Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Variants of the Graph Laplacian with Applications in Machine Learning

Sven Kurras

Dissertation

zur Erlangung des Grades
des Doktors der Naturwissenschaften (Dr. rer. nat.)
im Fachbereich Informatik
der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Universität Hamburg

Hamburg, Oktober 2016

Diese Promotion wurde gefördert durch die Deutsche Forschungsgemeinschaft,
Forschergruppe 1735 “Structural Inference in Statistics: Adaptation and Efficiency”.

Betreuung der Promotion durch:
Prof. Dr. Ulrike von Luxburg

Tag der Disputation:
22. März 2017

Vorsitzender des Prüfungsausschusses:
Prof. Dr. Matthias Rarey

1. Gutachterin:
Prof. Dr. Ulrike von Luxburg

2. Gutachter:
Prof. Dr. Wolfgang Menzel

Zusammenfassung

In sämtlichen Lebensbereichen finden sich Graphen. Zum Beispiel verbringen Menschen viel Zeit mit der Kantentraversierung des Internet-Graphen. Weitere Beispiele für Graphen sind soziale Netzwerke, öffentlicher Nahverkehr, Moleküle, Finanztransaktionen, Fischernetze, Familienstammbäume, sowie der Graph, in dem alle Paare natürlicher Zahlen gleicher Quersumme durch eine Kante verbunden sind. Graphen können durch ihre Adjazenzmatrix W repräsentiert werden. Darüber hinaus existiert eine Vielzahl alternativer Graphmatrizen. Viele strukturelle Eigenschaften von Graphen, beispielsweise ihre Kreisfreiheit, Anzahl Spannbäume, oder Random Walk Hitting Times, spiegeln sich auf die ein oder andere Weise in algebraischen Eigenschaften ihrer Graphmatrizen wider. Diese grundlegende Verflechtung erlaubt das Studium von Graphen unter Verwendung sämtlicher Resultate der Linearen Algebra, angewandt auf Graphmatrizen. Spektrale Graphentheorie studiert Graphen insbesondere anhand der Eigenwerte und Eigenvektoren ihrer Graphmatrizen. Dabei ist vor allem die Laplace-Matrix $L = D - W$ von Bedeutung, aber es gibt derer viele Varianten, zum Beispiel die normalisierte Laplacian, die vorzeichenlose Laplacian und die Diplacian. Die meisten Varianten basieren auf einer "syntaktisch kleinen" Änderung von L, etwa $D + W$ anstelle von $D - W$. Solcherart Modifikationen ändern meist vollständig die in den Eigenwerten und Eigenvektoren codierte Information. Auf diese Weise können sich neuartige Verbindungen zu Grapheigenschaften ergeben. Die vorliegende Doktorarbeit untersucht neue und existierende Varianten von Laplace-Matrizen. Die $\mathbf{f}$-adjusted Laplacian wird eingeführt und gezeigt, dass diese als eine spezielle Diagonalmodifikation der normalisierten Laplace-Matrix aufgefasst werden kann. Im Kontext zufälliger geometrischer Nachbarschaftsgraphen wird bewiesen, dass diese Matrix eine konkrete Manipulation der zugrundeliegenden Wahrscheinlichkeitsdichte beschreibt. Diese Intuition erlaubt neuartige Ansätze für verschiedene Problemstellungen des Maschinellen Lernens, zum Beispiel für die Bildsegmentierung im Falle nicht-uniform gesampelter Pixelpositionen. Diese Arbeit untersucht zudem die iterierte Anwendung des Normalisierungsschrittes $W \mapsto D^{-1/2} W D^{-1/2}$, welcher der normalisierten Laplace-Matrix zugrunde liegt. Das Ergebnis sind neue Resultate zum Konvergenzverhalten. Diese führen zur Definition der $\mathbf{f}$-fitted Laplacian, welche sich beispielsweise zur Behebung eines bestimmten Typs von Stichprobenverzerrung eignet. Das letzte Kapitel studiert die signed Laplacian. Dabei handelt es sich um eine Erweiterung der Laplace-Matrix auf Graphen mit sowohl positiven als auch negativen Kantengewichten. Diese Arbeit bietet eine Neuinterpretation des kleinsten Eigenwertes der signed Laplacian, wodurch sich der zugehörige Eigenvektor als der kanonische Kandidat für spektrales Korrelations-Clustering offenbart. Die Ergebnisse des so entwickelten Algorithmus sind State of the Art. Der Algorithmus wird in einer umfassenden Praxisanwendung implementiert, deren Ziel die automatisierte Identifikation von unfairem Verhalten in einem Multiplayer-Online-Spiel ist.

