the** (a, W)**-power diagram**.

Note that the induction of a clustering from a power diagram is not uniquely determined if there is an $x_i \in X$ such that $x_i \in \mathrm{bd}(P_j)$ for some $j \in \{1, \ldots, k\}$. We now have the terminology to formally denote the known relation of LSAs and power diagrams.

Theorem 2.49 (Aurenhammer, Hoffmann and Aronov 92)

1. *Let $C := (C_1, \ldots, C_k\}$ be a clustering of $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ induced by an (a, W)-power diagram $\mathcal{P} := (P_1, \ldots, P_k)$ for $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$. Then C is a LSA of X to a with respect to the induced cluster sizes.*

2. *Let $\kappa_1, \ldots, \kappa_k \in \mathbb{N}$ with $\sum_{i=1}^{k} \kappa_i = n$, let $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ and let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$. Then there is a LSA $C := (C_1, \ldots, C_k\}$ of X to a with respect to $\kappa_1, \ldots, \kappa_k$, and there is a set of weights $W := \{w_i : i \in \{1, \ldots, k\}\}$ such that C is the clustering induced by the (a, W)-power diagram.*

It is easy to see the connection between LSAs and power diagrams denoted in Theorem 2.49 1.): Assigning points that lie in a cell of a power diagram to the cluster of the respective cell yields a LSA with respect to the induced cluster sizes.

Lemma 2.50
Let $C := (C_1, \ldots, C_k\}$ be a clustering of $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ induced by an (a, W)-power diagram $\mathcal{P} := (P_1, \ldots, P_k)$ for $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$. Then C is a LSA of X to a with respect to the induced cluster sizes.

Proof. With C being induced by (a, W), we know that $x_l \in C_i \Rightarrow x_l \in P_i$. Thus, whenever $x_l \in C_i$, we have $\delta^2(x_l, a_i) - w_i \leq \delta^2(x_l, a_j) - w_j$ for all

$j \in \{1,\dots,k\}\backslash\{i\}$. By this, $\sum_{i=1}^{k}\sum_{x\in C_i}(\delta^2(x,a_i)-w_i)$ is minimal for C among all clusterings of X. Considering that

$$\sum_{i=1}^{k}\sum_{x\in C_i}(\delta^2(x,\frac{a_i}{\kappa_i})-w_i)=\sum_{i=1}^{k}\sum_{x\in C_i}\delta^2(x,\frac{a_i}{\kappa_i})-\sum_{i=1}^{k}\sum_{x\in C_i}w_i$$

and noting that $\sum_{i=1}^{k}\sum_{x\in C_i}w_i=\sum_{i=1}^{k}\kappa_i w_i$ is constant for fixed $\kappa_1,\dots,\kappa_k$ shows that C is a LSA of X to a with respect to the induced cluster sizes. □

Aurenhammer, Hoffmann and Aronov proved the second claim constructively, by developing an algorithm that constructs a LSA and a corresponding inducing power diagram satisfying the prerequisites.
In the following, we present an alternative, analytical proof for the second claim of Theorem 2.49 in the context of the gravity polytope. By doing so, we are able to connect LSAs with gravity vectors on the relative boundary of the gravity polytope, and derive another characterization of its vertices.
We start by showing that full a-induced cell decompositions can be interpreted as power diagrams, and vice versa.

Lemma 2.51
Let $a := (a_1^T,\dots,a_k^T)^T \in \mathbb{R}^{d\cdot k}$ with $a_i \neq a_j$ for all $i \neq j;\ i,j \in \{1,\dots,k\}$.

1. *Any full a-induced cell decomposition of $\mathbb{R}^d$ is an (a,W)-power diagram for some set of weights W.*
2. *Let $W := \{w_i : i \in \{1,\dots,k\}\}$ be a set of weights. The (a,W)-power diagram then is a full a-induced cell decomposition of $\mathbb{R}^d$.*

Proof. 1.) Let $\mathcal{H}$ be a full a-induced cell decomposition for $a := (a_1^T,\dots,a_k^T)^T \in \mathbb{R}^{d\cdot k}$ with $a_i \neq a_j$ for all $i \neq j;\ i,j \in \{1,\dots,k\}$. We use the separation directions $a_{ij} := a_j - a_i$. Such separating hyperplane directions are induced by using the sites $a_1,\dots,a_k$ in a power diagram. In the following, we prove the claim by showing that there is a set of weights $W := \{w_1,\dots,w_k\}$, such that the cells induced by $\mathcal{H}$ are the cells of the (a,W)-power diagram.
Consider the set of hyperplanes $\{H_{1j} : j \in \{1,\dots,k\}\} \subset \mathcal{H}$. It is easy to derive weights $w_1,\dots,w_k$ such that $H_{1j} := \{x \in \mathbb{R}^d : \delta^2(x,a_1)-w_1 = \delta^2(x,a_j)-w_j\}$: E.g. we can choose $w_1 = 0$ (arbitrarily), and set $w_j = \delta^2(x,a_j)-\delta^2(x,a_1)$ for any $x \in H_{1j} \in \mathcal{H}$. It remains to show that $\delta^2(x,a_i)-w_i = \delta^2(x,a_j)-w_j$ for all $x \in H_{ij} \in \mathcal{H}$ for all $i \neq j;\ i,j \in \{1,\dots,k\}$.

Suppose a_{1i} and a_{ij} are not collinear, then there is an $x \in H_{1i} \cap H_{ij} \cap H_{j1}$. By our choice of W, we have $\delta^2(x, a_1) - w_1 = \delta^2(x, a_i) - w_i$ and $\delta^2(x, a_1) - w_1 = \delta^2(x, a_j) - w_j$, and, by this, $\delta^2(x, a_i) - w_i = \delta^2(x, a_j) - w_j$.
Any $x' \in H_{ij}$ can be written as $x' = x + x''$ with $(a_j - a_i)^T x'' = 0$. We now show that $\delta^2(x', a_i) - w_i = \delta^2(x', a_j) - w_j$:

$$\begin{aligned}
\delta^2(x', a_i) - w_i &= \delta^2(x', a_j) - w_j \\
\Leftrightarrow \delta^2(x + x'', a_i) - w_i &= \delta^2(x + x'', a_j) - w_j \\
\Leftrightarrow -2(x + x'')^T a_i + a_i^T a_i - w_i &= -2(x + x'')^T a_j + a_j^T a_j - w_j \\
\Leftrightarrow 2(x + x'')^T (a_j - a_i) + a_i^T a_i - w_i &= a_j^T a_j - w_j \\
\Leftrightarrow 2x^T (a_j - a_i) + a_i^T a_i - w_i &= a_j^T a_j - w_j \\
\Leftrightarrow -2x^T a_i + a_i^T a_i - w_i &= -2x^T a_j + a_j^T a_j - w_j \\
\Leftrightarrow x^T x - 2x^T a_i + a_i^T a_i - w_i &= x^T x - 2x^T a_j + a_j^T a_j - w_j \\
\Leftrightarrow \delta^2(x, a_i) - w_i &= \delta^2(x, a_j) - w_j
\end{aligned}$$

With the statement in the last line satisfied, we see the claim for this case.
Let now a_{1i} and a_{ij} be collinear. By this, H_{1i}, H_{ij} and H_{j1} are parallel. Let $x_{ij} \in H_{ij}$ and $x_{1i} \in H_{1i}$, $x_{j1} \in H_{j1}$. We show that

$$\delta^2(x_{ij}, a_i) - w_i = \delta^2(x_{ij}, a_j) - w_j \Leftrightarrow \delta^2(x_{ij}, a_i) - \delta^2(x_{ij}, a_j) = w_i - w_j$$

by leading the assumption $\delta^2(x_{ij}, a_i) - \delta^2(x_{ij}, a_j) \neq w_i - w_j$ to a contradiction. We have

$$\begin{aligned}
\delta^2(x_{ij}, a_i) - \delta^2(x_{ij}, a_j) &\neq w_i - w_j \\
\Leftrightarrow x_{ij}^T x_{ij} - 2a_i^T x_{ij} + a_i^T a_i - x_{ij}^T x_{ij} + 2a_j^T x_{ij} - a_j^T a_j &\neq w_i - w_j \\
\Leftrightarrow a_i^T a_i - a_j^T a_j - 2(a_i - a_j)^T x_{ij} &\neq w_i - w_j
\end{aligned}$$

and with $a_i - a_j = a_{ji}$ and $x_{ij} \in H_{ij}$, $(a_i - a_j)^T x_{ij} = a_{ji}^T x_{ij} = \gamma_{ji}$. With $\mathcal{H}$ being a full a-induced cell decomposition, we know that $\gamma_{1i} + \gamma_{ij} + \gamma_{j1} = 0$, and thus $\gamma_{ji} = \gamma_{1i} + \gamma_{j1}$. Together, we derive

$$\begin{aligned}
\delta^2(x_{ij}, a_i) - \delta^2(x_{ij}, a_j) &\neq w_i - w_j \\
\Leftrightarrow a_i^T a_i - a_j^T a_j - 2\gamma_{1i} - 2\gamma_{j1} &\neq w_i - w_j
\end{aligned}$$

On the other hand $w_i - w_j = (w_i - w_1) + (w_1 - w_j)$, and with $x_{1i} \in H_{1i}$ and $x_{j1} \in H_{j1}$, we see that

$$\begin{aligned} w_i - w_j &= (w_i - w_1) + (w_1 - w_j) \\ &= \delta^2(x_{1i}, a_i) - \delta^2(x_{1i}, a_1) + \delta^2(x_{j1}, a_1) - \delta^2(x_{j1}, a_j) \\ &= x_{1i}^T x_{1i} - 2a_i^T x_{1i} + a_i^T a_i - x_{1i}^T x_{1i} + 2a_1^T x_{1i} - a_1^T a_1 \\ &+ x_{j1}^T x_{j1} - 2a_1^T x_{j1} + a_1^T a_1 - x_{j1}^T x_{j1} + a_j^T x_{j1} - a_j^T a_j \\ &= a_i^T a_i - a_j^T a_j - 2(a_i - a_1)^T x_{1i} - 2(a_1 - a_j)^T x_{j1} \\ &= a_i^T a_i - a_j^T a_j - 2\gamma_{1i} - 2\gamma_{j1} \end{aligned}$$

leading to a contradiction. This proves the first claim.

2.) Let, for any pair of indices $i \neq j \in \{1, \ldots, k\}^2$, $H_{ij} := \{x \in \mathbb{R}^d : \delta^2(x, a_i) - w_i = \delta^2(x, a_j) - w_j\}$. Clearly, H_{ij} is a hyperplane in $\mathbb{R}^d$, as $a_i \neq a_j$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$, and as such can be written in the form $H_{ij} := \{x \in \mathbb{R}^d : a_{ij}^T x = \gamma_{ij}\}$ for $a_{ij} := a_j - a_i$ and some $\gamma_{ij} \in \mathbb{R}$. By the definition of full a-induced cell decompositions, it remains to show that $\gamma_{i_1 i_2} + \gamma_{i_2 i_3} + \gamma_{i_3 i_1} = 0$ for any $\{i_1, i_2, i_3\} \subset \{1, \ldots, k\}$. For the sake of a simple notation, we assume $\{i_1, i_2, i_3\} = \{1, 2, 3\}$ without loss of generality.
With the H_{ij} being a-induced, if a_{12} and a_{23} are not collinear, neither are a_{12} and a_{31}, nor are a_{23} and a_{31}, recall Corollary 2.25.
Then there is an $x \in \mathbb{R}^d$ with $\delta^2(x, a_1) - w_1 = \delta^2(x, a_2) - w_2 = \delta^2(x, a_3) - w_3$, and this implies $x \in H_{12} \cap H_{23} \cap H_{31}$. We then obtain $0 = 0^T x = (a_{12} + a_{23} + a_{31})^T x = \gamma_{12} + \gamma_{23} + \gamma_{31}$.
Finally, suppose a_{12}, a_{23} and a_{31} are pairwise collinear. Let $x_{31} \in H_{31}$, i.e.

$$\delta^2(x_{31}, a_3) - w_3 = \delta^2(x_{31}, a_1) - w_1 \Leftrightarrow \delta^2(x_{31}, a_3) - \delta^2(x_{31}, a_1) = w_3 - w_1$$

We derive

$$\begin{aligned} w_3 - w_1 &= \delta^2(x_{31}, a_3) - \delta^2(x_{31}, a_1) \\ &= x_{31}^T x_{31} - 2x_{31}^T a_3 + a_3^T a_3 - x_{31}^T x_{31} + 2x_{31}^T a_1 - a_1^T a_1 \\ &= a_3^T a_3 - a_1^T a_1 + 2x_{31}^T (a_1 - a_3) \\ &= a_3^T a_3 - a_1^T a_1 + 2a_{31}^T x_{31} \\ &= a_3^T a_3 - a_1^T a_1 + 2\gamma_{31} \end{aligned}$$

On the other hand, for $x_{23} \in H_{23}$ and $x_{12} \in H_{12}$ we have

$$\begin{aligned} w_3 - w_1 &= (w_3 - w_2) + (w_2 - w_1) \\ &= \delta^2(x_{23}, a_3) - \delta^2(x_{23}, a_2) + \delta^2(x_{12}, a_2) - \delta^2(x_{12}, a_1) \\ &= x_{23}^T x_{23} - 2a_3^T x_{23} + a_3^T a_3 - x_{23}^T x_{23} + 2a_2^T x_{23} - a_2^T a_2 \\ &+ x_{12}^T x_{12} - 2a_2^T x_{12} + a_2^T a_2 - x_{12}^T x_{12} + a_1^T x_{12} - a_1^T a_1 \\ &= a_3^T a_3 - a_1^T a_1 - 2(a_3 - a_2)^T x_{23} - 2(a_2 - a_1)^T x_{12} \\ &= a_3^T a_3 - a_1^T a_1 - 2\gamma_{23} - 2\gamma_{12} \end{aligned}$$

and together

$$\begin{aligned} a_3^T a_3 - a_1^T a_1 + 2\gamma_{31} &= a_3^T a_3 - a_1^T a_1 - 2\gamma_{23} - 2\gamma_{12} \\ \Leftrightarrow 2\gamma_{31} &= -2\gamma_{23} - 2\gamma_{12} \\ \Leftrightarrow \gamma_{12} + \gamma_{23} + \gamma_{31} &= 0 \end{aligned}$$

This proves the claim. □

Lemma 2.51 shows that power diagrams and full a-induced cell decompositions are different representations of the same topological kind of partition of $\mathbb{R}^d$. This yields another characterization of power diagrams in our terminology.
Recalling Theorem 2.44, we directly obtain a second characterization of the vertices of the gravity polytope from Lemma 2.51.

Theorem 2.52
Let $C := (C_1, \ldots, C_k)$ be a PSC. Then $v(C)$ is a vertex of Q if and only if there is a power diagram $\mathcal{P} := (P_1, \ldots, P_k)$ with $C_i \subset int(P_i)$ for all $i \in \{1, \ldots, k\}$.

Lemma 2.51 further allows us to establish a direct correlation between LSAs with respect to $\kappa_1, \ldots, \kappa_k$ and gravity vectors on the relative boundary of the gravity polytope.

Theorem 2.53
Let $C := (C_1, \ldots, C_k)$ be a PSC, and let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $\frac{a_i}{\kappa_i} \neq \frac{a_j}{\kappa_j}$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$ and $a_\kappa := (\frac{a_1}{\kappa_1}^T, \ldots, \frac{a_k}{\kappa_k}^T)^T$. Then the following two statements are equivalent:

1. *$v(C) \in relbd(Q)$ and $a \in N(v(C))$*
2. *C is a LSA of X to a_κ with respect to $\kappa_1, \ldots, \kappa_k$*

Proof. 2) $\Rightarrow$ 1): Let $C := (C_1, \ldots, C_k)$ be a LSA of X to a_κ with respect to $\kappa_1, \ldots, \kappa_k$ for some a. The LSA is associated with a minimal value of

$$\sum_{i=1}^{k} \sum_{x \in C_i} \delta^2(x, \frac{a_i}{\kappa_i}) = \sum_{i=1}^{k} \sum_{x \in C_i} (x^T x - 2 \cdot \frac{a_i}{\kappa_i}^T x + \frac{a_i}{\kappa_i}^T \frac{a_i}{\kappa_i}).$$

With $\sum_{i=1}^{k} \sum_{x \in C_i} x^T x = \sum_{x \in X} x^T x$ and $\sum_{i=1}^{k} \sum_{x \in C_i} \frac{a_i}{\kappa_i}^T \frac{a_i}{\kappa_i} = \sum_{i=1}^{k} \frac{a_i^T a_i}{\kappa_i}$ constant for fixed X, a and cluster sizes $\kappa_1, \ldots, \kappa_k$, we look for a PSC belonging to

$$\arg\min_{\text{PSC } C} \sum_{i=1}^{k} \sum_{x \in C_i} -2 \cdot \frac{a_i}{\kappa_i}^T x = \arg\max_{\text{PSC } C} 2 \cdot \sum_{i=1}^{k} \sum_{x \in C_i} \frac{a_i}{\kappa_i}^T x = \arg\max_{\text{PSC } C} 2 \cdot \sum_{i=1}^{k} a_i^T c_i,$$

where c_i denotes the center of gravity of cluster C_i. This corresponds to a PSC $C = \arg\max_{\text{PSC } C} \sum_{i=1}^{k} a_i^T c_i = \arg\max_{\text{PSC } C} a^T v(C)$, implying $v(C) \in \text{relbd}(Q)$ and $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k} \in N(v(C))$.

1) $\Rightarrow$ 2): By Theorem 2.33 and Corollary 2.37, we can construct a full a_κ-induced cell decomposition $\mathcal{P} := (P_1, \ldots, P_k)$ of $\mathbb{R}^d$ with $C_i \subset P_i$. By Lemma 2.51, there is a set of weights $W := \{w_1, \ldots, w_k\}$, such that $\mathcal{P}$ is the (a_κ, W)-power diagram. With $C_i \subset P_i$, by Lemma 2.50, C is a LSA of X to a_κ with respect to $\kappa_1, \ldots, \kappa_k$. This proves the claim. □

Informally, Theorem 2.53 tells us that the LSAs with respect to $\kappa_1, \ldots, \kappa_k$ have gravity vectors on the relative boundary of the gravity polytope. The second claim of Theorem 2.49 follows (almost) immediately.

Lemma 2.54
Let $\kappa_1, \ldots, \kappa_k \in \mathbb{N}$ with $\sum_{i=1}^{k} \kappa_i = n$, let $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ and let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$. Then there is a LSA $C := (C_1, \ldots, C_k\}$ of X to a with respect to $\kappa_1, \ldots, \kappa_k$ and there is a set of weights $W := \{w_i : i \in \{1, \ldots, k\}\}$ such that C is the clustering induced by the (a, W)-power diagram.

Proof. Let $\kappa_1, \ldots, \kappa_k \in \mathbb{N}$ and let $a' := (\kappa_1 \cdot a_1^T, \ldots, \kappa_k \cdot a_k^T)^T \in \mathbb{R}^{d \cdot k}$. With Q being a polytope, there always is a PSC $C := (C_1, \ldots, C_k)$ with $a'^T v(C) \geq a'^T v'$ for all $v' \in Q \backslash \{v(C)\}$. By this, $v(C) \in \text{relbd}(Q)$ and $a' \in N(v(C))$, and we can apply Theorem 2.53 to see that C is a LSA of X to $a'_\kappa = a$ with respect to $\kappa_1, \ldots, \kappa_k$.
The rest of the claim follows analogously to the proof of 1) $\Rightarrow$ 2) for Theorem 2.53. □

We obtain another characterization of the vertices of the gravity polytope from the above results.

Theorem 2.55

Let $C := (C_1, \dots, C_k)$ be a PSC. Then $v(C)$ is a vertex of Q if and only if C is a strict LSA to some $a_\kappa := (\frac{a_1}{\kappa_1}^T, \dots, \frac{a_k}{\kappa_k}^T)^T \in \mathbb{R}^{d\cdot k}$ with respect to $\kappa_1, \dots, \kappa_k$.

Proof. Recalling Theorem 2.53, we know that the clustering $C := (C_1, \dots, C_k)$ associated with a vertex is a LSA to some $a_\kappa := (\frac{a_1}{\kappa_1}^T, \dots, \frac{a_k}{\kappa_k}^T)^T \in \mathbb{R}^{d\cdot k}$ with respect to $\kappa_1, \dots, \kappa_k$. We also see this from Lemma 2.50 and Theorem 2.52.
By Theorem 2.52, C is induced by an (a_κ, W)-power diagram such that $C_i \subset \text{int}(P_i)$ for all $i \in \{1, \dots, k\}$. For any $x \in X$ associated with a cluster C_i, this implies $\delta^2(x, \frac{a_i}{\kappa_i}) - w_i < \delta^2(x, \frac{a_j}{\kappa_j}) - w_j$ for any $j \in \{1, \dots, k\}\backslash\{i\}$. By

$$\sum_{i=1}^{k} \sum_{x \in C_i} (\delta^2(x, \frac{a_i}{\kappa_i}) - w_i) = \sum_{i=1}^{k} \sum_{x \in C_i} \delta^2(x, \frac{a_i}{\kappa_i}) - \sum_{i=1}^{k} \sum_{x \in C_i} w_i$$

and $\sum\limits_{i=1}^{k} \sum\limits_{x \in C_i} w_i$ being constant, this implies that $\sum\limits_{i=1}^{k} \sum\limits_{x \in C_i} \delta^2(x, \frac{a_i}{\kappa_i})$ is strictly minimal for C among all PSCs.
Conversely, let $\sum\limits_{i=1}^{k} \sum\limits_{x \in C_i} \delta^2(x, \frac{a_i}{\kappa_i})$ be strictly minimal for C among all PSCs. Following the proof of Theorem 2.53 $2) \Rightarrow 1)$, we see that C is the PSC uniquely maximal in Q with respect to $a^T v(C)$, where $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d\cdot k}$. This is only possible at a vertex of Q, for $a \in N(v(C))$, which proves the claim. □

We close this section by taking a closer look at the connection between power diagrams, LSAs and vertices of the gravity polytope. Our geometric approach directly leads us to some simple statements for the sites of a power diagram, respectively of strict LSAs.

Lemma 2.56

Let $\kappa_1, \dots, \kappa_k \in \mathbb{N}^k$ with $\sum\limits_{i=1}^{k} \kappa_i = n$, let $X := \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ and let $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d\cdot k}$ with $a_i \neq a_j$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$.
Let further $C := (C_1, \dots, C_k)$ be a PSC induced by an (a, W)-power diagram for some set of weights $W := \{w_i : i \in \{1, \dots, k\}\}$. If $C_i \subset$ int(P_i), then C is the only PSC that can be induced by an (a, W')-power diagram using the site vector a and any set of weights W'. In this case, further

1. *the site vectors $a := (a_1^T, \dots, a_k^T)^T$ and $a_b := ((a_1 + b)^T, \dots, (a_k + b)^T)^T$, for any $b := (b_1, \dots, b_d)^T \in \mathbb{R}^d$, yield the same PSC C.*

2. *the site vectors* $a := (a_1^T, \ldots, a_k^T)^T$ *and* $a_t := (t \cdot a_1^T, \ldots, t \cdot a_k^T)^T$*, for any* $0 < t \in \mathbb{R}$*, yield the same PSC* C*.*

Proof. By Theorem 2.44 and Lemma 2.51, we know that $C_i \subset \text{int}(P_i)$ for all $i \in \{1, \ldots, k\}$ can only be satisfied if $v^* := v(C)$ is a vertex of Q and $a' := (\kappa_1 \cdot a_1^T, \ldots, \kappa_k \cdot a_k^T)^T \in \mathbb{R}^{d \cdot k} \in \text{int}(N(v^*))$. Then $a' \notin N(v)$ for any $v \in Q \backslash \{v^*\}$, and Lemma 2.42 and Lemma 2.51 prove the uniqueness of C as being the only PSC which can be induced by an (a, W')-power diagram for any set of weights W'.

1.) With $a' = (\kappa_1 \cdot a_1^T, \ldots, \kappa_k \cdot a_k^T)^T \in N(v^*)$ for the vertex v^* associated with PSC C, we have

$$a'_b := (\kappa_1 \cdot (a_1 + b)^T, \ldots, \kappa_k \cdot (a_k + b)^T)^T \in N(v^*)$$

as well, due to

$$\begin{aligned}
a_b'^T v &= (\kappa_1 \cdot (a_1 + b)^T, \ldots, \kappa_k \cdot (a_k + b)^T)v \\
&= ((\kappa_1 \cdot a_1^T, \ldots, \kappa_k \cdot a_k^T) + (\kappa_1 \cdot b^T, \ldots, \kappa_k \cdot b^T))v \\
&= (a'^T + (\kappa_1 \cdot b^T, \ldots, \kappa_k \cdot b^T))v \\
&= a'^T v + (\kappa_1 b_1, \ldots, \kappa_1 b_d, \kappa_2 b_1, \ldots, \kappa_2 b_d, \ldots, \kappa_k b_1, \ldots, \kappa_k b_d)v \\
&= a'^T v + \sum_{i=1}^{k} (\kappa_i b_1, \ldots, \kappa_i b_d)c_i \\
&= a'^T v + \sum_{l=1}^{n} (b_1, \ldots, b_d)x_l \\
&= a'^T v + (b_1, \ldots, b_d)(\sum_{l=1}^{n} x_l),
\end{aligned}$$

as $(b_1, \ldots, b_d)(\sum_{l=1}^{n} x_l)$ is constant for fixed b.

2.) We have $a_t = t \cdot a$, and the claim follows from $a' \in N(v^*) \Leftrightarrow t \cdot a' \in N(v^*)$ for any vertex $v^* \in Q$. □

Informally, Lemma 2.56 shows that if the clusters of a PSC lie strictly in the interior of the cells of a power diagram, then there is no other PSC that can be induced by a power diagram using the same sites.

Additionally, the construction of the power diagram is invariant under an identical translation of all sites in $\mathbb{R}^d$ and a scaling of all sites by the same scalar. Note that 1.) implies that one of the sites can be chosen as $0 \in \mathbb{R}^d$, if we set $b = -a_i$ for some $i \in \{1, \dots, k\}$. This invariance can be transferred directly to LSAs by Lemma 2.50 and Lemma 2.54. See also [AHA98].

Corollary 2.57
Let $\kappa_1, \dots, \kappa_k \in \mathbb{N}$ with $\sum_{i=1}^{k} \kappa_i = n$, let $X := \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ and let $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ be a vector of sites with $a_i \neq a_j$ for all $i \neq j$; $i, j \in \{1, \dots, k\}$. Let further $C := (C_1, \dots, C_k)$ be a (strict) LSA of X to a with respect to $\kappa_1, \dots, \kappa_k$. Then

1. *C is a (strict) LSA of X to $a_b := ((a_1 + b)^T, \dots, (a_k + b)^T)^T$ with respect to $\kappa_1, \dots, \kappa_k$, for any $b := (b_1, \dots, b_d)^T \in \mathbb{R}^d$.*
2. *C is a (strict) LSA of X to $a_t := (t \cdot a_1^T, \dots, t \cdot a_k^T)^T$ with respect to $\kappa_1, \dots, \kappa_k$, for any $0 < t \in \mathbb{R}$.*

2.6 Summary and Outlook

In this chapter, we investigated the gravity polytope defined by the gravity vectors of prescribed-shape clusterings of an underlying point set, with our focus on a characterization of the vertices of this polytope. We started by transferring a result by Barners, Hoffman and Rothblum [BHR92] to our polytope, showing that the clusters of clusterings belonging to vertices of the gravity polytope are pairwise strictly linearly separable. On the other hand, there also are clusterings associated with vectors in the interior of the polytope satisfying this property.
Next, we turned to a result by Brieden and Gritzmann [BG09], showing that the clusterings of vertices allow cell decompositions of the space of the point set such that each cluster lies strictly in its own cell. Yet even this property does not suffice for a direct characterization of the vertices. We observed that the construction used for this cell decomposition satisfies some special properties, leading us to the notion of full a-induced cell decompositions, finally yielding the desired characterization of the vertices. The most important feature of full cell decompositions is that any subset of cells still yields a cell decomposition of the underlying space. We also showed that asking for this feature alone, under mild restrictions, directly leads to a full cell decomposition having a representation as a full a-induced cell decomposition, yielding another characterization of the vertices .

Finally, we showed that full a-induced cell decompositions correspond to power diagrams, a generalized class of Voronoi diagrams, and vice versa. Using this fact, we gave an alternative proof of a theorem of Aurenhammer, Hoffmann and Aronov [AHA98] in the context of the gravity polytope and hereby derived two additional characterizations of the vertices. As a byproduct, we obtained an alternative characterization of power diagrams, and observed that some properties of power diagrams and LSAs follow naturally from the point of view of our geometric approach.

There still are plenty of open questions concerning the gravity polytope and related polytopes. Of course not only the vertex structure of the polytope is of interest, but so are its edge-structure and its general facial structure. In Chapter 4, we will shortly turn to the edge-structure of the polytope.

The last section of this chapter also yields an interesting polytope to study, the polytope of gravity vectors of clusterings induced by power diagrams. It is the convex hull of gravity polytopes for all feasible combinations of cluster sizes and thus differs from both our gravity polytope and the unrestricted polytope considered by Barnes, Hoffman and Rothblum [BHR92]. Certainly, many properties of these two related polytopes can be transferred to this one.

In the next chapter, we turn to an application of the theoretical results of this chapter in the field of data classification.

3 Data Classification by Cell Decompositions

In the following, we use the characterization of the vertices of the gravity polytope described in the last chapter for a combinatorial optimization approach to data classification. As we see in the first section, vertex clusterings satisfy several nice properties when used this way.
We refer the reader to the **Notation and Symbols** appendix for a list of the most important symbols introduced in Chapter 2 and in this chapter.

3.1 Vertex Clusterings

In this section, we give some reasons as to why clusterings associated with vertices of a gravity polytope are especially useful for data classification purposes. First and foremost, such clusterings correspond to strict LSAs to some sites with respect to the given cluster sizes, by Theorem 2.55. Looking for LSAs of data vectors to clusters is an intuitively useful approach in many applications and thus a widely accepted way of clustering geometric data, whenever the Euclidean distance of data vectors is a suitable similarity measure [EE04; Mir05].
As we have seen, LSAs with respect to some cluster sizes satisfy a number of additional useful properties. Their clusters not only allow linear separation, but they also allow full a-induced cell decompositions (Theorem 2.44), or equivalently can be induced by power diagrams (Theorem 2.52, [AHA98]).
We hence obtain a geometric interpretation of the separating hyperplanes as 'differentiation criteria' of the clusters. The fact that, for any subsystem of clusters, we have a cell decomposition of space further emphasizes this interpretation.
If the structure of the underlying data does not adhere to linear separability of clusters, the use of **kernels** may amend the situation, by a transformation of the data such that linear separation of clusters is a valid approach. Kernel functions and the choice of the 'right' kernels for data sets are an active field of research in machine learning. See [SS02; Bor07] for surveys of the topic.
We begin by analyzing one of the most influential and classical methods for clustering without size restrictions, the **k-means algorithm** [Mac67], in the context of the gravity polytope. With its basic idea being to calculate LSAs to a set of sites in each iteration step, we only need a minimal adaption of the algorithm to relate the clusterings of each iteration step to vertices of the gravity polytope. We start by giving an informal description of the algorithm in pseudo-code (Algorithm 1).

Input : d, k, n, $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$
Output: Clustering $C := (C_1, \ldots, C_k)$, cluster sizes $\kappa_1, \ldots, \kappa_k$

Choose k cluster sites $a_1, \ldots, a_k$ (at random) in $\mathbb{R}^d$;
$(*)$ Assign each $x \in X$ to a cluster C_i for which $\delta^2(x, a_i)$ is minimal;
for $1 \leq i \leq k$ **do**
  $\kappa_i := |C_i|$;
  if $\kappa_i > 0$ **then**
    $a_i := \frac{1}{\kappa_i} \sum_{x \in C_i} x$;
  end
  else
    Choose a_i at random;
  end
end
if *the assignment of an* $x \in X$ *changed during the last iteration* **then**
  Go to step $(*)$;
end
return $C := (C_1, \ldots, C_k)$ and $\kappa_1, \ldots, \kappa_k$;

Algorithm 1: k-Means Algorithm

This basic version of the algorithm partitions a given point set $X \subset \mathbb{R}^d$ into k (not necessarily non-empty) clusters $C_1, \ldots, C_k$. In each iteration, a current set of cluster sites $a_1, \ldots, a_k$ is used to assign all $x \in X$ to a cluster C_i with $\delta^2(x, c_i) \leq \delta^2(x, c_j)$ for all $j \in \{1, \ldots, k\} \backslash \{i\}$. By this, $\sum_{i=1}^{k} \sum_{x \in C_i} \delta^2(x, a_i)$ is minimal for the clustering $C := (C_1, \ldots, C_k)$. This shows that C is a LSA of X to $a := (a_1^T, \ldots, a_k^T)^T$ for arbitrary cluster sizes, and thus certainly also with respect to the induced cluster sizes. By Theorem 2.53, we know that its gravity vector is on the relative boundary of the gravity polytope of the respective cluster sizes.

Lemma 3.1
Let $C := (C_1, \ldots, C_k)$ be a clustering of X derived after step $(*)$ of any iteration of Algorithm 1. Let k' be the number of non-empty clusters, let $C' := (C_1, \ldots, C_{k'})$ be derived from C by dropping all empty clusters (and reindexing), and let $\kappa_i := |C_i|$ for all $i \in \{1, \ldots, k'\}$. Then $v(C') \in relbd(Q(X, k'; \kappa_1, \ldots, \kappa_{k'}))$.

Informally, in each step, the k-means algorithm decomposes $\mathbb{R}^d$ into a Voronoi diagram with respect to $a_1, \ldots, a_k$ and induces a clustering C from it. With power diagrams being a direct generalization of Voronoi diagrams, this is just a simple special case of the clusterings induced by power diagrams described in the last chapter.

We now turn to the question of how to adapt Algorithm 1 to be able to guarantee that the resulting clusterings are vertices of their corresponding gravity polytope.

Input : d, k, n, $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$
Output: Clustering $C := (C_1, \ldots, C_k)$, cluster sizes $\kappa_1, \ldots, \kappa_k$

Choose k cluster sites $a_1, \ldots, a_k$ (at random) in $\mathbb{R}^d$;
$(*)$ Assign each $x \in X$ to the cluster C_i with minimal index i for which $\delta^2(x, a_i)$ is minimal;
for $1 \leq i \leq k$ **do**
  $\kappa_i := |C_i|$;
  if $\kappa_i > 0$ **then**
    $a_i := \frac{1}{\kappa_i} \sum_{x \in C_i} x$;
  end
  else
    Choose a random $x \in X$ (with $x \neq a_j$ for all $j \neq i$) and set $a_i := x$;
  end
end
if *the assignment of an $x \in X$ or an a_i changed during the last iteration* **then**
  Go to step $(*)$;
end
return $C := (C_1, \ldots, C_k)$ and $\kappa_1, \ldots, \kappa_k$;

Algorithm 2: Deterministic k-Means Algorithm

First, we always assign $x \in X$ to the cluster C_i of minimal index i with a closest site. By doing so, each pair of clusters C_i, C_j can be separated strictly in direction $a_j - a_i$ after each iteration of the algorithm.
If desired, we can also force the algorithm to run until all k clusters are non-empty by also checking whether any a_i changed during the last iteration. If so, either the assignment of an $x \in X$ changed, or a cluster was empty, so that its center was chosen differently at random. Instead of this random choice, we may set $a_i = x$ for some $x \in X$ with $x \neq c_j$ for all $j \in \{1, \ldots, k\}$. In the next step of the algorithm, x will then be assigned to C_i. Still, we impose no other restrictions on the cluster sizes (except for them summing up to n). The pseudo-code of Algorithm 2 adds the ideas mentioned. We obtain the following theorem.

Theorem 3.2
Let $C := (C_1, \ldots, C_k)$ be a clustering of X returned by Algorithm 2, and let $\kappa_i := |C_i|$ for all $i \in \{1, \ldots, k\}$. Then $v(C)$ is a vertex of $Q(X, k; \kappa_1, \ldots, \kappa_k)$.

Proof. With arbitrary starting sites, our k-means algorithm derives a clustering $C := (C_1, \ldots, C_k) \in Q(X, k; \kappa_1, \ldots, \kappa_k)$ of X with $\kappa_i \geq 1$ for all $1 \leq i \leq k$. It is a

PSC for k and the induced $\kappa_1, \ldots, \kappa_k$. We prove the claim by showing that C is a strict LSA to $a := (a_1^T, \ldots, a_k^T)$ with respect to $\kappa_1, \ldots, \kappa_k$, where $a_1, \ldots, a_k$ are the final cluster centers.
By the deterministic assignment used, we know that

$$x_l \in C_i \Rightarrow \delta^2(x_l, a_i) \leq \delta^2(x_l, a_j) \text{ for all } j \in \{1, \ldots, k\} \backslash \{i\}$$

and additionally that for

$$x_l \in C_k \Rightarrow \delta^2(x_l, a_k) < \delta^2(x_l, a_j) \text{ for all } j \in \{1, \ldots, k-1\}.$$

With C_k non-empty we obtain

$$\sum_{i=1}^{k} \sum_{x \in C_i} \delta^2(x, a_i) < \sum_{i=1}^{k} \sum_{x \in C_i'} \delta^2(x, a_i)$$

for any clustering $C' := (C_1', \ldots, C_k') \neq C$. This proves the claim. □

Note that the cluster sites of the last iteration step are the centers of gravity of the respective clusters after the termination of Algorithm 2. Note further that we do not need exactly k non-empty clusters to see that we have a strict LSA. Using $x_l \in C_{k'}$ for the highest index k' for which $C_{k'}$ is non-empty in the above proof yields the claim analogously. Due to this, we get the following corollary.

Corollary 3.3
Let $C := (C_1, \ldots, C_k)$ be a clustering of X derived after step $(*)$ of any iteration of Algorithm 2. Let k' be the number of non-empty clusters, let $C' := (C_1, \ldots, C_{k'})$ be derived from C by dropping all empty clusters (and reindexing), and let $\kappa_i := |C_i|$ for all $i \in \{1, \ldots, k'\}$. Then $v(C')$ is a vertex of $Q(X, k'; \kappa_1, \ldots, \kappa_{k'})$.

Of course, not all vertices of a gravity polytope Q can be derived from an application of Algorithm 2. The most obvious examples are clusterings where a pair of clusters C_i, C_j cannot be separated in direction $c_j - c_i$ for some $i \neq j;\ i, j \in \{1, \ldots, k\}$.
As we have seen, our vertex clusterings are strict LSAs with respect to some cluster sizes, and they may be the clusterings returned by an application of the k-means algorithm. It is known that such clusterings are 'optimal' clusterings in several geometric ways due to being induced by Voronoi diagrams, or power diagrams respectively. See [AK99] for a short survey.

The full a_κ-induced cell decompositions we construct for vertex clusterings as described in Theorem 2.33 are in a certain sense (Lemma 2.41 and the short remark after it) the only way to create a cell decomposition such that all subsets of indices still create cell decompositions. Later on, we will describe some advantages of having a full cell decomposition, instead of only a cell decomposition, for some stability measures for the classification and prediction algorithms we propose.
We now turn to a special type of gravity polytopes, where the cluster sizes satisfy $\kappa_1 = \kappa_2 = \cdots = \kappa_k$. If we look for a clustering into clusters of equal size and consider a vertex of the corresponding gravity polytope, we get an especially nice geometric property.

Lemma 3.4
Let $\kappa \in \mathbb{N}$ and $\kappa_i = \kappa$ for all $i \in \{1, \ldots, k\}$, let $C := (C_1, ..., C_k)$ be a clustering with gravity vector $v := v(C) = (c_1^T, \ldots, c_k^T)^T$ and let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $a^T v \geq a^T v'$ for any $v' \in Q \backslash \{v\}$.
Then v is optimal in Q for the objective function

$$\max_{PSC\ C} \sum_{i=1}^{k} \sum_{j=1}^{k} (a_i - a_j)^T (c_i - c_j)$$

Proof. v is optimal in Q for

$$\max a^T v = \max \sum_{i=1}^{k} a_i^T c_i$$

We have

$$\sum_{i=1}^{k} \sum_{j=1}^{k} (a_i - a_j)^T (c_i - c_j) =$$

$$\sum_{i=1}^{k} \sum_{j=1}^{k} (a_i^T c_i + a_j^T c_j - (a_j^T c_i + a_i^T c_j)) =$$

$$\sum_{i=1}^{k} (2k \cdot a_i^T c_i) - \sum_{i=1}^{k} \sum_{j=1}^{k} 2 \cdot a_j^T c_i =$$

$$2ka^T v - 2\sum_{i=1}^{k}\sum_{j=1}^{k} a_j^T c_i =$$
$$2ka^T v - 2\sum_{i=1}^{k} c_i^T(\sum_{j=1}^{k} a_j)$$

As k and a are fixed, $2k$ and $\sum_{j=1}^{k} a_j$ are constant, and as $\kappa_i = \kappa$ for all $i \in \{1, \ldots, k\}$, $\sum_{i=1}^{k} c_i^T(\sum_{j=1}^{k} a_j)$ also is constant. We get an objective function in the form

$$\max \text{ const } \cdot a^T v - \text{const},$$

proving the claim. □

Lemma 3.4 shows that if all clusters have the same number of points, when looking for a PSC C with optimal objective function value $a^T v(C)$, we 'push' the centers of gravity of each pair of clusters away from each other with respect to the vectors $a_i - a_j$, so that the sum over these pairwise distances is maximized. For the general case of different κ_i, similar arguments yield that the objective function $\max a^T v = \max \sum_{i=1}^{k} a_i^T c_i$ determines the same clustering as the objective function

$$\max_{\text{PSC } C} \sum_{i=1}^{k}\sum_{j=1}^{k} \left(\frac{a_i}{\kappa_i} - \frac{a_j}{\kappa_j}\right)^T (\kappa_i c_i - \kappa_j c_j).$$

In this case, the directions of the separating hyperplanes depend on the κ_i, and the centers of gravity c_i are scaled by their corresponding size.
Note that having cluster sizes such that $\frac{\kappa_i}{\kappa_j}$ is close to 1 for all $i \neq j;\ i, j \in \{1, \ldots, k\}$ implies that the clustering considered still is a close approximation of the best case for the above measure. Note further that

$$\kappa_i c_i = \kappa_i \cdot \frac{1}{\kappa_i} \sum_{x \in C_i} x = \sum_{x \in C_i} x,$$

and these are the coordinates of the vectors of clusterings in the (bounded-shape) partition polytopes investigated in [BHR92; HOR98; HOR99; HOR00].

We close this section by turning to a special vertex (and vertex clustering) in a gravity polytope for which $\kappa_1 = \cdots = \kappa_k$. It is tied to above measure for a 'good' separability of clusterings. We consider the 'total inter-cluster distance' [BG09]:

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{x_i \in C_i} \sum_{x_j \in C_j} (\frac{a_j}{\kappa_j} - \frac{a_i}{\kappa_i})^T (x_j - x_i)$$

Informally, it sums up the distances between all pairs of points of different clusters with respect to the vectors $\frac{a_i}{\kappa_i} - \frac{a_j}{\kappa_j}$. Note that if $x \in C_i$, then x is put into a relation to other vertices exactly $n - \kappa_i$ times. Thus, in general, the contribution of a vertex depends on the size of its cluster.

Let now X satisfy $\sum_{l=1}^{n} x_l = 0$. If necessary, an original point set X can be translated (as a whole) to satisfy this condition. In [BG09] it was shown that then a clustering C with maximal total inter-cluster distance (for 'normed' site vectors) not only is associated with a vertex $v = v(C) \in Q$, but also that $\|v\|$ is maximal in Q. Thus, this clustering can be determined by Euclidean norm maximization in Q, i.e. we have $v^T v > v'^T v'$ for any $v' \in Q \backslash \{v\}$. With $v = v(C)$ being maximal with respect to the Euclidean norm in Q and $\kappa_1 = \cdots = \kappa_k$, we know that the respective clustering is induced by a power diagram to sites $c_1, \ldots, c_k$ chosen as the centers of gravity of the clusters.

In this section, we investigated and validated the idea of considering vertex clusterings in the gravity polytope for the purpose of data classification from several points of view: First, we recalled that they are LSAs with respect to fixed cluster sizes. We showed that we only need to modify the classical k-means algorithm slightly to derive clusterings belonging to vertices of some gravity polytope. Both these facts relate the ideas following in this chapter to existing partitioning clustering algorithms.

Additionally, we proved that, in the case of equal cluster sizes, we optimize a geometric measure intuitively well-suited for data classification at the vertices of the polytope. The clustering of a vertex of maximal Euclidean norm in such a gravity polytope (with $\sum_{l=1}^{n} x_l = 0$) allows its clusters to be separated in direction of the difference of the centers of gravity of the clusters.

In the next section, we describe how we can use a cell decomposition of some geometric space such that each cell contains the data vectors of a single cluster for the two basic algorithms in the field of data classification. With our approach using purely geometric arguments, we do not rely on any knowledge about the

data, like cause-effect models or cross validation dependencies. Notationally easier in many cases, and closer to typical notation in the field of combinatorial optimization, we will talk about full a-induced cell decompositions (instead of the equivalent power diagrams) and full cell decompositions for the remainder of this chapter.

3.2 Data Classification by Cell Decompositions

Suppose we have a set of data vectors $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ that are partitioned into k clusters such that there is a k-cell decomposition of $\mathbb{R}^d$, where the interior of each cell contains the points of exactly one cluster.
If we consider our clustered set of points as our 'training set', we now have an immediate approach to assigning a new vector to one of the existing clusters: We simply check which cell the vector lies in, and assign it to the respective cluster. To do so, we only have to check the position of the new point x in relation to the hyperplanes of the hyperplane arrangement $\mathcal{H}$ inducing the cell decomposition. If x lies on the boundary of more than one cell, we choose the one of lowest index. Algorithm 3 describes this procedure, Figure 26 shows a simple application. It is easy to see that we only need $k-1$ comparisons:

Lemma 3.5
Let $\mathcal{H}$ induce a k-cell decomposition of $\mathbb{R}^d$ and let $x \in \mathbb{R}^d$. Algorithm 3 requires exactly $k-1$ comparisons to determine the cell of lowest index that x lies in. This can be done with $O(d \cdot (k-1))$ elementary operations and assignments.

Proof. With each comparison, one of the cells is ruled out. As we have a cell decomposition, i.e. a partition of $\mathbb{R}^d$, the last remaining cell will certainly contain x. If $x \in P_i \cap P_j$ for two cells P_i, P_j induced by $\mathcal{H}$, then x will be assigned to the one of lower index.
Each comparison contains a d-dimensional scalar product and leads to a single assignment. This yields the running time of $O(d \cdot (k-1))$. □

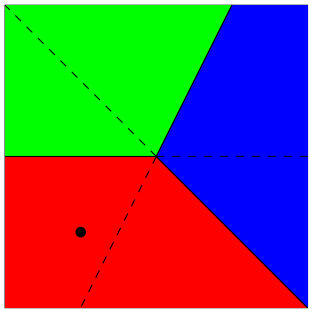
In the arithmetic model of computation, based on a hypothetical single-processor random-access machine, elementary arithmetic operations are assumed to require (constant) unit time. Thus only the number of operations performed is counted, see e.g. [AHU85; Cha03]. Using this model, we interpret Lemma 3.5 as follows.

Corollary 3.6
In the arithmetic model of computation, Algorithm 3 has a running time of $O(d \cdot (k-1))$.

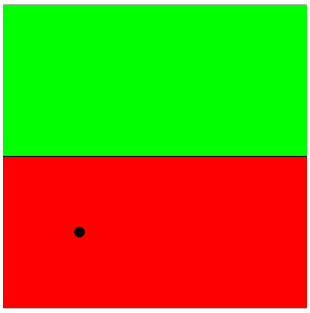
Input : $\mathcal{H}$ inducing a cell decomposition, vector x to be inserted
Output: Index i of the cell x lies in

$i := 1$;
$j := 2$;
while $j \leq k$ **do**
  if $a_{ij}^T x > \gamma_{ij}$ **then**
    $i = j$;
  end
  $j = j + 1$;
end
return i;

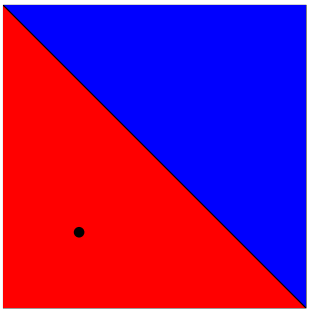
Algorithm 3: Cell of a Vector



(a) A 3-cell decomposition in $\mathbb{R}^2$, and a point to be inserted.



(b) The first comparison rules out a membership in the green cell.



(c) The second comparison rules out a membership in the blue cell.

Figure 26: An application of Algorithm 3 for inserting a point into a 3-cell decomposition of $\mathbb{R}^2$. P_1 is marked red, P_2 is marked green and P_3 is marked blue.

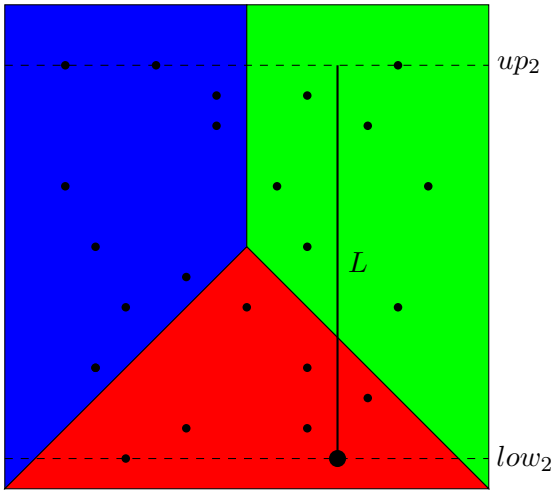


Figure 27: A 3-cell decomposition P_1 (red), P_2 (green), P_3 (blue) for of a point set X in $\mathbb{R}^2$. The big point represents the first coordinate of a new data vector x, which we want to complete to a vector on the line segment L by predicting a value for the second coordinate.

Another approach to using a cell decomposition arises from the desire to predict an 'expected' value for a data vector for which we only have incomplete information. Suppose $x \in \mathbb{R}^d$ is a data vector for which we only know the values of $d-1$ of its d parameters. We are interested in a **prediction** for the last, unknown parameter. This process is also called **data imputation** in the literature. See [WM04] for a short survey of commonly used imputation rules.
In the following, we describe how to use a cell decomposition for X for an approach to data imputation. Without loss of generality, let $x = (\xi_1, \ldots, \xi_{d-1}, \xi_d)^T$ with $\xi_1, \ldots, \xi_{d-1}$ already known and ξ_d to be derived. Let further $low_d := \min_{x \in X} \xi_d$ and $up_d := \max_{x \in X} \xi_d$ be lower and upper bounds on the value range of the d-th coefficients of points in X. If we know the possible data range to be different from the minima or maxima taken by data vectors in X, or if we only want to consider a narrower range, we, of course, are free to do so.
Let us now consider the line segment

$$L := \{(\xi_1, \ldots, \xi_{d-1}, low_d)^T + \lambda(0, \ldots, 0, up_d - low_d)^T : \lambda \in [0, 1]\}.$$

It corresponds to the set of vectors where x is complemented with a value $\xi_d \in [low_d, up_d]$.

The idea is to look at the length of the intersection of this line with the cells of our cell decomposition. Each cell P_i contains exactly the points of a single cluster C_i with gravity vector c_i. Taking the d-th entry $(c_i)_d$ of c_i for each cluster means taking the average d-th coefficient of all points in the cluster. If we weigh these average values according to the length of the intersection of L with the respective cell, we obtain a 'natural' prediction for ξ_d. Figure 27 and 28 depict this process. We now turn to an algorithmic description of this process. We need low_d, up_d and $(c_i)_d$ for all $i \in \{1, \dots, k\}$ as input. As L is a line segment, $L \cap H_{ij} = \emptyset$, $|L \cap H_{ij}| = 1$ or $L \subset H_{ij}$ for all hyperplanes H_{ij}.
For an $x' \in L$ to be in cell P_i, we need $a_{ij}^T x' \leq \gamma_{ij}$ for all $j \in \{1, \dots, k\}$. Knowing this, we introduce values $low_{ij}, up_{ij} \in [0, 1]$ for all $i \neq j;\ i, j \in \{1, \dots, k\}$ defined as follows. Let

$$low_{ij} := \arg_\lambda [\min_{\lambda \in [0,1]} a_{ij}^T((\xi_1, \dots, \xi_{d-1}, low_d)^T + \lambda(0, \dots, 0, up_d - low_d)^T) \leq \gamma_{ij}].$$

If there is no $\lambda \in [0, 1]$ satisfying this inequality, we set $low_{ij} = 1$. This can happen if $L \cap H_{ij} = \emptyset$. Analogously, we define

$$up_{ij} := \arg_\lambda [\max_{\lambda \in [0,1]} a_{ij}^T((\xi_1, \dots, \xi_{d-1}, low_d)^T + \lambda(0, \dots, 0, up_d - low_d)^T) \leq \gamma_{ij}].$$

If there is no $\lambda \in [0, 1]$ satisfying this inequality, we set $up_{ij} = low_{ij}$. Again, this can happen if $L \cap H_{ij} = \emptyset$. By these definitions, we always have $low_{ij} \leq up_{ij}$, and $low_{ij} = up_{ij}$ if and only if there is no $\lambda \in [0, 1]$ with

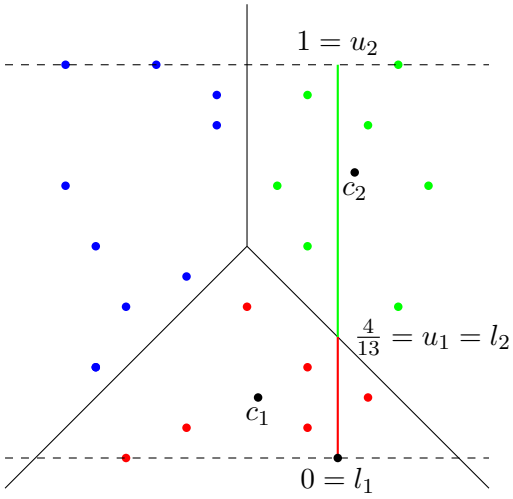
$$a_{ij}^T((\xi_1, \dots, \xi_{d-1}, low_d)^T + \lambda(0, \dots, 0, up_d - low_d)^T) \leq \gamma_{ij}.$$

In the end, $[low_{ij}, up_{ij}]$ describes the range of λs for which the respective vector in L is in $H_{ij}^{\leq}$. Algorithm 4 describes this preprocessing step. Therein, we avoid the calculation of minima or maxima with the knowledge that each H_{ij} separates $\mathbb{R}^d$ linearly into two halfspaces.
Defining vectors

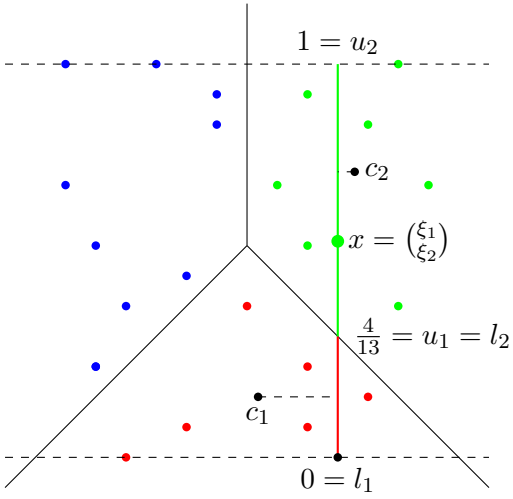
$$l_i := \max_{j \in \{1,\dots,k\} \setminus \{i\}} low_{ij} \text{ and } u_i := \min_{j \in \{1,\dots,k\} \setminus \{i\}} up_{ij} \text{ for } i \in \{1, \dots, k\},$$

we know that, if $l_i \neq u_i$,

$$\lambda \in [l_i, u_i] \Rightarrow (\xi_1, \dots, \xi_{d-1}, low_d)^T + \lambda(0, \dots, 0, up_d - low_d) \in P_i.$$



(a) L intersects with P_1 (red) and P_2 (green). $(c_1)_2$ and $(c_2)_2$ are weighted according to the relative lengths of these intersections for a prediction of ξ_2.



(b) The expected value ξ_2 for the incomplete data vector $x = \binom{\xi_1}{\xi_2}$ is calculated as $(u_1 - l_1) \cdot (c_1)_2 + (u_2 - l_2) \cdot (c_2)_2$.

Figure 28: An example for data value prediction using the cells of the cell decomposition of Figure 27.

Input : $\mathcal{H}$ inducing a cell decomposition, $\xi_1, \ldots, \xi_{d-1}$, low_d, up_d
Output: low_{ij}, up_{ij} for all $i \neq j;\ i, j \in \{1, \ldots, k\}$

for $1 \leq i \leq k$ **do**
  for $1 \leq j \leq k; j \neq i$ **do**
    if $a_{ij}^T(\xi_1, \ldots, \xi_{d-1}, low_d)^T \leq \gamma_{ij}$ **then**
      $low_{ij} = 0$;
    end
    else
      if $a_{ij}^T((\xi_1, \ldots, \xi_{d-1}, up_d)^T > \gamma_{ij}$ **then**
        $low_{ij} = 1$;
      end
      else
        $low_{ij} = \arg_\lambda[a_{ij}^T((\xi_1, \ldots, \xi_{d-1}, low_d)^T + \lambda(0, \ldots, 0, up_d - low_d)^T) = \gamma_{ij}]$;
      end
    end
  end
  for $1 \leq j \leq k; j \neq i$ **do**
    if $a_{ij}^T(\xi_1, \ldots, \xi_{d-1}, up_d)^T \leq \gamma_{ij}$ **then**
      $up_{ij} = 1$;
    end
    else
      if $a_{ij}^T((\xi_1, \ldots, \xi_{d-1}, low_d)^T > \gamma_{ij}$ **then**
        $up_{ij} = low_{ij}$;
      end
      else
        $up_{ij} = \arg_\lambda[a_{ij}^T((\xi_1, \ldots, \xi_{d-1}, low_d)^T + \lambda(0, \ldots, 0, up_d - low_d)^T) = \gamma_{ij}]$;
      end
    end
  end
end
return low_{ij}, up_{ij} for all $i \neq j;\ i, j \in \{1, \ldots, k\}$;

Algorithm 4: Calculation of low_{ij} and up_{ij}

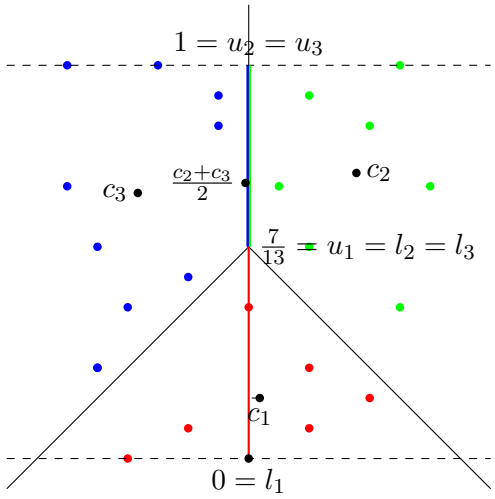


Figure 29: The cell decomposition of Figure 27, and an incomplete x with $L \subset H_{23}$. Here $\frac{1}{2}(c_2 + c_3)$ is weighted with the length of the common intersection of P_2 and P_3 with L.

We use these intervals as weights for our prediction, namely we set $w_i := (u_i - l_i) \cdot (c_i)_d$ and $\xi_d := \sum_{i=1}^{k} w_i$, unless $L \subset H_{ij}$.
We can avoid the case of $L \subset H_{ij}$ either by using a (possibly slightly perturbed) cell decomposition for which $a_{ij}^T(0, \ldots, 0, 1)^T \neq 0$ for $i \neq j;\ i, j \in \{1, \ldots, k\}$, or we consider it during the algorithm as follows: We check whether there are intervals $[l_i, u_i] = [l_j, u_j]$ with $i \neq j;\ i, j \in \{1, \ldots, k\}$. If so, we identify the number s of clusters that share such an interval. Their respective weight contribution value is then just divided by s, i.e.

$$w_i := \frac{1}{s}(u_i - l_i) \cdot (c_i)_d$$

Figure 29 depicts this approach. This can be done easily if we sort the l_i and u_i by ascending values.
Algorithm 5 sums up the whole prediction process (without a sorting routine for the l_i and u_i). Its running time is as follows.

Lemma 3.7
Let $\mathcal{H}$ induce a k-cell decomposition of $\mathbb{R}^d$ and let $x \in \mathbb{R}^d$. Then Algorithm 5 can be performed with $O(d \cdot k \cdot (k-1))$ elementary operations and assignments.

Proof. We start by analyzing Algorithm 4, used as a subroutine of Algorithm 5. For any of the $k \cdot (k-1)$ combinations of $i \neq j$, we determine low_{ij} and

Input : $\mathcal{H}$ inducing the cell decomposition, $\xi_1, \ldots, \xi_{d-1}$, low_d, up_d, $(c_i)_d$ for all $i \in \{1, \ldots, k\}$
Output: Predicted value ξ_d

Calculate up_{ij} and low_{ij} with Algorithm 4;
for $1 \leq i \leq k$ **do**
  $l_i := \max_{j \in \{1,\ldots,k\} \setminus \{i\}} low_{ij}$;
  $u_i := \min_{j \in \{1,\ldots,k\} \setminus \{i\}} up_{ij}$;
  $w_i := (u_i - l_i) \cdot (c_i)_d$;
end
Index set $I := \{1, \ldots, k\}$;
for $i \in I$ **do**
  $I = I \setminus \{i\}$;
  $s := 1$;
  Index set $I' := \{i\}$;
  for $j \in I$ **do**
    if $l_i = l_j$ *and* $u_i = u_j$ **then**
      $s = s + 1$;
      $I = I \setminus \{j\}$;
      $I' = I' \cup \{j\}$;
    end
  end
  for $j \in I'$ **do**
    $w_j = \frac{1}{s} w_j$;
  end
end
return $\xi_d := \sum_{i=1}^{k} w_i$;

Algorithm 5: Prediction

up_{ij}. Both calculations work analogously, we turn to the one for low_{ij}. In a worst case, we have to check the conditions in both if-statements and solve the equality in the last else-case. For this, the two d-dimensional scalar products and $\gamma_{ij} - a_{ij}^T(\xi_1, \ldots, \xi_{d-1}, low_d)^T$ have to be calculated. The latter has to be set equal to $\lambda a_{ij}^T(0, \ldots, 0, up_d - low_d)^T$. This is an equation in $\mathbb{R}$, which can be solved by a single division. Further, there is exactly one assignment for each combination of $i \neq j$. This results in a total of at most $2k \cdot (k-1)$ comparisons and $O(d \cdot k \cdot (k-1))$ elementary operations and assignments.
In Algorithm 5, we then continue with the calculation of l_i, u_i and w_i for all $i \in \{1, \ldots, k\}$. For l_i and u_i we need to consider the values low_{ij} and up_{ij} for all $j \in \{1, \ldots, k\} \backslash \{i\}$, and thus require $2k \cdot (k-1)$ comparisons (and up to that many assignments) for this step. The w_i are then derived by $2k$ arithmetic operations. In the second for-loop, each index $i \in \{1, \ldots, k\}$ is removed from I and added to I' exactly once. By this, we have $2k$ set operations. We perform both comparisons of the if-statement at most $\frac{k \cdot (k-1)}{2}$ times, for a total of at most $k \cdot (k-1)$ comparisons. We have $O(k)$ assignments and increments of s in total. The second interior for-loop is again entered exactly once for each index in $\{1, \ldots, k\}$, yielding another k arithmetic operations. Finally, the return-part yields another k arithmetic operations.
This results in a total of at most $3(k \cdot (k-1))$ comparisons and $O(k)$ set operations, elementary arithmetic operations and assignments for Algorithm 5 without the operations in Algorithm 4. The running time is dominated by the $O(d \cdot k \cdot (k-1))$-term from Algorithm 4, yielding the claim. □

Corollary 3.8
In the arithmetic model of computation, Algorithm 5 has a running time of $O(d \cdot k \cdot (k-1))$.

We described the process for a single incomplete criterion. Similar ideas can be used if we have more than one 'missing' data entry. In this case L will be higher-dimensional, so that it is necessary to calculate areas or volumina instead of line segment lengths. Such calculations are considerably more difficult.
Additionally, if we are willing to increase our computational effort in our original case of only one single incomplete criterion, we may increase the stability of our prediction as follows: Instead of using a line segment L, we choose a (small) $\epsilon \in \mathbb{R}$ and use a box

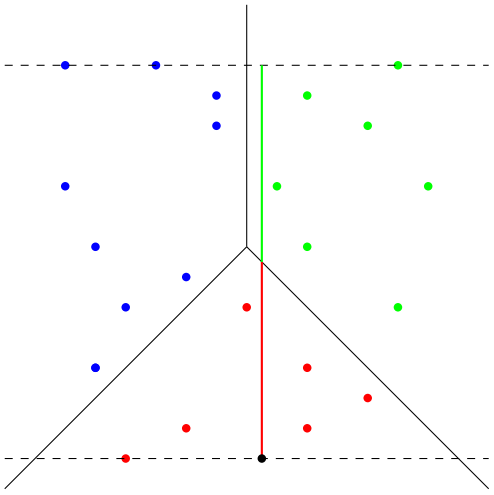


Figure 30: An application of Algorithm 5 with a problematic position of x. Even though the first $d-1$ coefficients of x put it close to a membership in the blue cluster, the prediction is done entirely with information from the red and green one.

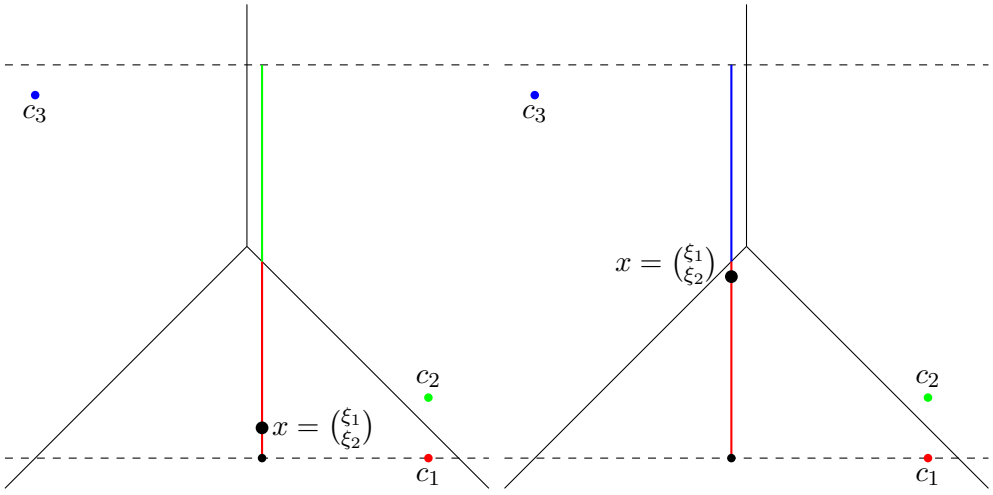
$$L' := \{(\xi_1 + \epsilon_1, \ldots, \xi_{d-1} + \epsilon_{d-1}, 0)^T + \lambda(0, \ldots, 0, up_d - low_d) : \\ \lambda \in [0,1], \epsilon_i \in [-\epsilon, \epsilon] \text{ for all } i \in \{1, \ldots, d-1\}\}.$$

Again, this means that we have to calculate volumina instead of a line segment. Doing so is especially useful if our situation is similar to the one depicted in Figure 30. Without this addition, we will solely predict the value of ξ_d according to the red and green cluster, neglecting how close the variables $\xi_1, \ldots, \xi_{d-1}$ put it to a membership in the blue cluster.
In the next section, we turn to further options and advantages that we have for data classification if we have a full cell decomposition instead of only a cell decomposition.

3.3 Data Classification by Full a-induced Cell Decompositions

Constructing a clustering of a data set $X \subset \mathbb{R}^d$ and a corresponding cell decomposition as described in the previous chapter not only yields a cell decomposition, but a full a-induced cell decomposition of $\mathbb{R}^d$, see Corollary 2.36. This fact means some good news and some bad news. The bad news is that we restrict our investigations to this special type of cell decompositions. The good news is

that the nice structure of full a-induced cell decompositions yields several ideas to improve our data classification methods. Also, it yields an approach to a visualization due to the importance of the subsystems of three cells. We turn to both these advantages and disadvantages of our approach in the following.
We consider a PSC $C := (C_1, \ldots, C_k)$ and a full a-induced cell decomposition $\mathcal{P} := (P_1, \ldots, P_k)$ with $C_i \subset \text{int}(P_i)$ for all $i \in \{1, \ldots, k\}$ for some $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $a_i \neq a_j$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$. $v := v(C)$ is optimal in Q with respect to $a'^T v$, where $a' = (\kappa_1 \cdot a_1^T, \ldots, \kappa_k \cdot a_k^T)^T$.
Recalling the construction of the a-induced separating hyperplane directions $a_{ij} := a_j - a_i$ in Theorem 2.33, we see that these $\frac{k \cdot (k-1)}{2}$ vectors (recall $a_{ij} = -a_{ji}$) are defined by the k sites $a_1, \ldots, a_k \in \mathbb{R}^d$, or rather by only $k - 1$ sites $a_2, \ldots, a_k$, if we (without loss of generality) set $a_1 = 0$. (Recall Lemma 2.56.)
Thus choosing the system of separation directions by our a-induction is a restriction due to our self-imposed reduced number of degrees of freedom. Consider the example in Figure 20. It is a cell decomposition, and the cells $P_1, \ldots, P_k$ can be a-induced. Each cell is the intersection of only two halfspaces, only $k = 6$ hyperplanes are needed for the definition of the cells.
If we only are interested in any cell decomposition, the other $\frac{6 \cdot 5}{2} - 6 = 9$ hyperplanes could be chosen 'arbitrarily' (such that they do not intersect the interior of the respective cells). If we want a full cell decomposition, we only have to choose them within the respective cones, see Lemma 2.24. But if we derive the directions from an a-induction, they are fixed to specific values.
Yet this specific structure is also a big advantage in that all subsets of indices also yield cell decompositions of the respective data space. We now turn to the question how we can use this fact profitably in our data classification efforts.
The algorithms we described in the last section for inserting a new point into an existing cluster and predicting a data entry only require a cell decomposition of the data space. If we have a full cell decomposition, we have several opportunities to extend this basic approach.
First, assume we want to complete a missing variable of a data vector according to Algorithm 5, in a situation like depicted in Figure 30 (compare Figures 28, 29). Even though a large part of L is very close to the blue cell, the prediction value is generated entirely with the information of the green and red clusters.
It is easy to construct an example for which the prediction result differs greatly if the known data entries of the x to be complemented are only slightly different. Consider the example in Figure 31. It depicts the high sensitivity of our approach.



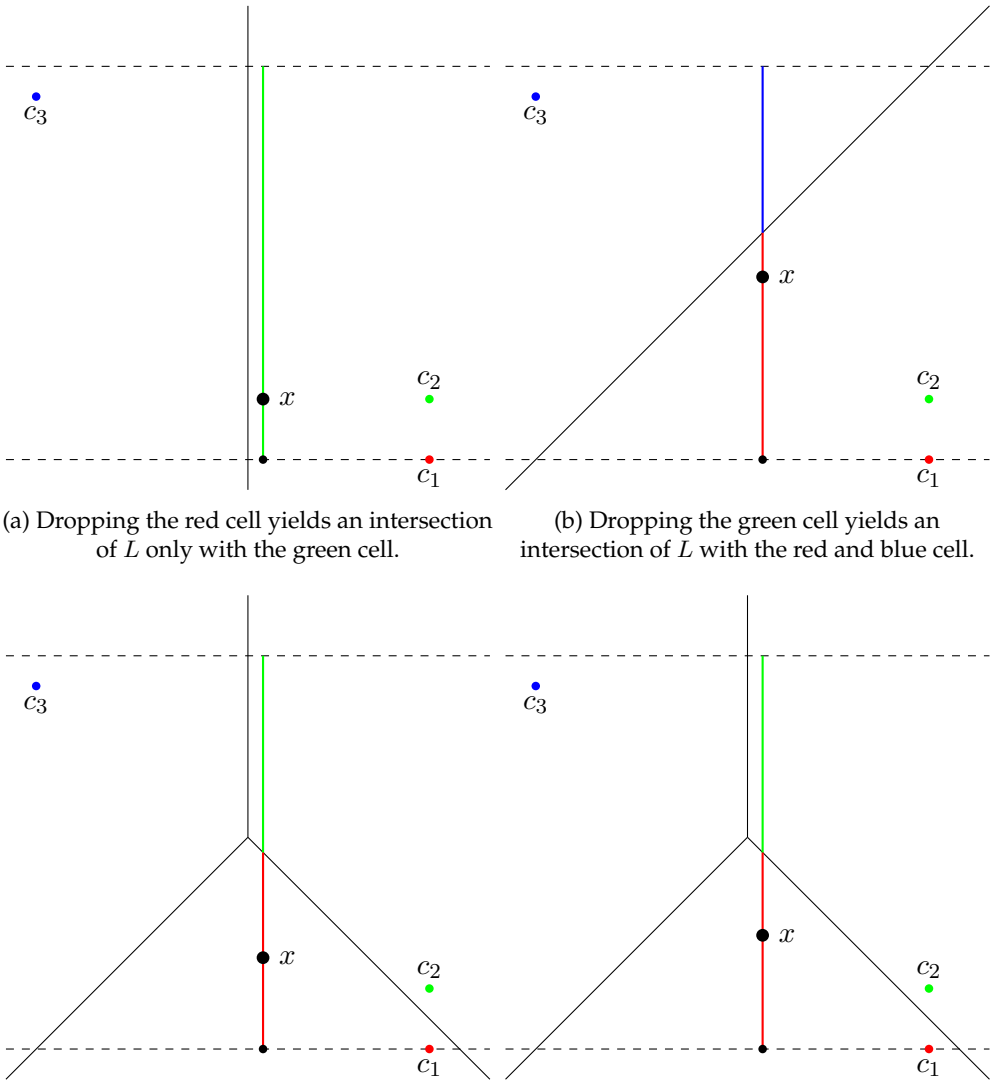
(a) Prediction with L intersecting the red and green cells.

(b) Prediction with L intersecting the red and blue cells.

Figure 31: An application of Algorithm 5 in a 'worst case' example. The prediction is very sensitive to small changes in the x to be complemented.

Mostly, it results from an intuitively 'bad' cell decomposition we are working with - we will turn to this in detail later on. A first idea of reducing this sensitivity was mentioned earlier: Integrating a whole area around the incomplete vector x yields a less sensitive approach, yet it is computationally expensive.
We now turn to how a full cell decomposition helps us with measuring and reducing the sensitivity of the prediction. Recall that the cells of any subset of indices still form a cell decomposition of $\mathbb{R}^d$. As our algorithm is sensitive to L being close to, but not in a specific cell, we choose the following approach:
First, we apply the algorithm normally, and derive a prediction in the usual way. We note which cells had a non-empty intersection with L. For each index i identifying such a cell, we perform our algorithm for the cell decomposition $P_1^I, \ldots, P_{i-1}^I, P_{i+1}^I, \ldots, P_k^I$, where $I := \{1, \ldots, k\} \backslash \{i\}$. We get a prediction result for each of these special cases, and can now add these up with appropriate weights to derive a prediction value. If we have no further information, we may try so with e.g. equal weights for all indices. If we have further information on the original cell decomposition and the original performance of the algorithm, we may choose more sophisticated ones: E.g. if we know that a long line segment of L is very close to the separating hyperplane between two clusters in the

original cell decomposition, we can assign a higher weight to the respective cell decomposition without the index of this cell.
Figure 32 depicts the procedure. We have a non-empty intersection of L with the red and green cells and thus also look at the cell decompositions only consisting of the red and blue, and of the green and blue cluster. In $c)$, we sum up the predicted values of ξ_2 with equal weights for the original and both subsystem values.
From a closer look at the original cell decomposition, we may know that a long section of L lies in the green cell, very close to the blue cell, in the original cell decomposition. By this, we know that we can reduce the sensitivity with respect to that hyperplane by increasing the weight of the cell decomposition subsystem where we leave out the green cell. In $d)$, we show the prediction for this subsystem having double its usual weight (and thus yielding (a rather extreme) 50% of the total information in our example).
Even if we do not want to influence the prediction for the unknown coefficient of x in our cell decomposition directly, we obtain a number of prediction values that we can compare the one using the full original system to. By this, we get an intuitive measure for the stability of the prediction done. Checking how much our prediction value changes if the factors of the contributing cell decomposition subsystems are adjusted slightly yields another measure for the sensitivity and stability of our prediction.
Another useful property of full cell decompositions arises with respect to the additional classification possibilities we have. As every subsystem of indices is a cell decomposition by itself, we may introduce 'secondary' classifications for new data points as follows.
Suppose a new data vector x lies in cell P_i and thus is associated with cluster C_i. If we consider the cell decomposition of index set $I := \{1, \dots, k\}\backslash\{i\}$, x lies in some cell P_j^I with $j \in I$. This information can be interpreted as x being most similar to the data vectors in C_i, but also being rather similar to the ones in C_j. Naturally, this idea can be continued to add a 'tertiary' classification or more for such a point.
Doing so might lead to some deeper insight into the underlying data structure: Suppose we have clusters C_i and C_j, and in the subsystem for indices $I := \{1, \dots, k\}\backslash\{i\}$, almost all data vectors of C_i now are in P_j^I. Additionally for the subsystem $I' := \{1, \dots, k\}\backslash\{j\}$, almost all data vectors of C_j' now are in P_i^I. Then it is clear that the two clusters differ almost only by the combination of data



(a) Dropping the red cell yields an intersection of L only with the green cell.

(b) Dropping the green cell yields an intersection of L with the red and blue cell.

(c) The prediction for even weights of the original cell decomposition and both subsystem cell decompositions.

(d) The prediction for the cell decomposition skipping the (critical) green cell having double the normal weight factor.

Figure 32: How to use full cell decompositions for a sensitivity reduction of Algorithm 5 for the example of Figure 31 a).

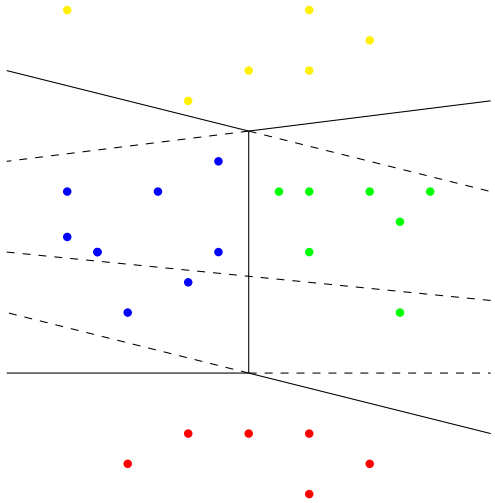
entries induced by the separating hyperplane H_{ij}. Figure 33 shows an example for such a dependency.
The fact that we consider a full cell decomposition further allows us to visualize our high-dimensional cell decompositions by two-dimensional projections. Each triple $I := \{i, j, t\}$ of indices still yields a cell decomposition.
By Lemma 2.24, their separating vectors have a two-dimensional span. By this, we can depict our data set in the plane

$$E_{ijt} := \{\lambda_{it}a_{it} + \lambda_{ij}a_{ij} + \lambda_{jt}a_{jt} : \lambda_{it}, \lambda_{jt}, \lambda_{ij} \in \mathbb{R}\} = \{\lambda_{it}a_{it} + \lambda_{ij}a_{ij} : \lambda_{it}, \lambda_{ij} \in \mathbb{R}\}.$$

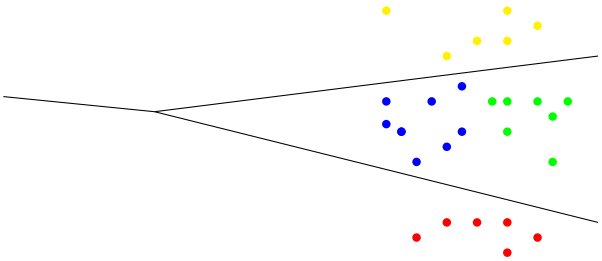
Therein the intersection of the hyperplanes H_{ij}, H_{jt}, H_{ti} is a single point, if the hyperplanes are not parallel to each other (see Corollary 2.25 and Lemma 2.26). We know that these triples of indices describe our full cell decompositions fully, by definition and Lemma 2.27 . Thus the respective figures suffice to understand the structure of the full cell decomposition at hand. Almost all figures throughout Chapter 2 and 3 are two-dimensional examples, to be pictographically intelligible. Interpreting them as the two-dimensional projections of higher-dimensional cell decompositions, we know that they do not describe special cases, but depict all of the necessary information.
We will describe another advantage of our full cell decompositions when turning to the choice of size restrictions $\kappa_1, \ldots, \kappa_k$ for the clusters. Next, we discuss the construction of a PSC and a corresponding full a-induced cell decomposition according to Theorem 2.33 algorithmically.

3.4 Calculation of a Full a-induced Cell Decomposition

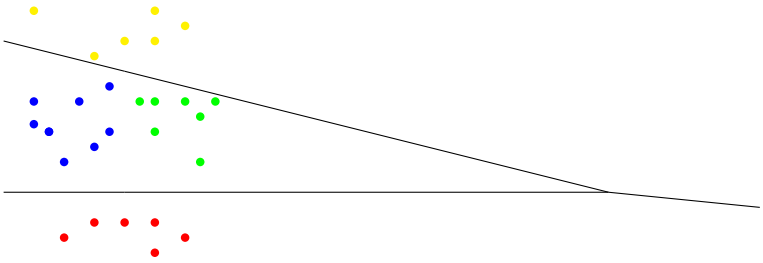
If we have a clustering of a point set $X \subset \mathbb{R}^d$ and a cell decomposition such that each cell contains the points of exactly one cluster, we get a natural, intuitive and efficient approach to data classification and prediction by using the algorithms described in Section 3.2.
Suppose that in our application the number of desired clusters k is predetermined, as are the sizes $\kappa_1, \ldots, \kappa_k$ of the clusters. Theorem 2.33 and Corollary 2.36 explain that, by calculating a vertex of $Q(X, k, \kappa_1, \ldots, \kappa_k)$, we get a PSC that allows a full a-induced cell decomposition. Analogously, it is induced by an (a, W)- power diagram for some set of weights W (Theorem 2.52), and is a (strict) LSA to a with respect to $\kappa_1, \ldots, \kappa_k$ (Theorem 2.55).



(a) The cell decomposition for all four clusters.



(b) In the cell decomposition 'skipping' the blue cluster, the points of the blue cluster now lie in the cell of the green one.



(c) In the cell decomposition 'skipping' the green cluster, the points of the green cluster now lie in the cell of the blue one.

Figure 33: A full cell decomposition for which the green and blue clusters are separated similarly from all other clusters.

On one hand, this allows us to use existing algorithms for the calculation of power diagrams, respectively LSAs. Aurenhammer, Hoffman and Aronov gave a constructive proof of Theorem 2.49 $2.)$ in the form of an algorithm that constructs the weights of a power diagram inducing a clustering respecting the given cluster size constraints [AHA98]. Their algorithm is strongly polynomial, and runs in time $O(k^2 n \log n + kn \log^2 n)$.
In an alternative approach, Balzer and Heck gave an iterative algorithm for the calculation of a LSA with respect to given cluster sizes. Each iteration step runs in time $O(k^2 + kn \log \frac{n}{k})$ [BH08], with empirical results showing that only a low number of such steps usually is required.
In the following, we will show that the calculation of a vertex of Q can be modeled as a $0,1$-integer linear program which can be solved in the form of a linear program. Further, having a vertex of Q and the associated PSC, the (separate) calculation of the hyperplane positions for a full cell decomposition belonging to this PSC can be modeled by a linear program, as well. We here cover the basis for an algorithm, by constructing the associated linear programs. There are many advantages of using this approach, with the most important ones being a simple implementation, the wide availability of efficient and powerful solvers for linear programs and the favorable running time of the algorithms used by these solvers. (See e.g. [Bor82; Bor87] for an analysis of the expected running time of the Simplex algorithm.)
We start by modeling the constraints to derive a PSC. We introduce decision variables $y_{ij} \in \{0,1\}$ for all $i \in \{1,\dots,k\}$ and all $j \in \{1,\dots,n\}$, indicating whether C_i contains x_j ($y_{ij} = 1$) or not ($y_{ij} = 0$). As each x_j lies in exactly one cluster, we need the constraints

$$\sum_{i=1}^{k} y_{ij} \quad = \quad 1 \qquad (j \leq n).$$

Additionally, cluster C_i has to contain exactly κ_i vertices. We can denote this in the form

$$\sum_{j=1}^{n} y_{ij} \quad = \quad \kappa_i \qquad (i \leq k)$$

Summed up, we obtain the following set of constraints for the decision variables in $\mathbb{R}^{k \cdot n}$:

$$\begin{aligned} \sum_{j=1}^{n} y_{ij} &= \kappa_i && (i \leq k) \\ \sum_{i=1}^{k} y_{ij} &= 1 && (j \leq n) \\ y_{ij} &\in \{0,1\} && (i \leq k, j \leq n) \end{aligned}$$

A relaxation of the $y_{ij} \in \{0,1\}$ to $0 \leq y_{ij} \leq 1$, and noting that $\sum_{i=1}^{k} y_{ij} = 1$ and $0 \leq y_{ij}$ for all $i \leq k$ and $j \leq n$ already implies $y_{ij} \leq 1$ for all $i \leq k, j \leq n$, yields the following polytope in $\mathbb{R}^{k \cdot n}$. As we will see, no information is lost due to this relaxation.

Definition 3.9 (Partition Polytope (PP))
Let $k, \kappa_1, ..., \kappa_k \in \mathbb{N}$ with $\sum_{i=1}^{k} \kappa_i = n$. We call the polytope constructed above the **partition polytope** $PP(k, \kappa_1, \ldots, \kappa_k)$*:*

$$\begin{aligned} \sum_{j=1}^{n} y_{ij} &= \kappa_i && (i \leq k) \\ \sum_{i=1}^{k} y_{ij} &= 1 && (j \leq n) \\ y_{ij} &\geq 0 && (i \leq k, j \leq n) \end{aligned}$$

If the context is clear, we will only use the term PP instead of $PP(k, \kappa_1, \ldots, \kappa_k)$, and use it as an abbreviation for 'a partition polytope', 'the partition polytope', or 'the corresponding partition polytope'. We note that the partition polytope and the gravity polytope are connected by a linear transformation adding the geometric information.

Lemma 3.10
Let $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$. Then there is a linear transformation that, applied to the partition polytope, derives the corresponding gravity polytope.

Proof. The partition polytope lies in $\mathbb{R}^{n \cdot k}$, as we have a $0,1$-decision variable for each item and cluster. The gravity polytope lies in $\mathbb{R}^{d \cdot k}$ with its defining vectors listing the centers of gravity of each cluster. In the partition polytope, the items have no geometric interpretation, it is added during the transformation.
Let y_{ij} again denote the decision variable whether item j belongs to cluster C_i, and let $x_j \in \mathbb{R}^d$ be the geometric vector associated with item j. Suppose we have a PSC of our data set, from which we obtain the values of the y_{ij} and the gravity vector v. From this, we can identify the linear transformation used. We prove the

claim by denoting it in the form $v = Ay$, where $y = (y_{11}, \ldots, y_{1n}, y_{21}, \ldots, y_{kn})^T$. We have

$$A = \begin{pmatrix} \frac{x_1}{\kappa_1} & \frac{x_2}{\kappa_1} & \ldots & \frac{x_n}{\kappa_1} & 0 & 0 & \ldots & 0 & \ldots & 0 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 & \frac{x_1}{\kappa_2} & \frac{x_2}{\kappa_2} & \ldots & \frac{x_n}{\kappa_2} & \ldots & 0 & 0 & \ldots & 0 \\ & & & & \ldots & & & & & & & & \\ & & & & \ldots & & & & & & & & \\ 0 & 0 & \ldots & 0 & 0 & 0 & \ldots & 0 & \ldots & \frac{x_1}{\kappa_k} & \frac{x_2}{\kappa_k} & \ldots & \frac{x_n}{\kappa_k} \end{pmatrix}$$

Each of the lines in this notation of A is actually d-dimensional due to the x_j being d-dimensional. A consists of k blocks of dimensions $(d \cdot k) \times n$, put side to side. □

Note that the partition polytope does not use any geometric information. By Lemma 3.10, we may try to construct an X corresponding to a set of n (non-geometric) items to investigate the structure of both polytopes further. A basic approach to doing so will be described towards the end of Chapter 4.

The partition polytope defines equalities that represent that we need to assign κ_i items to the i-th cluster and that each item must be assigned to exactly one cluster. The y_{ij}-values capture this information. This informal interpretation is just a special case of the constraints of the general **transportation problem**, a classical problem in the field of operations research:

Informally, we have m suppliers and n demanders of a resource. Each supplier i has some quantity $a_i > 0$, each demander j asks for some quantity $b_j > 0$. The sum of available quantities equals the sum of demanded quantities. The goal is to minimize the costs of transporting these items from the suppliers to the demanders, where c_{ij} denotes the cost of sending one unit from supplier i to demander j.

Transportation problems arise in many fields of operational research and economics, see [KW68; KYK84; Loe06] for surveys of the field. Modeled in the same way, we note that the partition polytope is a special **transportation polytope**. These are well-studied and understood, and are the basis for efficient algorithms for transportation problems due to their constraint matrix being totally unimodular, which implies that all vertices are integral [KW68].

With our $0, 1$-decision variables, they only consist of coefficients 0 and 1, and thus directly correspond to a PSC. By this, no information is lost due to the relaxation used. This allows a calculation of vertices of Q optimal with respect to a linear objective function by optimizing a linear objective function over PP, as follows:

Theorem 3.11
Let $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, and let $k, \kappa_1, ..., \kappa_k \in \mathbb{N}$ with $\sum_{i=1}^{k} \kappa_i = n$. Let Q be the corresponding gravity polytope, and $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$.
Then we can find a vertex v of Q with $a^T v \geq a^T v'$ for any $v' \in Q$ by solving a linear program.

Proof. Let $C := (C_1, \ldots, C_k)$ be a PSC and $v := (c_1^T, \ldots, c_k^T)^T$. We prove the claim by showing that we only need to optimize a linear objective function over the corresponding PP.

To do so, we first look at the objective function $a^T v = \sum_{i=1}^{k} a_i^T c_i$. The c_i are constructed as the arithmetic mean of the points of cluster C_i. We see that $c_i = \frac{1}{\kappa_i} \sum_{j=1}^{n} y_{ij} x_j$ for all $i \in \{1, \ldots, k\}$ and thus

$$a^T v = \sum_{i=1}^{k} a_i^T c_i = \sum_{i=1}^{k} \frac{1}{\kappa_i} a_i^T (\sum_{j=1}^{n} y_{ij} x_j) = \sum_{i=1}^{k} \sum_{j=1}^{n} \frac{1}{\kappa_i} y_{ij} a_i^T x_j$$

Being a special transportation polytope, we know that the coefficient matrix of PP is totally unimodular. With $\kappa_i \in \mathbb{N}$ for all $i \in \{1, \ldots, k\}$, we know that any vertex of PP is integral, and thus is a $0, 1$-vector identifying a PSC directly. Thus, it suffices to optimize the following relaxed linear program.

$$\begin{aligned}
\max \sum_{i=1}^{k} \sum_{j=1}^{n} y_{ij} \tfrac{1}{\kappa_i} a_i^T x_j & \\
\sum_{j=1}^{n} y_{ij} &= \kappa_i && (i \leq k) \\
\sum_{i=1}^{k} y_{ij} &= 1 && (j \leq n) \\
y_{ij} &\geq 0 && (i \leq k, j \leq n)
\end{aligned}$$

The clustering associated with the values y_{ij} of the optimal solution of this linear program is the PSC with optimal $a^T v$ in Q, by construction of PP and the objective function. This proves the claim. □

We note that the linear program in Theorem 3.11 has $k \cdot n + 2k + 2n$ constraints (if we consider each constraint with $=$ as two constraints with $\leq$ and $\geq$). By this linear program, we calculate a PSC belonging to a vertex of Q, or respectively a LSA to some set of sites $a = (a_1^T, \ldots, a_k^T)^T$. We are now also able to calculate a full a-induced cell decomposition inducing this PSC using a linear program.

Theorem 3.12
Let $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, and let $k, \kappa_1, ..., \kappa_k \in \mathbb{N}$ with $\sum_{i=1}^{k} \kappa_i = n$. Let $C := (C_1, \ldots, C_k)$ be a clustering such that $v := v(C)$ is a vertex of Q and $a \in \mathbb{R}^{d \cdot k}$ such that $a^T v > a^T v'$ for any $v' \in Q$.
Then we can calculate a full a-induced cell decomposition $\mathcal{P} := (P_1, \ldots, P_k)$ with $C_i \subset int(P_i)$ by solving a linear program.

Proof. The directions of the separating hyperplanes are trivially given as $a_{ij} := \frac{a_j}{\kappa_j} - \frac{a_i}{\kappa_i}$. We prove the claim by constructing a linear program to calculate the positions γ_{ij} of the hyperplanes H_{ij} based on the given clustering.
Again, we use $y_{ij} \in \{0, 1\}$ for all $i \in \{1, \ldots, k\}$ and all $j \in \{1, \ldots, n\}$, indicating whether C_i contains x_j ($y_{ij} = 1$) or not ($y_{ij} = 0$). Note that these are not variables in our linear program, but the fixed values corresponding to our clustering.
By Corollary 2.36, we know that we are able to choose the γ_{ij} such that any cycle of γ_{ij} sums up to 0. Using the relations $\gamma_{ij} = \gamma_{i1} - \gamma_{j1}$ (equivalent to $\gamma_{1i} + \gamma_{ij} + \gamma_{j1} = 0$) represents the positioning of the hyperplanes fully, by the definition of full a-induced cell decompositions.
The positions of the hyperplanes must be according to the conditions $C_i \subset P_i$, thus we add constraints $y_{il} \cdot (a_{ij}^T x_l) \leq y_{il} \cdot \gamma_{ij}$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$ and all $l \in \{1, \ldots, n\}$. With $y_{il} = 0$ if $x_l \notin C_i$, in this case, the constraint is satisfied automatically by $0 \leq 0$.
For ease of notation, we complement the system of γ_{ij} with γ_{ii} for all $i \in \{1, \ldots, k\}$. They will automatically be set to $\gamma_{ii} = 0$. With the constraints mentioned above, we have the following linear feasibility problem:

$$\begin{aligned} y_{il} \cdot (a_{ij}^T x_l) &\leq y_{il} \cdot \gamma_{ij} && (i \neq j, i \leq k, j \leq k, l \leq n) \\ \gamma_{ij} &= \gamma_{i1} - \gamma_{j1} && (i \leq k, j \leq k) \end{aligned}$$

With $\gamma_{ij} = -\gamma_{ji}$, it suffices to only use $j > i$. As C is associated with a vertex v of Q, and $a^T v > a^T v'$ for any $v' \in Q \backslash \{v\}$, we know that the clustering is strictly separable. For a cell decomposition, we have $C_i \subset \text{int}(P_i)$. Thus, we know that above system is solvable such that $a_{ij}^T x_l < \gamma_{ij}$ for all $i \neq j;\ i, j \in \{1, \ldots, k\}$ and $l \in \{1, \ldots, n\}$ with $x_l \in C_i$.
Such a solution for the system of γ_{ij} can be found by maximizing the minimal (normed) distance of a hyperplane to the closest vectors of the clusters it separates.

To do so, we compute the values $maxA_{ij} := \max_{l:y_{il}=1} a_{ij}^T x_l$ for all $i \neq j;\ i,j \in \{1,\dots,k\}$. We then add the constraints $\gamma_{ij} = maxA_{ij} + \delta_{ij}$ for all $i \neq j;\ i,j \in \{1,\dots,k\}$ to our linear program. Recalling the remark after Lemma 2.32, we know that we can choose all $\delta_{ij} > 0$ at the same time. We guarantee that we do so by using a lower bound $\delta \leq \delta_{ij}$ for all $i \neq j;\ i,j \in \{1,\dots,k\}$, and maximizing δ in our objective function.
In the resulting linear program, the conditions $y_{il} \cdot (a_{ij}^T x_l) \leq y_{il} \cdot \gamma_{ij}$ can be replaced by the conditions $\gamma_{ij} = maxA_{ij} + \delta_{ij}$. This time, we need to consider all $i \neq j$, as the distances of H_{ij} to both clusters are important.

$$\begin{array}{rcll} & \max \delta & & \\ maxA_{ij} + \delta_{ij} & = & \gamma_{ij} & (i \neq j, i \leq k, j \leq k, l \leq n) \\ \frac{\delta_{ij}}{\|a_{ij}\|} & \geq & \delta & (i \neq j, i \leq k, j \leq k) \\ \gamma_{ij} & = & \gamma_{i1} - \gamma_{j1} & (i \leq k, j \leq k) \end{array}$$

By the objective function, δ will certainly be chosen as $\delta > 0$, yielding $C_i \subset \text{int}(P_i)$ for the a-induced system of hyperplanes using the calculated γ_{ij}. Note that the $\|a_{ij}\|$ do not necessarily have a finite bit-representation. Note further that dividing δ_{ij} by $\|a_{ij}\|$ is not necessary to find a $\delta > 0$. Thus, to avoid computational problems, we can also use $\delta_{ij} \geq \delta$ in the above linear program. This proves the claim. □

Depending on the $a \in \mathbb{R}^{d \cdot k}$ used, the linear program of Theorem 3.11 returns a PSC optimal with respect to the objective function, but not necessarily uniquely optimal. Recall Corollary 2.43 and Figure 25.
Intuitively, a cell decomposition without strict separation of the clusters is 'bad' for our data classification purposes. We can identify this problem case by observing an objective function value of $\delta = 0$ in the linear program of Theorem 3.12.
A randomly chosen vector $a := (a_1^T, \dots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ will generally satisfy $\frac{a_i}{\kappa_i} \neq \frac{a_j}{\kappa_j}$ for all $i,j \in \{1,\dots,k\}$, avoiding the problem described in Lemma 2.18. Due to the finite number of facets of Q, respectively PP, we will also almost certainly find a vertex v uniquely optimal with respect to $a^T v$.
On the other hand, with the geometric interpretation of our boundary clusterings as LSAs, and with our separating hyperplane directions as linear differentiation criteria of the clusters, such a random choice of a may not be viable: We might

have a fixed a for which several vertices are (not uniquely) optimal, i.e. several PSCs with weakly separating cell decompositions.
In this case, the algorithm in [AHA98] converges to a single 'arbitrary' one of these multiple PSCs. To identify all optimal PSCs, we then have to identify the $x \in X$ on the boundary of any cell and investigate all PSCs derived by an application of one or more cyclical exchanges of these items to the original PSC.
With our polytopal approach, we not only obtain a direct criterion for the existence of multiple optimal PSCs by observing $\delta = 0$ for the linear program of Theorem 3.12 (see above), but we also have an alternate way to identify these PSCs by enumerating the vertices of $PP \cap H$ (see e.g. [AF92] for a short survey on vertex enumeration algorithms), where H is the hyperplane of the objective function vector with $v \in H$ for an optimal vertex $v \in PP$.
We close this section with a description of our approach for random vectors $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$. Calculating a fixed number of PSCs allowing full a-induced cell decompositions for different randomly chosen a and choosing the one with the highest δ-value out of a fixed number of these 'tries', we obtain a cell decomposition intuitively useful for our data classification purposes. Algorithm 6 describes these steps in pseudo-code.

Input : $d, k, n, \kappa_1, \ldots, \kappa_k \in \mathbb{N}$, $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, $t \in \mathbb{N}$
Output: Clustering $C := (C_1, \ldots, C_k)$ and Cell Decomposition $\mathcal{P} := (P_1, \ldots, P_k)$ with $C_i \subset \text{int}(P_i)$ for all $i \in \{1, \ldots, k\}$

$z := 0$;
$\delta := 0$;
while $\delta = 0$ *or* $z < t$ **do**
  $z = z + 1$;
  Choose $a(z) := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$;
  Calculate a vertex $v^* := v^*(z)$ of Q with $a^T v^* \geq a^T v$ for all $v \neq v^* \in Q$ and corresponding clustering $C(z)$ according to Theorem 3.11;
  Calculate a cell decomposition $\mathcal{P}(z)$ and corresponding $\delta(z)$ for the clustering associated with v^* according to Theorem 3.12;
  if $\delta(z) > \delta$ **then**
    $C := C(z)$;
    $\mathcal{P} := P(z)$;
  end
end
return C and $\mathcal{P}$;

Algorithm 6: Full a-induced Cell Decomposition

In the next section, we turn to some possible criteria and characterizations of 'good' cell decompositions for our data classification purposes.

3.5 Good Cell Decompositions

Up to this point, we described how to use a full cell decomposition for the basic algorithms in data classification and how to calculate a clustering and cell decomposition by identifying a vertex of the gravity polytope optimal with respect to some linear objective function. For the positions of the hyperplanes of this cell decomposition, we solve the LP constructed in the proof of Theorem 3.12:

$$
\begin{array}{rcll}
 & \max \delta & & \\
maxA_{ij} + \delta_{ij} & = & \gamma_{ij} & (i \neq j, i \leq k, j \leq k, l \leq n) \\
\gamma_{ij} & = & \gamma_{i1} - \gamma_{j1} & (i \leq k, j \leq k) \\
\frac{\delta_{ij}}{\|a_{ij}\|} & \geq & \delta & (i \neq j, i \leq k, j \leq k)
\end{array}
$$

In its objective function $\max \delta$, we maximize the minimal Euclidean distance δ_{ij} between a point in a cluster C_i and a separating hyperplane H_{ij}. The above is but one of many ideas of what could be a 'nice' property of a cell decomposition. In the following we discuss different ideas for what a 'good' cell decomposition could look like.

We start with a fixed vector of sites $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ and turn to the positioning of the hyperplanes of an a-induced cell decomposition. Assume we have a given k-clustering $C := (C_1, \ldots, C_k)$ of $X \subset \mathbb{R}^d$. Let the associated gravity vector $v := v(C)$ be a vertex of Q such that v is uniquely optimal in Q with respect to $a^T v$. By the choice of a, the directions of our a_κ-induced hyperplanes are fixed. We now evaluate different options for the construction of our cell decomposition with these hyperplane directions.

Intuitively, we want our hyperplanes to separate the clusters 'well', i.e. a cell decomposition is the better the further the points of the clusters are 'away' from the separating hyperplanes. There are many ideas as to how to measure this, yet they all have something in common: With v being a vertex uniquely optimal with respect to $a^T v$, C allows strict separation, and by this, the positioning of the cell decomposition hyperplanes can be adjusted (at least slightly) while still keeping the strict separation property. Any translation of a hyperplane H_{ij} directly affects both clusters C_i and C_j (and indirectly all others if we have a full

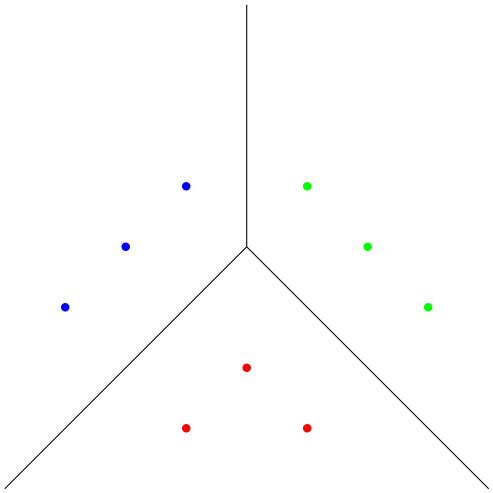
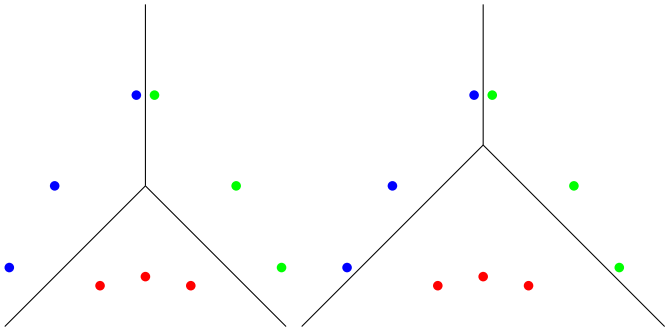


Figure 34: A cell decomposition optimal for all linear objective functions discussed in this section.

cell decomposition). When H_{ij} is moved further away from the points in C_i, it is simultaneously moved closer to the ones in C_j, and vice versa. Thus, for each hyperplane, we have to respect the distances to both its separated clusters.
We start by describing some straightforward ideas that can be realized in a linear program. First, consider the objective function described in Theorem 3.12. We maximize the minimal distance of the cluster points to the separating hyperplanes of their clusters.
Figure 34 shows an example where this works well. On the other hand, it is easy to find examples where this approach can lead to an undesirable cell decomposition. Suppose there are two clusters C_i, C_j with $x_i \in C_i$, $x_j \in C_j$ very close to each other. Then $\delta \leq \frac{1}{2}\|x_i - x_j\|$, and thus δ is forced to a very small value. With our objective function, the positions of the other separating hyperplanes then are pretty arbitrary. Figure 35 shows two cell decompositions yielding the same objective function value, yet, in many applications, we desire the positioning of the left one over the one to the right.
This raises two ideas about how to extend this first objective function: First, measuring distances to the points of the clusters may not be a good approach. As our clustering is fixed, we have no control about the absolute distances between points of the different clusters and thus using an objective function that is 'independent' from these distances may be preferable.



(a) The red-blue and red-green separating hyperplanes are equally far away from the closest points in the respective clusters.

(b) The red-blue and red-green separating hyperplanes are very close to the blue, respectively green cluster.

Figure 35: Two cell decompositions showing that only considering the minimal distance of a hyperplane to one of its respective clusters can yield an undesirable positioning of the hyperplanes.

Informally, we could ask for our hyperplanes H_{ij} to be 'in the middle' in between clusters C_i and C_j. We can account for this property by punishing the objective function value whenever a hyperplane is not exactly equally far away from the closest points of both clusters. To do so, we introduce a variable ν that is an upper bound on $\frac{\delta_{ij}-\delta_{ji}}{\|a_{ij}\|}$ and $\frac{\delta_{ji}-\delta_{ij}}{\|a_{ij}\|}$ and minimize it in the objective function. The optimal objective function value is achieved if ν is chosen as a tight bound. The corresponding linear program looks as follows:

$$
\begin{array}{rcll}
\min \nu & & & \\
max A_{ij} + \delta_{ij} & = & \gamma_{ij} & (i \neq j, i \leq k, j \leq k, l \leq n) \\
\gamma_{ij} & = & \gamma_{i1} - \gamma_{j1} & (i \leq k, j \leq k) \\
\nu & \geq & \frac{\delta_{ij}-\delta_{ji}}{\|a_{ij}\|} & (i \leq k, j \leq k) \\
\nu & \geq & \frac{\delta_{ji}-\delta_{ij}}{\|a_{ij}\|} & (i \leq k, j \leq k)
\end{array}
$$

Note that, in this model, we still measure the quality of separation achieved using an absolute (Euclidean) distance, but this distance does not (directly) depend on the absolute distance between points of the clusters. Figure 34 again is an example where this approach works well, but the example of Figure 35 $b)$ shows that we may want to consider not only a single (extremal) distance for a pair of clusters, but instead the positions of more or of all of the hyperplanes.

A viable approach would be to only look at the positions of the hyperplanes with non-empty intersections with their respective cells. The identification of this property requires additional computational effort. Conversely, as we have full cell decompositions, we may turn to subsystems of cells for our data classification algorithms, so all of our hyperplanes can be important. Due to this, we follow the latter case where we consider all hyperplanes.
For the sake of simplicity, we describe how to do this with respect to our original measure: For each hyperplane H_{ij}, its 'quality of separation' is tied to the lower one of the distances $\frac{\delta_{ij}}{\|a_{ij}\|}$ to the nearest point in C_i and $\frac{\delta_{ji}}{\|a_{ij}\|}$ to the nearest point in C_j. Within a linear model, we can now sum up these lower values for each combination of $i < j \in \{1, \dots, k\}$. The example of Figure 34 again is optimal with respect to this objective function, and thus again is an example where this approach works well. Constructing a linear program such that $d_{ij} := \min\{\frac{\delta_{ij}}{\|a_{ij}\|}, \frac{\delta_{ji}}{\|a_{ij}\|}\}$ for $i < j;\ i, j \in \{1, \dots, k\}$ and such that these d_{ij} are summed up in the objective function yields the following description:

$$
\begin{aligned}
\max \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} d_{ij} & & \\
maxA_{ij} + \delta_{ij} &= \gamma_{ij} & (i \neq j, i \leq k, j \leq k, l \leq n) \\
\gamma_{ij} &= \gamma_{i1} - \gamma_{j1} & (i \leq k, j \leq k) \\
d_{ij} &\leq \tfrac{\delta_{ij}}{\|a_{ij}\|} & (i < j \leq k) \\
d_{ij} &\leq \tfrac{\delta_{ji}}{\|a_{ij}\|} & (i < j \leq k)
\end{aligned}
$$

On the other hand, such an objective function again greatly favors a good positioning of hyperplanes where the respective clusters are far away from each other above the positioning of hyperplanes where $\frac{\delta_{ij}}{\|a_{ij}\|}$, $\frac{\delta_{ji}}{\|a_{ij}\|}$ cannot be very large due to the structure of the underlying clustering. It may be useful to turn to the relative positions of hyperplanes to fight this effect.
For this purpose, let

$$m_{ij} := |\min_{x \in C_j} a_{ij}^T x - \max_{x \in C_i} a_{ij}^T x| = |-maxA_{ji} - maxA_{ij}| = |maxA_{ij} + maxA_{ji}|.$$

We then identify $0 \leq \frac{\delta_{ij}}{m_{ij}} \leq 1$ as the relative distance of C_i to H_{ij} (see Figure 36). Summing up $\min\{\frac{\delta_{ij}}{m_{ij}}, \frac{\delta_{ji}}{m_{ij}}\}$ for all $i < j;\ i, j \in \{1, \dots, k\}$ yields the following linear program.

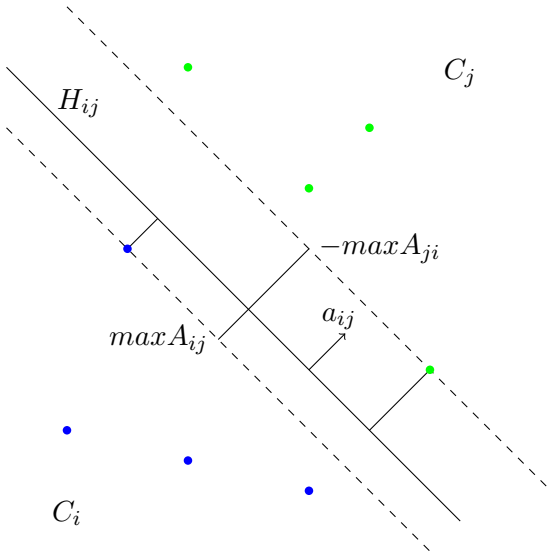


Figure 36: The construction of $d_{ij} = \min\{\frac{\delta_{ij}}{m_{ij}}, \frac{\delta_{ji}}{m_{ij}}\}$. In this example, $d_{ij} = \frac{1}{3}$.

$$
\begin{array}{rcll}
\max \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} d_{ij} & & & \\
maxA_{ij} + \delta_{ij} & = & \gamma_{ij} & (i \neq j, i \leq k, j \leq k, l \leq n) \\
\gamma_{ij} & = & \gamma_{i1} - \gamma_{j1} & (i \leq k, j \leq k) \\
d_{ij} & \leq & \frac{\delta_{ij}}{m_{ij}} & (i < j \leq k) \\
d_{ij} & \leq & \frac{\delta_{ji}}{m_{ij}} & (i < j \leq k)
\end{array}
$$

A combination of the measures described above in the objective function is a useful approach within the power of a linear model. For a given cell decomposition, we can measure its 'quality' with respect to several of the measures described by introducing a 'hierarchy' of the measures:
E.g we could combine the search for a big δ, and a punishing factor for hyperplanes not being in the middle in between their clusters. We use δ as our primary measure and subtract the secondary measure ν, divided by a very big factor, e.g. $10^{10} \cdot const$ if $const$ is the biggest absolute of a number appearing in the input, to respect the hierarchy. By this, we informally look for a cell decomposition for which the minimal distance between a hyperplane and one of its clusters is big and, among these, for one with central hyperplane positions whenever possible.

$$
\begin{array}{rcll}
\max \delta - \frac{1}{10^{10} \cdot const}\nu & & & \\
maxA_{ij} + \delta_{ij} & = & \gamma_{ij} & (i \neq j, i \leq k, j \leq k, l \leq n) \\
\gamma_{ij} & = & \gamma_{i1} - \gamma_{j1} & (i \leq k, j \leq k) \\
\frac{\delta_{ij}}{\|a_{ij}\|} & \geq & \delta & (i \neq j, i \leq k, j \leq k) \\
\nu & \geq & \frac{\delta_{ij} - \delta_{ji}}{\|a_{ij}\|} & (i \leq k, j \leq k) \\
\nu & \geq & \frac{\delta_{ji} - \delta_{ij}}{\|a_{ij}\|} & (i \leq k, j \leq k)
\end{array}
$$

In our models up to this point, we always used linear punishing factors for a non-central positioning of the hyperplanes between their clusters. In practice, it may be useful to use a more extreme, e.g. quadratic punishing factor, such that only hyperplanes that lie far from a central position contribute significantly. Such a quadratic objective function was introduced in this context in [BG09] under the name 'quadratic best fit':

$$
\begin{array}{rcll}
\min \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} (d_{ij} - \frac{1}{2})^2 & & & \\
maxA_{ij} + \delta_{ij} & = & \gamma_{ij} & (i \neq j, i \leq k, j \leq k, l \leq n) \\
\gamma_{ij} & = & \gamma_{i1} - \gamma_{j1} & (i \leq k, j \leq k) \\
d_{ij} & \leq & \frac{\delta_{ij}}{m_{ij}} & (i < j \leq k) \\
d_{ij} & \leq & \frac{\delta_{ji}}{m_{ij}} & (i < j \leq k)
\end{array}
$$

The fixed clustering being the basis for the calculation of a cell decomposition, we can try to improve the quality of a cell decomposition by choosing different clusterings as starting points. This can be done by a random choice of vectors a and the computation of the corresponding optimal vertices of Q (recall Theorem 3.11). Algorithm 6 describes this procedure for the original criterion, looking for a big δ. The resulting cell decompositions for these vertex clusterings can then be compared with respect to the above measures.

As we have seen, the sites $a \in \mathbb{R}^{d \cdot k}$ used for a full a-induced cell decomposition play a fundamental role in the 'quality' of the cell decomposition. In this section, up to now, we considered a fixed a, and only optimized the positioning of the hyperplanes separating the clusters according to the directions fixed by the a-induction. In general, with a fixed, so is the PSC $C := (C_1, \ldots, C_k)$ which allows a full a_κ-induced cell decomposition.

On the other hand, having a vertex $v := v(C)$ of Q means that any $a \in N(v)$ can be used for a full a_κ-induced cell decomposition. First, we note that we

can calculate such a vector a for a vertex v using a linear program. This linear program can also be interpreted as a corollary of Theorem 5 in [BHR92].

Lemma 3.13

Let $C := (C_1, \ldots, C_k)$ be a PSC and let $v := v(C)$ be a vertex in Q. A vector $a := (a_1, \ldots, a_k)^T \in \mathbb{R}^{d \cdot k}$ with $a^T v > a^T v'$ for any $v' \in Q \backslash \{v\}$ can be identified by solving a linear program.

Proof. We only look for any viable $a \in \mathbb{R}^{d \cdot k}$ and thus only have a feasibility problem. Recall the constraints of the problem for the computation of a cell decomposition from Theorem 3.12:

$$\begin{aligned} y_{il} \cdot ((\tfrac{a_j}{\kappa_j} - \tfrac{a_i}{\kappa_i})^T x_l) &\leq y_{il} \cdot \gamma_{ij} && (i \neq j, i \leq k, j \leq k, l \leq n) \\ \gamma_{ij} &= \gamma_{i1} - \gamma_{j1} && (i \leq k, j \leq k) \end{aligned}$$

Everything but the a_{ij} and γ_{ij} is fixed. We want strict separation, which we enforce by introducing a small $\epsilon > 0$ to make the first type of constraints 'strict'. We get our feasibility problem as

$$\begin{aligned} y_{il} \cdot ((\tfrac{a_j}{\kappa_j} - \tfrac{a_i}{\kappa_i})^T x_l) &\leq y_{il} \cdot ((1-\epsilon)\gamma_{ij}) && (i \neq j, i \leq k, j \leq k, l \leq n) \\ \gamma_{ij} &= \gamma_{i1} - \gamma_{j1} && (i \leq k, j \leq k) \end{aligned}$$

This proves the claim. □

Note that we have $d \cdot k + k \cdot k$ variables and a total of $k(k-1) \cdot n + k^2$ constraints in above LP. With some (linear-time) preprocessing to remove the inequalities that are trivially satisfied due to $y_{il} = 0$, we can cut the latter number to $(k-1) \cdot n + k^2$ constraints.

In general, if we calculate a vertex of Q using a nonlinear model, e.g. by using a quasi-convex program, we (still) need to identify a vector in its outer cone of normals before we can apply our construction of a cell decomposition. Lemma 3.13 shows that we can do so by solving a linear program. Only in special cases, like if we use some k-means variant to derive a vertex of some gravity polytope (see Theorem 3.2), we already know a corresponding vector in the outer cone of normals.

Additionally, we can use the feasibility problem in Lemma 3.13 as a basis for finding a 'good' vector for the separation directions of a fixed vertex clustering. If we calculate a vertex v to be optimal with respect to $a^T v$ for some random $a \in \mathbb{R}^{d \cdot k}$, a is just one of all the options we have from $N(v)$. If we add an objective function, we may actively search for a site vector of 'nice' properties.

Note that this is not as simple as just using the objective functions described in the beginning of this section: With $a_1, \ldots, a_k \in \mathbb{R}^{d \cdot k}$ being variables, in all models, we would obtain nonlinear constraints, due to the fractions using $\|a_{ij}\|$ or m_{ij} being put in relation to decision variables. This makes the corresponding problems difficult to solve for large k.

Thus, it is natural to take a close look at some explicit vectors $a \in N(v)$ satisfying interesting geometric properties, and to examine whether these geometric properties transfer to nice properties for the corresponding full a_κ-induced cell decomposition. We here give two examples that will be investigated in detail in upcoming work:

Let v be a vertex of Q, and let $b_1, \ldots, b_t \in \mathbb{R}^{d \cdot k}$ with $\|b_i\| = 1$ be the normals of all facets of Q containing v. Recall that Q is not full-dimensional (Lemma 2.17), which implies that we consider the facets of Q in the affine hull space of Q.

Let $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ with $\|a\| = 1$ be such that $\max_{i \in \{1,\ldots,t\}} a^T b_i$ is minimal. In a certain sense, such an a is maximally different from the vectors defining the cone of outer normals of v. Further, looking at $a := \sum_{i=1}^{t} b_i$, we get another interesting choice of a in the cone.

For both of these choices of a, we require an explicit description of the outer cone of normals. It remains to explore how to best obtain this description. Both the fact that the gravity polytope is derived from the partition polytope by a linear transformation (Lemma 3.10), as well as the knowledge about the edges of the gravity polytope presented towards the end of the next chapter may help with this.

In this section, up to now, we first considered cell decompositions with the separating hyperplane directions fixed, and then cell decompositions for a fixed vertex of Q. Both of these fixations have a great impact on the possible quality of the cell decomposition to be constructed.

It would be a great advantage to be able to leave this two-step approach of first calculating some arbitrary PSC, and then a good cell decomposition for this fixed clustering, for a combined optimization of both the clustering and the separating hyperplane directions and positions. A direct modeling approach for this yields not much hope for performing such a computation in practice. We use the notation of Theorems 3.11 and 3.12.

For the sake of simplicity, we only turn to the constraints of such a model. First, we need to make sure that each vertex in X is assigned to exactly one cluster, and that each cluster size is correct. We again obtain the constraints

$$\begin{array}{rcll} \sum_{j=1}^{n} y_{ij} & = & \kappa_i & (i \leq k) \\ \sum_{i=1}^{k} y_{ij} & = & 1 & (j \leq n) \\ y_{ij} & \in & \{0,1\} & (i \leq k, j \leq n) \end{array}$$

As both the assignments of vertices to clusters as well as the hyperplane directions and positions are variables to be optimized, we need to directly work with the constraints

$$\begin{array}{rcll} y_{il} \cdot (a_{ij}^T x_l) & \leq & y_{il} \cdot \gamma_{ij} & (i \neq j, i \leq k, j \leq k, l \leq n) \\ \gamma_{ij} & = & \gamma_{i1} - \gamma_{j1} & (i \leq k, j \leq k) \end{array}$$

The first type of constraints is problematic. Rephrasing it to

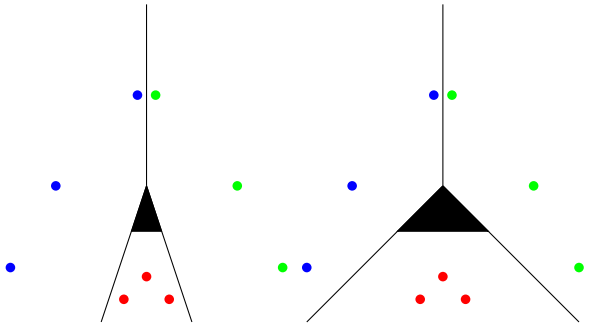
$$y_{il} \cdot (\frac{a_j}{\kappa_j}^T x_l - \frac{a_i}{\kappa_i}^T x_l - \gamma_{ij}) \leq 0,$$

we see that these are quadratic constraints and, without any further knowledge about the vectors a_i, in general the problem unfortunately is neither all convex nor all concave. Additionally, the y_{il} are $0,1$-decision variables. Due to this inherent difficulty, it is unlikely that it is possible to construct an efficient (exact) algorithm for the combination of the two steps.

We close this section by turning to the site vectors $a := (a_1^T, \ldots, a_k^T)^T \in \mathbb{R}^{d \cdot k}$ a final time, without the context of a PSC allowing a full a-induced cell decomposition.

Interpreting the directions of the separating hyperplanes as differentiation criteria, one may ask for these criteria to be 'maximally' different from each other. Consider the two cell decompositions in Figure 37. The cell decomposition in $a)$ has a small angle between the hyperplane separating the blue and red, and the red and green clusters. The cell decomposition in $b)$ is, by a pictographical impression, better with the lowest angle of two hyperplanes still being the one between the same hyperplanes as in Figure $a)$, but being a lot bigger.

If we are not restricted to some fixed set of sites, we could try to define our sites such that the minimal angle between two hyperplanes H_{li}, H_{ij} is large. Recalling



(a) A small angle between the blue-red and red-green separating hyperplanes.

(b) A big angle between the blue-red and red-green separating hyperplanes.

Figure 37: A big size of the smallest angle between some hyperplanes H_{li} and H_{ij} may be a criterion for a 'good' cell decomposition.

our a-induction, this is equivalent to asking for the biggest angle between some a_{li} and a_{ij} to be small. Figure 38 depicts this connection.

In this section, we discussed several ideas about what could make a cell decomposition desirable from a data classification point of view. We started by considering only the positioning of the hyperplanes for a fixed PSC and fixed separating hyperplanes. Then we turned to the more difficult search for useful separating hyperplane directions for a fixed PSC.

Both of these cases have in common that they are a two-step approach of first calculating a PSC, and then a good cell decomposition for this fixed clustering. We showed that a direct modeling approach falls short of being viable for a combined optimization of both the clustering and the separating hyperplane directions and positions. Finally, we considered the separating hyperplanes of a cell decomposition as differentiation criteria, and gave an example as to how this might affect our choice of the sites for the construction of a cell decomposition.

In the following, we consolidate our data classification approach by turning to the role that the geometric information of the data vectors and the fixed size restrictions take in our model.

3.6 Geometric Information and Size Restrictions

The data classification ideas described in this chapter use a set of geometric data vectors $X := \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ as basis. From the geometric information, we

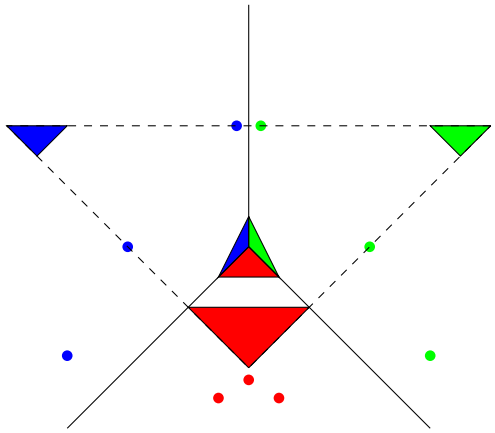


Figure 38: Big angles between hyperplanes H_{li} and H_{ij} correspond to small ones between a_{li} and a_{ij}.

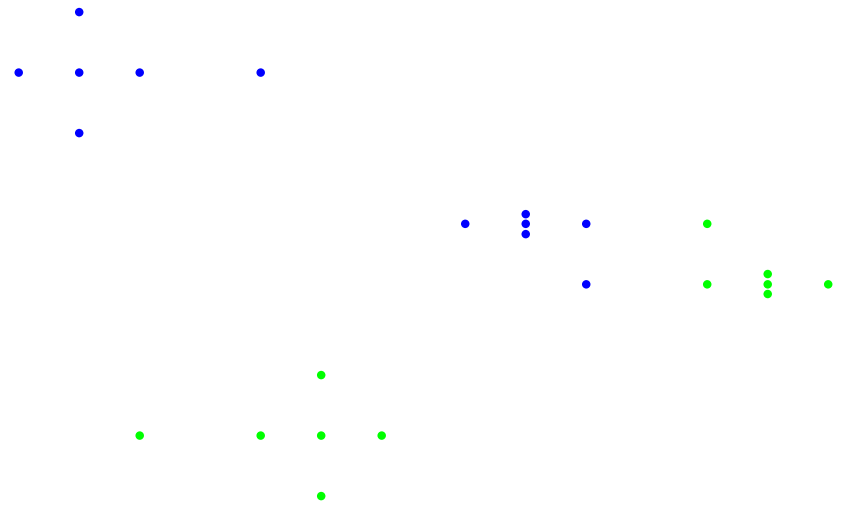
compute a cell decomposition of $\mathbb{R}^d$ which then is used for the algorithms for data classification and prediction described in Section 2.

Each data vector x_i identifies d features, which are encoded as the coefficients of $x_i \in \mathbb{R}^d$. This **feature representation** is central to our whole data classification approach.

If all of the features of the data vectors are of a common 'number type', or at least of similar ones, a mapping of these attributes to geometric coordinates often can be done easily. In many applications though, the types of the features of each data vector are very heterogenous. Some of them may be continuous values, like percentiles. Some of them may be discrete, but ordered, values that still can be mapped to geometric values, like the age of patients. Other attributes can be difficult to transfer, e.g. binary ones like the gender of a patient or an indicator whether a patient suffered from a specific disease before. In a 'worst case', a feature does not allow a (direct) geometric interpretation, like a categorization of objects according to different colors.

Additionally, we work in a Euclidean space $\mathbb{R}^d$, and thus we rely on the Euclidean distance as a valid **similarity measure** of the data vectors. Further, we interpret separating hyperplanes as differentiation criteria between clusters. If the structure of the underlying data is known, the use of kernel functions may help with deriving a geometric representation of X where linear separation is useful [SS02; Bor07].

Even if the features of the data vectors allow a geometric interpretation, a different scaling of the corresponding data ranges for the geometric representation



(a) An intuitive association of points with two clusters according to their Euclidean distance.

(b) A different data range scaling changes the relative distances of the points (and their association).

Figure 39: A different geometric representation of the same data set yields a different notion on which data entries are similar. The two representations only differ by the scaling of the second data vector entry.

of the data vectors will yield different cell decompositions that we use for our algorithms. Figure 39 shows how a different representation can artificially induce structure into a data set. Note that in some cases, this may be desired. Similarly to instance-level constraints, we can use this effect to 'guide' our algorithms to find PSCs and cell decompositions of some desired properties.

Outliers with respect to the rest of the data set will influence the centers of gravity of their corresponding clusters greatly. While the characterization of what makes a data vector an outlier in a given data set already is a difficult statistical question [Haw80], some way to identify them and 'deal' with them is necessary. For a survey of outlier detection methods, see [HA04].

Finally, real-world data recordings will often be incomplete, to a degree where our prediction algorithm may not be readily usable anymore. Finding a way to deal with this problem, as well, is an important step to the consolidation of our approach. All of the issues described are common for the application of clustering and data classification algorithms in practice, and will not be described here in further detail.

Another very important aspect is the choice of the number of clusters k, and of their sizes $\kappa_1, \ldots, \kappa_k$. The ideas described in this chapter were designed for the

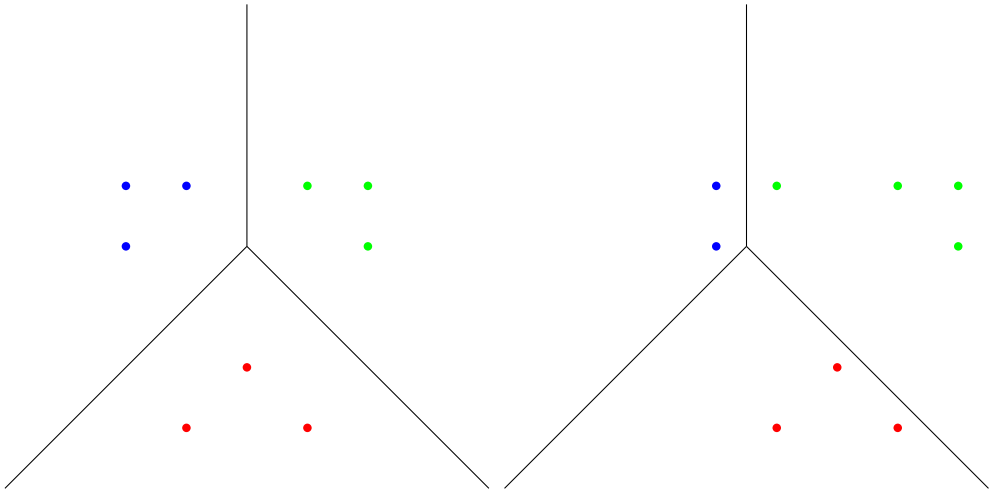
case where these parameters are given, and clearly they work best in this case. There are many applications for constrained clustering for which the parameters are given, but even many more where the number of clusters, and especially the cluster sizes, are not given explicitly. In the following, we discuss some ideas about how to apply our algorithms if we only have partial information about the parameters beforehand, and some problems that arise when doing so.

Our data classification algorithms (Algorithm 3 and 5) can be used for any clustering induced by a power diagram, respectively a full cell decomposition, and are not tied to any size restrictions, or the number of clusters. On the other hand, the calculation of a full a-induced cell decomposition (Algorithm 6) relies on this information. Dropping the size restrictions on the clusters and calculating a vertex of the corresponding polytope yields a clustering where all clusters lie in disjoint cones in the geometric space [BHR92]. In most data classification applications, such a structure of a clustering or cell decomposition is too restrictive.

Often, the choice of k is not problematic, as there is a lot of information on what could be a useful number of clusters to generate. When considering patient data in medicine, like in the example of dementia prediction, each cluster defines a center of gravity, interpreted as a 'virtual patient'. Clearly, we do not want too few of these, but not too many either. Additional information from past data analysis processes gives us a good estimate of what k could look like. With our data classification algorithms being efficient, and the calculation of a full a-induced cell decomposition (Algorithm 6) being possible by means of two linear programs, we have the opportunity of calculating cell decompositions for different values of k.

When choosing $\kappa_1, \ldots, \kappa_k$, we have to be especially careful. While we have some information about acceptable values, e.g. by wanting each 'virtual patient' to be constructed from not too few, and not too many data vectors, we restrict ourselves to specific values of sizes. This may influence our results greatly, as we can see in Figure 40, and thus it is very important to use a 'good' choice for $\kappa_1, \ldots, \kappa_k$. Suppose we only know about some lower bound κ on $\kappa_1, \ldots, \kappa_k$, and let k be fixed. This corresponds to the example where any virtual patient should be constructed from the data of at least κ patients.

By calculating a vertex of a gravity polytope Q, we derive a k-clustering which allows a full cell decomposition. Thus any subset of clusters still yields a cell decomposition, recall Lemma 2.38. For an index set $I \subset \{1, \ldots, k\}$, we obtain an



(a) With all clusters having exactly 3 points, we are able to find a desirable cell decomposition for some fixed hyperplane directions.

(b) With the blue cluster having only 2, and the green cluster having 4 vertices, we are not capable of finding a cell decomposition as 'good' as in a).

Figure 40: The fixed cluster sizes have a great impact on the obtainable 'quality' of a cell decomposition.

$|I|$-cell decomposition which induces an $|I|$-clustering canonically by the data vectors being in the respective cells of the subsystem.
For the sake of simplicity, and without loss of generality, let $\frac{n}{\kappa} = k'$ be integral. We have $k' \geq k$ and in non-trivial cases $k' > k$. Let us now calculate a full a-induced k'-cell decomposition of $\mathbb{R}^d$ like described in Algorithm 6, using cluster sizes equal to κ.
As a byproduct of this calculation, we obtain $\binom{k'}{|I|}$ $|I|$-cell decompositions for index sets $I := \{i_1, \ldots, i_t\} \subset \{1, \ldots, k'\}$. For all of these, the respective numbers $\kappa^I_{i_1}, \ldots, \kappa^I_{i_t}$ satisfy $\kappa^I_i \geq \kappa$ for all $i \in I$. The membership of data vectors with respect to a subsystem of indices $|I|$ is easily calculated: First, if $x \in C_i$ in the original cell decomposition, and $i \in I$, then $x \in C^I_i$. Otherwise, the cell of an x can be calculated using Algorithm 3.
Thus, with each randomized algorithm step, we obtain $\binom{k'}{k}$ cell decompositions that have exactly k clusters and satisfy the given lower bound on the cluster sizes. We can use one of the measures described in the last section to identify their 'quality', and choose one of these cell decompositions. Using this approach, we are capable of guaranteeing lower bounds on the cluster sizes.

We close this section with a comment on the case where we have no knowledge about $\kappa_1, \ldots, \kappa_k$ at all. We might combine the classical k-means algorithm and our polytopal approach to find a good cell decomposition. Performing the k-means algorithm as denoted in Algorithm 2 yields k non-empty clusters $C_1, \ldots, C_k$ of sizes $\kappa_1 := |C_1|, \ldots, \kappa_k := |C_k|$, and a vertex v of the corresponding gravity polytope $Q = Q(X, k; \kappa_1, \ldots, \kappa_k)$, recall Theorem 3.2. We can then measure the 'quality' of the a-induced cell decomposition belonging to this solution, where $a := (c_1^T, \ldots, c_k^T)^T$. With the ideas of the last section, we can then try to construct another vector a' in $N(v)$ to obtain a 'better' cell decomposition for the same clustering, or relate it to the cell decompositions constructed for other vertices in Q.

3.7 Summary and Outlook

In this chapter, we described how to use the results of our polytopal studies of the gravity polytope in Chapter 2 for data classification. The clusterings belonging to vertices of the gravity polytope satisfy a number of geometric properties which make them desirable from a data classification point of view.

Most importantly, they allow cell decompositions such that each cell contains the data vectors of exactly one cluster. We showed how such cell decompositions can be used for efficient data classification and prediction algorithms. Further, we related our approach to existing data classification methods by showing that the classical k-means algorithm, in each step, calculates a clustering which is a vertex of the gravity polytope corresponding to the induced cluster sizes.

In fact, the clusterings of vertices of a gravity polytope allow special cell decompositions, namely full a-induced cell decompositions. We described some advantages of having a full cell decomposition instead of only a cell decomposition for the stability of our algorithms. In future work, we will investigate how to further increase the stability of our prediction value by using a box or cylinder instead of a line segment in Algorithm 5.

We showed that the calculation of a vertex of the gravity polytope can be done by means of a linear program, as the constraints that have to be satisfied form a special transportation polytope, the partition polytope. Similarly, the calculation of the hyperplane positions of a corresponding full a-induced cell decomposition can be done by solving a linear program.

The question as to what makes a cell decomposition a 'good' one for our data classification purposes is difficult. We gave some absolute and relative measures

obtainable by linear programs and turned to some ideas as to how to extend these measuring ideas by nonlinear programming. In this context, we will investigate the geometric meaning and possible advantage of using vectors that, in a certain sense, lie in the 'center' of the cone of outer normals of a vertex for the construction of the cell decompositions for our algorithms.
We will also turn to the construction of site vectors yielding big angles between the separating hyperplane directions. With the interpretation of these directions as differentiation criteria, we hereby obtain cell decompositions where pairs of clusters are separated by 'maximally' different criteria.
For the sake of simplicity and efficient algorithms, we chose a two-step approach of first identifying a clustering allowing a cell decomposition, and then finding a good cell decomposition for it. It seems to be difficult to deterministically find a clustering that allows a cell decomposition that is optimal with respect to some measure. An investigation as to whether our two-step approach can be unified to an efficient one-step approximation algorithm with a good approximation factor with respect to any of our measures is of high interest.
All of the geometric interpretation we give and use in this chapter heavily relies on the representation of the features of our data vectors as a set $X \subset \mathbb{R}^d$. A different geometric representation will result in a different structure of the point set, and thus in very different clusterings and cell decompositions. We have to be careful about an induction of artificial structure into a data set.
On the other hand, with our purely geometric approach, the possibility of intentionally inducing geometric structure into a data set might help us to 'guide' our algorithms to find clusterings and cell decompositions of desired properties, similarly to the use of instance-level constraints. Some of these possibilities will be explored in future work.
Finally, our approach is designed for the case that we have fixed values for the number of clusters and the sizes of the clusters. As there are many applications in which the cluster sizes are not given explicitly, we closed this chapter by discussing some ideas about how to handle this case.
Many of the theoretical results of Chapter 2, as well as the ideas about data classification presented in this chapter, can be transferred to clustering with different types of size restrictions. This will be discussed in upcoming work.

4 Edge-Structure of the Partition and Gravity Polytope

In many applications, not only the PSC optimal with respect to some objective function is of interest, but so are its relation to other PSCs, and the means by which it is derived.
We noted that two PSCs differ by a set of cyclical exchanges that share no items (Lemma 2.11). The special structure of the partition polytope, being a transportation polytope, implies that we are able to associate its edges with such cyclical exchanges between PSCs. The calculation of a vertex of the gravity polytope can be done by an application of the Simplex algorithm for the system of constraints defining the partition polytope (Theorem 3.11). This allows us to interpret an edge-walk (determined by the Simplex algorithm) as a sequence of applications of cyclical exchanges to PSCs. From this, the desire for a deeper understanding of the edge-structure of the partition polytope and the gravity polytope arises naturally.
While the edge-structure of the gravity polytope will prove tough to analyze, we will be able to derive a tight bound on the combinatorial diameter of the partition polytope (in a worst case). We first turn to the partition polytope.

4.1 Skeleton of the Partition Polytope

We begin with the terminology required for our purposes. For some basic notation and a survey of the elementary graph theoretic arguments we use in this chapter, see e.g. [Die06]. When talking about a graph with no further information, we consider an undirected graph without loops and without multiple edges.
Further, we refer the reader to the **Notation and Symbols** appendix for a list of the symbols used in Chapter 2, 3 and in this chapter. In the following, we will also often use the abbreviation PP introduced in Definition 3.9, if the parameters of the considered polytope are clear from the context.
The partition polytope is a special transportation polytope, recall the remark after Lemma 3.10. Transportation polytopes are well-studied objects in operations research and statistics, see e.g. [KW68; KYK84; Loe06; LKOS09] and references therein. We require only some basic facts about its polytopal structure in the following, established in [KW68]. As a service to the reader, we summarize these results.
For the remainder of this chapter, let $X := \{x_1, \ldots, x_n\}$ be a set of (non-geometric) items, and consider a PSC of X into k clusters $C := (C_1, \ldots, C_k)$ of sizes

$\kappa_1, \ldots, \kappa_k$. Let further in the following $K_{k,n}(C, X)$ be the complete bipartite graph on two sets of (graph) vertices $C := \{c_1, \ldots, c_k\}$ and $X := \{x_1, \ldots, x_n\}$. The ambiguous use of C and the c_i, as well as X and the x_i is intentional: In the following, we assume that vertex c_i corresponds to cluster C_i and vertex x_i to the equivalently-named item. When C and X are clear from the context, we use the shorter notation $K_{k,n}$ for $K_{k,n}(C, X)$.
We first turn to a classical graph-theoretical representation of PSCs.

Lemma 4.1
There is a set $\mathcal{E}$ of subsets of edges in $K_{k,n}(C, X)$ such that a PSC $C := (C_1, \ldots, C_k)$ of $X := \{x_1, \ldots, x_n\}$ corresponds bijectively to an $E \in \mathcal{E}$.

Proof. We obtain the desired one-to-one correspondence by creating a subset E of edges by

$$\{c_i, x_j\} \in E \Leftrightarrow x_j \in C_i$$ □

By the construction in Lemma 4.1, we obtain exactly one edge for each vertex x_j, yielding $|E| = n$. Each c_i is connected to κ_i vertices in X. Each x_j is connected to exactly one c_i. We denote this edge-subset E of $K_{k,n}$ as follows

Definition 4.2 (Assignment of X to C)
*Let $C := \{c_1, \ldots, c_k\}$, $X := \{x_1, \ldots, x_n\}$ and $\kappa_1, \ldots, \kappa_k \in \mathbb{N}$. We call a set of edges $E \subset E(K_{k,n}(C, X))$ a $(\kappa_1, \ldots, \kappa_k)$-**assignment of X to C** if it satisfies the following criteria:*

- $deg(x_j)_{|E} = 1$ *for all* $j \in \{1, \ldots, n\}$
- $deg(c_i)_{|E} = \kappa_i$ *for all* $i \in \{1, \ldots, k\}$

Here $deg(y)_{|E}$ denotes the degree of $y \in C \cup X$ with respect to edge-set E. If $\kappa_1, \ldots, \kappa_k$ are clear from the context, we also use the term **assignment of X to C**. Whenever we talk about an **assignment** in the following, we think of the corresponding assignment of X to C as a subset of $K_{k,n}(C, X)$, if the context is clear. Figure 41 shows an assignment of 5 items to 3 clusters.
By Definition 3.9 and the fact that PP is a $0, 1$-polytope with a totally unimodular constraint matrix, PSCs bijectively correspond to vertices of the associated partition polytope. We obtain the following immediate corollary.

Corollary 4.3
A vertex of the partition polytope corresponds bijectively to a $(\kappa_1, \ldots, \kappa_k)$-assignment of $X := (x_1, \ldots, x_n)$ to $C := (c_1, \ldots, c_k)$.

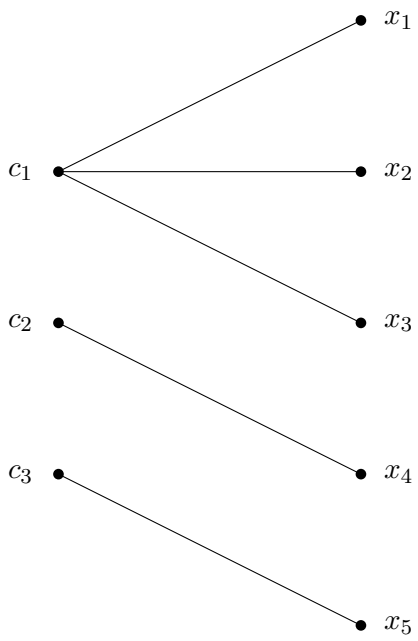


Figure 41: A $(3, 1, 1)$-assignment of 5 items to 3 clusters.

Next, we turn to the edge-structure of the partition polytope. By Lemma 2.11, two PSCs differ by a set of cyclical exchanges $\mathcal{CE}$ that do not move any item twice. It is easy to see that a cyclical exchange corresponds to a cycle in $K_{k,n}(C, X)$ and that $\mathcal{CE}$ is a set of edge-disjoint cycles such that cycles may only share vertices in C. We use the notation $E \Delta E' := (E \cup E') \backslash (E \cap E')$ for the **symmetric difference** of assignments E and E'.

Lemma 4.4
Let $C := (C_1, \ldots, C_k)$ be a PSC, and let $C' := (C'_1, \ldots, C'_k)$ be derived from C by an application of a cyclical exchange $CE := (x_{i_1}, \ldots, x_{i_t})$, where $x_{i_j} \in C_{i_j}$ for all $j \in \{1, \ldots, t\}$. Then CE corresponds bijectively to a cycle in $E \Delta E'$, where E and E' are the assignments corresponding to C, respectively C'.

Proof. E and E' differ only by the edges corresponding to CE. These clearly form a cycle in $K_{k,n}$, as $\{c_{i_j}, x_{i_j}\} \in E \backslash E'$ and $\{x_{i_j}, c_{i_{j+1}}\} \in E' \backslash E$ for all $j \in \{1, \ldots, t\}$. Such a cycle identifies both a cyclical exchange to derive C' from C, as well as the inverse one to derive C from C'. □

Knowing this, we can transfer Lemma 2.11 directly to the assignments in $K_{k,n}$

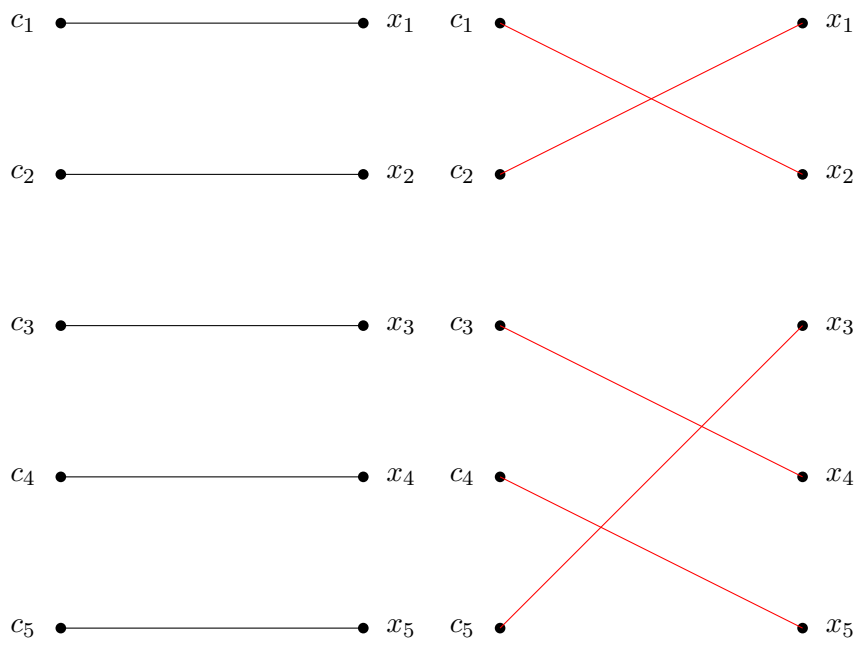
Lemma 4.5
Let C and C' be two PSCs, and let E and E' be the corresponding assignments in $K_{k,n}$. Then $E\Delta E'$ is a set of edge-disjoint cycles $\mathcal{CY} = \{CY_1, \dots, CY_r\}$ that may only share vertices in C.

Proof. By Lemma 4.4 and Lemma 2.11, $E\Delta E'$ is a set of cycles in $K_{k,n}$. The rest of the claim follows directly from $deg(x_j)_{|E} = deg(x_j)_{|E'} = 1$ for all $j \in \{1, \dots, n\}$.□

Note that this also implies that we can use the greedy construction of cyclical exchanges described in Lemma 2.11 to obtain corresponding cycles in $K_{k,n}$. Figure 42 shows two assignments E and E' and their corresponding symmetric difference $E\Delta E'$.
We are now able to identify the edge-structure of the partition polytope.

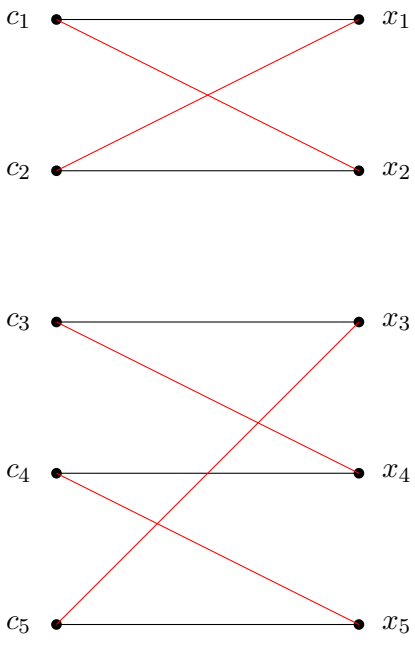
Lemma 4.6
Let C and C' be two PSCs, let v, v' be the corresponding vertices in PP and let E, E' be the corresponding assignments. Then v and v' are connected by an edge in PP if and only if $E\Delta E'$ contains exactly one cycle.

Proof. To show that two vertices v and v' in PP are connected by an edge, it suffices to find a hyperplane touching the polytope in v and v', but 'cutting off' all other vertices strictly.
To do so, we introduce a labeling function $f : E(K_{k,n}) \to \{0, 1\}$ by setting $f(e) = 0$ for $e \in E \cup E'$, and $f(e) = 1$ otherwise. We obtain $f(E) = f(E') = 0$. By Lemma 4.5, assignments differ by cycles.
Suppose E and E' differ by only one cycle, and suppose we have another $v'' \in PP$ with $f(E'') = 0$, $v'' \neq v$ and $v'' \neq v'$. By Corollary 4.3, $E \neq E'' \neq E'$, and by $f(E'') = 0$, we know that $E'' \subset E \cup E'$. As $E'' \neq E$, we have an edge $e = \{c_i, v_j\} \in E''\backslash E$. Then $e \in E'$, as $E'' \subset E \cup E'$, implying that e lies in the single cycle by which E and E' differ. Using the same argument for all $e \in E''\backslash E$ yields $E = E'$, in contradiction to $v \neq v'$.
Conversely, let now $E \cup E'$ contain at least two cycles CY_1 and CY_2. Suppose we have a labeling function $f' : E(K_{k,n}) \to \mathbb{R}$ such that $f'(E) = f'(E') > f'(E'')$ for all $v'' \in PP$ with $v \neq v'' \neq v'$. Note that this implies $f'(E \cap CY_i) = f'(E' \cap CY_i)$ for $i \in \{1, 2\}$.
If we choose v'' such that $e \in E \cap E' \Rightarrow e \in E''$, and $E'' \cap CY_1 = E \cap CY_1$, $E'' \cap CY_2 = E' \cap CY_2$, then $f'(E'') = f'(E) = f'(E')$, a contradiction. This proves that v and v' are not connected by an edge in PP. □



(a) An assignment E in $K_{5,5}$.

(b) An assignment E' in $K_{5,5}$.



(c) $(E \cup E')\backslash(E \cap E')$.

Figure 42: Two assignments E, E' of 5 items to 5 clusters, and their symmetric difference $E \Delta E'$.

Note that $E \cup E'$ contains exactly one cycle if and only if $E \Delta E'$ contains exactly one cycle, as such a cycle cannot lie in $E \cap E'$.
With the above lemmata, we know both the vertex-structure, as well as the edge-structure of the partition polytope: By Corollary 4.3, each assignment is a vertex in PP (and vice versa), and two vertices in PP are connected by an edge if and only if the assignments differ by exactly one cycle, by Lemma 4.6, respectively if the associated PSCs differ by exactly one cyclical exchange, by Lemma 4.4. With this correspondence, we will also talk about an application of cyclical exchanges to vertices of PP or assignments in the following.
We now are ready to investigate the combinatorial diameter of the partition polytope. First, we give a formal definition of the skeleton and the combinatorial diameter of a polytope.

Definition 4.7 (Skeleton of a Polytope)
Let $P \subset \mathbb{R}^d$ be a polytope defined as the convex hull of vectors $v_1, \dots, v_n \in \mathbb{R}^d$. We call the graph $G(P) := (V, E)$ derived from P by $V := \{v_1, \dots, v_n\}$ and E containing an edge between $v_i, v_j \in V$ if and only if v_i and v_j are connected by an edge in P the **skeleton of** *P.*

In the literature the above graph often is called the 1**-skeleton of** P. There are other ways of creating a skeleton-structure for a polytope, from its higher-dimensional faces.

Definition 4.8 (Diameter)
Let $G = (V, E)$ be a graph with $V = \{v_1, \dots, v_n\}$ and let l_{ij} denote the length of a shortest path from v_i to v_j in G. The **diameter of** *G then is defined as*

$$\max_{i \neq j;\ i,j \in \{1,\dots,n\}} l_{ij}.$$

The **(combinatorial) diameter of a polytope** *P is the diameter of the corresponding skeleton $G(P)$.*

Let $G = (V, E)$ be a graph. We call the length l_{ij} of a shortest path from v_i to v_j the **(combinatorial diametral) distance** of v_i and v_j. We use the same term for two vertices of a polytope P if $G = G(P)$ is the corresponding skeleton.
With these definitions, we can now state the main result of this chapter.

Theorem 4.9
$PP(k, \kappa_1, \dots, \kappa_k)$ has a diameter of at most

$$\min \left\{ \kappa_{i_1} + \kappa_{i_2}, \lfloor \frac{n}{2} \rfloor \right\},$$

where $i_1 := \arg\max_{i\in\{1,\dots,k\}} \kappa_i$ *and* $i_2 := \arg\max_{i\in\{1,\dots,k\}\setminus\{i_1\}} \kappa_i$.

Informally, the first part of the bound in Theorem 4.9 is the sum of the sizes of the two biggest clusters.

For the proof of this claim in this chapter, we introduce a graph representing the difference of two PSCs. We derive it from the classical representation of PSCs as assignments in $K_{k,n}$ (see Lemma 4.1) as follows.

Consider two PSCs C and C' of $X := \{x_1, \dots, x_n\}$ and their assignments E and E', and $E\Delta E'$. Whenever there is an edge $\{c_i, x_l\} \in E\setminus E'$, there is an edge $\{x_l, c_j\} \in E'\setminus E$ for some $j \neq i$.

We construct a labeled digraph $G = (V, E'', w)$ with vertex set $V \subset \{c_1, \dots, c_k\}$, edge-set E'' and a labeling function $w : E'' \to X$. We start with $V := \emptyset$ and $E'' := \emptyset$.

Then E'' and V are constructed using a single rule:

$$\{c_i, x_l\} \in E\setminus E' \wedge \{x_l, c_j\} \in E'\setminus E \Rightarrow V := V \cup \{c_i, c_j\} \wedge E'' := E'' \cup \{(c_i, c_j)\}.$$

The labeling function w then is defined for such an edge $e := (c_i, c_j)$ by $w(e) = x_l$. Note that we may derive multiple edges of type (c_i, c_j) (with different labels) like this, yielding a multigraph.

Informally, this construction can be interpreted as moving an item x_l from cluster C_i to C_j, whenever there is an edge $e = (c_i, c_j)$ with label $w(e) = x_l$ between the corresponding vertices.

Let us assign a formal name to this construction.

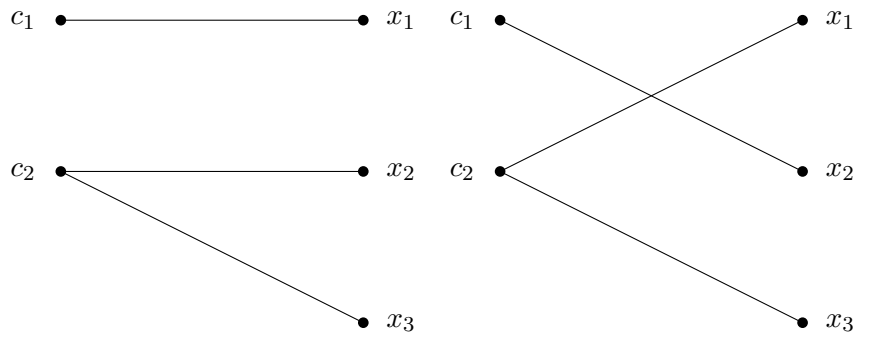
Definition 4.10 (Clustering Difference Graph (CDG))

Let C and C' be two PSCs, and E and E' be the corresponding assignments in $K_{k,n}$. We call the labeled digraph $G = (V, E'', w)$ constructed as explained above the **clustering difference graph** $CDG = CDG(C, C')$ **from** C **to** C' *(or of C and C').*

With vertices of PP being in one-to-one correspondence with PSCs, we will also use the notation $CDG(v, v')$ if v is the vertex of C and v' the vertex of C' in PP, or $CDG(E, E')$ for the corresponding assignments. Note that $CDG(C', C)$ can be derived from $CDG(C, C')$ by switching the direction of all edges in E''.

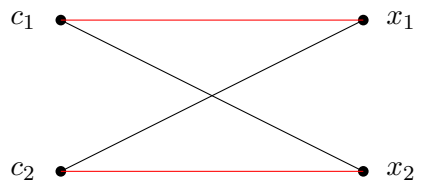
Figure 43 shows an example for the construction of a CDG. The figure shows an auxiliary step where the edges of $E\Delta E'$ are oriented in the 'direction' of the item movement desired. We informally say that they are **oriented from** E **to** E'.

With $E\Delta E'$ being the symmetric difference between the PSCs C and C', by Lemma 2.11, Lemma 4.4 and Lemma 4.5, the cycles in G describe the cyclical

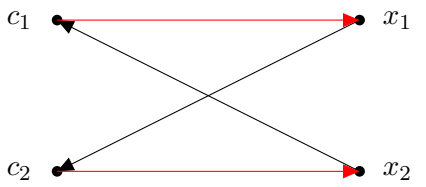


(a) Assignment E.

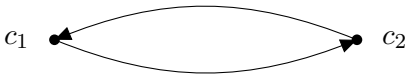
(b) Assignment E'.



(c) $E \Delta E'$. The edges of E are red.



(d) $E \Delta E'$ with edges oriented from E to E'.



(e) $CDG(E, E')$

Figure 43: The construction of a CDG from assignment E to assignment E'.

exchanges required to derive C' from C. By this, we obtain an immediate corollary.

Corollary 4.11
A CDG decomposes into cycles. Each of these cycles corresponds bijectively to a cyclical exchange.

Proof. Let C, C' be two PSCs with corresponding assignments E and E'. They differ by a set of cyclical exchanges, by Lemma 2.11. These cyclical exchanges correspond directly to the cycles in $E \Delta E'$, by Lemma 4.4. Thus $E \Delta E'$ decomposes into cycles.
It remains to prove that each cycle in $E \Delta E'$ corresponds bijectively to a cycle in $CDG(C, C') = (V, E'', w)$. By construction, for any $e = (c_i, c_j) \in E''$ with $w(e) = x_l$, there are two edges $\{c_i, x_l\} \in E \backslash E'$ and $\{x_l, c_j\} \in E' \backslash E$, and vice versa. Further, the cycles in $E \Delta E'$ do not share any items in X, by Lemma 4.5. This proves the claim. □

Analogously to the remark after Lemma 2.11, note that a cycle decomposition of a CDG is not necessarily uniquely defined, not even the number of cycles of a decomposition is fixed. Such a set of cyclical exchanges can be identified greedily, following the proof of Lemma 2.11.
Note further that there is an inherent relationship between clustering difference graphs and clustering graphs (see Definition 2.31). If two clusterings C, C' belong to neighbouring vertices v, v' in PP, they differ by a single cyclical exchange, by Lemma 4.6. Then the CDG from C to C' contains a single cycle, and its edges yield a subgraph of the clustering graph for C.
Informally, if we do not consider the labeling function w, a CDG for PSCs C and C' identifies how many items need to be changed from which cluster to which cluster to derive C' from C. The only information lost is which items have to be changed. Due to this loss of information, without w, the same CDG can correspond to multiple pairs of PSCs or vertices of PP. On the other hand, the property of two PSCs belonging to neighbouring vertices in PP can be investigated without considering w.

Corollary 4.12
Let C and C' be two PSCs, and let v, v' be the corresponding vertices in PP. Then v and v' are connected by an edge in PP if and only if $CDG(C, C')$ contains exactly one cycle.

Proof. Let E, E' be the assignments corresponding to C, C'. The claim is a direct corollary of Lemma 4.6 and the one-to-one association of cycles in $CDG(C, C')$ and $E \Delta E'$ (Lemma 4.4, Corollary 4.11). □

In the next section, we turn to a graph theoretical question. Its solution is our first step to an investigation of the diameter of the skeleton of the partition polytope. We will be able to apply the results to our CDGs.

4.2 Vertex-Disjoint Cycle Covering

Covering graphs with cycles is a well-studied problem in the literature, for a survey see e.g. [Zha97]. Common questions involve finding the minimal number of edge-disjoint or vertex-disjoint cycles required to cover graphs or subgraphs of specific properties [IJ03]. We here use the following understanding of the term 'cover'.

Definition 4.13 ((Edge) Cover)
Let $G = (V, E)$ be a graph. A set of edges $E' \subset E$ is an **(edge) cover** *of V if for all $v \in V$ there is an edge $e \in E'$ such that v is incident to e.*

Informally, we also say that E **covers** V or that E **is covering** V. The term is transferred canonically to subsets of vertices $V' \subset V$ of a graph.
In this section, we investigate the existence of a set of vertex-disjoint cycles covering a given set of vertices in a digraph. We state the problem as follows.

Problem 4.14 (Vertex-Disjoint Cycle Covering)
Let $G = (V, E)$ be a digraph (not necessarily without loops or without multiple edges) and let $M \subset V$.
Question: *Is there a set of (pairwise) vertex-disjoint cycles in E covering M?*

For a certain class of graphs, including clustering difference graphs, and a certain kind of vertex sets M, we can answer the question posed positively. In the following, we assume our graphs to have at least two vertices.

Theorem 4.15
Let $G = (V, E)$ be a digraph (not necessarily without loops or multiple edges) with $deg_-(v) = deg_+(v) \geq 1$ for all $v \in V$. Let $D_{max} = \max_{v \in V} deg_-(v)$ and let $M \subset V$ with $\sum_{v \in M} deg_-(v) \geq (|M| - 1)D_{max} + 1$. Then M can be covered by a set of vertex-disjoint cycles.

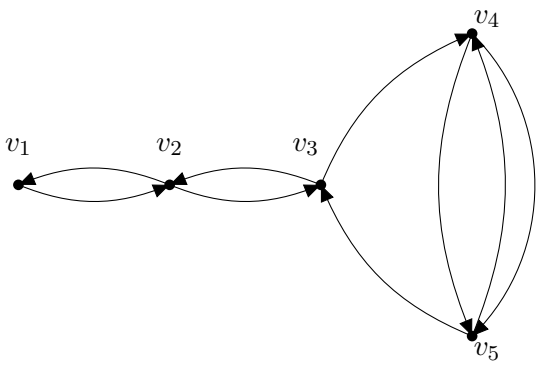
Here $deg_-(v)$ and $deg_+(v)$ denote the indegree and outdegree of a vertex v. Note that any CDG satisfies $deg_-(v) = deg_+(v) \geq 1$ for all $v \in V$ by Corollary 4.11, and the fact that the CDG only contains vertices for which there is at least one edge.
In the next section, we will only need the following immediate corollary.

Corollary 4.16
Let $G = (V, E)$ be a digraph (not necessarily without loops or multiple edges) with $deg_-(v) = deg_+(v) \geq 1$ for all $v \in V$. Then the set M of vertices of maximal $deg_-(v)$ can be covered by a set of vertex-disjoint cycles.

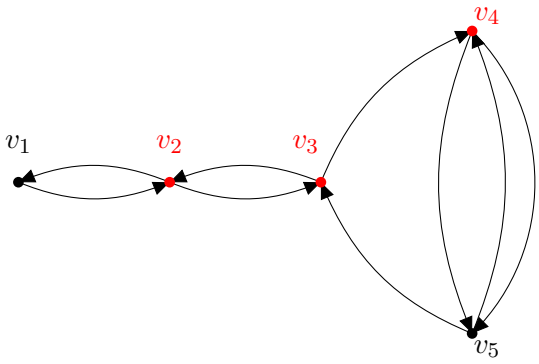
Proof. The set M of vertices of maximal $deg_-(v)$ satisfies $\sum_{v \in M} deg_-(v) = |M| D_{max}$ $\geq (|M| - 1)D_{max} + 1$. The claim then follows directly from Theorem 4.15. □

The remainder of this section is dedicated to proving Theorem 4.15. Suppose $G = (V, E)$ is a digraph (not necessarily without loops or multiple edges) with $deg_-(v) = deg_+(v) \geq 1$ for all $v \in V$, and let $M \subset V$ with $\sum_{v \in M} deg_-(v) \geq (|M| - 1)D_{max} + 1$.
Figure 44 shows a graph $G = (V, E)$ and a set $M \subset V$ to be covered. Clearly, both G and M satisfy the required prerequisites: G is connected and $deg_-(v) = deg_+(v)$ for all $v \in V$, and M consists of three (of the four) vertices of maximal degree.
With G satisfying $deg_-(v) = deg_+(v)$ for all $v \in V$, we know that G decomposes into (edge-disjoint) cycles, as removing the edges of any cycle in G yields a subgraph $H = (V, E')$ satisfying $deg_-(v)_{|E'} = deg_+(v)_{|E'}$ for all $v \in V$, as well. By this, G clearly has a cover by edge-disjoint cycles, and if it is connected, even a Eulerian cycle, see [Die06].
To identify a cover by vertex-disjoint cycles, we have to take a different approach. We start by constructing a network $G' := (V', E', s, t)$ from $G = (V, E)$ using the following rules:

- Split up all vertices $v \in V$ into two vertices v_{in} and v_{out}. For each edge $(v, w) \in E$, add an edge (v_{out}, w_{in}) to E'.
- For each $v \in V \backslash M$, add (v_{in}, v_{out}) to E'.
- Add a start vertex s and a target vertex t to V', and add $deg_-(v)$ edges (s, v_{out}) to E' for all $v \in M$ and $deg_+(v)$ edges (v_{in}, t) to E' for all $v \in M$.



(a) A connected graph $G = (V, E)$ with $deg_-(v) = deg_+(v)$ for all $v \in V$.



(b) The set $M = \{v_2, v_3, v_4\}$ of vertices to be covered.

Figure 44: A graph $G = (V, E)$ and $M \subset V$ satisfying the prerequisites of Theorem 4.15.

Note that we do not consider a capacity measure for the edges in E'. Figure 45 $a)$ shows the network G' for the graph G of Figure 44. We avoid using double subscripts by writing i_{in} and i_{out} instead of $v_{i_{in}}$ and $v_{i_{out}}$.
We obtain an interesting lower bound on the size of the maximal flow in G'.

Lemma 4.17
$G' := (V', E', s, t)$ has a maximal flow f of size at least $(|M| - 1)D_{max} + 1$.

Proof. By construction, we have $deg_-(v) = deg_+(v)$ for all $v \in V'\backslash\{s, t\}$ and $deg_+(s) = deg_-(t) \geq (|M| - 1)D_{max} + 1$. Any (s, t)-cut in G' then is at least of size $(|M| - 1)D_{max} + 1$, and we get the claim by the well-known max flow-min cut Theorem [FF56]. □

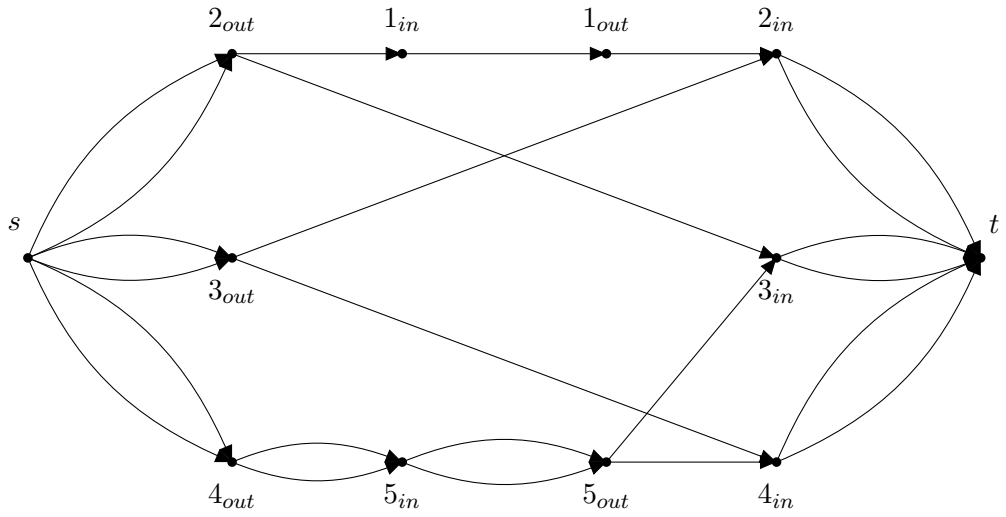
We continue by constructing another network $G'' := (V', E'', s, t)$ from $G' = (V', E', s, t)$ using a single rule:

- Multiple edges of type $(s, v_{out}), (v_{in}, v_{out}), (v_{in}, t)$ are replaced by a single edge.

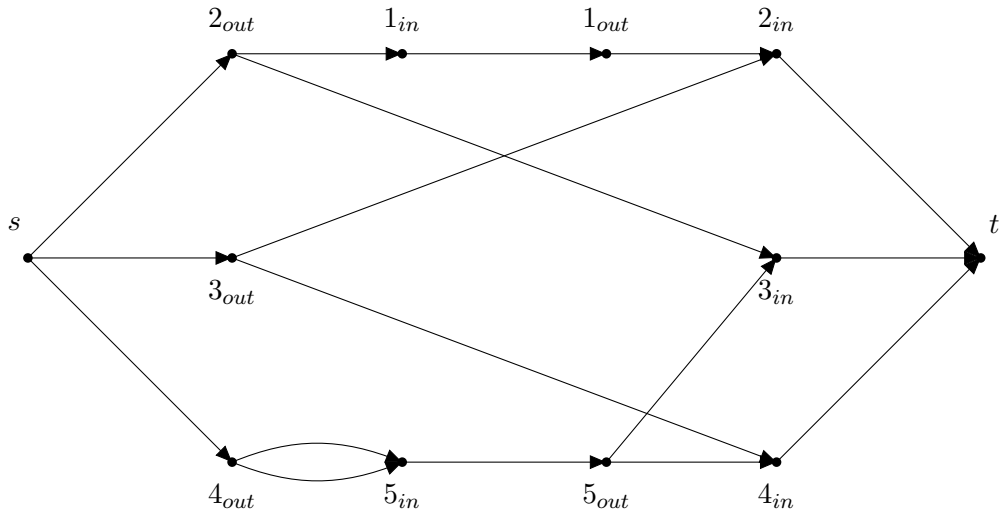
Figure 45 $b)$ shows the network G'' for the graph G of Figure 44.
We obtain an exact value for the size of a maximal flow in G''.

Lemma 4.18
$G'' := (V', E'', s, t)$ has a maximal flow f of size $|M|$.

Proof. As $deg_+(s) = deg_-(t) = |M|$, $|f| \leq |M|$. Let (V_1, V_2) with $V_1 \cup V_2 = V'$, $V_1 \cap V_2 = \emptyset$ and $s \in V_1$, $t \in V_2$, be an arbitrary (s, t)-cut in G''.
By Lemma 4.17, we have at least $(|M| - 1)D_{max} + 1$ edge-disjoint (s, t)-paths in G'. In G'', the corresponding (s, t)-paths are edge-disjoint except for the use of the contracted multiple edges of types $(s, v_{out}), (v_{in}, v_{out})$ or (v_{in}, t).
We choose such a path P in G''. P 'crosses' from V_1 to V_2 for a final time at a vertex $v \in V_1$, using an edge $e = (v, w) \in E''$ with $w \in V_2$. Thus there is at least one edge $e = (v, w)$ with $v \in V_1$ and $w \in V_2$, i.e. the capacity of the cut (V_1, V_2) is at least one.
We remove v (and all incident edges) from G'' and know that, by this, we lose at most D_{max} (s, t)-paths from G'', as $deg_-(v) = deg_+(v) \leq D_{max}$, by our construction. We now iteratively choose a new path in the remaining network, and count the edges found.
As we started with at least $(|M| - 1)D_{max} + 1$ (s, t)-paths, after $|M - 1|$ iterations, we still have at least one (s, t)-path remaining, so we can do so at least $|M|$ times. By this, the capacity of the (V_1, V_2)-cut is at least $|M|$. As (V_1, V_2) is an arbitrary (s, t)-cut in G'', by the max flow-min cut Theorem, we get $|f| = |M|$. □



(a) The network G' for the graph of Figure 44.



(b) The network G'' for the graph of Figure 44.

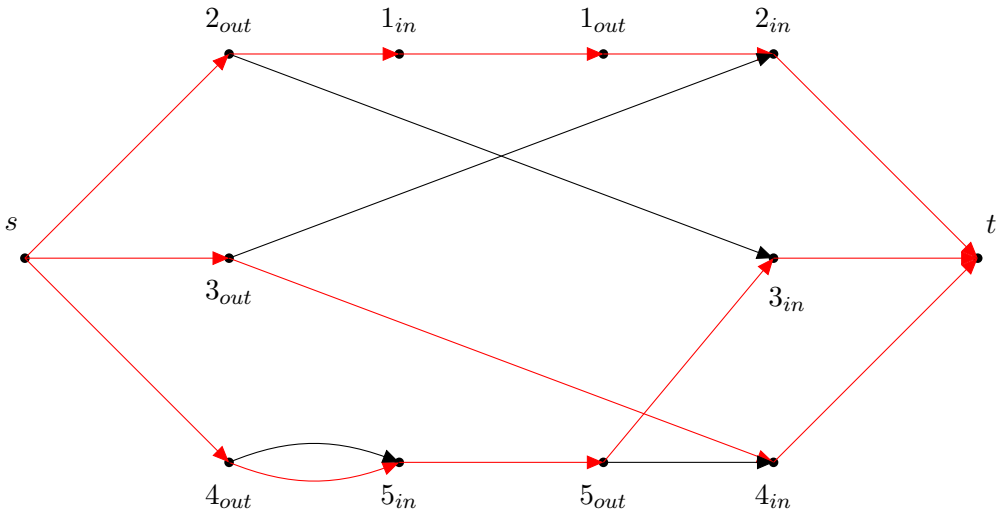
Figure 45: The networks G' and G'' for the graph G of Figure 44.

The flow identified in $G'' = (V', E'', s, t)$ has a special interpretation in the original graph G.

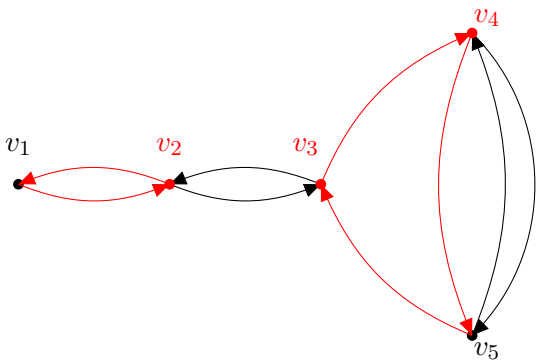
Lemma 4.19
A flow of size $|M|$ in G'' corresponds bijectively to a cover of M in G by a set of vertex-disjoint cycles.

Proof. Let f be a flow of size $|M|$ in G''. All edges in G'' except for the ones of type $(s, v_{out}), (v_{in}, v_{out})$ or (v_{in}, t) correspond bijectively to edges in G. Of these, the ones used by f are a cover $f' \subset E$ of M by vertex-disjoint cycles. This can be seen as follows:
Due to our construction of G' and G'', where we split up all vertices v into two vertices v_{in}, v_{out}, we have $deg_-(v)_{|f'} = deg_+(v)_{|f'} \leq 1$.
In G'', we connect each v_{out} of a $v \in M$ to a v'_{in} of a $v' \in M$ by a path in f. For $v = v'$, the v_{out}, v_{in}-path corresponds to a cycle in G covering v that does not contain any other vertices in M.
If $v \neq v'$, the v_{out}, v'_{in}-path corresponds to a cycle-part in G. In this case, there are more cycle-parts in G, or respectively paths in G'' connecting v' to some v'', and so on, until some v^{l-1} is connected to v. Together, the edges used form a cycle in G which covers $v, v', \ldots, v^{l-1} \in M$. Thus, we get a set of cycles f' in G covering M. With $deg_-(v)_{|f'} = deg_+(v)_{|f'} \leq 1$, no vertex lies on two different cycles. By this, f' is a set of pairwise vertex-disjoint cycles.
Conversely, the association of a maximal flow f in G'' with a given set of vertex-disjoint cycles f' in G is canonical: Apart from the edges of type (v_{out}, w_{in}) if $(v, w) \in f'$, we just need to add all edges of type (s, v_{out}) and (v_{in}, t) and all edges (v_{in}, v_{out}) for all $v \in V \backslash M$ covered by f' to obtain f. □

Figure 46 a) shows the maximal flow of size $|M| = 3$ in G'' of Figure 45 b), and Figure 46 b) the corresponding vertex-disjoint cycle cover of M in the original graph G of Figure 44.
We close this section by noting that, due to the contraction of multiple edges of types $(s, v_{out}), (v_{in}, v_{out}), (v_{in}, t)$ to single edges, there is no structural change to the flow f in G'' if we replace all multiple edges of type (v_{out}, w_{in}) by single ones as well, as each vertex can be used only once anyway. If we do so, when translating the flow f in G'' to f' in G, we then choose an arbitrary edge of the contracted multiple edges.
In this section, we saw that for a certain type of graphs $G = (V, E)$ and certain subsets $M \subset V$ of vertices in G, it is possible to cover M by a set of vertex-disjoint



(a) A flow of size $|M| = 3$ in G'' of Figure 45 b).



(b) The vertex-disjoint cycle cover corresponding to the maximal flow in a).

Figure 46: The construction of a vertex-disjoint cycle cover of M in G of Figure 44 from a maximal flow in G'' of Figure 45.

cycles. In the next section, we use this fact for CDGs to derive an upper bound on the diameter of partition polytopes.

4.3 Diameter of the Partition Polytope

Before using the results of the last section to derive a tight bound on the diameter of the partition polytope (in a worst case), let us turn to some known diameter statements.

First of all, the partition polytope is a special transportation polytope. This allows us to transfer the diameter bounds established for the general transportation polytope [BHS02; HS02; BHS03], yielding a diameter bound of $8(k+n-2)$ for the partition polytope. Using unpublished work, this bound can be reduced to $4(k+n-1)$ [LKOS09] or even $3(k+n-2)$ [Loe06].

Additionally, the partition polytope is a $0,1$-polytope, and by this the famous **Hirsch Conjecture** [Dan63] holds for it [Nad89]. It states that its diameter can be no higher than $fac - dim$, where dim is the dimension of the polytope and fac its number of facets.

Recalling Definition 3.9, and noting that $y_{in} = \kappa_i - \sum_{j=1}^{n-1} y_{ij}$ for all $i \in \{1,\dots,k\}$, as well as $y_{kj} = 1 - \sum_{i=1}^{k-1} y_{ij}$ for all $j \in \{1,\dots,n\}$, it is easy to see that $dim = (k-1)\cdot(n-1)$ (as the remaining y_{ij} are chosen 'freely').

With this direct dependence of the y_{in} and y_{kj} from the y_{ij} with $i < k, j < n$, it suffices to use $(k-1)+(n-1)$ of the 'equality' constraints (in the form $\sum_{j=1}^{n-1} y_{ij} \leq \kappa_i$, respectively $\sum_{i=1}^{k-1} y_{ij} \leq 1$) and $(k-1)\cdot(n-1)$ of the inequality constraints $y_{ij} \geq 0$ to describe all of the facets of the polytope. Each of these constraints can induce at most a single facet of the polytope, thus $fac \leq (k-1)+(n-1)+(k-1)\cdot(n-1)$. By this, the partition polytope has a diameter of at most $fac - dim \leq k+n-2$.

In the following, we derive a much tighter bound on the diameter of the partition polytope by exploiting its special structure.

We start by examining the graph theoretical result of the last section in the context of the partition polytope and CDGs. Let C and C' be two PSCs. By Theorem 4.15 and Corollary 4.16, we know that we can cover the set M of vertices of maximal degree in $CDG(C,C')$ by a set of vertex-disjoint cycles.

Corollary 4.20

Let C and C' be two PSCs. Let M be the set of vertices of maximal degree in $CDG(C,C')$. Then M can be covered by a set of pairwise vertex-disjoint cycles in $CDG(C,C')$.

Proof. We already noted that CDGs satisfy the prerequisites of Theorem 4.15. The statement then is a direct rewording of Corollary 4.16. □

As we will see (in Lemma 4.22 and Corollary 4.24), having a set of vertex-disjoint cycles in $CDG(C, C')$ covering the vertices of maximal degree allows us to identify at most two cyclical exchanges that, applied to C, yield a PSC C'' such that the vertices of maximal degree in $CDG(C'', C')$ have indegree one less than in $CDG(C, C')$. This corresponds to 'walking along' the edges associated with these cyclical exchanges (recall Lemma 4.6, Corollary 4.12) in PP, which allows us to talk about the diameter of PP.
The construction we describe first was used in [BR74] to prove the diameter of the **Birkhoff polytope** to be two. The Birkhoff polytope is a special transportation polytope, and a special case of our partition polytope, satisfying $\kappa_1 = \cdots = \kappa_k = 1$. Informally, k items are assigned to k clusters.
In algebraic terminology, the construction we will use also corresponds to the decomposition of a permutation into two indecomposable permutations, used to derive a bound on the diameter of permutation polytopes for permutation groups of the form $\min\{2 \cdot t, \lfloor \frac{n}{2} \rfloor\}$ in [GP06], where t is the number of the non-trivial orbits of the permutation group.
The close relationship of this bound to the results of this section can be seen by using the **Young representation** [Jam78; Onn93] of the symmetric group to identify a permutation group underlying the combinatorial structure of the partition polytope.
With our graph-theoretical terminology and our constructive approach, it is not difficult to derive the diameter bound as denoted in Theorem 4.9. We first establish the trivial bound of $\lfloor \frac{n}{2} \rfloor$.

Lemma 4.21
$PP(k, \kappa_1, \ldots, \kappa_k)$ has a diameter of at most $\lfloor \frac{n}{2} \rfloor$.

Proof. Two PSCs C and C' differ by a set of cyclical exchanges which do not move any $x \in X$ more than once, by Lemma 2.11. At most (all) n items participate in these cyclical exchanges, and each cyclical exchange contains at least two items. Thus, the number of cyclical exchanges to derive C' from C is bounded by $\lfloor \frac{n}{2} \rfloor$. The claim follows from the correspondence of cyclical exchanges and edges in PP, by Corollary 4.11 and Corollary 4.12. □

Let us now turn to the construction used in [BR74] and [GP06]. We describe it using cluster difference graphs. Recall that its edges are all labeled with the

specific items to be moved. For the sake of a simple representation, most of the time, we will not denote these labels and refer to a CDG in the form $G = (V, E)$.

Lemma 4.22
Let C and C' be two PSCs such that $CDG(C, C')$ decomposes into a set of vertex-disjoint cycles. Applying at most two cyclical exchanges to C, we can then transform C into C'.

Proof. Note that the fact that $CDG(C, C')$ decomposes into a set of cycles follows from Corollary 4.11. The special assumption is that these cycles do not share any vertices in $CDG(C, C')$.
We prove the claim constructively. Let $CDG(C, C') = G(V, E)$ and let $\mathcal{CY} := \{CY_1, \dots, CY_t\}$ be the given set of vertex-disjoint cycles. If $\mathcal{CY}$ consists of a single cycle CY, we use this cycle in the following. Otherwise, we define a cyclical exchange as follows:

- Start with the cycle decomposition $\{CY_1, \dots, CY_t\}$.
- Remove an edge $e_i = (c_{s_i}, c_{u_i})$ from CY_i for all $i \in \{1, \dots, t\}$.
- Add an edge $e_i' = (c_{s_i}, c_{u_{i+1}})$ for all $i \in \{1, \dots, t\}$ to CY_i. We assume $t + 1 = 1$ with respect to the indices i.

The result of this construction is that $CY := CY_1 \cup CY_2 \cup \dots \cup CY_t$ is a single cycle through all vertices of the CDG. By applying this cyclical exchange to C, we obtain a PSC C''. Let us now consider $CDG(C'', C')$. If $\mathcal{CY}$ consisted of only a single cycle CY, $C'' = C'$ and we are done.
Otherwise, by our construction, all item movements but the ones associated with edges e_i and e_i' were 'performed correctly' by the application of CY. The remaining difference lies in the items that should have been moved from c_{s_i} to c_{u_i}, but have been moved to $c_{u_{i+1}}$ instead. Thus $CDG(C'', C')$ only consists of edges of type $(c_{u_{i+1}}, c_{u_i})$ for all $i \in \{1, \dots, t\}$. Clearly, these edges form a single cycle in $CDG(C'', C')$. Applying the corresponding cyclical exchange, i.e. moving the items x_i that were associated with the edge (c_{s_i}, c_{u_i}) in $CDG(C, C')$ now from cluster $C_{u_{i+1}}$ to C_{u_i} yields the PSC C'. □

We informally said that we 'apply at most two cyclical exchanges to C'. Note that the order of the two cyclical exchanges constructed is important, as they share items. We actually have to talk about a sequence of these cyclical exchanges being applied.

Note that the prerequisites of Lemma 4.22 are always satisfied in the Birkhoff polytope, due to $\kappa_i = 1$ for all $i \in \{1, \ldots, k\}$, i.e. each cluster having only a single item.
Let us now consider the two PSCs C and C' with corresponding assignments E and E' depicted in Figure 42. They assign 5 items to 5 clusters, and thus yield vertices of the Birkhoff polytope. There must be two cyclical exchanges to derive C' from C. We construct $CDG(C, C')$ in Figure 47.
Figure 48 depicts the construction of a first cyclical exchange according to Lemma 4.22 to derive a PSC C'' with corresponding assignment E''. Figure 49 shows the resulting assignment E'' and the difference of E'' to E'.
We then construct $CDG(E'', E')$ in Figure 50. Recall that the proof of Lemma 4.22 describes its direct construction from the information in Figure 48 without using the information depicted in Figure 49. As expected, it only contains a single cycle. Applying this single cyclical exchange to E'' yields E'.
Recall that applying a cyclical exchange to a PSC (or the corresponding assignment) is equivalent to walking from the associated vertex in PP along an edge to a neighbouring vertex. We get the following immediate corollary.

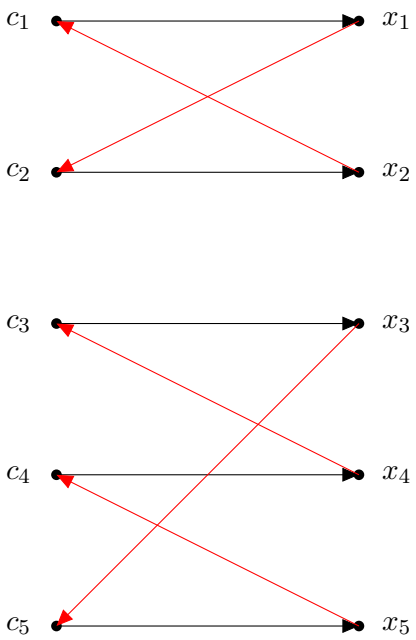
Corollary 4.23
Let C and C' be two PSCs such that $CDG(C, C')$ decomposes into a set of vertex-disjoint cycles $\mathcal{CY}$, and let v and v' be the corresponding vertices in PP.
If $\mathcal{CY}$ only contains a single cycle, v and v' are connected by an edge in PP. If $\mathcal{CY}$ contains more than one cycle, v and v' have a combinatorial distance of exactly two in PP.

Proof. The construction described in Lemma 4.22 explains that we have to apply one, or respectively two cyclical exchanges to derive C' from C. Corollary 4.11 and Corollary 4.12 establish the connection of this fact to the combinatorial distance, as explained above. □

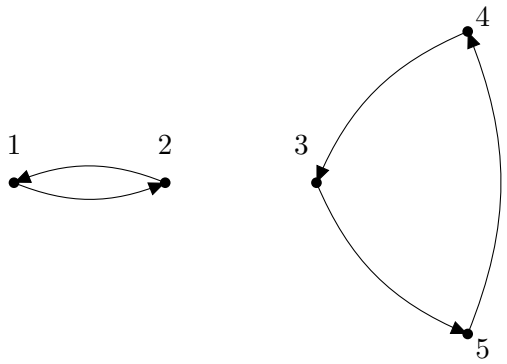
We get another immediate corollary.

Corollary 4.24
Let C and C' be two PSCs. By applying at most two cyclical exchanges we can derive a PSC C'' such that $\max_{c_i \in CDG(C'', C')} deg_-(c_i) = \max_{c_i \in CDG(C, C')} deg_-(c_i) - 1$.

Proof. We know that the set M of vertices of maximal degree in $CDG(C, C')$ has a cover by vertex-disjoint cycles, by Corollary 4.20. With the construction from Lemma 4.22, we can 'delete' this set of cycles from $CDG(C, C')$, reducing the in- and outdegree of all vertices covered by exactly one. This proves the claim. □

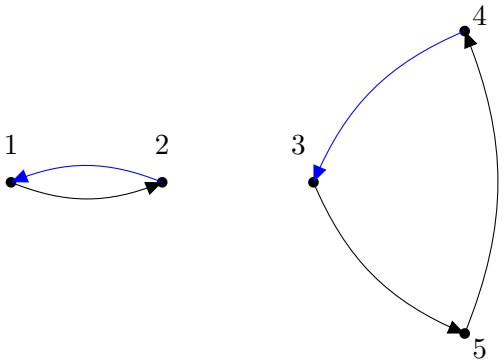


(a) The 'orientated' difference of E and E'.

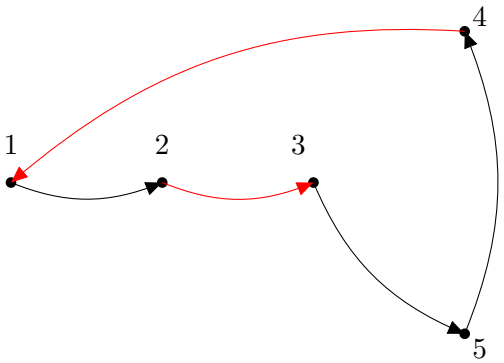


(b) The corresponding $CDG(E, E')$.

Figure 47: Construction of $CDG(E, E')$ for the assignments of Figure 42. Here, it is a set of vertex-disjoint cycles.

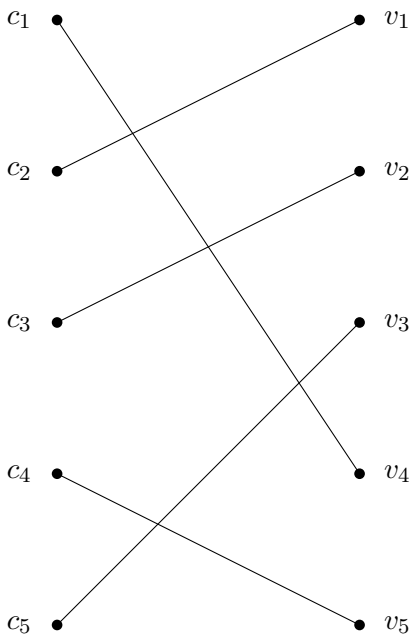


(a) One edge in every cycle of $CDG(E, E')$ of Figure 47 marked blue.

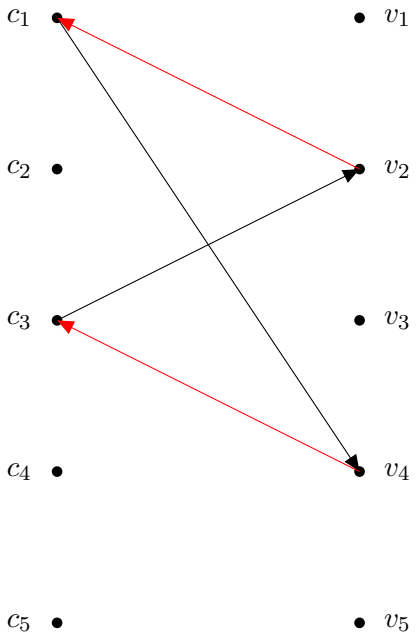


(b) The cyclical exchange that we apply to derive a PSC C'' from C.

Figure 48: Construction of a first cyclical exchange according to Lemma 4.22 for $CDG(E, E')$ of Figure 47.

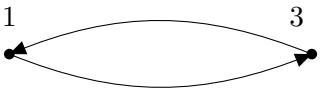


(a) The assignment E'' derived from an application of the cyclical exchange in Figure 48 to E.

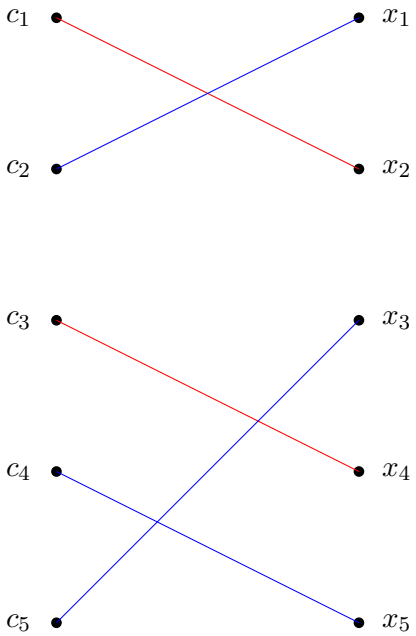


(b) The 'orientated' difference of E'' and E'.

Figure 49: The assignment E'' derived from E by an application of the cyclical exchange of Figure 48 to E.



(a) The $CDG(E'', E)$ corresponding to Figure 49 b). The CDG consists only of a single cycle.



(b) The application of the cyclical exchange in a) to E'' yields E'. The blue assignments were 'corrected' by the first cyclical exchange, the red assignment by the second one.

Figure 50: The application of the single cyclical exchange in $CDG(E'', E')$ to E'' yields E'

Informally, Corollary 4.24 states that for PSCs C and C', we can reduce the maximal indegree in $CDG(C, C')$ by one by applying at most two cyclical exchanges to C, and considering the resulting PSC C'' and $CDG(C'', C')$.
We now are ready to prove the diameter bound of the partition polytope of Theorem 4.9:

Theorem 4.25

$PP(k, \kappa_1, \ldots, \kappa_k)$ has a diameter of at most

$$\min\left\{\kappa_{i_1} + \kappa_{i_2}, \lfloor\frac{n}{2}\rfloor\right\},$$

where $i_1 := \arg\max_{i\in\{1,\ldots,k\}} \kappa_i$ and $i_2 := \arg\max_{i\in\{1,\ldots,k\}\setminus\{i_1\}} \kappa_i$.

Proof. Let C and C' be two PSCs, and consider $CDG(C, C')$. We know that $deg_-(c_i) \leq \kappa_i \leq \kappa_{i_1}$ for all $c_i \in CDG(C, C')$. The case $deg_-(c_i) = \kappa_i$ corresponds to the 'worst case' that there is no $x \in C_i$ with $x \in C'_i$.
By Corollary 4.24, by applying at most two cyclical exchanges to C, we are able to reduce this maximal indegree by one. Doing so iteratively, we can derive C' from C by applying a sequence of at most $2 \cdot \max_{i\in\{1,\ldots,k\}} \kappa_i$ cyclical exchanges, each representing the transition from a vertex of PP to a neighbouring vertex along an edge.
Additionally, if the set M of vertices of maximal degree in $CDG(C, C')$ only consists of a single vertex, we can clearly cover M with a single cycle, and thus can reduce the maximal indegree in $CDG(C, C')$ by applying only a single cyclical exchange.
We can do so $\kappa_{i_1} - \kappa_{i_2}$ times, before M contains at least two vertices. This results in a total of at most $(\kappa_{i_1} - \kappa_{i_2}) + 2\kappa_{i_2} = \kappa_{i_1} + \kappa_{i_2}$ cyclical exchanges needed.
The other part of the given bound is proven in Lemma 4.21. □

We formally note that the diameter result for the Birkhoff polytope is a special case of this result.

Corollary 4.26

The Birkhoff polytope has diameter at most 2.

Proof. We have $\kappa_1 = \cdots = \kappa_k = 1$ for the Birkhoff polytope, so, by Theorem 4.9, its diameter is at most $\kappa_{i_1} + \kappa_{i_2} = 2$. □

It is easy to see that the bound of Corollary 4.26 becomes tight for $n \geq 4$. We now turn to a more general example showing that the bound given in Theorem 4.9 can be tight for any κ_{i_1}.

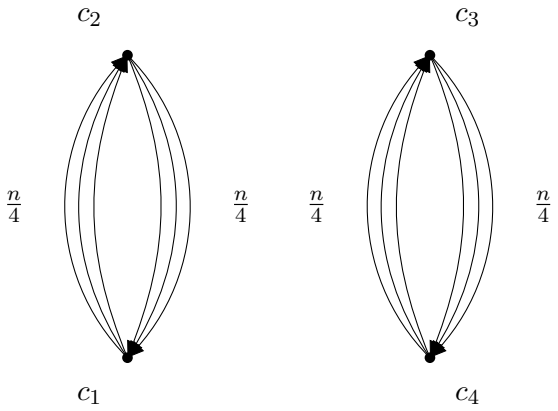


Figure 51: The CDG for a tight bound example for Theorem 4.9.

Lemma 4.27
The bound given in Theorem 4.9 can be tight for any value of κ_{i_1}.

Proof. We prove the claim by constructing a tight example. Let $k = 4$ be arbitrary and $\kappa_1 = \kappa_2 = \kappa_3 = \kappa_4$. Thus, each of the 4 clusters contains $\frac{n}{4}$ points. Consider $C := (C_1, C_2, C_3, C_4)$, where

$$x_1, \dots, x_{\frac{n}{4}} \in C_1,\ x_{\frac{n}{4}+1}, \dots, x_{\frac{n}{2}} \in C_2,\ x_{\frac{n}{2}+1}, \dots, x_{\frac{3n}{4}} \in C_3,\ x_{\frac{3n}{4}+1}, \dots, x_n \in C_4.$$

Additionally, let $C' := (C'_1, C'_2, C'_3, C'_4)$ be defined by

$$x_{\frac{n}{4}+1}, \dots, x_{\frac{n}{2}} \in C'_1,\ x_1 \dots, x_{\frac{n}{4}} \in C'_2,\ x_{\frac{3n}{4}+1}, \dots, x_n \in C'_3,\ x_{\frac{n}{2}+1}, \dots, x_{\frac{3n}{4}} \in C'_4.$$

Informally, C and C' have no item-assignment in common. Instead, all $\frac{n}{4}$ items that are in C_1 are in C'_2, and vice versa. The same holds for C_3 and C'_4. The corresponding $CDG(C, C')$ is depicted in Figure 51. $CDG(C, C')$ decomposes into $\frac{n}{2} = \lfloor \frac{n}{2} \rfloor = 2 \cdot \kappa_{i_1}$ cyclical exchanges, yielding a combinatorial distance of the two clusterings in PP of at most this value. We now show that C cannot be transferred to C' with less applications of cyclical exchanges: By induction, we prove that we can at most reduce the number of (edge-disjoint) cycles in the remaining CDG by $\frac{n}{2}$ by applying $\frac{n}{2}$ cyclical exchanges.
It is easy to see that the claim holds for $n = 4$ or $\kappa_i = 1$ for $i \in \{1, \dots, 4\}$. In this case, the graph consists of two vertex-disjoint cycles. With the construction from Lemma 4.22, we need at least two cyclical exchanges to 'delete' the two cycles of the original CDG. Figure 52 shows that we have no advantage by using different,

longer cyclical exchanges than the two canonical ones. So let the claim now hold for n, and consider the graph with $n+4$ vertices or $\kappa'_i := \kappa_i + 1$.
Using our induction hypothesis, we know that by applying $\frac{n}{2}$ cyclical exchanges we can only reduce the number of (edge-disjoint) cycles in the remaining CDG by at most $\frac{n}{2}$. Thus, after doing so, we still have at least 2 cycles. We need at least 2 more cyclical exchanges to 'delete' these. This proves the claim. □

Note that in the example of Lemma 4.27, $\kappa_{i_1} = \kappa_{i_2}$. This is not necessary for the bound of Theorem 4.9 to be tight.
We derived our diameter bound by considering the maximally possible indegree of a vertex of the CDG of two PSCs. By this approach, for two given PSCs, we obtain a direct corollary to their combinatorial distance.

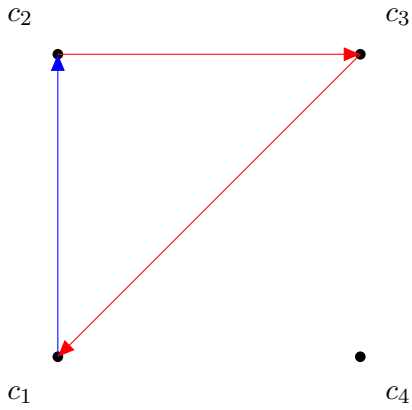
Corollary 4.28
Let $C := (C_1, \dots, C_k)$ and $C' := (C'_1, \dots, C'_k)$ be two PSCs of X, let v and v' be the corresponding vertices in PP, and let $\eta_1, \dots, \eta_k \in \mathbb{N}$, where η_i is the number of items $x \in X$ with $x \in C_i \wedge x \notin C'_i$.
The combinatorial distance of v and v' then is at most $\min\{\eta_{i_1} + \eta_{i_2}, \lfloor \frac{n}{2} \rfloor\}$, where $i_1 := \arg \max_{i \in \{1,\dots,k\}} \eta_i$ and $i_2 := \arg \max_{i \in \{1,\dots,k\} \setminus \{i_1\}} \eta_i$.

Proof. The indegrees of the vertices of $CDG(C, C')$ correspond to the values of $\eta_1, \dots, \eta_k$ rather than to the worst-case bounds $\kappa_1, \dots, \kappa_k$. The claim then follows analogously to the proof of Theorem 4.9. □

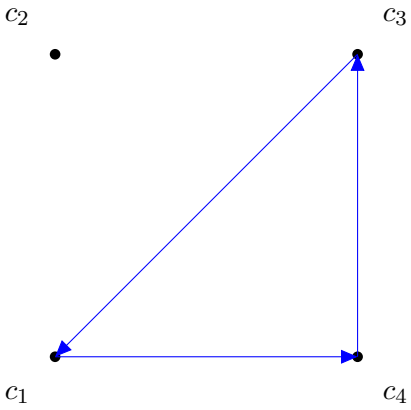
Note that trivially $\eta_{i_1} \leq \lfloor \frac{n}{2} \rfloor$.
In this section, we derived a bound on the diameter of the partition polytope. The bound is tight with respect to 'worst-case' examples for $\max_{i \in \{1,\dots,k\}} \kappa_i$. In the next section, we finish our discussion of the edge-structure of the partition polytope by summarizing the steps necessary to construct a corresponding edge-walk, and by showing that this can be done in polynomial time in the arithmetic model of computation.

4.4 Algorithmic Application

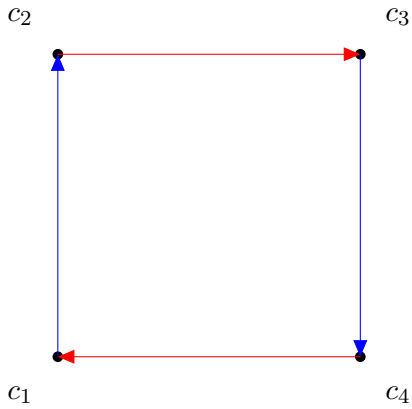
Our constructive algorithms in the last sections explain algorithmic means for transitions between clusterings. This section is dedicated to summarizing these algorithms and investigating their efficiency.
Assume we have PSCs $C := (C_1, \dots, C_k)$ and $C' := (C'_1, \dots, C'_k)$ and that we want to derive C' from C. To do so, we iteratively transform C to new clusterings



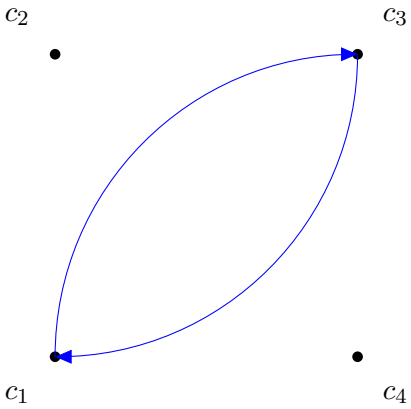
(a) A cyclical exchange of length 3 'corrects' only one (the blue) assignment.



(b) It leads to another CDG with a cyclical exchange of length 3 'correcting' 3 item assignments.



(c) A cyclical exchange of length 4 'corrects' two (the blue) assignments.



(d) It leads to another CDG with a cyclical exchange of length 2 'correcting' two item assignments.

Figure 52: Cyclical exchanges of length 3 and 4 applied to the CDG of Figure 51. By using two cyclical exchanges, in both examples, the number of cycles in the remaining CDG is reduced by exactly two.

that are 'less different' from C', until we are done. This can be done in polynomial time.

Theorem 4.29

Let C and C' be two PSCs of $X := \{x_1, \ldots, x_n\}$. A sequence $\mathcal{T}$ of at most $\min\{\eta_{i_1} + \eta_{i_2}, \lfloor \frac{n}{2} \rfloor\}$ cyclical exchanges to derive C' from C can be determined in polynomial time in n in the arithmetic model of computation. (The η_i are defined as in Corollary 4.28.)

Proof. We start by calculating $CDG(C, C')$. The description of the construction for Definition 4.10 is based on the transformation of $E\Delta E'$ corresponding to C and C' to $CDG(C, C')$. Here, we will not create $E\Delta E'$ for C and C' explicitly, but instead describe how to create $CDG(C, C')$ directly: We start with an empty digraph $G = (V, E, w)$ with labeling function w. For each $x_1, \ldots, x_n$, we test whether $x_l \in C_i$ and $x_l \in C'_j$ for $i \neq j$. If so, we add c_i and c_i to V, $e := (c_i, c_j)$ to E and set $w(e) := x_l$. We need to look at each vertex exactly once, and obtain a graph with $|E| \leq n$ and $|V| \leq k$. Clearly, the calculation of the CDG can be done by performing n comparisons and a constant number of graph and set operations belonging to each comparison result. We here assume that it takes constant time to identify the cluster of a vertex, and use the notation $C_i(x_l)$ for the index of the cluster C_i with $x_l \in C_i$. Algorithm 7 lists the steps in pseudo-code.

Input : PSCs $C := (C_1, \ldots, C_k), C' := (C'_1, \ldots, C'_k)$ of point set $X := (x_1, \ldots, x_n)$
Output: $CDG(C, C') = G(V, E, w)$

$V := \emptyset, E := \emptyset$;
for $1 \leq l \leq n$ **do**
  if $C_i(x_l) \neq C'_j(x_l)$ **then**
    $V = V \cup \{c_i, c_j\}$;
    $e := (c_i, c_j)$;
    $E = E \cup \{e\}$;
    $w(e) = x_l$;
  end
end
return $CDG(C, C') = G(V, E, w)$;

Algorithm 7: Construction of a CDG

A cycle decomposition of $CDG(C, C')$ can be calculated greedily as seen in the proof of Lemma 2.11. $CDG(C, C')$ has at most n edges. It is easy to describe an implementation for this where we only need to look at each edge up to two times, yielding polynomial running time for this step.

Specifically, when closing a cycle CY during our greedy collection of edges in the CDG, we remove the corresponding edges from E, and, at the same time, record that we removed an edge for each $c_i \in CY$ so that we know $deg_-(c_i)$ for each $c_i \in V$ (as well as η_{i_1} and η_{i_2}) as soon as $E = \emptyset$.
In the end, we then can identify the set $M \subset V$ of vertices of maximal degree by performing k comparisons and at most k set operations. Algorithm 8 describes this procedure in pseudo-code.

Input : $CDG(C, C') = G(V, E, w)$
Output: Cycle decomposition $\mathcal{T}'$ of E; $deg_-(c_i)$ for all $c_i \in V$, η_{i_1} and set $M \subset V$ of vertices of maximal deg_-

$\mathcal{T}' := \emptyset$;
$deg_-(c_i) := 0$ for all $c_i \in V$;
while $E \neq \emptyset$ **do**
  Greedily identify a cycle CY in E (Lemma 2.11);
  $E = E \backslash CY$;
  $deg_-(c_i) = deg_-(c_i) + 1$ for all $c_i \in CY$;
  $\mathcal{T}' = \mathcal{T}' \cup \{CY\}$;
end
$\eta_{i_1} := 0$;
$M := \emptyset$;
for $c_i \in V$ **do**
  if $deg_-(c_i) = \eta_{i_1}$ **then**
    $M = M \cup \{c_i\}$;
  end
  if $deg_-(c_i) > \eta_{i_1}$ **then**
    $M = \{c_i\}$;
    $\eta_{i_1} = deg_-(c_i)$;
  end
end
return $\mathcal{T}'$, $deg_-(c_i)$ for all $c_i \in V$, η_{i_1} and M;

Algorithm 8: Greedy Decomposition of CDG

For the sake of simplicity, we only denoted the calculation of η_{i_1} in the pseudo-code. Clearly, the value of η_{i_2} can be determined at the same time, without losing the polynomial running time of this step.
If we obtain a decomposition $\mathcal{T}'$ of the CDG into at most $\min\{\eta_{i_1} + \eta_{i_2}, \lfloor \frac{n}{2} \rfloor\}$ cycles this way, we set $\mathcal{T}$ to be (an arbitrarily ordered) sequence of corresponding cyclical exchanges, and are done. Otherwise we apply the construction of Lemma 4.22 to reduce $deg_-(v)$ of all vertices $v \in M$ by one by applying at most two cyclical exchanges. The first step to do so is to find a vertex-disjoint cycle

cover of M in $CDG(C, C')$. We start with the construction of the network $G'' = (V', E'', s, t)$ used for the calculation of a max flow to do so (see Lemma 4.18 and Lemma 4.19).
Starting with $CDG(C, C')$, we follow the construction described in Section 4.2 and denote it in pseudo-code in Algorithm 9. We simplify the description by contracting it to one step. For each edge of type (v_{out}, w_{in}), we have a label identifying the $x \in X$ belonging to the corresponding movement. Recalling the comment after Lemma 4.19, it would suffice to have single edges between all of the vertices of the network.

Input : $CDG(C, C') = G(V, E, w)$, set M of vertices of maximal deg_-
Output: Network $G'' = (V', E'', s, t)$, (partial) labeling function $w' : E'' \to X$

$V' := \{s, t\}$, where $s, t \notin V$;
$E'' := \emptyset$;
For each $v \in V$, $V' = V' \cup \{v_{in}, v_{out}\}$;
For each $(v, w) \in E$, $e := (v_{out}, w_{in})$, $E'' = E'' \cup \{e\}$ and $w'(e) = w((v, w))$;
For each $v \in V \backslash M$, $E'' = E'' \cup \{(v_{in}, v_{out})\}$;
For each $v \in M$, $E'' = E'' \cup \{(s, v_{out}), (v_{in}, t)\}$;
return $G'' = (V', E'', s, t)$, w';

Algorithm 9: max flow Network Construction

We know that $|V| \leq k$, so we create a set of vertices V' with $|V'| \leq 2k + 2$. Trivially $|M| \leq |V|$ and $|E| \leq n$. We add at most n edges of type (v_{out}, w_{in}) to the graph, and additionally either one edge (v_{in}, v_{out}) or two edges $(s, v_{out}), (v_{in}, t)$, depending on whether $v \in V \backslash M$ or $v \in M$. This yields at most another $2n$ edges, so that $E'' \leq 3n$. We again obtain a polynomial running time for this step.
We can use e.g. the algorithm of Dinic [Din70] to solve the max flow problem in $G'' = (V', E'', s, t)$. A straightforward implementation of the algorithm, as described by Dinic, yields a running time of $O(|V'^2||E''|) \subset O((2k + 2)^2 \cdot 3n) \subset O(n^3)$. We derive a flow f which corresponds bijectively to a vertex-disjoint cycle cover of $CDG(C, C')$, see Lemma 4.19.
This correspondence can be established by collecting all (at most n) edges of type (v_{out}, w_{in}) used for f. We identify the different cycles of the cover like in the proof of Lemma 4.19: Starting at a vertex $v_{out} \in M$, we follow the s, t-path in f to a vertex $v'_{in} \in M$. If $v = v'$, we have one of the cycles of the cover, and start over again at a $w \in M$ we did not 'use' yet. Otherwise, we follow the s, t-path in f through v'_{out}, until we hit another v''_{in}. If $v'' = v$, we are done with this cycle. Otherwise, we continue with another path until we hit v_{in}. In total, we need to

consider each edge in f at most once, and $|f| \leq |E''| \leq 3n$ yields a running time bound of $O(n)$. We obtain a set of at most $\lfloor \frac{n}{2} \rfloor$ cycles $\mathcal{CY}$. Algorithm 10 describes these steps in pseudo-code.

Input : Network $G'' = (V', E'', s, t)$, M, flow f in G''
Output: Vertex-disjoint cycle cover $\mathcal{CY} = \{CY_1, \ldots, CY_t\}$ of M in $CDG(C, C')$

$\mathcal{CY} := \emptyset$;
$s := 0$;
while $M \neq \emptyset$ **do**
  $s = s + 1$;
  Pick a random $v \in M$, $M = M \backslash \{v\}$;
  $v'' := v$, set v' to a dummy value;
  while $v' \neq v$ **do**
    Follow the s,t-path in f through v''_{out} to a $v'_{in} \in M$, identify all edges of type (w_{out}, w'_{in}) used and add them to CY_s;
    $v'' = v'$;
    $M = M \backslash \{v'\}$;
  end
  $\mathcal{CY} = \mathcal{CY} \cup \{CY_s\}$;
end
return $\mathcal{CY}$;

Algorithm 10: Vertex-Disjoint Cycle Covering

If $\mathcal{CY}$ contains only a single cycle, we can apply the respective cyclical exchange directly, and, in one step, reduce the maximal deg_- in $CDG(C, C')$ by one. Otherwise we have to perform the construction of Lemma 4.22 to identify two cyclical exchanges CE_1, CE_2 that achieve the same reduction.
First, we define CE_1 by removing an edge $e_i = (c_{s_i}, c_{u_i})$ from each cycle CY_i for all $i \in \{1, \ldots, t\}$, and instead adding an edge $e'_i = (c_{s_i}, c_{u_{i+1}})$ connecting it to the 'next' cycle. This requires $2t$ set operations.
Second, we define CE_2 as the collection of edges of type $(c_{u_{i+1}}, c_{u_i})$ for all $i \in \{1, \ldots, t\}$. This requires t set operations. We get a running time of $O(t) \subset O(n)$. Algorithm 11 describes these steps in pseudo-code. Note that the labels of all edges but those of type $(c_{s_i}, c_{u_{i+1}})$ and $(c_{u_{i+1}}, c_{u_i})$ are clear from w.
These two cyclical exchanges CE_1, CE_2 (or just the single one CE_1) then have to be applied to $CDG(C, C')$ to reduce the maximal deg_- of the vertices by one. Note that the application of these two cyclical exchanges has the same effect as the application of the set of cyclical exchanges associated with the cycle cover $\mathcal{CY}$. We know the specific items to move due to the labeling function w.

Input : Vertex-disjoint cycle cover $\mathcal{CY} = \{CY_1, \ldots, CY_t\}$ of M in $CDG(C, C')$, labeling function w
Output: Cyclical exchange CE_1 or cyclical exchanges CE_1, CE_2

$CE_1 := \emptyset$, $CE_2 := \emptyset$;
if $t = 1$ **then**
  return $CE_1 = CY_1$;
end
Choose indices $s_1 \neq u_1, \ldots, s_t \neq u_t$ such that $(c_{s_i}, c_{u_i}) \in CY_i$ for all $i \in \{1, \ldots, t\}$;
for $1 \leq i \leq t$ **do**
  Add all edges but $e_i := (c_{s_i}, c_{u_i})$ from CY_i to CE_1;
  Add $e'_i := (c_{s_i}, c_{u_{i+1}})$ to CE_1 and set $w(e'_i) := w(e_i)$;
end
for $1 \leq i \leq t$ **do**
  Add $e'_i := (c_{u_{i+1}}, c_{u_i})$ to CE_2 and set $w(e'_i) := w(e_i)$;
end
return CE_1 and CE_2;

Algorithm 11: Construction of Cyclical Exchanges

The result of an application of CE_1 and then CE_2 to C is a new clustering C'' derived from C by at most $2 \cdot 2k$ set operations in the form of item transfers (deletions and additions of items). Algorithm 12 describes this step in pseudo-code.

Input : PSC C of $X := \{x_1, \ldots, x_n\}$, cyclical exchanges CE_1, CE_2
Output: PSC $C'' := (C''_1, \ldots, C''_k)$ derived from C by an application of CE_1, CE_2

$C'' := C$;
For each x_l associated with an edge $(c_i, c_j) \in CE_1$, remove x_l from C''_i and add x_l to C''_j;
For each x_l associated with an edge $(c_i, c_j) \in CE_2$, remove x_l from C''_i and add x_l to C''_j;
return C'';

Algorithm 12: Application of Cyclical Exchanges

Using the terminology of Corollary 4.28, we now have a PSC C'' such that η_{i_1} for the new $CDG(C'', C')$ is one less than for $CDG(C, C')$. If the new $\eta_{i_1} > 0$, we are not done yet. Instead, in a straightforward implementation, we now set $C = C''$ and perform all the algorithms described in this proof for the updated C. As the original $\eta_{i_1} \leq \lfloor \frac{n}{2} \rfloor$, and as we reduce it by one in each iteration step, we need to perform our algorithms at most $\lfloor \frac{n}{2} \rfloor$ times. With all of the algorithms

used having a polynomial running time, it thus is clear that the total running time is polynomial in n.

Input : PSCs $C := (C_1, \ldots, C_k), C' := (C'_1, \ldots, C'_k)$ of point set $X := (x_1, \ldots, x_n)$
Output: Sequence of cyclical exchanges $\mathcal{T}$ to derive C' from C

$\mathcal{T} := \emptyset$;
Construct $CDG(C, C')$ (Algorithm 7);
Decompose $CDG(C, C')$ into a set of cycles $\mathcal{T}'$ and identify the set of vertices of maximal deg_- M in $CDG(C, C')$ (Algorithm 8);
while $C \neq C'$ **do**
  if *the number of cyclical exchanges in* $\mathcal{T} \cup \mathcal{T}'$ *is* $\leq \min\{\eta_{i_1} + \eta_{i_2}, \lfloor \frac{n}{2} \rfloor\}$ **then**
    add the set $\mathcal{T}'$ (arbitrarily ordered) to the end of $\mathcal{T}$;
    return $\mathcal{T}$;
  end
  else
    Construct network $G'' = (V', E'', s, t)$ (Algorithm 9);
    Use the algorithm of Dinic for a max flow f in G'';
    Derive a vertex-disjoint cycle cover $CY \subset E(CDG(C, C''))$ of M from f (Algorithm 10);
    Construct cyclical exchanges CE_1, CE_2 for CY (Algorithm 11);
    if $CE_2 = \emptyset$ **then**
      Apply CE_1 to C to derive a PSC C'' (Algorithm 12);
      $C = C''$, add CE_1 to the end of $\mathcal{T}$;
    end
    else
      Apply CE_1 and CE_2 to C to derive a PSC C'' (Algorithm 12);
      $C = C''$, add CE_1, and then CE_2, to the end of $\mathcal{T}$;
    end
    Construct $CDG(C, C')$ (Algorithm 7);
    Decompose $CDG(C, C')$ into cycles $\mathcal{T}'$ and identify the set of vertices of maximal deg_- M in $CDG(C, C')$ (Algorithm 8);
  end
end

Algorithm 13: Derivation of C' from C by Cyclical Exchanges

Note that, as long as the current $\eta_{i_1} > \eta_{i_2}$, we find only one cyclical exchange CE_1 to apply to the current PSC C. Thus, we satisfy the desired bound on the number of cyclical exchanges in $\mathcal{T}$. We close the proof by summarizing the complete Algorithm 13 in pseudo-code. Note that the return-part in the if-clause is necessarily reached as soon as $\mathcal{T}' = \emptyset$. □

Throughout the proof of Theorem 4.29, we used a high-level description of the algorithms in pseudo-code. We did not consider any underlying data structures and used some assumptions for operations being possible in constant time. As we only wanted to prove the polynomial running time of the whole algorithm, these assumptions are unproblematic, as all these operations trivially can be implemented in at least polynomial time in n. For an efficient implementation in practice, a good choice of these data structures and operations is necessary.
Our constructive approach to the calculation of a small set of cyclical exchanges to derive a PSC from another PSC allows us to apply some constraints to the construction, like avoiding item movements between specific clusters. In our graph-theoretical terminology, it is easy to model these constraints, e.g. by deleting edges from graphs and networks we use. Some of these possibilities will be explored in future work.
We close this chapter with a short look at the edge-structure of the gravity polytope.

4.5 Edge-Structure of the Gravity Polytope

Up to this point in this chapter, we investigated the edge-structure and diameter of the partition polytope. We were able to derive a tight upper bound on its diameter (in a worst case). An analysis of the edge-structure of the gravity polytope is equally interesting, but, as we will see, quickly proves to be difficult. With the main difference between the polytopes being the addition of geometric information, it is clear that the set of vectors $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ clustered takes a central role in our investigation.
Recall that our gravity polytope is based on having fixed values for the cluster sizes $\kappa_1, \ldots, \kappa_k$. In the (bounded-shape) partition polytope without size restrictions as considered in [BHR92], walking in direction of an edge of the polytope (starting from a vertex v) is associated with moving an item $x \in X$ from some cluster C_i to another cluster C_j for $i \neq j;\ i, j \in \{1, \ldots, k\}$ [FOR03]. Clearly, not all item movements of this type belong to an edge, e.g. if we move an $x \in int(C_i)$ to another cluster C_j. Additionally, it is not clear that we arrive at a new vertex of the polytope like this, we just know that the new clustering derived lies on the respective edge.
In the gravity polytope, the situation is even more difficult. Instead of simply moving an item from one cluster to another, we always have to apply cyclical exchanges to satisfy the cluster size restrictions, see Lemma 2.11.

Nonetheless, under mild assumptions on X, we will be able to associate the transition of one vertex of the gravity polytope to a neighbouring vertex of the polytope with the application of a single cyclical exchange. We formulate our main result as follows.

Theorem 4.30
Let C, C' be two PSCs, $v = v(C), v' = v(C')$ be two vertices of Q connected by an edge and X be a point set in general position. Then C' can be derived from C by applying a single cyclical exchange.

We here use the following definition of the term 'general position' for X.

Definition 4.31 (General Position of X)
A set $X := \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ of points is **in general position** *if $x_i \neq x_j$ for all $i \neq j;\ i, j \in \{1, \dots, n\}$ and if no four points lie on a single line in $\mathbb{R}^d$.*

In the following, we will informally say that we apply a cyclical exchange to a vertex $v \in Q$ when the PSC C with $v = v(C)$ is clear from the context. We split the proof into several subclaims. First, we need to complement our terminology.

Definition 4.32 (Vector of a Cyclical Exchange)
Let $v \in Q$ and let CE be a cyclical exchange applied to v to derive $v' = v + y$. Then y is the **vector of the cyclical exchange** *CE.*

Note that a cyclical exchange contains information about the original and derived assignment, and so does the vector belonging to it. We informally call a cyclical exchange CE_i to have **weight** 0 **with respect to** $a = (a_1^T, \dots, a_k^T)^T$ if and only if the vector y_i of CE_i satisfies $a^T y_i = 0$.
If two vertices v, v' of Q are connected by an edge, there is an $a \in \mathbb{R}^{d \cdot k}$, such that $a^T v = a^T v' > a^T v''$ for any vertex $v'' \in Q \backslash \{v, v'\}$. The clusterings associated with these two vertices differ by a set of cyclical exchanges, by Lemma 2.11. Their clustering difference graph contains only cycles of an interesting property. In the following, we denote such cycles as CE instead of CY, and sets of cycles as $\mathcal{CE}$ instead of $\mathcal{CY}$, to emphasize their interpretation as cyclical exchanges.

Lemma 4.33
Let $C := (C_1, \dots, C_k)$, $C' := (C'_1, \dots, C'_k)$ be two PSCs with $v = v(C), v' = v(C')$ being vertices of Q connected by an edge, and let a be a vector with $a^T v = a^T v' > a^T v''$ for any vertex $v'' \in Q \backslash \{v, v'\}$. Let further $\mathcal{CE} := (CE_1, \dots, CE_t)$ be a cycle decomposition of $CDG(C, C')$, and let y_i be the vector of CE_i for all $i \in \{1, \dots, t\}$. Then $a^T y_i = 0$ for all $i \in \{1, \dots, t\}$.

Proof. By Corollary 4.11, we know that C and C' differ by a set of cyclical exchanges $\mathcal{CE} := (CE_1, \ldots, CE_t)$ bijectively corresponding to the given cycle decomposition of $CDG(C, C')$. We have $v' = v + y$ for $y = \sum_{i=1}^{t} y_i$.
There is a vector $a = (a_1^T, \ldots, a_k^T)^T$ such that $a^T v = a^T v' > a^T v''$ for any other vertex $v'' \in Q$. We have

$$a^T y = \sum_{i=1}^{t} a^T y_i = 0 \quad \Leftrightarrow \quad \sum_{i=2}^{t} a^T y_i = -a^T y_1$$

Suppose $a^T y_1 < 0$. Then $\sum_{i=2}^{r} a^T y_i > 0$. Thus there is a $j \in \{2, \ldots, r\}$ with $a^T y_j > 0$. Applying only the cyclical exchange CE_j to v yields a $v'' \in Q$ with $a^T v'' = a^T(v + y_j) > a^T v = a^T v'$, a contradiction to v and v' being optimal with respect to $a^T v = a^T v'$. If $a^T y_1 > 0$, we obtain the same contradiction. Thus $a^T y_1 = 0$, and the claim follows analogously for all $i \in \{1, \ldots, t\}$. This proves the claim. □

Any cycle in a CDG can be greedily completed to a cycle decomposition of the CDG (see the proof of Lemma 2.11), so the weight of any cycle in the CDG of neighbouring vertices in Q is 0, by Lemma 4.33.
Next, we show that the cyclical exchanges (in the CDG) to derive a vertex from a neighbouring vertex move items of the same clusters in the same **'order'**. We use this informal term as follows:
Let $C_i \to C_j$ denote that we move some item $x \in C_i$ to cluster C_j. Two cyclical exchanges have a different order if in one of them we have an item movement of type $C_i \to C_j$ and in the other $C_i \to C_l$ for $j \neq l;\ j, l \in \{1, \ldots, k\} \backslash \{i\}$.

Lemma 4.34
Let $C := (C_1, \ldots, C_k)$, $C' := (C'_1, \ldots, C'_k)$ be two PSCs with $v = v(C), v' = v(C')$ being vertices of Q connected by an edge, and let a be a vector with $a^T v = a^T v' > a^T v''$ for any vertex $v'' \in Q \backslash \{v, v'\}$. Let further $X := \{x_1, \ldots, x_n\}$ be a point set with $x_i \neq x_j$ for all $i \neq j;\ i, j \in \{1, \ldots, n\}$, and let $\mathcal{CE} := (CE_1, \ldots, CE_t)$ be a cycle decomposition of $CDG(C, C')$. Then all CE_i move items of the same clusters in the same order.

Proof. We have $v' = v + y$ for some vector $y \in \mathbb{R}^{d \cdot k}$. As $CDG(C, C')$ only contains cyclical exchanges of weight 0 with respect to a by Lemma 4.33, all gravity vectors of clusterings that are derived from applying any subset of cyclical exchanges in $\mathcal{CE}$ to v (or their inverse to v') must be on the edge between v and v'. This implies

that all cyclical exchanges between v and v' have to move the centers of gravity of the clusters componentwisely in direction of y.
Suppose that there are two cyclical exchanges $CE_1 \neq CE_2$ in $CDG(C, C')$. CE_1 and CE_2 have to move items between the same subset of clusters. Otherwise, as $x_i \neq x_j$ for $i \neq j$, the vectors of CE_1 and CE_2 leave different cluster centers unchanged, and thus cannot be collinear. Due to this, CE_1 and CE_2 move items between the same clusters, and respectively between the same associated vertices in $CDG(C, C')$, w.l.o.g. $c_1, \ldots, c_r$.
Suppose w.l.o.g. that CE_1 is of type $C_1 \to C_2 \to ... \to C_r \to C_1$ and that CE_2 contains a movement of type $C_i \to C_j$ with $j \neq i+1$ and not both $i = r, j = 1$. If $i < j-1$, we get a cyclical exchange $C_1 \to ...C_i \to C_j(\to ...) \to C_1$ 'skipping' at least one C_l for $i < l < j$. If $j < i$, we get a cyclical exchange of type $C_j \to ... \to C_i \to C_j$ skipping all clusters C_l with $l < j$ or $l > i$.
Thus there is a cyclical exchange CE_3 in $CDG(C, C')$ that leaves different cluster centers unchanged. As we have seen, its vector cannot be collinear to the vector of CE_1. This is a contradiction to all cyclical exchanges yielding clusterings on the edge between v and v'.
By this, all cyclical exchanges of any cycle decomposition CE of $CDG(C, C')$ move items of the same clusters in the same order. This proves the claim. □

Finally, we use the general position of X to complete the proof of Theorem 4.30.

Lemma 4.35
Let $C := (C_1, \ldots, C_k)$, $C' := (C'_1, \ldots, C'_k)$ be two PSCs with $v = v(C), v' = v(C')$ being vertices of Q connected by an edge, and let a be a vector with $a^T v = a^T v' > a^T v''$ for any vertex $v'' \in Q \backslash \{v, v'\}$. Let further X be a point set in general position. Then v and v' differ by only a single cyclical exchange, i.e. $CDG(C, C')$ contains only one cycle.

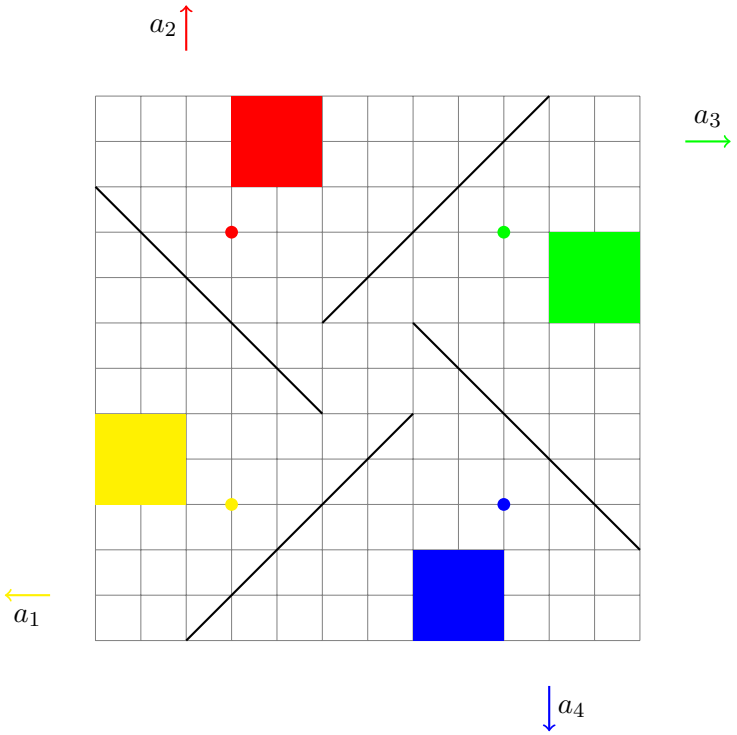
Proof. In Lemma 4.33 and Lemma 4.34, we showed that v and v' differ by a set of cyclical exchanges that w.l.o.g. all are of type $C_1 \to C_2 \to ... \to C_r \to C_1$. Recall that these cyclical exchanges do not move any item twice (Lemma 2.11), and thus, if such a cyclical exchange contains a cluster of size $\kappa_i = 1$, $CDG(C, C')$ only contains a single cyclical exchange.
So suppose now that $\kappa_1, \ldots, \kappa_r > 1$, and suppose that CE_1 swaps items $x_1 \to x_2 \to ... \to x_r \to x_1$ where $x_i \in C_i$ and CE_2 swaps items $x'_1 \to x'_2 \to ... \to x'_r \to x'_1$ where $x'_i \in C_i$.

Thus, the $CDG(C, C')$ contains all cycles of type $C_1 \to C_2 \to ... \to C_r \to C_1$ that use any combination of edges in $CDG(C, C')$ with which either x_i or x'_i is associated for all $i \in \{1, \dots, r\}$.
The vectors of all of these cyclical exchanges must be collinear. Only looking at cluster C_1, we see that

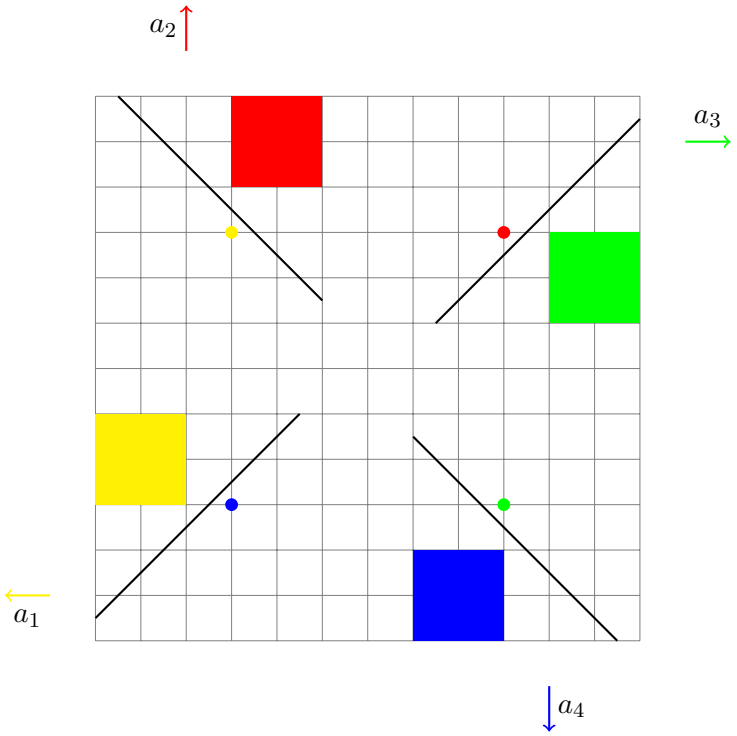
$$\frac{1}{\kappa_1}(x_r - x_1), \frac{1}{\kappa_1}(x_r - x'_1), \frac{1}{\kappa_1}(x'_r - x_1) \text{ and } \frac{1}{\kappa_1}(x'_r - x'_1)$$

are collinear. This can only be the case if $x_1, x'_1, x_r, x'_r \in X$ are on a line, which is a contradiction, as we assumed X to be in general position. This proves the claim. □

Theorem 4.30 follows directly from Lemma 4.35. Figure 53 shows two vertex clusterings connected by an edge in the gravity polytope. Both PSCs are optimal with respect to the $a := (a_1^T, a_2^T, a_3^T, a_4^T)^T \in \mathbb{R}^{d \cdot k}$ depicted in the figure. The filled rectangles contain all but the explicitly shown vertices of the respective clusters. Note that X in general position rules out the trivial case of being in dimension $d = 1$. The restrictions $x_i \neq x_j$ for $i \neq j$ are necessary to guarantee that a vertex of Q is in one-to-one correspondence to a PSC, see Lemma 2.12. Also, they are necessary for the proof of Lemma 4.34.
The prerequisite of X having no four points on a line is necessary, as well. Figure 54 shows an example of what may happen if this condition is not satisfied. The PSCs C and C' of the two-dimensional X are vertices of the respective Q, as C is uniquely optimal with respect to $b = \binom{b_1}{b_2}$, and C' is uniquely optimal with respect to $b' = \binom{b'_1}{b'_2}$ in Q. The corresponding vertices are neighbours in Q, as both C and C' are optimal with respect to $a := \binom{a_1}{a_2}$ in Q.
Any PSC C'' of equivalent objective function value $a^T v(C'') = a^T v(C) = a^T v(C')$ has a gravity vector $v(C)$ lieing between $v(C)$ and $v(C')$. Thus $v(C)$ and $v(C')$ are connected by an edge, but they differ by two cyclical exchanges. Examples for higher-dimensional X can be constructed analogously.
Unfortunately, the edge-structure of the gravity polytope is not as simple as the edge-structure of the partition polytope, where edges corresponded bijectively to cyclical exchanges. In the gravity polytope, edges are associated with certain cyclical exchanges under the assumption that we cluster a set of vectors X in general position, but only some cyclical exchanges belong to edges.
A next step to understanding the edge-structure of the gravity polytope might be to be able to construct a set X associated with a set of non-geometric items

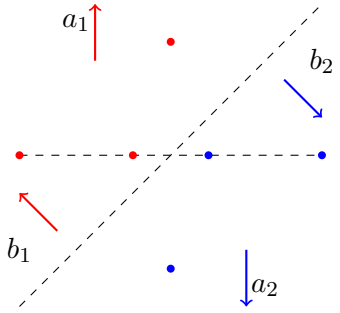


(a) A PSC connected by an edge to the PSC in b).

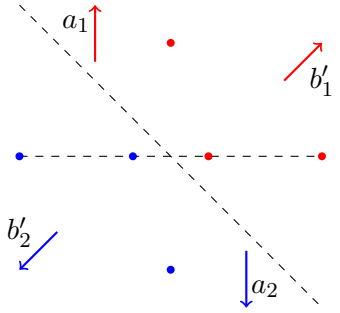


(b) A PSC connected by an edge to the PSC in a).

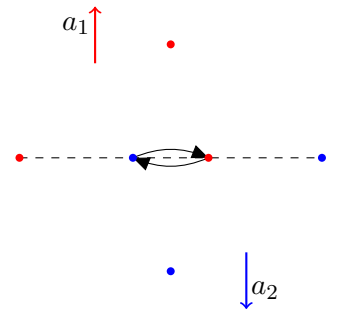
Figure 53: Two PSCs belonging to vertices in the gravity polytope that are connected by an edge.



(a) A PSC C optimal with respect to $a = \binom{a_1}{a_2}$ and uniquely optimal with respect to $b = \binom{b_1}{b_2}$ in Q.



(b) Another PSC C' optimal with respect to $a = \binom{a_1}{a_2}$ and uniquely optimal with respect to $b' = \binom{b'_1}{b'_2}$ in Q.



(c) A PSC C'' optimal with respect to $a = \binom{a_1}{a_2}$, but not uniquely optimal with respect to any vector in $\mathbb{R}^{d \cdot k}$.

Figure 54: X being in general position may be necessary for edges to correspond bijectively to certain cyclical exchanges. The PSCs C and C' are neighbouring vertices in the gravity polytope, but C'' lies in the interior of the corresponding edge.

such that a specific cyclical exchange (or edge in the partition polytope) also belongs to an edge in the gravity polytope. By Lemma 3.10, we know how we can transform the partition polytope to the gravity polytope, and that this linear transformation depends on the choice of X. The above remarks tell us that we should use an X in general position to avoid unnecessary complications. We have to create X such that the two PSCs associated with the edge in the partition polytope are neighbouring vertices in the gravity polytope as well.
We close this section by showing that this is possible by only using a two-dimensional set X, as follows.

Lemma 4.36
Let C, C' be two PSCs of a non-geometric set I of n items, let $v, v' \in PP$ be the respective vertices and let them be connected by an edge $e' = (v, v')$. There is a mapping of I to a two-dimensional geometric set $X := \{x_1, \ldots, x_n\}$ such that $v(C)$ and $v(C')$ are vertices in Q, and such that they are connected by an edge e corresponding to the same cyclical exchange as e'.

Proof. We prove the claim constructively. C and C' differ by a single cyclical exchange CE. For now, assume that $CE := (x_1, \ldots, x_k)$ with $x_i \in C_i$ for all $i \in \{1, \ldots, k\}$ involves all k clusters.
Figure 55 shows the construction of a cell decomposition $P := (P_1, \ldots, P_k)$ of $\mathbb{R}^2$, with $C_i \subset (P_i)$ for all $i \in \{1, \ldots, k\}$. In the figure, we have $k = 8$ clusters, each having its own cell.
The cells are constructed by choosing k points $a_1, \ldots, a_k$ on the unit sphere in $\mathbb{R}^2$, and then creating a full a-induced cell decomposition by calculating a Voronoi diagram for these vectors. We choose the a_i by walking along the unit-sphere clockwise, such that the pairs of cells P_i, P_{i+1} have a common one-dimensional intersection for all $i \in \{1, \ldots, k\}$.
We put all of the geometric points of each cluster C_i in the interior of their respective cell P_i (and in the interior of the unit ball), except for the vertices $x_1, \ldots, x_k$. x_i is chosen as the intersection of the line from a_i to a_{i+1} and the line $P_i \cap P_{i+1}$.
It is clear that both C and C' then are optimal with respect to $a^T v(C) = a^T v(C')$, where $a = (a_1^T, \ldots, a_k^T)^T$, and that $a^T v(C'') < a^T v(C)$ for any other PSC C''.
Finally, let us consider the case where the cyclical exchange does not involve all k clusters of the clustering. Then we perform above construction analogously for the clusters of the cyclical exchange. For all other clusters C_i, we choose arbitrary sites $a_i \in \mathbb{R}^2$ with $\|a_i\| > 2$, and complement the Voronoi diagram analogously,

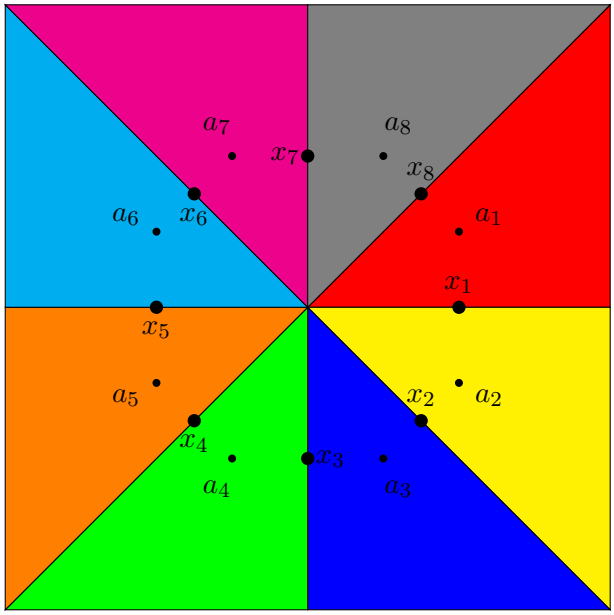


Figure 55: A cyclical exchange $(x_1, \ldots, x_k)$ corresponds to walking along an edge in Q. The different colors represent different cells.

putting the respective items into the interior of the cells created. By $\|a_i\| > 2$, the items $x \in X$ lieing in a C_i 'participating' in the cyclical exchange still satisfy $x \in P_i$ (due to being in the unit ball), and the x_i of the cyclical exchange still are in the common intersection of their respective cells after 'finishing' the Voronoi diagram, as they also lie in the interior of the unit ball. We again obtain a vector a being optimal for both C and C', but for no other PSC. This yields the claim.□

Note that the construction of Lemma 4.36 is valid independently of having fixed cluster sizes.

4.6 Summary and Outlook

In this chapter, we turned to the edge-structure of both the partition polytope and the gravity polytope. In both polytopes, there is a relation of edges to cyclical exchanges. Still, their edge-structure is quite different.
In the partition polytope, every vertex is bijectively associated with a PSC, and edges bijectively belong to cyclical exchanges. This simple structure allowed us to identify an upper bound on the diameter of the partition polytope as the sum of sizes of the two biggest clusters. This bound can be tight. It can be interpreted

as a direct generalization of the diameter bound of the Birkhoff polytope [BR74], and is related to a diameter bound for permutation polytopes [GP06].
The bound was derived by considering a graph-theoretic problem of covering the set of vertices of maximal degree by vertex-disjoint cycles in graphs that decompose into cycles. The constructive proof of this approach yields an efficient algorithm for the calculation of a small set of cyclical exchanges deriving one PSC from another.
It remains to investigate the 'best' data structures (and corresponding operations) for an implementation of this algorithm in practice, and a corresponding 'optimal' running time. Further, the constructive approach can be exploited to satisfy some additional constraints on the transition from one PSC to another. An example for this would be to avoid item movements of specific types by dropping certain edges during the construction of the utility networks used.
The edge-structure of the gravity polytope is more difficult to analyze: First of all, not all PSCs are vertices in the gravity polytope. Only under some (mild) assumptions, edges correspond to certain cyclical exchanges. On the other hand, only some cyclical exchanges correspond to edges. An efficient identification of these cyclical exchanges in the gravity polytope is of high interest with respect to diameter questions, and might contribute to an approach of some of the open problems described in Chapter 3.
Finally, we showed how to transform the partition polytope to the gravity polytope by choosing a vertex set X in $\mathbb{R}^2$ for the items such that a single specified edge is transformed to an edge in the gravity polytope. This is a small step to understanding the transformation connecting the two polytopes. Future work will resolve whether a set of edges in the partition polytope can be transformed to a set of edges in the gravity polytope. A first impression indicates that choosing X in a higher-dimensional space allows us to do so.

5 The Consolidation of Farmland

This chapter is based on [BBG09], a joint work with Prof. A. Brieden and Prof. P. Gritzmann. We extend its contents to a more complete view of the field.
In the first section, a typical problem situation of many agricultural regions is described. In the second section, we turn to the advantages and disadvantages of the classical land consolidation often initiated in such a situation and propose an alternative approach that can be applied individually in practice or can be used to complement the classical methods of land consolidation. Then we turn to some ideas for an economic analysis of the value-added process in an agricultural area, in Section 3. In Section 4, we identify how to transfer these economic measuring ideas to an algorithmic model. In Section 5, we derive a formal problem statement and investigate an integer linear programming approach to it. We further describe how to adapt this approach to derive an efficient near-feasible algorithm. In Section 6, we modify the classical k-means algorithm [Mac67] (see Algorithm 1) to another algorithm for our real-world problem. Finally, we turn to an evaluation of the empirical results of both algorithmic ideas. As we will see, our model leads to flexible and efficient algorithms performing favorably with respect to all relevant economic objective functions given by the Bavarian State Institute for Agriculture.
For earlier work on this real-world problem from a combinatorial optimization point of view, see [Bri03; BG04; BG06]. We refer the reader to an overview of our notational conventions and a list of the symbols used in this chapter in the **Notation and Symbols** appendix.

5.1 Real-World Situation

In many rural communities, farmers cultivate a large number of small-sized lots that are scattered over an extended region. A **lot** is an individually registered piece of land used for agricultural purposes. For many agricultural regions, the geographical information is available in the form of a shapefile [ESR98] used by many geographical information systems (GIS). We refer to these shapefiles as **GIS data**. They are databases that, among other things, capture the geographical location and geometric shape of the lots of a region, their attributes like size or bonity, i.e. quality of soil, and their cultivators. With this information, a visual representation is possible. Figure 56 shows an example. The small bounded areas are the lots of the region, the different colors represent different cultivating

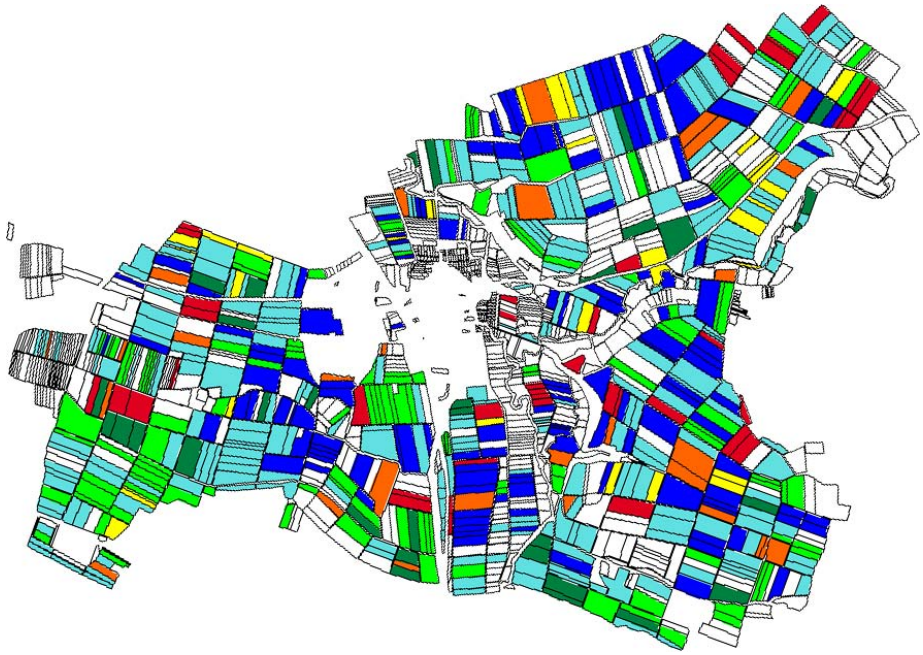


Figure 56: A typical agricultural region. Different colors represent different cultivating farmers.

farmers. To keep data privacy, it is an artifically altered region, yet it closely resembles the situation of many regions in Northern Bavaria.
The problem situation is pictographically clear. The farmers face two main disadvantages. First, their single separate lots are small. When cultivating lots, farming machines need space to turn around. This is why typically there is a small strip of **headland** along the border of the lots where no crop is cultivated. Additionally, the machines consume fuel and may lose dung when turning around. These negative effects are called **headland effects** and **turning effects**. Of course, they are especially bad when the lots have long borders relative to their total size. This is the case if they are small or not of rectangular shape. Then modern heavy machinery cannot be used profitably, and this is why the cost of cultivation of such lots is much higher than it would be for fewer, larger lots of the same total size and bonity. The second main issue is that the farmers' lots are scattered all over the area. Due to this, there is considerable overhead driving. This directly implies additional transportation cost and time. Calculations of the Bavarian State Institute for Agriculture show that the two factors mentioned often add up to more than 30% of the farmers' income coming from their agricultural production.

5.2 Classical Approach and Algorithmic Idea

In a situation like in Figure 56, typically a classical land consolidation process is initiated. Once it is decided on, the farmers of the region are obliged to participate. The process consists of a complete restructuring of the agricultural area. This includes discarding the current lot structure and creating a new one. The process involves extensive legal reassignments of ownership. Additionally, the existing infrastructure may be extended.

While a complete restructuring of the area offers the opportunity to create good solutions for the farmers, there are several disadvantages. Due to the extended surveying and planning needed, it is a lengthy and costly process. Many years may pass before the farmers benefit from the process. There are reports of land consolidation processes of ten or more years. If the process takes such a long time, the results of the replanning vary a lot in quality. Information used may become outdated during the process, and may severely hamper the quality of the resulting agricultural structure.

A further disadvantage is the focus of the classical land consolidation on legal assignments and reassignments of ownership. In agricultural areas where the majority of lots is cultivated by farmers with lend-lease agreements instead of by the owners, any reassignment of the ownership of lots does not directly improve the farmers' cultivation situation. Additionally, lend-lease contracts are more prone to change than the ownership of lots. During a year-long land consolidation process, the cultivation rights might change greatly. Hence, a scattered agricultural structure may remain in regions which undergo a classical kind of land consolidation. Figure 57 shows an agricultural region in Northern Bavaria after a classical land consolidation process. An advantage over the original structure is neither pictographically intelligible, nor appears in the cost structure of the value-added process of the region.

In the following sections we develop an algorithmic approach using the lend-lease agreement-structure to an advantage: A farmer who cultivates a lot that he does not own is generally rather willing to 'trade' his leasing contract. Keeping the existing lot structure, the efficiency of the agricultural production in the area can be improved on by reassigning the cultivators of the lots. A further advantage of this approach is that each farmer of the region can decide individually whether he wants to participate in the process, and if so, can 'fix' some of the lots most important to him, e.g. those close to his home.

Figure 57: The structure of an agricultural region in Northern Bavaria after a classical land consolidation process. The lots of the farmers still are scattered over the region.

Of course, no farmer will accept more than a marginal decrease in total farm size or bonity. Under these constraints, the two algorithms proposed later in this chapter yield 'good' redistributions of the lots. We will examine what a good redistribution of lots looks like in the next two sections, where we first turn to an economic analysis of agricultural regions and then talk about our direct goals for the redistribution.

The running time of the proposed algorithms is less than a minute for typical input sizes, on a typical laptop. The design of these algorithms is accompanied by the development of a software tool called **Agriopt** providing a simple-to-use calculation, editing and visualization platform. Due to the short running time of the algorithms, the possibility of editing lot distributions manually in the tool, and due to the pictographically intelligible visualization, it is easy to demonstrate possible redistributions of the existing lots to the farmers of a community in a meeting. In fact, a series of such meetings was held in two communities in Northern Bavaria. As it turned out, the discussions, propositions by farmers and redistribution tweaking during these meetings greatly improve the quality and reception of the results. For one, farmers sceptical of the methods used often fix most, if not all, of their lots at first. But when they get to see the fairness and power of the methods they decide to participate with more and more lots. Also, special constraints and personal preferences can be respected that are not easily representable abstractly.

5.3 Economic Analysis

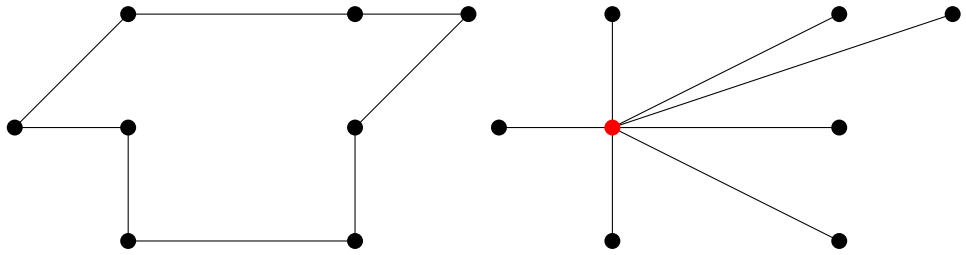
A profound understanding of the economic value-added process in an agricultural region is central to the improvement of its cost structure. This section is dedicated to summarizing such an economic analysis. The ideas presented here were established in a cooperation of Prof. Andreas Brieden and Dr. Paul Rintelen of the Bavarian State Institute for Agriculture. They are implemented in the Agriopt tool.

We are only concerned about the total cost and time required for a fixed plan of crop-growing. Each crop has different properties, and cultivation processes may differ greatly. Lots of higher bonity may be used for different crops than those of low bonity. They may require different machinery and work scheduling. The decision which crop to use a certain lot for may be influenced by which other lots a farmer cultivates. In the following we reduce and simplify the complex processes involved to mathematically analyzable and approximable ones.

Our focus is on two factors: The driving cost and time and the cultivation cost and time.

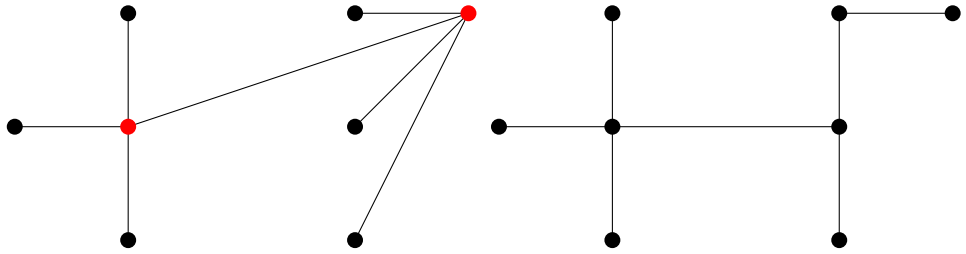
5.3.1 Driving

We use reference points in the Euclidean plane for each lot. A possible measure for the total driving distance of a farmer could be a tour of minimal length through all of his lots. Such a measure would be based on the assumption that the farmer has to inspect all of his lots in a single tour. The Traveling Salesman Problem however is NP-hard [Kar72], as is the Euclidean Traveling Salesman Problem [GGJ76] [Pap77].
Another extreme would be to add up the lengths of the shortest paths from a farmer's base to all of his lots. This would reflect a scenario where each lot requires a full day's work, so there is no synergy in driving from one lot to the other. Another possible measure would be to sum up all distances of a farmer's lots to the closest 'center lot', and to add the distances of a minimum spanning tree of the center lots. This would model the assumption that material is deposited at each center lot and that it then is distributed to all other nearby lots one by one.
In practice each of these scenarios may be realistic, and all of them may represent some of the actual tours done. A good compromise is to relate the driving distance with the length of a minimum spanning tree (MST) for the lots of each farmer: A minimum spanning tree is a lower bound on the lengths of the second and third proposition, and twice its length is an upper bound on the length of a minimal tour. Figure 58 depicts the proposed measures.
An MST of the lots of a farmer C_i can be calculated in $O(\kappa_i^2)$, where κ_i is the number of lots of farmer C_i, by considering the complete graph of lot distances (see e.g. [CT76]). See [Bor06] for a survey of MST algorithms on graphs. Computing a Delaunay triangulation of the reference points of the lots in the Euclidean plane and applying Prim's algorithm to its edges yields a running time of $O(\kappa_i \log(\kappa_i))$ [SH75].
Depending on the cultivated crop, a farmer has to visit his lots a certain number of times per year. We will not distinguish which crop is cultivated on which lot. Taking an average value for the mix of cultivated crops in the agricultural region, we multiply the length of the minimum spanning tree by the average number of tours needed. We assume the driving cost and time to be linear in the



(a) A minimal tour through the lots of a farmer.

(b) The red base lot connected to all other lots.



(c) An MST of the red center lots (a single edge), and stars to the other lots.

(d) An MST of the lots.

Figure 58: The lots of a farmer connected by the four presented measures for the driving distance. The MST in d) is a lower bound on the measures in b) and c) and twice its length is an upper bound on the one in a).

driving distance, allowing several parameters like cost of fuel, machinery aging or driving speed to be adjusted.

5.3.2 Cultivation

The second important measure is the cultivation cost and time. As was mentioned briefly in the starting section of this chapter, bigger lots are less cost- and time-intensive to cultivate than smaller lots of the same total size. This is due to the headland and turning effects, as, in general, the bigger a lot is, the shorter the border is relative to its size. By this, connected lots belonging to the same farmer are a great advantage. The shapes of lots also are important. They usually are polyhedral, but identifying the exact shape of a lot or of several lots connected to each other that can be cultivated together is difficult. Thus, we assume all lots or sets of connected lots of single farmers to have a relation of size and border length as if they were rectangular with a 4 to 1 ratio of length to width. Headland and turning effects then are modeled as regressions based on the size of the lots or total size of connected lots belonging to a single farmer. The regressions differ considerably for different crops. Like with the driving cost and time, we use an average number derived from the mix of cultivated crops in the agricultural region.

5.4 Good Farmland Distributions

We start with some more formal terminology. The lot data is given as part of the GIS data for the region. Therein, the shape of the lots is represented by a set of edges (and vertices) in the Euclidean plane. Each of these edges identifies either one or two lots it belongs to. If an edge identifies two lots, they share the edge as a common border. This means that there is no natural or artifical obstacle between them like a road, hedgerow, or a river. We call such lots **connected** or adjacent. See Figure 59 for an example.
From this representation it is easy to calculate a contiguity graph identifying connected components of lots. We call a set of lots that forms an inclusion-maximal connected subgraph of this graph a **(connected) lot group**. When we additionally talk about the assignments of lots, we use the term **(connected) lot component** for an inclusion-maximal set of lots cultivated by the same farmer forming a connected subgraph in the contiguity graph.

Figure 59: A contiguity graph of connected lots is identified from the edge-structure in the GIS data for the region.

Additionally, lots identify their **size** in hectare and a bonity measure, which is a positive integral number up to 100. We define the **value** of a lot as the product of its size and bonity, which is quite common in agriculture. The **center** of a lot is the arithmetic mean point of the vertices of its representation in the Euclidean plane. We then approximate the **distance** of two lots by the air-line distance of their respective centers.

Farmers will be able to designate **fixed lots**, i.e. lots that are not subject to redistribution. These will play a major role in the algorithms of the following sections as **center lots** which should be integrated into significantly bigger connected lot components after the redistribution.

Let us now turn to our goals for a combinatorial redistribution of lots. As seen in the sections before, the two main characteristics of a good farmland distribution are short driving distances and big lot component sizes. Equivalently to the latter criterion, we can aim for a low number of connected lot components. This directly leads to a large average size of the connected lot components for each farmer.

The driving distances between fixed lots are unavoidable and not subject to any optimization. By minimizing the distances of the farmers' free lots to the center lots, we get a good and easily computable measure for the driving distance (see the previous section).

A redistribution of the lots is only viable if it respects any fixations of lots, and the total size and value of each farmer's lots stays within some attribute bounds, as, of course, no farmer will accept more than a marginal decrease. As the following lemma and corollary show, it generally is NP-hard to decide whether there is another feasible distribution of lots but the original one even if we restrict ourselves to a single attribute and two farmers.

Problem 5.1 (Feasible Lot Distribution)
Given two farmers, a set S of n lots with attributes $s_1, \ldots, s_n \in \mathbb{N}$, $\kappa_1, \kappa_2 \in \mathbb{N}$ with $\kappa_1 + \kappa_2 = \sum_{i=1}^{n} s_i$ and $\epsilon_1^-, \epsilon_1^+, \epsilon_2^-, \epsilon_2^+ \in]0,1[$, decide whether there is a partition (S_1, S_2) of S, such that

$$\begin{aligned}(1-\epsilon_1^-)\kappa_1 \leq \sum_{s\in S_1} s &\leq (1+\epsilon_1^+)\kappa_1\\ (1-\epsilon_2^-)\kappa_2 \leq \sum_{s\in S_2} s &\leq (1+\epsilon_2^+)\kappa_2\end{aligned}$$

Lemma 5.2
Feasible Lot Distribution *is NP-hard.*

Proof. We show that Subset Sum (Given a set S of positive integers and $\sigma \in \mathbb{N}$, decide whether there is a subset of S which sums up to σ), can be polynomially transformed to the given problem. Since, by [Kar72], the former is NP-hard, so is the latter.
To do so, we set

$$\kappa_1 := \sigma, \quad \kappa_2 := -\sigma + \sum_{s\in S} s, \qquad \epsilon_1^- = \epsilon_1^+ = \epsilon_2^- = \epsilon_2^+ := \frac{1}{\sigma+1}.$$

By using the set S as attributes of the lots to be distributed among the two farmers, the constructed instance of the given problem then clearly is a 'yes'-instance (i.e. there is a feasible solution) if and only if the given instance of Subset Sum is a 'yes'-instance. □

Our claim follows from a variant of the above problem.

Problem 5.3 (Feasible Lot Redistribution)
Given two farmers, a set S of n lots with attributes $s_1, \ldots, s_n \in \mathbb{N}$, a partition S_1, S_2 of S with $\kappa_1 = \sum_{s\in S_1} s$, $\kappa_2 = \sum_{s\in S_2} s$ and $\epsilon_1^-, \epsilon_1^+, \epsilon_2^-, \epsilon_2^+ \in]0,1[$, decide whether there is a partition $(S_1', S_2') \neq (S_1, S_2)$ of S, such that

$$\begin{aligned}(1-\epsilon_1^-)\kappa_1 \leq \quad \sum_{s\in S_1^*} s \quad &\leq (1+\epsilon_1^+)\kappa_1\\ (1-\epsilon_2^-)\kappa_2 \leq \quad \sum_{s\in S_2^*} s \quad &\leq (1+\epsilon_2^+)\kappa_2\end{aligned}$$

Corollary 5.4
Feasible Lot Redistribution *is NP-hard.*

Proof. We prove the claim by performing a polynomial transformation of Subset Sum (Given a set S of positive integers and $\sigma \in \mathbb{N}$, decide whether there is a subset of S which sums up to σ) to the given problem analogously to the proof of Lemma 5.2.
To do so, we set

$$\kappa_1 := \sigma, \quad \kappa_2 := \sum_{s\in S} s, \qquad \epsilon_1^- = \epsilon_1^+ = \epsilon_2^- = \epsilon_2^+ := \frac{1}{\sigma+1}.$$

By using a lot s^* with attribute κ_1 and $S' := S \cup \{\kappa_1\}$ as the vector of attributes of the lots to be distributed among the two farmers, (S_1, S_2) with $S_1 = \{s^*\}$ and $S_2 = S$ is clearly a feasible partition of S'. By our construction, another feasible partition (S_1', S_2') satisfies $s^* \in S_2'$, so S_1' only contains values in S, and thus our constructed instance is a 'yes'-instance if and only if the given instance of Subset Sum is a 'yes'-instance. □

Due to the general difficulty of finding feasible redistributions of lots even when only considering a single attribute, and due to the fact that the given bounds are only soft bounds of acceptance by the participating farmers, in the following sections, we will be content with only near-feasible solutions.
Further, if we directly try to optimize both driving distances and the number of lot components, we end up with a multi-criteria optimization problem and, in general, such problems are hard to solve. Thus, we tackle the problem by using a hierarchical approach. We account for the second criterion by not splitting up connected lot groups unless absolutely necessary, and then determine a redistribution of the preprocessed connected lot groups that is nearly optimal with respect to the first criterion. Doing so allows us to construct quick algorithms yielding good results.
In the following two sections, we discuss two algorithmic approaches for calculating good redistributions. We first turn to the core algorithm of the chapter,

where we model the problem as an integer linear program and exploit its 'nice' properties. Thereafter we adapt a classical clustering algorithm to our real-world problem, and finally compare some empirical results of the two algorithms.

5.5 A $0,1$-Integer Programming Approach

In this section, we follow the construction of a $0,1$-integer program in [BBG09]. To obtain a formal, general problem statement, we gather our basic information in an (undirected) weighted graph $G := (V, E, w_V, w_E)$.
For the geographical location of each lot, we calculate a reference point in the Euclidean plane by taking the arithmetic mean of the vertices of its GIS representation. The resulting reference points $v_1, ..., v_n \in \mathbb{R}^2$ constitute the vertices of G. A vector-valued weight function $w_V : V \to (\mathbb{R}^+)^d$ with $d \in \mathbb{N}$ models the attributes of the lots. In our real-world application we have $d = 2$, using the size and value of lots as attributes. For a subset $C \subset V$, $w_V(C)$ is defined to denote the componentwise sum of the weights of all vertices contained in C.
The graph G is complete, i.e. E is the set of all two-element subsets of V. We use the function $w_E : E \to \mathbb{R}^+$ to represent the distances between the respective lots. In the simplest case, it just identifies the Euclidean distance of the respective reference points, i.e. $w_E(e) = \|v - w\|$ for $e = \{v, w\}$. If we have more detailed information about the 'real' distances, like from the border of one lot to the border of another lot, or that natural obstacles between two lots result in longer driving distances, we define w_E according to this information.
Referring to the wording and notation of the earlier chapters, the parts of a partition P of V will be called **clusters**, P itself is a **clustering**. Let k denote the number of participating farmers, and suppose that the i-th farmer fixes $\mu_i \in \mathbb{N}$ center lots. We assume that each farmer designates a non-empty set of center lots that are not subject to redistribution. If, in practice, a farmer does not designate such a lot, we identify his most valuable one, and use it as his only center lot. All other lots of the same farmer should be close to one of his center lots. Conceptually, this does not only imply small driving distances, but also rewards adjacent lots being assigned to the same farmer.
We represent the center lots of the i-th farmer by a set $\{c_{i1}, \ldots, c_{i\mu_i}\}$, where $c_{il} = v_{l'}$ for some $l' \in \{1, \ldots, n\}$. For each lot c_{ij}, we create a cluster C_{ij}. We call $C_i := \bigcup_{j \in \{1,\ldots,\mu_i\}} C_{ij}$ the i-th **cluster group**. It contains all clusters of the i-th farmer.

Let $\kappa_i \in (\mathbb{R}^+)^2$ denote the vector that contains the original farm size and value of the i-th farmer as components. Let further ϵ_i^- and $\epsilon_i^+ \in (\mathbb{R}^+)^2$ denote lower and upper bounds on the relative tolerance that the i-th farmer accepts with respect to these two measures.
Using the notation $\mathbb{1} = (1, \ldots, 1)^T \in \mathbb{Z}^d$ and $(a_1, \ldots, a_d)^T \circ (b_1, \ldots, b_d)^T \to (a_1 b_1, \ldots, a_d b_d)^T$, we can write these (vector-valued) tolerance constraints as $(\mathbb{1} - \epsilon_i^-) \circ \kappa_i \leq \sum_{j=1}^{\mu_i} w_V(C_{ij}) \leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i$.
Our goal is to minimize the driving distances in our objective function. Recall that for an approximation of the driving distances in an economic evaluation, we choose the MST of all lots of a farmer. At this point, this measure is too difficult and indirect, so we turn to the simple-to-model measure depicted in Figure 58 c). Note that the minimum spanning tree of the center lots are unavoidable distances. Thus, it suffices to only look at the sum of distances of a farmer's lots to their respective center lot.
For a center lot $c_{ij} = v$, we call the set of edges $\{\{c_{ij}, w\} : w \in C_{ij}, w \neq c_{ij}\}$ the **star** of c_{ij}. We minimize the sum of edge-weights $\sum_{i=1}^{k} \sum_{j=1}^{\mu_i} \sum_{w \in C_{ij}} w_E(\{c_{ij}, w\})$ in these stars of all fixed vertices c_{ij}, respecting the above balancing constraints. The following problem statement summarizes all necessary information.

Problem 5.5 (Constrained Minimum-*k*-Star Clustering (CSC))
Let $G = (V, E, w_V, w_E)$ be a complete weighted graph with $V := \{v_1, ..., v_n\}$ and $w_E : E \to \mathbb{R}^+$ with $w_E(\{v, v\}) = 0$ for $v \in V$. Let $\kappa_1, \ldots, \kappa_k \in (\mathbb{R}^+)^d$ with $d \in \mathbb{N}$ and $w_V : V \to (\mathbb{R}^+)^d$ with $\sum_{i=1}^{k} \kappa_i = w_V(V)$. Let further $\mu_1, ..., \mu_k \in \mathbb{N}$ and $\{c_{11}, \ldots, c_{1\mu_1}, \ldots, c_{k1}, \ldots, c_{k\mu_k}\} \subset V$. Finally, let $\epsilon_1^{\pm}, \ldots, \epsilon_k^{\pm} \in (\mathbb{R}^+)^d$.
Then the problem is to compute a partition $P = (C_{11}, \ldots, C_{1\mu_1}, \ldots, C_{k1}, \ldots, C_{k\mu_k})$ of V into clusters with $c_{ij} \in C_{ij}$ for $i \in \{1, \ldots, k\}, j \in \{1, \ldots, \mu_i\}$, and

$$(\mathbb{1} - \epsilon_i^-) \circ \kappa_i \leq \sum_{j=1}^{\mu_i} w_V(C_{ij}) \leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i \quad (i \in \{1, \ldots, k\})$$

such that

$$val(P) := \sum_{i=1}^{k} \sum_{j=1}^{\mu_i} \sum_{w \in C_{ij}} w_E(\{c_{ij}, w\})$$

is minimal among all such partitions.

With our notation, it is possible to model CSC as an integer linear program, as follows.

Lemma 5.6
CSC can be modeled as an integer linear program.

Proof. The decision variables

$$x_{ijl} \in \{0,1\} \quad (i \leq k,\, j \leq \mu_i,\, l \leq n)$$

are used to identify whether the vertex v_l belongs to C_{ij} ($x_{ijl} = 1$), or not ($x_{ijl} = 0$). Since each vertex must be assigned to exactly one cluster, we have the constraints

$$\sum_{i=1}^{k} \sum_{j=1}^{\mu_i} x_{ijl} = 1 \quad (l \leq n).$$

Each cluster C_{ij} is associated with a single center vertex $c_{ij} = v_l$. Naturally, $c_{ij} \in C_{ij}$. We model these restrictions by fixing the corresponding variables, i.e.

$$x_{ijl} = 1 \quad (i \leq k, j \leq \mu_i, l \leq n : c_{ij} = v_l).$$

Finally, the vector-valued size restrictions read

$$(\mathbb{1} - \epsilon_i^-) \circ \kappa_i \leq \sum_{j=1}^{\mu_i} \sum_{l=1}^{n} x_{ijl} w_V(v_l) \leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i \quad (i \leq k)$$

while the objective function becomes

$$\min \sum_{i=1}^{k} \sum_{j=1}^{\mu_i} \sum_{l=1}^{n} x_{ijl} w_E(\{c_{ij}, v_l\}).$$

Summed up, we have the following 0-1-integer linear program (ILP):

$$\begin{aligned}
&\min \sum_{i=1}^{k} \sum_{j=1}^{\mu_i} \sum_{l=1}^{n} x_{ijl} w_E(\{c_{ij}, v_l\}) \\
(\mathbb{1} - \epsilon_i^-) \circ \kappa_i \leq &\sum_{j=1}^{\mu_i} \sum_{l=1}^{n} x_{ijl} w_V(v_l) \leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i && (i \leq k) \\
&\sum_{i=1}^{k} \sum_{j=1}^{\mu_i} x_{ijl} = 1 && (l \leq n)
\end{aligned}$$

$$x_{ijl} = 1 \qquad (i \leq k, j \leq \mu_i, l \leq n \text{ s.t. } c_{ij} = v_l)$$
$$x_{ijl} \in \{0,1\} \qquad (i \leq k, j \leq \mu_i, l \leq n). \qquad \square$$

As a direct corollary of Lemma 5.2 and Corollary 5.4, we know that CSC is NP-complete.

Corollary 5.7
When restricted to all rational instances, CSC is NP-hard. The hardness persists if the instances are restricted to those with $k = 2$ and $d = 1$.

Proof. We only need to reword the proof of Lemma 5.2 slightly. $\square$

On the other hand, CSC is related to the following problem SSP, which allows an exact polynomial-time algorithm for fixed k. It was studied in [GBH98].

Problem 5.8 (Size-restricted Minimum-k-Star Partition (SSP))
Let $G = (V, E, w_E)$ be a complete weighted graph with $V := \{v_1, ..., v_n\}$ and $w_E : E \to \mathbb{R}^+$ with $w_E(\{v, v\}) = 0$ for $v \in V$. Let $\kappa_1, ..., \kappa_k \in \mathbb{N}$ with $\sum_{i=1}^{k} \kappa_i = n$.
Then the problem is to compute k vertices $c_1, ..., c_k \in V$ and a partition $P = (C_1, \ldots, C_k)$ of V with $|C_i| = \kappa_i$ and $c_i \in C_i$ for $i \in \{1, \ldots, k\}$, such that

$$val(P, c_1, \ldots, c_k) := \sum_{i=1}^{k} \kappa_i \sum_{v \in C_i} w_E(\{c_i, v\})$$

is minimal among all such partitions and choices of $c_1, ..., c_k$.

While related, there are several obvious and significant differences between CSC and SSP. In SSP, we do not consider any vertex-weights, and thus only have cardinality constraints. Further, we do not have cluster groups, but instead just single clusters, to which these cardinality constraints apply.
On the other hand, while we fix the center vertices beforehand in CSC, an optimal choice for the center vertices is part of the output of SSP. The key observation of the proof of the polynomial running time of an exact algorithm is that once the center vertices $c_1, \ldots, c_k$ of the k stars are chosen, the remaining problem is a transportation problem, hence can be solved in polynomial time as an LP. There are $\binom{n}{k}$ choices for $c_1, ..., c_k$. Thus, the number of transportation problems that have to be solved is $O(n^k)$ yielding a polynomial-time algorithm for fixed k.

Due to its running time being exponential in k, this algorithm is, however, not practical for even moderate problem sizes. Additionally, its assumptions are too restrictive for an application to our real-world problem.
Noting that SSP can be solved in polynomial time for any fixed k, and recalling that CSC is NP-hard, we know that the differences between CSC and SSP have to create this tractability gap. From Corollary 5.7 (and the proofs of Lemma 5.2 and Corollary 5.4), we know that the numerical weights alone already are a reason for CSC to be NP-hard.
We now extend SSP to use cluster groups instead of single clusters to investigate their role in the running time. The following problem is such a generalization of SSP, and by this already much closer to our practical application, yet it still avoids numerical weights. It also accounts for the fact that some vertices may be fixed to be in certain cluster groups. We have one cluster for each fixed vertex, but we still choose the center vertices of the clusters 'freely'.
As it turns out, this combinatorial problem with all vertices having uniform weight is as close as we can get to practical land consolidation before the problem becomes NP-hard. This ties the NP-hardness of CSC exactly to the use of numerical weights for the vertices.

Problem 5.9 (Size-restricted Minimum-k-Star Group Partition (SGP))
Let $G = (V, E, w_E)$ be a complete weighted graph with $V := \{v_1, ..., v_n\}$ and $w_E : E \to \mathbb{R}^+$ with $w_E(\{v, v\}) = 0$ for $v \in V$. Let $\kappa_1, ..., \kappa_k \in \mathbb{N}$ with $\sum_{i=1}^{k} \kappa_i = n$. Let further $\mu \in \mathbb{N}$, $\mu_1, \ldots, \mu_k \in \mathbb{N}$ with $\mu = \sum_{i=1}^{k} \mu_i$, and $\{v_{11}, \ldots, v_{1\mu_1}, \ldots, v_{k1}, \ldots, v_{k\mu_k}\} \subset V$. Finally, let $\epsilon_1^{\pm}, \ldots, \epsilon_k^{\pm} \in \mathbb{R}^+$.
Then the problem is to compute vertices $c_{11}, \ldots, c_{k\mu_k} \in V$, subcluster sizes $\sigma_{11}, \ldots, \sigma_{k\mu_k}$ with

$$(1 - \epsilon_i^-)\kappa_i \leq \sum_{j=1}^{\mu_i} \sigma_{ij} \leq (1 + \epsilon_i^+)\kappa_i \quad (i \in \{1, \ldots, k\})$$

and $\sum_{i=1}^{k} \sum_{j=1}^{\mu_i} \sigma_{ij} = n$, and a partition $P = (C_{11}, \ldots, C_{1\mu_1}, \ldots, C_{k1}, \ldots, C_{k\mu_k})$ of V with $|C_{ij}| = \sigma_{ij}$ and $c_{ij}, v_{ij} \in C_{ij}$ for $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, \mu_i\}$, such that

$$val(P, c_{11}, \ldots, c_{k\mu_k}, \sigma_{11}, \ldots, \sigma_{k\mu_k}) := \sum_{i=1}^{k} \sum_{j=1}^{\mu_i} \sigma_{ij} \sum_{v \in C_{ij}} w_E(\{v, c_{ij}\})$$

is minimal among all such partitions, choices of $c_{11}, \ldots, c_{k\mu_k}$, and choices of $\sigma_{11}, \ldots, \sigma_{k\mu_k}$.

Indeed, SGP can still be solved in polynomial time.

Lemma 5.10
For fixed μ, all feasible combinations of cluster sizes $\sigma_{11}, ..., \sigma_{k\mu_k}$ in SGP can be detected in polynomial time.

Proof. We trivially have $1 \leq \sigma_{ij} \leq n$, so n^μ is an upper bound on the number of feasible combinations of cluster sizes.
The subcluster sizes $\sigma_{11}, ..., \sigma_{k\mu_k}$ are feasible if and only if $\sum_{i=1}^{k} \sum_{j=1}^{\mu_i} \sigma_{ij} = n$ and $(1 - \epsilon_i^-)\kappa_i \leq \sum_{j=1}^{\mu_i} \sigma_{ij} \leq (1 + \epsilon_i^+)\kappa_i$ for $i \in \{1, \ldots, k\}$. Hence, feasibility can be decided by means of $2k + 1$ arithmetic tests per combination. □

Theorem 5.11
SGP is solvable in polynomial time for fixed μ.

Proof. By Lemma 5.10, there is a polynomial number of feasible subcluster sizes $\sigma_{11}, ..., \sigma_{k\mu_k}$. Also, there are at most $\binom{n}{\mu}$ different combinations of subcluster center choices. Thus, it suffices to solve a polynomial number of problems where the σ_{ij} and c_{ij} are fixed, and we proceed by showing that each of these problems can be solved in polynomial time. Of course, each problem with fixed σ_{ij} and c_{ij} can be formulated as an ILP.

$$
\begin{aligned}
\min \sum_{i=1}^{k} \sum_{j=1}^{\mu_i} \sigma_{ij} \sum_{l=1}^{n} x_{ijl} w_E(\{v_l, c_{ij}\}) & \\
\sum_{l=1}^{n} x_{ijl} = \sigma_{ij} & \qquad (i \leq k, j \leq \mu_i) \\
\sum_{i=1}^{k} \sum_{j=1}^{\mu_i} x_{ijl} = 1 & \qquad (l \leq n) \\
x_{ijl} = 1 & \qquad (i \leq k, j \leq \mu_i, l \leq n \text{ s.t. } v_{ij} = v_l) \\
x_{ijl} = 1 & \qquad (i \leq k, j \leq \mu_i, l \leq n \text{ s.t. } c_{ij} = v_l) \\
x_{ijl} \in \{0, 1\} & \qquad (i \leq k, j \leq \mu_i, l \leq n)
\end{aligned}
$$

Let A denote the coefficient matrix, so that the ILP reads

$$\min c^T x \qquad \text{s.t.} \quad Ax = b,\ x \in \{0, 1\}^{\mu n}.$$

We show that A is totally unimodular, so that we know that it suffices to solve its LP-relaxation, where the $0,1$-condition is replaced by $0 \leq x$ (see e.g. [Sch98]). Due to the constraints of type $\sum_{i=1}^{k} \sum_{j=1}^{\mu_i} x_{ijl} = 1$ and due to $x_{ijl} \geq 0$, the constraint $x \leq 1$ is redundant and can be omitted.
The matrix $A = (A_1^T, A_2^T, A_3^T)^T$ decomposes into three parts. A_1 fixes the size of each C_{ij} to σ_{ij}, A_2 forces all vertices to be assigned to exactly one cluster and A_3 fixes some individual vertices to a corresponding cluster.
A matrix M is totally unimodular if and only if $\binom{M}{E}$ is totally unimodular, where E denotes the unit matrix of appropriate size. Since A_3 is just a subset of the rows of the unit matrix, it thus suffices to only consider $A' := \binom{A_1}{A_2}$. In the $0,1$-matrix A', each column has exactly two 1-entries, one in A_1 and one in A_2. Thus A_1 and A_2 provide a partition of the rows of A' satisfying a well-known criterion [HT56] for total unimodularity of A'. □

We get the following direct corollary.

Corollary 5.12
When restricted to instances with $\mu = \sum_{i=1}^{k} \mu_i$ fixed and uniform vertex-weights, CSC is solvable in polynomial time.

Informally, in the case of uniform-weight vertices, i.e. lots of equal size, and a fixed number of center vertices, i.e. fixed lots, CSC can be solved in polynomial time. Still, the running time of the algorithm described in the proof of Theorem 5.11 is prohibitive for all relevant real-world problem sizes. Additionally, the assumption of having lots of equal size is much too restrictive for an application in practice.
However, the above results show the way to a practical algorithm for the consolidation of farmland described in the general CSC. The main idea is a trade-off of running time and feasibility of the redistribution: The relative tolerances $\epsilon_i^{\pm}$ of the farmers' original attributes provide only soft constraints. In practice, it is enough to only aim for near-feasibility of our solution. Knowing this, we will obtain a fast algorithm that performs very well in practice and achieves all relevant goals in the consolidation of farmland, namely a reduction of the driving distances for the farmers, and a reduction of the cost of cultivation of lots by creating larger connected lot components.
We begin with some 'preprocessing' to simplify the underlying ILP. The conditions $x_{ijl} = 1$ or respectively $x_{ij'l} = 1$ both mean that the vertex v_l belongs to C_{ij} or respectively $C_{ij'}$. However, C_{ij} and $C_{ij'}$ are subclusters of the same

cluster group C_i. The balancing constraints do not distinguish in which of the two subclusters v_l lies. An optimal solution with respect to our objective function only considers its distance to the closest of the centers c_{ij} with $j \in \{1, ..., \mu_i\}$. This implies that our problem can be modeled by only considering these lowest edge-weights to some cluster center in C_i. Thus it suffices to only use two-index decision variables $x_{il} \in \{0, 1\}$ with $i \in \{1, \ldots, k\}, l \in \{1, \ldots, n\}$ that identify whether vertex v_l belongs to cluster C_i. The new weight function $w' : (V \times \{1, ..., k\}) \to \mathbb{R}^+$ then is defined by

$$w'(v_l, i) := \min_{j \in \{1,\ldots,\mu_i\}} w_E(\{c_{ij}, v_l\}).$$

This preprocessing can be performed in $O(n^2)$ time, as it suffices to consider the distance $w_E(v_i, v_j)$ only once for $i \neq j;\ i, j \in \{1, \ldots, n\}$. We obtain the following ILP.

$$\min \sum_{i=1}^{k} \sum_{l=1}^{n} x_{il} w'(v_l, i)$$

$$(\mathbb{1} - \epsilon_i^-) \circ \kappa_i \leq \sum_{l=1}^{n} x_{il} w_V(v_l) \leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i \qquad (i \leq k)$$

$$\sum_{i=1}^{k} x_{il} = 1 \qquad (l \leq n)$$

$$x_{il} = 1 \qquad (i \leq k, l \leq n \text{ s.t. } c_{ij} = v_l)$$

$$x_{il} \in \{0, 1\} \qquad (i \leq k, l \leq n)$$

In this model, the goal of minimizing driving distances is represented directly in the objective function. Since the distances between the fixed (center) lots of a farmer are unavoidable and not subject to any optimization, we minimize the distances of the farmers' free lots to their closest center lots. Our second goal of generating a low number of connected lot components for each farmer is only represented indirectly by this objective function. While it is obvious that looking for low driving distances will usually lead to many adjacent lots being assigned to the same farmer, we want to reward such assignments further. To do so, we use some instance-level constraints and modified lot distances in our hierarchical approach.

Suppose there is a center lot v_l that is connected to a lot $v_{l'}$. Additionally, there is a lot $v_{l''}$ of size equal to that of $v_{l'}$, not connected to v_l, e.g. on the other side of a road, but with slightly smaller geographical distance to v_l. Of course, we would then like to assign lot $v_{l'}$ to the cultivator of v_l with priority. Above model would prefer $v_{l''}$ over $v_{l'}$. We modify our model by introducing distance weights that prefer lots in the same connected lot group.

Another, rather extreme, idea is to only assign connected lot groups to farmers as a whole. This certainly reduces the problem size dramatically. However, it means that if one of the lots of the connected lot group is fixed by a farmer, the whole group has to be assigned to that farmer. Naturally, there is a conflict whenever two farmers fix lots in the same connected lot group that can only be resolved by somehow breaking up the respective group. But even if no such conflicts occur, such a radical approach generally leads to a prohibitive imbalance.

In practice, we obtain the best results from combining the two ideas above: We reward the assignment of lots belonging to the connected lot group of a center lot of a farmer to this farmer by a distance factor less than 1. In addition, we try to assign connected lots to single farmers. However, rather than working with complete connected lot groups, we split them into several connected components that can be assigned individually. In particular, each fixed lot becomes an individual component to avoid the 'hard' conflicts mentioned above. Additionally, components consisting of inclusion-maximal sets of connected non-fixed lots are created. We then always assign all lots of a component to the same farmer. Note that this is the coarsest possible partition of the lots into such components, and that it is not difficult to realize this partition in a more sophisticated way. Other ways of creating these components have different effects on our optimization goals. Algorithm 14 describes the steps taken in pseudo-code.

We denoted Algorithm 14 using the same variables as before. However, except for one, each variable referring to the same farmer and the same connected lot component is redundant. In practical computations, these are left out.

For typical smaller problem sizes, the reduced ILP has about 5000 decision variables, and can be solved very quickly to optimality with the aid of a state of the art branch-and-cut approach.

Of course, in its basic form, Algorithm 14 may still create a too large deviation in the balancing constraints. We now turn to this issue in more detail. Additionally, we address the issue of how to proceed if the instances are too large to solve the ILP directly.

Input : GIS data for the lots of an agricultural region, original distribution of lots to cultivators, partition into connected lot groups
Output: Near-feasible 'optimal' redistribution of lots

Preprocess w':

- Compute the Euclidean distances between the lot coordinates
- Scale distances that belong to the same connected lot group by a factor less than one, yielding the distance function w_E
- Set $w'(v_l, i) := \min_{j \in \{1,\ldots,\mu_i\}} w_E(\{c_{ij}, v_l\})$ for all $l \leq n, i \leq k$

Determine the partition $\mathcal{C} = \{comp_1, \ldots, comp_u\}$ of lots into connected lot components:

- Fixed lots create a component on their own
- Inclusion-maximal sets of non-fixed connected lots form a component

Solve the following ILP:

$$\min \sum_{i=1}^{k} \sum_{l=1}^{n} x_{il} w'(v_l, i)$$

$$(\mathbb{1} - \epsilon_i^-) \circ \kappa_i \leq \sum_{l=1}^{n} x_{il} w_V(v_l) \leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i \qquad (i \leq k)$$

$$\sum_{i=1}^{k} x_{il} = 1 \qquad (l \leq n)$$

$$x_{il} = 1 \qquad (i \leq k, l \leq n \text{ s.t. } c_{ij} = v_l)$$

$$x_{il} = x_{il'} \qquad (l, l', c \leq u \text{ s.t. } v_l, v_{l'} \in comp_c)$$

$$x_{il} \in \{0, 1\} \qquad (i \leq k, l \leq n)$$

return partition P associated with the lowest objective function value;

Algorithm 14: Consolidation of Farmland (ILP)

A useful approach for the latter problem is to work with the LP-relaxation. In the reduced formulation where the decision variables y_{ic} signify the assignment of connected lot components and the corresponding distances are

$$w''(comp_c, i) := \sum_{v_l \in comp_c} w'(v_l, i),$$

the LP-relaxation takes the following form:

$$\begin{aligned}
&\min \sum_{i=1}^{k}\sum_{c=1}^{u} y_{ic} w''(comp_c, i) \\
(\mathbb{1} - \epsilon_i^-) \circ \kappa_i \leq \sum_{c=1}^{u} y_{ic} w_V(comp_c) &\leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i && (i \leq k) \\
\sum_{i=1}^{k} y_{ic} &= 1 && (c \leq u) \\
y_{ic} &= 1 && (i \leq k, c \leq u, l \leq n \\
& && \text{s.t. } c_{ij} = v_l \in comp_c) \\
y_{ic} &\geq 0 && (i \leq k, c \leq u)
\end{aligned}$$

Using this relaxation, we will in general obtain only fractional solutions. Fractional variables $0 < y_{ic} < 1$, respectively $0 < x_{il} < 1$ signify assignments of fractional parts of lots to farmers. In practice, it may be ok to split the cultivation rights of some lots, and drawing new lot boundaries, yet this is in conflict with our purely combinatorial approach to keep the original lot structure chosen as a foundation of our algorithm.
The good news is that we can keep the number of such fractional assignments so small that they can be handled in a suitable postprocessing step [Bri03].

Lemma 5.13
Let y^* be an optimal vertex of the polytope P of feasible points of the LP-relaxation. Then at most $2k$ connected lot components are not assigned integrally by y^*.

Proof. In each vertex of P at least $k \cdot u$ constraints are active. Of course, the u constraints $\sum_{i=1}^{k} y_{ic} = 1$ are active, by definition. Also, each fixation constraint $y_{i'c} = 1$ for some $i' \leq k$ already implies $\sum_{i=1}^{k} y_{ic} = 1$.

Further at most half of the $4k$ componentwise constraints

$$(\mathbb{1} - \epsilon_i^-) \circ \kappa_i \leq \sum_{c=1}^{u} y_{ic} w_V(comp_c), \quad \sum_{c=1}^{u} y_{ic} w_V(comp_c) \leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i,$$

can be active if $(\epsilon_i^+)_j + (\epsilon_i^-)_j > 0$ for $i \leq k$ and $j \leq 2$, where $(\epsilon_i^\pm)_j$ denotes the j-th component of $\epsilon_i^\pm$. Otherwise, for each i and j with $(\epsilon_i^\pm)_j = 0$, one of the constraints is redundant.

All other active constraints are of the type $y_{ic} \geq 0$. This implies that at most $u + 2k$ of the y_{ic} are not 0 since otherwise the number of active constraints is less than $k \cdot u$.

Now, let us show that at most $4k$ of them are less than 1, hence fractional. For this, let s denote the number of fractional assignments i.e. $0 < y_{ic} < 1$, and let t be the number of assignments $y_{ic} = 1$. Suppose that $s > 4k$. Since it takes at least 2 of the s fractional variables to satisfy an equality $\sum_{i=1}^{k} y_{ic} = 1$ that involves a fractional variable we have $t \geq u - \frac{s}{2}$. Hence $u + 2k \geq s + t \geq s + u - \frac{s}{2} = u + \frac{s}{2} > u + 2k$, a contradiction. So there are at most $4k$ fractional decision variable assignments y_{ic}, and thus at most $2k$ connected lot components not assigned uniquely to a farmer. □

In general, rounding fractional variables optimally to integrality is an NP-hard task. Since we have only very few fractionally assigned lots, and since we are content with a near-feasible solution, we can use the following conceptual strategy. We solve the LP-relaxation and fix all integral lot assignments. Then we distribute the fractionally assigned lots by solving a suitable ILP to optimality (or by means of some heuristic). The design of the method used depends on the goals we want to optimize. Figure 60 shows the connected lot components that are assigned fractionally in one of the LP-relaxations we calculated for the agricultural region shown in Figure 56.

An example for a very simple heuristic is based on the relative loss of value for each farmer in the partial assignment given by the integral components of the solution produced by the LP-relaxation. Successively, a farmer with the greatest relative loss then gets a lot of the not-yet assigned ones that puts his value as close as possible to his original value.

Algorithm 15 sums up the steps taken. As we only solve a single LP, and then either solve a very small ILP or apply a heuristic to a very low number of lots, this algorithm runs very quickly, in a couple of seconds for typical input sizes.

Input : GIS data for the lots of an agricultural region, original distribution of lots to cultivators, partition into connected lot groups
Output: Near-feasible 'optimal' redistribution of lots

Preprocess w':

- Compute the Euclidean distances between the lot coordinates
- Scale distances that belong to the same connected lot group by a factor less than one, yielding the distance function w_E
- Set $w'(v_l, i) := \min_{j \in \{1,\ldots,\mu_i\}} w_E(\{c_{ij}, v_l\})$ for all $l \leq n, i \leq k$

Determine the partition $\mathcal{C} = \{comp_1, \ldots, comp_u\}$ of lots into connected lot components:

- Fixed lots create a component on their own
- Inclusion-maximal sets of non-fixed connected lots form a component

Solve the following LP:

$$\min \sum_{i=1}^{k} \sum_{c=1}^{u} y_{ic} w''(comp_c, i)$$

$$(\mathbb{1} - \epsilon_i^-) \circ \kappa_i \leq \sum_{c=1}^{u} y_{ic} w_V(comp_c) \leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i \qquad (i \leq k)$$

$$\sum_{i=1}^{k} y_{ic} = 1 \qquad (c \leq u)$$

$$y_{ic} = 1 \qquad (i \leq k, c \leq u, l \leq n \text{ s.t. } c_{ij} = v_l \in comp_c)$$

$$y_{ic} \geq 0 \qquad (i \leq k, c \leq u)$$

Let P be the partition associated with the lowest objective function value;
Fix all integrally assigned lot components in P (for a partition P');
Distribute the remaining lot components assigned fractionally in P using some heuristic to derive P' and return P';

Algorithm 15: Consolidation of Farmland (Relaxed)

Figure 60: Connected lot components assigned fractionally by a LP-relaxation.

In practice, it returns a farmland redistribution that keeps farmer values about the same, but typically yields a considerable reduction in driving and cultivation cost. We will perform an economic evaluation of some empirical results towards the end of this chapter.

5.6 A k-means Approach

In this section, we discuss an approach to the clustering problem explained in Sections 5.4 and 5.5 using an adaption of the k-means algorithm [Mac67], see Algorithm 1.

The fundamental idea of each iteration of the k-means algorithm is to assign the lots closest to a virtual center to this center, and by that to the same cluster. Intuitively it is clear that, in our application, this will lead to clusterings where lots close to each other are likely to be put in the same cluster. By this, we derive a clustering which will be favorable both with respect to the driving distances of the farmers as well as the creation of large lot components for the farmers. Due to this, k-means is a valid approach to our real-world problem.

Note that k-means is a deterministic algorithm, but the output clustering depends on the starting centers chosen. Instead of doing so randomly, we choose one starting center for each center lot, with the geometric coordinates of the

center lot. In the first iteration of the algorithm, we thus directly measure the distances of the lots to these center lots. By this, we emphasize that we want to integrate the center lots into big lot components. Additionally, like in our integer programming approach, we further emphasize our desire for large connected lot components by assigning inclusion-maximal sets of non-fixed connected lots as a whole. Note that we cannot apply a scaling factor to the distances of the lot components in a lot group in the same way as in the $0, 1$-integer approach.
Note further that in a basic version (Algorithm 1, Algorithm 2), k-means does not consider any size constraints on the clusters. To do so, instead of simply calculating which center a lot component is closest to and performing the canonical association, we now iteratively add the lot components to the clusters. First, we order the lot components descendingly by their value. Before a lot component is added to a cluster, we then perform multiple layers of checks as follows.
We start by assigning the fixed center lots to the respective farmers. We track the total 'current possession' of all farmers with respect to the different attributes of the lots that were assigned to them. When looking for the farmer to assign a lot component to, we at first only consider all farmers whose current total lot attributes neither sum up to their lower accepted bound of total value nor to their lower accepted bound of size yet. Additionally, we only consider those farmers for which the total lot value and size does not rise above any of their upper accepted bounds of deviation if we assign the lot component to them. We then assign the lot component to the closest center of such a farmer.
After distributing most of the lot components, we cannot guarantee that there is such a farmer anymore. We first drop the restriction that we do not want a farmer to get significantly more lot value or size than he originally had, and choose the one whose upper deviation in an attribute is smallest after the assignment. If we still find no farmer we can assign the current lot component to, this means that all farmers are above the lower deviation bounds for at least one lot attribute. We then try to assign the lot component to a farmer who already has 'enough' lots, but whose upper possession deviation in an attribute is smallest after the assignment. Note that the farmer might still have less than his original possession with respect to each attribute after the assignment, so that these types of assignments still may yield feasible redistributions of the lots.
On the other hand, we cannot guarantee the feasibility of the resulting clustering. The assignment of whole lot components together may imply that there actually is no feasible redistribution. Additionally, another problem arises if the bonity

of the lots varies greatly: Recall that the value of a lot is defined as its size times its bonity measure. If the bonity of the lots differs a lot, the two measures for size and value are not directly (linearly) correlated to another. Then many assignments will be feasible with respect to only one of the two measures, while being infeasible with respect to the other. Still, problems will only arise during the assignment of the last few lots at the end of the process. Recall that we ordered the lot components by descending value, so, especially with respect to this measure, the last lots to be assigned are rather small. This fact, together with the choices of the assignment restrictions and the order in which they are dropped lead to near-feasibility with respect to the soft bounds of deviation in practice. Algorithm 16 sums up the steps taken in pseudo-code.
We investigate the performance of this approach with an example in the next section, and compare it with our $0, 1$-integer approach.

5.7 Empirical Results

Let us now give an impression of the practical performance of our two algorithms. While we have performed our computations for many real-world data sets and have worked together with farmers on successfully implementing our approach in practice, we will here restrict ourselves to artificially altered input to keep data privacy. The example region depicted in Figure 56 does, however, represent a situation that is typical for many Bavarian communities. It is further small enough to be pictographically intelligible in the available resolution. In this example, we have 7 farmers cultivating a total of 651 lots, which are scattered over a large area.
We use (rather restrictive) 3% as the accepted soft bounds of deviation for each farmer in our calculations. This results in our solutions 'trying' to be very close to the original possession of the farmers. By this approach, even if we have only near-feasible solutions, we will have deviations that, in practice, generally will be accepted by the farmers. In the following, we will show a number of lot distributions calculated and investigate the feasibility or near-feasibility of our solutions. Further, we will estimate the economic advantages of our redistributions in Table 1 and 2 towards the end of this section. The cost measures in the tables are derived according to the ideas described in Section 5.3.
We start by applying our two algorithms with 32% fixed lots. We begin with such a high percentage for various reasons. First, we want to indicate how the quality

Input : GIS data for the lots of an agricultural region, original distribution of lots to cultivators, partition into connected lot groups
Output: Near-feasible 'optimal' redistribution of lots

Choose μ centers $c_{11}, \ldots, c_{k\mu_k}$ as the coordinates of the center lots of the farmers;
Order the free lot components by descending value to derive $\mathcal{C} = \{comp_1, \ldots, comp_u\}$ and calculate their representing coordinates in $\mathbb{R}^2$ as an arithmetic mean of the lots in the component;
while *Lot component assignment changed during the last iteration and a fixed number of iterations is not reached yet* **do**
    Define a vector $\sigma := (\sigma_1^T, \ldots, \sigma_k^T)^T \in \mathbb{R}^{2 \cdot k}$ of current size σ_{i_1} and value σ_{i_2} of each farmer as the sum of sizes and values of their fixed lots;
    for $1 \leq c \leq u$ **do**
        Calculate distance $w'(comp_c, c_{ij})$ of $comp_c$ to each center;
        Find the farmer C_i with the closest center c_{ij} with respect to $w'(comp_c, c_{ij})$, with C_i satisfying $\sigma_i \leq (\mathbb{1} - \epsilon_i^-) \circ \kappa_i$ and $\sigma_i + w_v(comp_c) \leq (\mathbb{1} + \epsilon_i^+) \circ \kappa_i$;
        if *such a farmer C_i exists* **then**
            Assign $comp_c$ to the j-th cluster of C_i;
            $\sigma_i = \sigma_i + w_v(comp_c)$;
        end
        else
            Find the farmer C_i satisfying $\sigma_i \leq (\mathbb{1} - \epsilon_i^-) \circ \kappa_i$ and for which $\sigma_i + w_v(comp_c) \leq (\mathbb{1} + \epsilon) \circ \kappa_i$ holds for minimal ϵ;
            if *such a farmer C_i exists* **then**
                Assign $comp_c$ to the j-th cluster of C_i, if $w'(comp_c, c_{ij})$ is minimal for C_i;
                $\sigma_i = \sigma_i + w_v(comp_c)$;
            end
            else
                Find the farmer C_i for which $\sigma_i + w_v(comp_c) \leq (\mathbb{1} + \epsilon) \circ \kappa_i$ holds for minimal ϵ;
                Assign $comp_c$ to the j-th cluster of C_i, if $w'(comp_c, c_{ij})$ is minimal for C_i ;
                $\sigma_i = \sigma_i + w_v(comp_c)$;
            end
        end
    end
    Recalculate the centers $c_{11}, \ldots, c_{k\mu_k}$ as arithmetic means of the lots associated with the respective clusters;
end
return redistribution of lots according to the assignment of the lots to the farmers' centers;

Algorithm 16: Consolidation of Farmland (k-means)

of the reassignment depends on the degree of freedom one allows. Second, we want to show that even with only little flexibility, one can already obtain some improvement in the cost structure. This is an important feature since, in practice, farmers at first tend to fix many lots, while later, when they are confident in the fairness of the method and its performance, many additional lots will be subject to redistribution.

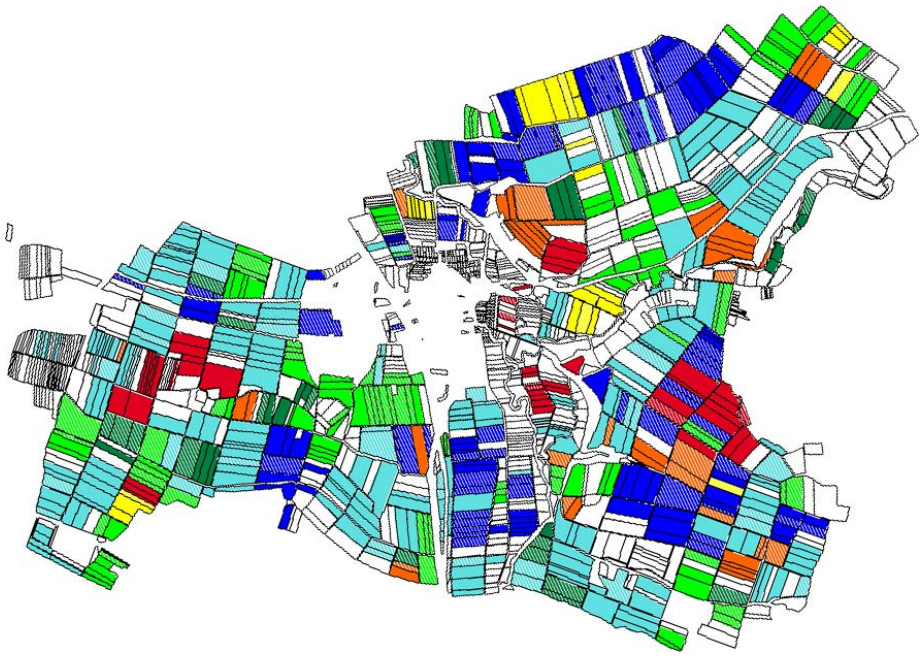
Figure 61 $a)$ shows the redistribution of lots returned by our $0,1$-integer approach. Note that all farmers participate in the redistribution process. Of course, with such a high number of fixations there is no wide margin for improvement. Also, with many of his lots fixed, a farmer has a large number of center lots. These create many small clusters scattered over the area. The same effect is visible in the resulting distribution of our k-means approach, see Figure 61 $b)$. But while the lots still look pretty scattered, both Table 1 and Table 2 already show some significant economic improvement over the original assignment.

A similar, slightly better effect is recognizable if we only have 24% fixed lots, see Figure 62. The lots already are a bit less scattered, and the economic measures indicate an even greater improvement than before.

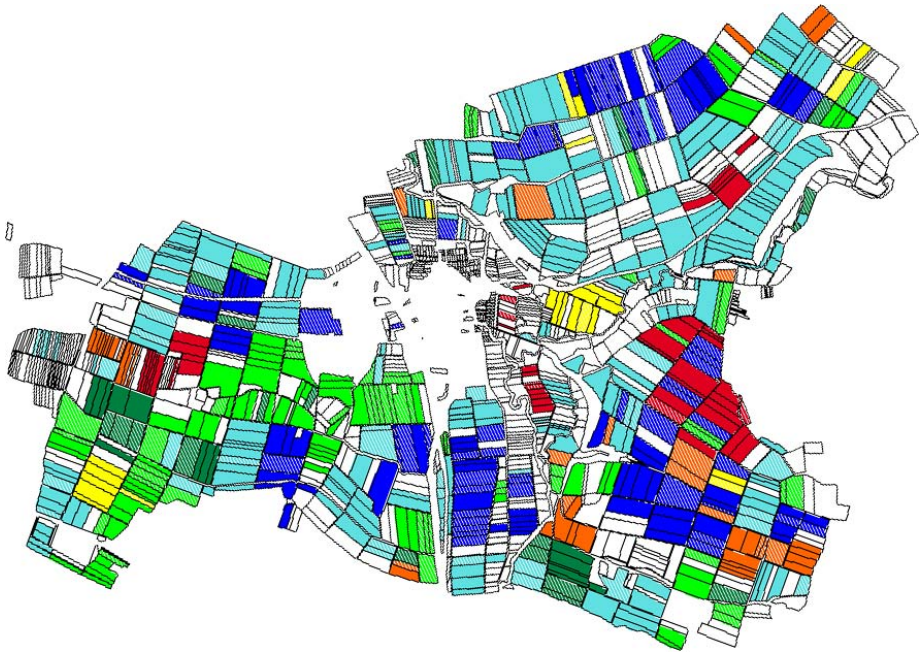
Figure 63 shows redistributions for 16% fixed lots, and Figure 64 redistributions for 8% fixed lots. Even on a first visual impression, the lots look substantially less scattered than in the version with 32% fixed lots. Tables 1 and 2 confirm this impression, which is not surprising due to the increased number of 'tradeable' lots.

Finally, Figure 65 shows two calculated redistributions of lots for 4% fixed lots, and Figure 66 shows the calculated redistributions where only a single lot is fixed for each farmer. The pictographical impression and economic measures improve even further.

Let us now relate our results for our two algorithms. Especially in the examples with a low percentage of fixations, the k-means algorithm yields redistributions of lots that leave a slightly better visual impression than the ones of our $0,1$-integer approach. This is due to two reasons: First, the main idea of k-means is to associate fields with centers that are close to them in the Euclidean plane. Doing so directly creates large areas of lots assigned to single farmers, and these are the ones that intuitively induce the good visual impression. Second, our k-means approach is significantly worse with respect to the deviations of farmer size than our $0,1$-integer approach, thus 'allowing' itself better-looking solutions.



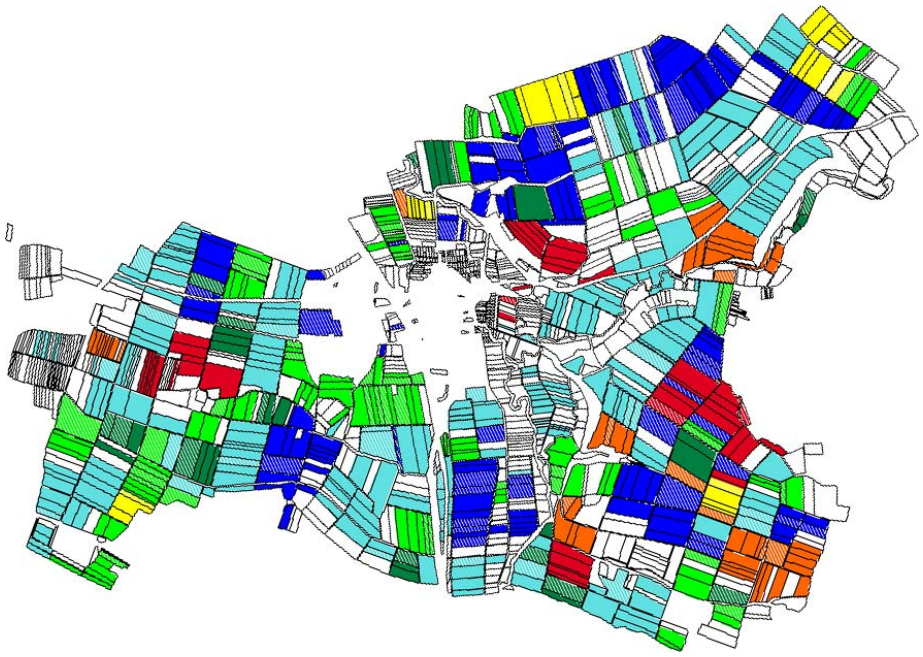
(a) 0, 1-Integer Program



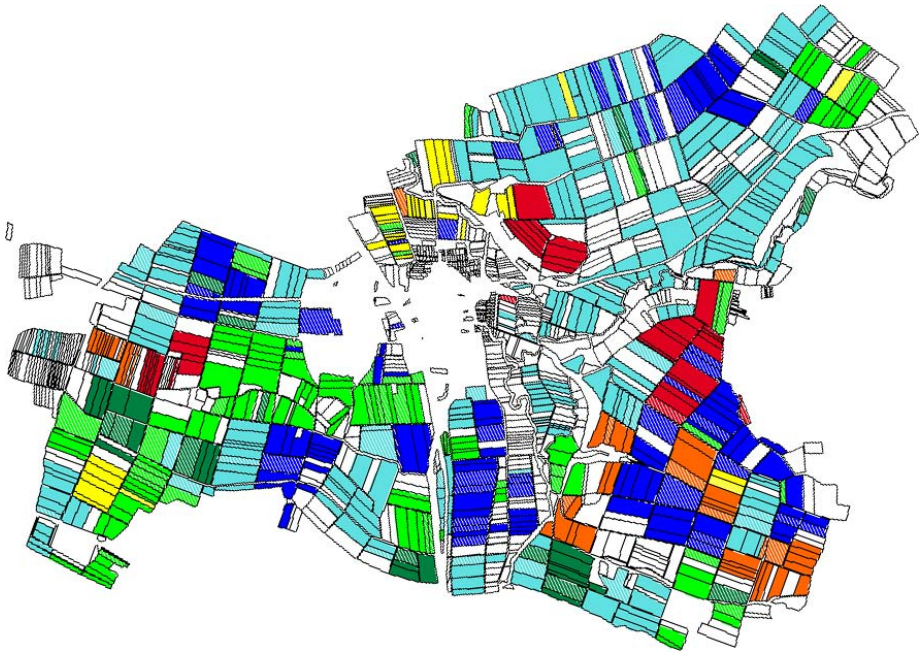
(b) Constrained k-means

Figure 61: The redistribution of lots returned by our two algorithms for 32% fixed lots. Fixed lots are shaded.

.



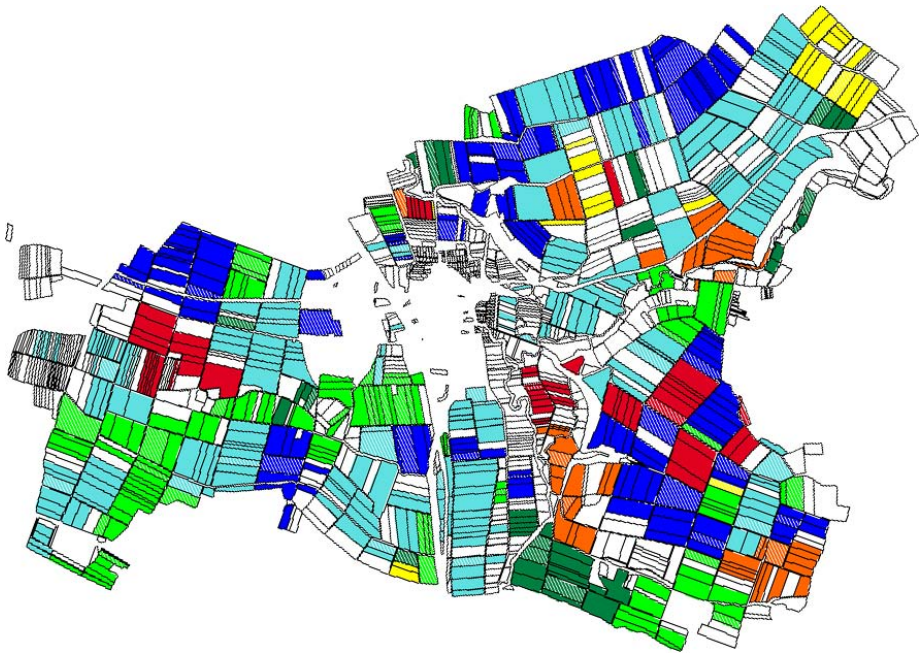
(a) 0, 1-Integer Program



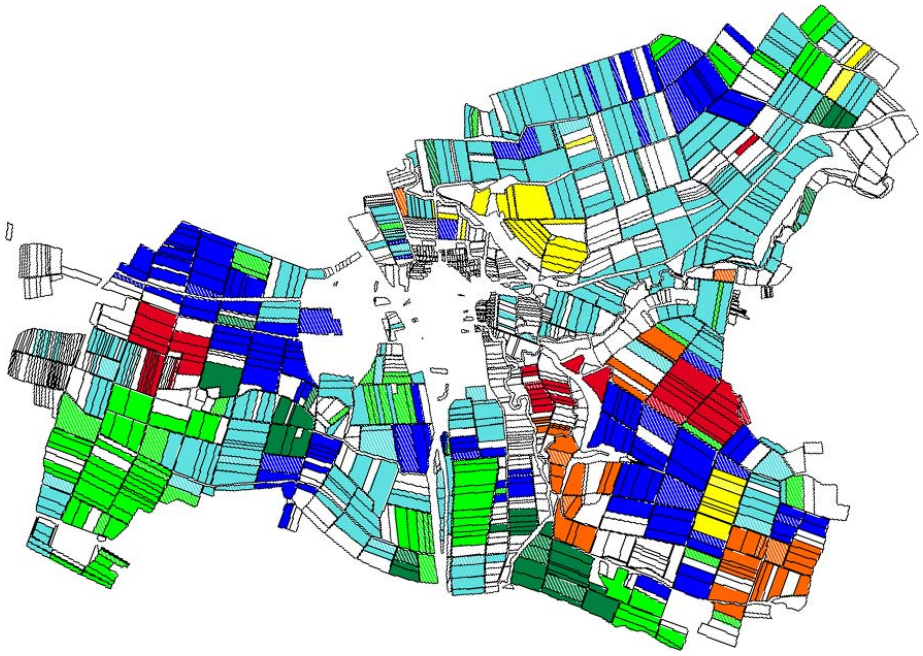
(b) Constrained k-means

Figure 62: The redistribution of lots returned by our two algorithms for 24% fixed lots. Fixed lots are shaded.

.



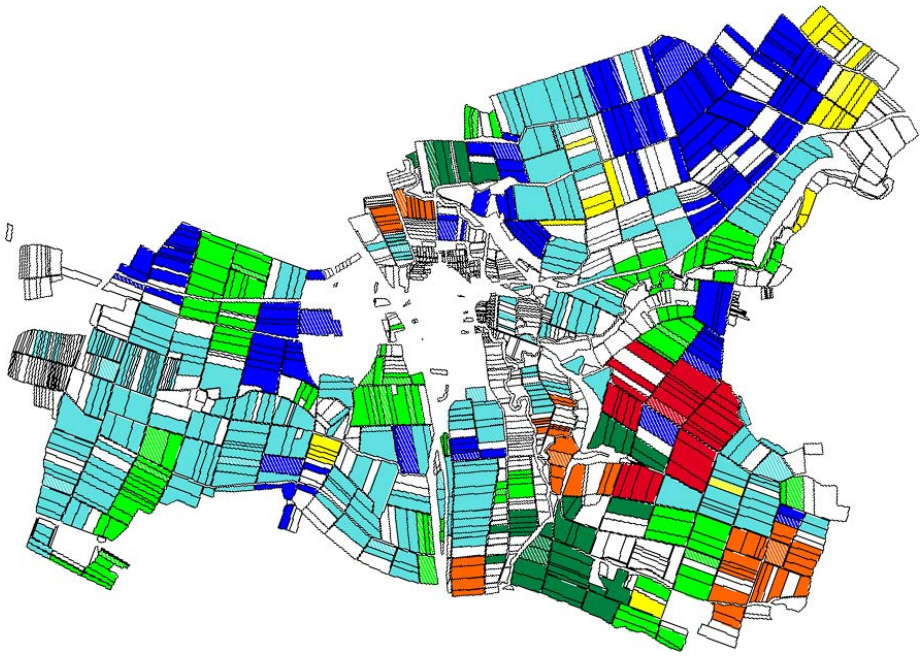
(a) 0, 1-Integer Program



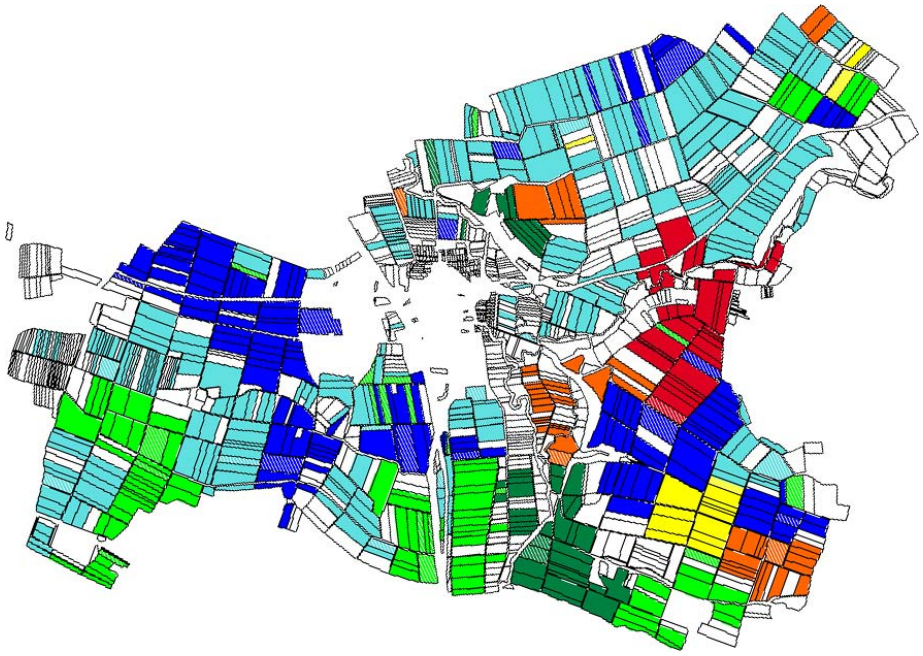
(b) Constrained k-means

Figure 63: The redistribution of lots returned by our two algorithms for 16% fixed lots. Fixed lots are shaded.

.



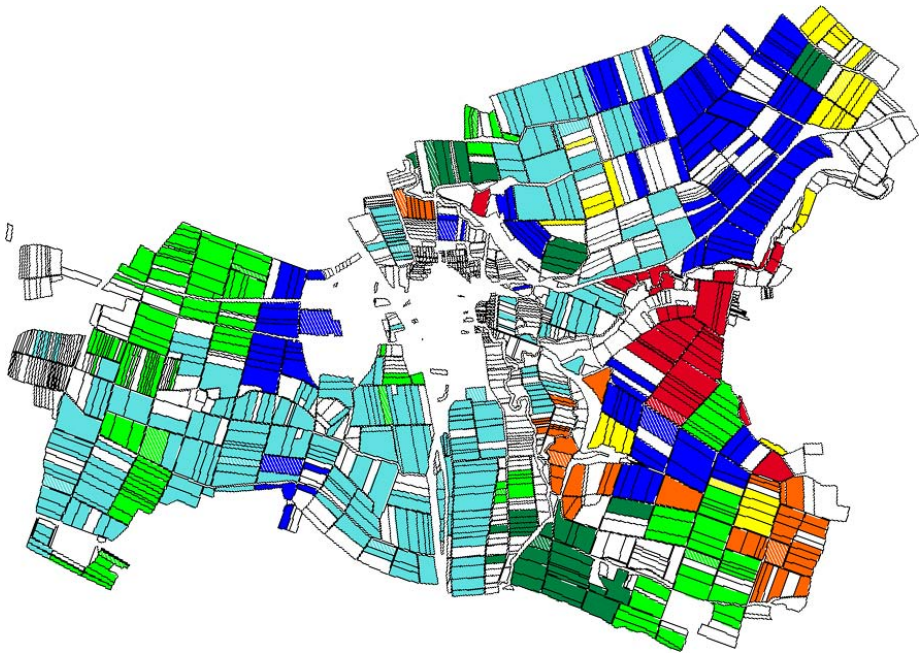
(a) 0, 1-Integer Program



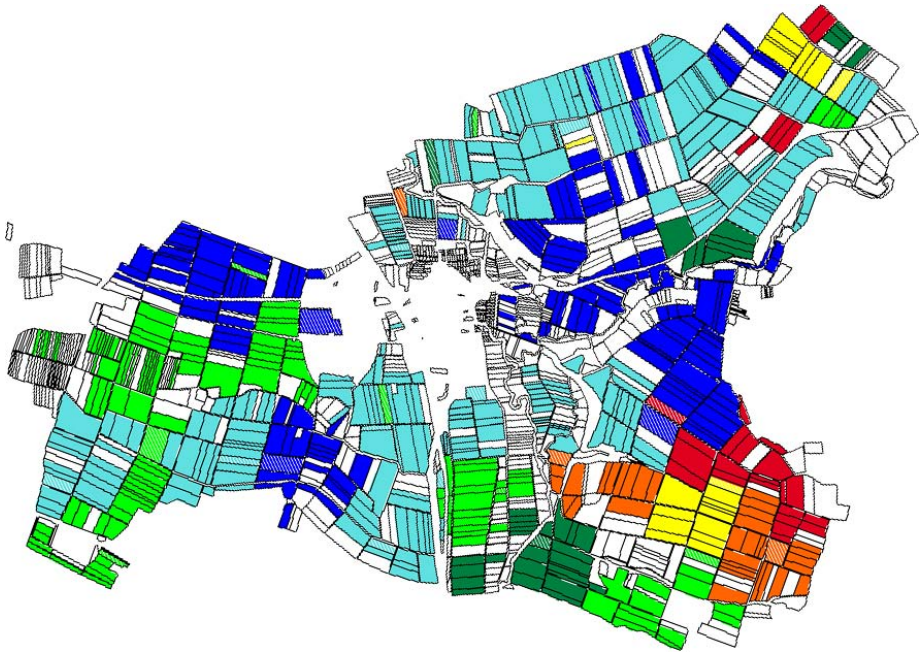
(b) Constrained k-means

Figure 64: The redistribution of lots returned by our two algorithms for 8% fixed lots. Fixed lots are shaded.

.



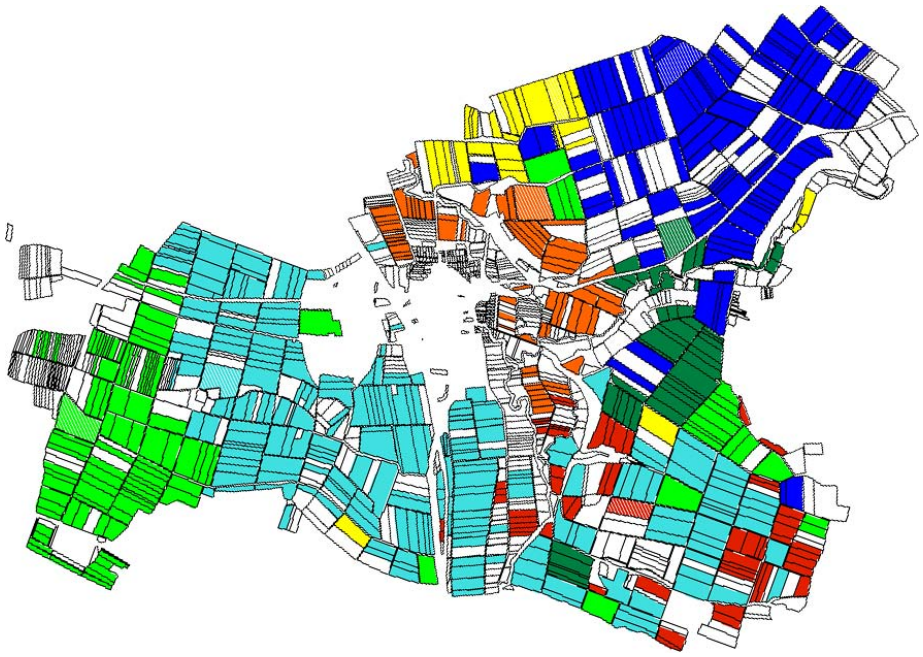
(a) $0, 1$-Integer Program



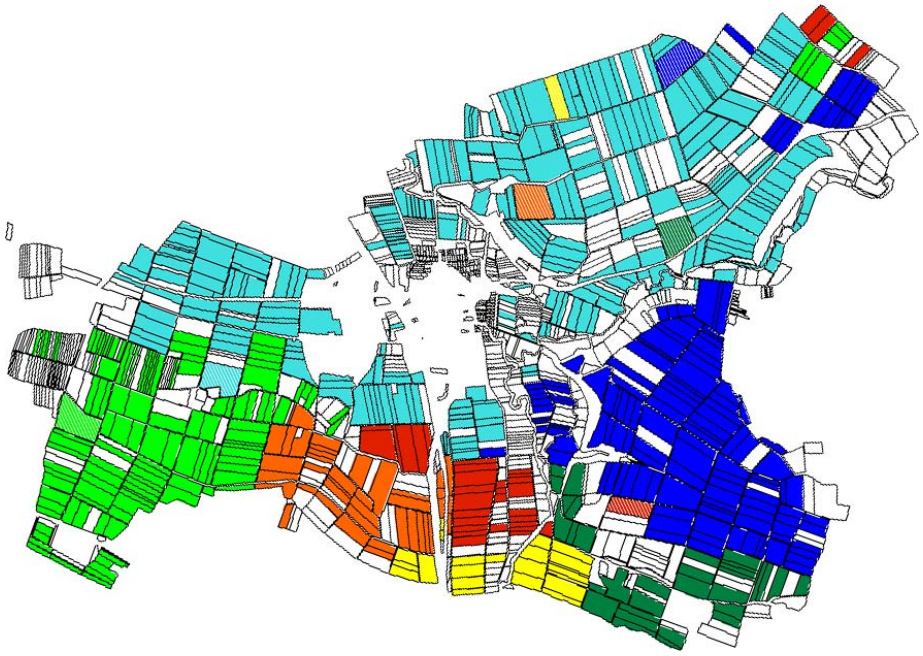
(b) Constrained k-means

Figure 65: The redistribution of lots returned by our two algorithms for 4% fixed lots. Fixed lots are shaded.

.



(a) 0, 1-Integer Program



(b) Constrained k-means

Figure 66: The redistribution of lots returned by our two algorithms for only one single fixed lot per farmer (about 1%). Fixed lots are shaded.

.

Note that the quality of our two algorithms with respect to the feasibility or near-feasibility of the solutions is not correlated to the percentage of fixations. In our $0,1$-integer model, it only depends on the number of farmers (see Lemma 5.13). Similarly, our k-means heuristic does not consider this information either. Let us take a look at the farmer value deviations of the solutions of our algorithms, not distinguishing the different fixations. In the six examples calculated, our $0,1$-integer approach yields a feasible solution two times. The worst-case example has a deviation of $-5,2\%$ for a very small farmer's value. Typically, deviations are less than -4%, and (in five out of the six examples) at most a single farmer is below the lower threshold of -3% with respect to his total size or value in each example. More often than not, these numbers are accompanied by an increase in the respective other measure for the farmer's possession. These are very good numbers with respect to an application in practice. Other data sets confirm the impression gained that the $0,1$-integer approach returns redistributions of the lots 'good enough' (with respect to the near-feasibility) to be applied in practice. The deviations of the k-means approach are somewhat worse. Out of the six calculations, no redistribution satisfies hard feasibility with respect to our soft bounds of deviation. Between one and three farmers usually are below the lower threshold of deviation, with worst-case deviations being around -9%. Again, more often than not, farmers that have a loss of total lot value higher than the accepted deviation gained in total lot size. Performing our heuristic only considering a single attribute (like the value) yields significantly better results with respect to feasibility. Still, this approach has to be improved on with respect to the feasibility of the returned redistributions of lots before an application in practice is realistic.

The most important aspect of quality of our redistribution next to the realizability of our solutions in practice is the advantage we obtain with respect to the economic measures. Table 1 lists some of the measures for our $0,1$-integer solutions, Table 2 lists them for the k-means approach.

The columns 'total cost', 'cultivation cost' and 'driving cost' list the percentage of reduction in the respective cost measure (approximated like described in section 5.3) from our original lot assignment. The column 'number of lot components' contains the percentage of reduction in the total number of connected lot components of all farmers. For the entries in the column titled 'possible reduction in lot components', we first calculate the minimal number of lot components that our region must contain due to its structure. It depends on the number of connected

fixed lots	total cost	cultivation cost	driving cost	number of lot components	poss. reduction in lot comp.
32%	$-8,76\%$	$-8,06\%$	$-27,07\%$	$-32,08\%$	$86,31\%$
24%	$-9,73\%$	$-8,95\%$	$-30,25\%$	$-35,40\%$	$92,49\%$
16%	$-10,84\%$	$-9,83\%$	$-37,15\%$	$-38,50\%$	$95,08\%$
8%	$-11,53\%$	$-10,26\%$	$-44,50\%$	$-40,27\%$	$96,30\%$
4%	$-11,65\%$	$-10,27\%$	$-47,81\%$	$-40,27\%$	$96,30\%$
1%	$-12,26\%$	$-10,77\%$	$-50,91\%$	$-42,04\%$	100%

Table 1: Measures for the quality of the LP-method redistributions for different percentages of lot fixations. (See Algorithm 15.)

fixed lots	total cost	cultivation cost	driving cost	number of lot components	poss. reduction in lot comp.
32%	$-8,34\%$	$-7,72\%$	$-24,45\%$	$-29,65\%$	$79,76\%$
24%	$-8,87\%$	$-8,09\%$	$-29,18\%$	$-30,97\%$	$80,92\%$
16%	$-9,00\%$	$-9,01\%$	$-34,87\%$	$-31,20\%$	$77,05\%$
8%	$-10,27\%$	$-9,05\%$	$-42,14\%$	$-35,40\%$	$84,66\%$
4%	$-11,09\%$	$-9,76\%$	$-45,76\%$	$-38,27\%$	$91,53\%$
1%	$-12,12\%$	$-10,58\%$	$-52,31\%$	$-41,37\%$	$98,43\%$

Table 2: Measures for the quality of the k-means redistributions for different percentages of lot fixations. (See Algorithm 16.)

lot groups and on the chosen fixations. Then we compare the reduction in the number of lot components we achieve to the (theoretically) maximally possible reduction to this lower bound.

Note how lowering the number of fixed lots affects the quality of our solution positively. Note also that even with the highest degree of fixations, we already achieve significant improvements over the original assignment. Note further that the $0,1$-integer results are generally slightly better than the ones of the (sometimes pictographically better) k-means approach. We observed the same effect for different agricultural regions. The most significant differences are in the number of connected lot components created. For the biggest part, this is due to the advantage of the $0,1$-integer approach being able to assign a scaling factor to decrease the distance of lot components in the same lot group, which we cannot do in the k-means approach.

It is clear why our $0,1$-integer approach does so well with respect to our economic measures, recalling that we transferred our economic measuring ideas directly to derive the algorithmic model. Interestingly, the numbers favor the $0,1$-integer results over the k-means results, in contrast to the first visual impression gained and to the fact that it performs better with respect to the farmers' size and value deviations. As an example, reconsider the lot distribution calculated by the k-means approach in Figure 66 b). While most of a farmer's lots are very close to each other, some of the fixed center lots lie very far from the rest of the farmer's lots. During the k-means iterations, the coordinates of the centers moved further and further away from the coordinates of the center lots.
The running time of our algorithms for these examples was less than a couple of seconds on a typical laptop. Similar results were obtained for a large number of practical instances. Generally, the $0,1$-integer approach yields lower farmer size and value deviations and better economic measures, and as such, it is our approach of choice in practice.
With a view towards future applications of even larger problem sizes, let us point out again that the proposed algorithm decomposes into a large LP and (e.g.) a small subsequent ILP. The LP can be solved efficiently within the standard running time of state-of-the-art LP-algorithms and is hence capable of handling instances of millions of variables and restrictions. As to the ILP, recall that only $2k$ lot components are fractionally assigned by the LP, where k denotes the number of participating farmers. Hence the ILP is small and can also be solved quickly for all relevant practical problem sizes.

5.8 Summary and Outlook

In this chapter, we investigated the economic cost structure of agricultural regions. Based on this model, we developed two algorithms for the consolidation of farmland based on a combinatorial redistribution of the lots in a fixed, existing lot structure. The main algorithm of the chapter is a $0,1$-integer approach, where we use the power of relaxation and heuristic rounding to obtain near-feasible solutions of significantly better economic measures, ready to be applied in practice.
The main open questions concerning this approach center around the rounding used. Generally, rounding a fractional solution to a 'good' integer solution is a NP-hard task. It remains to investigate whether the special structure of our problem allows us to do so efficiently. A possible first step might be to consider

whether defixing some of the lots assigned fractionally by the LP step allows us to achieve better results with our rounding routine. Near-feasibility instead of feasibility most often appears when the farmers involved in the consolidation process have lots of vastly different total value. Especially the smallest farmers are prone to lose or gain significantly more than their desired lower or upper bound of deviation. Due to this, we are working on a multi-step approach, where we first calculate a redistribution of lots for the farmers of low total value. Only then we fix these farmers' lots and continue with the farmers of higher total value.

The second algorithm of this chapter is a sophisticated adaption of the classical k-means algorithm. While the results of this second algorithm are pictographically impressing, problems with farmer size and value deviations and worse economic measures make it less useful in practice than our $0,1$-integer approach. Open questions include whether it is possible to adapt the heuristic constraints of the k-means approach to get closer to feasible redistributions of the lots and to obtain better economic measures for the redistributions.

Land consolidation is a wide field, and is not only of interest in the context of the consolidation of farmland, but also in the forest and similar areas. Different types of lots imply different types of constraints. With our abstract underlying model, we are investigating its applicability to some related fields. Additionally, we consider an application of our algorithmic approach to other clustering problems outside of the field of consolidation, like to the scheduling of room planning in a facility. All of these ideas are accompanied by extensions of the Agriopt software tool which unifies and visualizes our algorithmic results.

Notation and Symbols

Whenever we want to emphasize that we define a data structure A to be equivalent to some B, we write $A := B$. When this fact is clear, we just use the 'normal' notation $A = B$.
When considering an index set $I := \{i_1, \ldots, i_p\}$, for the sake of simplicity, we use $i_0 = i_p$ and $i_{p+1} = i_1$. Additionally, we denote an index range of $i \in \{1, \ldots, t\}$ as $i \leq t$, and $i \in \{t, \ldots, t'\}$ as $t \leq i \leq t'$ for a simple-to-read description of the constraints of (linear) programs.
The following paragraphs contain an informal (non-exhaustive) list of the most important symbols used in this thesis. We start with general ones, and then list the symbols according to the chapter in which they are defined. Note that Chapters 3 and 4 use a lot of the symbols defined in Chapter 2, while Chapter 5 mostly is a stand-alone chapter with respect to the notation used.

General Symbols

$\mathbb{N}$	the set of natural numbers
$\mathbb{R}$, $\mathbb{R}^+$	the set of real numbers, respectively strictly positive real numbers
$[a, b]$	the closed interval of real numbers between a and b
$d \in \mathbb{N}$	the dimension of a vector space
$\mathbb{R}^d$	the d-dimensional Euclidean vector space over the real numbers
v	a vector in a vector space
$\|v\|$	the Euclidean norm of v
$\delta(v, v')$	the Euclidean distance of v and v'
$H_{a,\beta}$	the hyperplane of vectors $x \in \mathbb{R}^d$ with $a^T x = \beta$ for $a \in \mathbb{R}^d$ and $\beta \in \mathbb{R}$
$H^{\leq}_{a,\beta}, H^{\geq}_{a,\beta}$	the halfspace of vectors $x \in \mathbb{R}^d$ with $a^T x \leq \beta$, respectively $a^T x \geq \beta$
$H^{<}_{a,\beta}, H^{>}_{a,\beta}$	the strict halfspace of vectors $x \in \mathbb{R}^d$ with $a^T x < \beta$, respectively $a^T x > \beta$
$N(v)$	the cone of outer normals for v with respect to some geometric body

int, bd	the interior, respectively boundary of a geometric body
relint, relbd	the relative interior, respectively relative boundary of a geometric body
dim	the dimension of a geometric body
span	the span of a set of vectors
$G = (V, E)$	a graph with vertex set V and edge set E
CY	a cycle in a graph
$O(\cdot)$	the Landau-symbol
const	a constant value in $\mathbb{R}$

Symbols of Chapter 2

$n \in \mathbb{N}$	the number of data vectors
$X = \{x_1, \ldots, x_n\}$	a set of (geometric) data vectors in $\mathbb{R}^d$
k	the number of clusters
$C = (C_1, \ldots, C_k)$	a k-clustering of X
C_i	the i-th cluster of C
$\|C_i\|$	the size, i.e. the number of items of C_i
$\|C\| = (\|C_1\|, \ldots, \|C_k\|)$	the shape of C
$\kappa_1, \ldots, \kappa_k$	the prescribed-number of items in clusters $C_1, \ldots, C_k$
PSC C	a prescribed-shape clustering C, i.e. a k-clustering of prescribed shape $(\kappa_1, \ldots, \kappa_k)$
$v = v(C)$	the gravity vector of clustering C
$\mathcal{C}$	the set of gravity vectors of PSCs $\mathcal{C} = \mathcal{C}(X, k, \kappa_1, \ldots, \kappa_k)$
Q	the gravity polytope $Q = Q(X, k, \kappa_1, \ldots, \kappa_k)$
$a = (a_1^T, \ldots, a_k^T)^T$	a vector in $\mathbb{R}^{d \cdot k}$ with $a_i \in \mathbb{R}^d$ for all $i \in \{1, \ldots, k\}$
$a_\kappa = (\frac{a_1}{\kappa_1}^T, \ldots, \frac{a_k}{\kappa_k}^T)^T$	a vector in $\mathbb{R}^{d \cdot k}$ with $a_i \in \mathbb{R}^d$ for all $i \in \{1, \ldots, k\}$
$\|I\|$	the number of indices of an index set I
CE	a cyclical exchange $CE = (x_{i_1}, \ldots, x_{i_t})$ for an index set $I = \{i_1, \ldots, i_t\}$ and $x_{i_j} \in C_{i_j}$ for all $i_j \in I$
$\mathcal{CE} = (CE_1, \ldots, CE_t)$	a set of cyclical exchanges
$a \perp b$	symbolic notation of two vectors a, b being orthogonal

$a \| b$	symbolic notation of two vectors a, b being collinear
$H_1 \| H_2$	symbolic notation of two hyperplanes H_1, H_2 being parallel to each other, i.e. having collinear normal vectors
$\mathcal{H}$	a k-cell arrangement
a_{ij}	the vector $a_{ij} = a_j - a_i$ or $a_{ij} = \frac{a_j}{\kappa_j} - \frac{a_i}{\kappa_i}$, depending on a construction from a or a_κ
γ_{ij}	the (right-hand side)-value of a hyperplane in direction a_{ij} separating clusters C_i and C_j
H_{ij}	the hyperplane $H_{a_{ij},\gamma_{ij}}$
$\mathcal{P} = (P_1, \ldots, P_k)$	a cell decomposition
P_i	the i-th cell of $\mathcal{P}$
LSA	a least-squares assignment of X to a with respect to $\kappa_1, \ldots, \kappa_k$
(a, W)-power diagram	a power diagram for vector of sites a and set of weights W
W	a set of weights $W = \{w_i : i \in \{1, \ldots, k\}\}$

Symbols of Chapter 3

x	a data vector $x = (\xi_1, \ldots, \xi_d)^T \in \mathbb{R}^d$ or an 'incomplete' data vector with only $\xi_1, \ldots, \xi_{d-1}$ known
ξ_d	the d-th coefficient predicted for an incomplete data vector x
$\text{low}_d, \text{up}_d$	a tight lower bound, respectively upper bound to ξ_d for all $x \in X$
L	the line segment $L = L(x) = \{(\xi_1, \ldots, \xi_{d-1}, \text{low}_d) + \lambda(0, \ldots, 0, \text{up}_d - \text{low}_d)^T : \lambda \in [0, 1]\}$ for an incomplete x
PP	the partition polytope $PP = PP(X, k, \kappa_1, \ldots, \kappa_k)$

Symbols of Chapter 4

$X = \{x_1, \ldots, x_n\}$	a set of (non-geometric) items
$K_{k,n}$	the complete bipartite graph $K_{k,n} = K_{k,n}(C, X)$ of vertex sets $C = (c_1, \ldots, c_k)$, $X = (x_1, \ldots, x_n)$
E, E'	the assignments in $K_{k,n}$ corresponding to clusterings C, C'
v, v'	the vertices in PP corresponding to clusterings C, C'
$E \Delta E'$	the symmetric difference $E \Delta E' = (E \cup E') \backslash (E' \cap E) = (E \backslash E') \cup (E' \backslash E)$ of assignments E, E'
i_1, i_2	the indices of the clusters of biggest, respectively second biggest size
$CDG(C, C')$	the cluster difference graph corresponding to clusterings C and C'
$CDG(E, E'), CDG(v, v')$	the equivalent of $CDG(C, C')$ with v and E associated with C, and v' and E' associated with C'
M	a set of vertices to be covered by a set of vertex-disjoint cycles
$\mathcal{CY}$	a set of cycles in a graph
f, f', f''	a maximal flow in a network or a cycle covering in a graph
$deg(v)$	the degree of a vertex v in a graph
$deg_{\|E}(v)$	the degree of a vertex v with respect to edge set E
$deg_-(v), deg_+(v)$	the indegree, respectively outdegree of a vertex v in a digraph
D_{max}	the maximal indegree $D_{max} = \max_{v \in V} deg_-(v)$ of a vertex in a graph
η_i	the number of items x with $x \in C_i$, but $x \notin C'_i$ for some clustering $C' = (C'_1, \ldots, C'_k) \neq C$
$C_i \to C_j$	a movement of an item $x \in C_i$ to cluster C_j

Symbols of Chapter 5

n	the number of lots
$v_1, \ldots, v_n$	reference points for lots in the Euclidean plane
k	the number of farmers
d	the number of attributes of lots
$w_V(v_i)$	the attribute vector $w_V(v_i) \in (\mathbb{R}^+)^d$ for the i-th lot
κ_i	the vector of total attributes of the i-th farmer's original lots
$\epsilon_i^-, \epsilon_i^+$	lower bounds $\epsilon_i^- \in (\mathbb{R}^+)^d$, respectively upper bounds $\epsilon_i^+ \in (\mathbb{R}^+)^d$ to the allowed relative deviation from the i-th farmers original possession with respect to w_V
$w_E(v_i, v_j)$	the distance of the i-th and j-th lot
μ_i	the number of center lots of the i-th farmer
μ	the total number of center lots $\mu = \sum_{i=1}^{k} \mu_i$
P	a partition (or clustering) $P = (C_{11}, \ldots, C_{1\mu_1}, \ldots, C_{k1}, \ldots, C_{k\mu_k})$ of the lots
C_{ij}	the j-th cluster of the i-th farmer
c_{ij}	j-th center lot of the i-th farmer
C_i	the cluster group of the i-th farmer $C_i = \bigcup_{j \in \{1,\ldots,\mu_i\}} C_{ij}$
$\mathbb{1}$	the vector $\mathbb{1} = (1, \ldots, 1)^T \in \mathbb{Z}^d$
$(a_1, \ldots, a_d)^T \circ (b_1, \ldots, b_d)^T$	the operation on the d-dimensional vectors $(a_1, \ldots, a_d)^T$ and $(b_1, \ldots, b_d)^T$ defined by $(a_1, \ldots, a_d)^T \circ (b_1, \ldots, b_d)^T \to (a_1 b_1, \ldots, a_d b_d)$
$comp_c$	the c-th connected lot component
u	the number of connected lot components
$\mathcal{C} = \{comp_1, \ldots, comp_u\}$	a partition of lots into connected lot components
GIS	a geographical information system
MST	Minimum Spanning Tree
TSP	Traveling Salesman Problem
CSC	Constrained Minimum-k-Star Clustering
SSP	Size-restricted Minimum-k-Star Partition
SGP	Size-restricted Minimum-k-Star Group Partition

List of Definitions

List of Figures

All figures in this thesis were created by the author. The figures showing agricultural regions were exported from the Agriopt tool mentioned in Chapter 5. For all other figures, the TEX-packages TikZ and PGF were used, see [Tan08].

List of Algorithms

List of Tables

References

[AF92] D. Avis and K. Fukuda. "A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra". In: *Discrete and Computational Geometry* 8 (1), 1992, pp. 295–313.

[AHA98] F. Aurenhammer, F. Hoffman, and B. Aronov. "Minkowski-type theorems and least-squares clustering". In: *Algorithmica* 20, 1998, pp. 61–76.

[AHU85] A. Aho, J. Hopcroft, and J. Ullman. *Data Structures and Algorithms*. Addison-Wesley, 1985.

[AK99] F. Aurenhammer and R. Klein. "Voronoi diagrams". In: *Handbook of Computational Geometry*. Elsevier Science, 1999, pp. 201–290.

[And73] M. Anderberg. *Cluster Analysis for Applications*. Academic Press, 1973.

[Aur87] F. Aurenhammer. "Power diagrams: Properties, algorithms and applications". In: *SIAM J. Comput.* 16 (1), 1987, pp. 78–96.

[BBG09] S. Borgwardt, A. Brieden, and P. Gritzmann. "Constrained Minimum-k-Star Clustering and its application to the consolidation of farmland". In: *Operational Research* 54, 2009.

[BDW08] S. Basu, I. Davidson, and K. Wagstaff. *Clustering with Constraints: Advances in Algorithms, Theory and Applications*. Chapman & Hall, 2008.

[Ber02] P. Berkhin. *Survey of Clustering Data Mining Techniques*. Tech. rep. Accrue Software, 2002.

[BG04] A. Brieden and P. Gritzmann. "A quadratic optimization model for the consolidation of farmland by means of lend-lease agreements". In: *Operations Research Proceedings 2003: Selected Papers of the International Conference on Operations Research (OR 2003)*. Ed. by D. Ahr, R. Fahrion, M. Oswald, and G. Reinelt. Springer-Verlag, 2004, pp. 324–331.

[BG06] A. Brieden and P. Gritzmann. "Von Ackerbau und polytopalen Halbnormen: Diskrete Optimierung für die Landwirtschaft". In: *Mathematik erleben - Kombinatorische Optimierung lehren und lernen*. Ed. by S. Huß mann and B. Lutz-Westphal. Vieweg Verlag, 2006.

[BG09] A. Brieden and P. Gritzmann. *On the Voronoi property of extremal clusterings with prescribed cluster sizes. Manuscript*. 2009.

[BH08] M. Balzer and D. Heck. "Capacity-constrained Voronoi diagrams in finite spaces". In: *Proceedings of the 5th Annual International Symposium on Voronoi Diagrams in Science and Engineering*. Vol. 2. 2008, pp. 44–56.

[BHR92] E. R. Barnes, A. J. Hoffman, and U. G. Rothblum. "Optimal partitions having disjoint convex and conic hulls". In: *Math. Program.* 54 (1), 1992, pp. 69–86.

[BHS02] G. Brightwell, J. Heuvel, and L. Stougie. *A polynomial bound on the diameter of the transportation polytope*. Tech. rep. SPOR report 2002-15. Tech. Univ. Eindhoven, 2002.

[BHS03] G. Brightwell, J. Heuvel, and L. Stougie. *A linear bound on the diameter of the transportation polytope*. Tech. rep. SPOR report 2003-12. Tech. Univ. Eindhoven, 2003.

[Bor06] S. Borgwardt. "Nearly optimal algorithms on minimum spanning trees". Diplomarbeit. Technische Universität München, 2006.

[Bor07] K. Borgwardt. "Graph Kernels". PhD Thesis. Ludwig-Maximilians-University Munich, 2007.

[Bor82] K. H. Borgwardt. "The Average number of pivot steps required by the Simplex-Method is polynomial". In: *Mathematical Methods of Operations Research* 26 (1), 1982, pp. 157–177.

[Bor87] K. H. Borgwardt. *The Simplex Method. A Probabilistic Analysis*. Springer-Verlag, 1987.

[BR74] M. Balinski and A. Russakoff. "On the assignment polytope". In: *SIAM Review* 16 (4), 1974.

[Bre93] G. E. Bredon. *Topology and Geometry*. Springer-Verlag, 1993.

[Bri03] A. Brieden. "On the approximability of (discrete) convex maximization and its contribution to the consolidation of farmland". Habilitationsschrift. Technische Universität München, 2003.

[Cha03] R. Chaudhuri. "Do the arithmetic operations really execute in constant time?" In: *SIGCSE Bull.* 35 (2), 2003, pp. 43–44.

[CT76] D. Cheriton and R. E. Tarjan. "Finding minimum spanning trees". In: *SIAM J. Comput.* 5, 1976, pp. 724–742.

[Dan63] G. B. Dantzig. *Linear Programming and Extensions*. Princeton Univ. Press, 1963.

[DB08] I. Davidson and S. Basu. *A Detailed List of Constrained Clustering Papers*. 2008.

[DH73] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[Die06] R. Diestel. *Graph theory*. Springer-Verlag, 2006.

[Din70] E. A. Dinic. "Algorithm for solution of a problem of maximum flow in a network with power estimation". In: *Soviet Math. Dokl.* 11 (5), 1970, pp. 1277–1280.

[EE04] H. M. M. ten Eikelder and A. A. van Erk. "Unification of some least squares clustering methods". In: *Journal of Mathematical Modelling and Algorithms* 3, 2004, pp. 105–122.

[ELL01] B. Everitt, S. Landau, and M. Leese. *Cluster Analysis*. John Wiley & Sons, 2001.

[Els97] U. Elsner. *Graph Partitioning - A survey*. Tech. rep. 97-27. Tech. Univ. Chemnitz, 1997.

[ESR98] ESRI. *Shapefile Technical Description. An ESRI White Paper*. 1998.

[FF56] L. R. Ford and D. R. Fulkerson. "Maximal flow through a network". In: *Canadian Journal of Mathematics* 8, 1956, pp. 399–404.

[FOR03] K. Fukuda, S. Onn, and V. Rosta. "An Adaptive Algorithm for Vector Partitioning". In: *Journal of Global Optimization* 25, 2003, pp. 305–319.

[GBH98] N. Guttmann-Beck and R. Hassin. "Approximation algorithms for min-sum p-clustering". In: *Discrete Applied Mathematics* 89, 1998, pp. 125–142.

[GGJ76] M. R. Garey, L. Graham, and D. S. Johnson. "Some NP-complete geometric problems". In: *Proc. ACM Symposium of Theory of Computing*. 1976, pp. 10–22.

[GP06] R. Guralnick and D. Perkinson. "Permutation polytopes and indecomposable elements in permutation groups". In: *Journal of Combinatorial Theory, Series A 113*. Elsevier Science, 2006, pp. 1243–1256.

[HA04] V. Hodge and J. Austin. "A survey of outlier detection methodologies". In: *Artificial Intelligence Review* 22 (2), 2004, pp. 85–126.

[Haw80] D. M. Hawkins. *Identification of outliers*. Vol. 29. Chapman & Hall, 1980.

[HJ97] P. Hansen and B. Jaumard. "Cluster analysis and mathematical programming". In: *Math. Program.* Vol. 79. 1997, pp. 191–215.

[HK08] F. Höppner and F. Klawonn. "Clustering with Size Constraints". In: *Computational Intelligence Paradigms, Innovative Applications*, 2008.

[HOR00] F. K. Hwang, S. Onn, and U. G. Rothblum. "Linear-shaped partition problems". In: *Operations Research Letters* 26 (4), 2000, pp. 159–163.

[HOR98] F. K. Hwang, S. Onn, and U. G. Rothblum. "Representations and characterizations of vertices of bounded-shape partition polytopes". In: *Linear Algebra and its Applications* 278 (1-3), 1998, pp. 263–284.

[HOR99] F. K. Hwang, S. Onn, and U. G. Rothblum. "A Polynomial Time Algorithm for Shaped Partition Problems". In: *SIAM J. on Optimization* 10 (1), 1999, pp. 70–81.

[HS02] J. Heuvel and L. Stougie. *A quadratic bound on the diameter of the transportation polytope*. Tech. rep. SPOR 2002-17. Tech. Univ. Eindhoven, 2002.

[HT56] I. Heller and C.B. Tompkins. "An Extension of a Theorem of Dantzig's". In: *Linear Inequalities and Related Systems, Annals of Mathematics Studies*. Ed. by H.W. Kuhn and A.W. Tucker. Vol. 38. Princeton University Press, 1956, pp. 247–254.

[IJ03] Y. Ishigami and T. Jiang. "Vertex-Disjoint Cycles Containing Prescribed Vertices". In: *Journal of Graph Theory* 42 (4), 2003, pp. 276–296.

[Jam78] G. D. James. *The Representation Theory of the Symmetric Group*. Springer-Verlag, 1978.

[JD88] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. "Data Clustering - A Review". In: *ACM Computing Surveys*. Vol. 31-3. 1999, pp. 264–323.

[Kar72] R. M. Karp. "Reducibility among combinatorial problems". In: *Complexity of Computer Computations*. Ed. by R.E. Miller and J.W. Thatcher. Plenum Press, 1972, pp. 85–103.

[Kin67] B. King. "Step-wise clustering procedures". In: *J. Amer. Stat. Assoc.* Vol. 69. 1967, pp. 86–101.

[KP04] S. B. Kotsiantis and P. E. Pintelas. "Recent Advances in Clustering: A Brief Survey". In: *WSEAS Transactions on Information Science and Applications*. Vol. 1-1. 2004, pp. 73–81.

[KW68] V. Klee and C. Witzgall. "Facets and vertices of transportation polyhedra". In: *Mathematics of the decision sciences, Part 1*. Ed. by G.B. Dantzig and A.F. Veinott. American Mathematics Society. 1968, pp. 257–282.

[KYK84] M. Kovalev, V. Yemelichev, and M. Kratsov. *Polytopes, graphs and optimization*. Cambridge Univ. Press, 1984.

[LKOS09] J. A. De Loera, E. D. Kim, S. Onn, and F. Santos. "Graphs of transportation polytopes". In: *J. Comb. Theory, Ser. A* 116 (8), 2009, pp. 1306–1325.

[Loe06] J. A. De Loera. *Transportation Polytopes: A Twenty Year Update*. Talk at Workshop on Convex Sets and their Applications, Banff. 2006.

[Mac67] J. B. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proc. of the Fifth Symposium on Math, Statistics and Probability*. University of California Press, 1967, pp. 281–297.

[Mir05] B. Mirkin. *Clustering for data mining: A data recovery approach*. Chapman & Hall, 2005.

[MTA08] D. Man and M. Tosa-Abrudan. "Fuzzy Clustering Algorithms: A Survey". In: *7th Joint Conference on Mathematics and Computer Science*. 2008.

[Nad89] D. Naddef. "The Hirsch Conjecture is true for $(0, 1)$-polytopes". In: *Math. Program.* 45 (1), 1989, pp. 109–110.

[Onn93] S. Onn. "Geometry, Complexity, and Combinatorics of Permutation Polytopes". In: *J. Comb. Theory, Ser. A* 64 (1), 1993, pp. 31–49.

[Pap77] C. H. Papadimitriou. "Euclidean TSP is NP-complete". In: *Theoretical Computer Science* 4, 1977, pp. 237–244.

[Sch98] A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[Sco82] J. Scoltock. "A Survey of the Literature of Cluster Analysis". In: *Comput. J.* 25 (1), 1982, pp. 130–134.

[SH75] M. I. Shamos and D. J. Hoey. "Closest-point problems". In: *Proc. 16th Annual IEEE Symposium on Foundations of Computer Science*. 1975, pp. 151–162.

[SS02] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, 2002.

[SS73] P. H. A. Sneath and R. R. Sokal. *Numerical Taxonomy*. Freeman, 1973.

[Tan08] T. Tantau. *The TikZ and PGF Packages. Manual for version 2.00*. 2008. URL: `http://sourceforge.net/projects/pgf`.

[War63] J. H. Ward. "Hierarchical Grouping to Optimize an Objective Function". In: *J. Amer. Stat. Assoc.* Vol. 58. 1963, pp. 236–244.

[WC00] K. Wagstaff and C. Cardie. "Clustering with Instance-level Constraints". In: *Proc. of the Seventeenth International Conference on Machine Learning*. 2000, pp. 1103–1110.

[WCRS01] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. "Constrained K-means Clustering with Background Knowledge". In: *Proc. of the 17th International Conference on Machine Learning*. 2001, pp. 577–584.

[WM04] I. Wasito and B. Mirkin. "Nearest neighbour approach in the least-squares data imputation algorithms". In: *Information Sciences* 169, 2004, pp. 1–25.

[XW05] R. Xu and D. Wunsch. "Survey of Clustering Algorithms". In: *IEEE Transactions on Neural Networks*. Vol. 16:3. 2005, pp. 645–678.

[ZG03] S. Zhong and J. Ghosh. "Scalable, Balanced Model-based Clustering". In: *SIAM International Conference on Data Mining*. Vol. SDM-03. 2003, pp. 71–82.

[Zha97] C. Q. Zhang. *Integer flows and cycle cover of graphs*. Marcel Dekker, 1997.