Abstract

Graphs are everywhere. For example, people spend a lot of time on traversing the hyperlink-edges of the web graph. Other examples of graphs are social networks, public transportation maps, molecules, financial transactions, fishing nets, family trees, and the graph that you get by connecting any two natural numbers that have the same digit sum. A graph can be represented by its adjacency matrix W, but there exist several alternative graph matrices. Many structural properties of a graph such as the absence of cycles, number of spanning trees or random walk hitting times, reflect in these graph matrices as all kinds of algebraic properties. This fundamental relation allows to study graph properties by applying all the machinery from linear algebra to matrices. Spectral graph theory focuses particularly on the eigenvalues and eigenvectors of graph matrices. In this field of research, the graph Laplacian matrix $L = D - W$ is particularly well-known, but there are numerous variants such as the normalized Laplacian, the signless Laplacian and the Diplacian. Most variants are defined by a "syntactically tiny" modification of L, for example $D + W$ instead of $D - W$. Nevertheless, such modifications can drastically affect the information that is encoded in the eigenvalues and eigenvectors. This can finally lead to new relations to graph properties. This thesis studies novel and existing variants of Laplacian matrices. The **f**-adjusted Laplacian is introduced and proven to be a specific diagonal modification of the normalized Laplacian matrix. In the context of random geometric neighborhood graphs, this matrix can be understood as a specific distortion that is applied to the underlying probability density. This intuition allows for new approaches to various applications in machine learning, for example to image segmentation in case of non-uniformly sampled pixel positions. This thesis further studies the repeated application of the normalization step $W \mapsto D^{-1/2}WD^{-1/2}$ that underlies the normalized Laplacian. It contributes a novel convergence result that finally leads to the **f**-fitted Laplacian as another strategy to remove a certain type of sampling bias. The last chapter studies the signed Laplacian, which generalizes the Laplacian to graphs of both positive and negative edge weights. This thesis contributes a novel re-interpretation of the smallest eigenvalue of the signed Laplacian. It identifies the corresponding eigenvector as the canonical candidate for a spectral approach to correlation clustering. The suggested algorithm is shown to compete with state-of-the-art. The algorithm is implemented in an extensive application from practice that aims at automatically detecting unfair user behavior in a multi-player online game.

Acknowledgements

First of all, I want to thank my advisor Ulrike von Luxburg for guiding my research with the right balance between clear orientation and total freedom. Thank you for introducing me to the fascinating world of machine learning by sharing your knowledge and intuition in many inspiring discussions. These three years of research in your working group were really a great time in which I learned interesting stuff at a higher rate than ever before! Special thanks go also to my office mates Matthäus Kleindessner and Morteza Alamgir. I really enjoyed the time with you, and I am grateful for your helpful input, for the nice conference travels, for your pleasant scientific spirit, and for all the sweet pastries from Austria and Iran!

I am grateful to the German Research Foundation who funded my research in the brilliant Research Unit 1735, *Structural Inference in Statistics: Adaptation and Efficiency*. I highly appreciated all our meetings, in particular the excellent spring schools! I say "thank you for everything" to all the wonderful interim group members, workmates, students, professors, IT admins, and the administration staff, in particular to Hildegard Westermann. A thousand thanks go further to all the people that I met at conferences, guest talks, poster sessions or colloquia, whose names I forgot or never knew, but who nevertheless contributed to great scientific and non-scientific conversations. Such a friendly and competent research community is really something that is worth paying a lot of taxes for!

Since I made the almost-mistake to start working in industry in parallel to finishing "just the last 10%" of this thesis, big thanks go to my employer Risk.Ident GmbH who gave me the opportunity to solely focus on finishing this thesis during the hot final phase. Special thanks go to Marco Fisichella, who never stopped asking me on the progress of my thesis, and to the terrific data science team — you all rock!

A million thanks go to my parents, family and friends, who sustained me during the last year of living on a submarine, surfacing only every now and then. Last but absolutely not least my warmest thanks go to my wonderful wife-to-be Julia and our sweet lovely baby Charlotte, who were patient with my tight time-window scheduling for such a long time. Now the windows are open and our full-time life begins! I ♥ you!

Contents

List of Illustrations

Figures

Tables

List of Symbols

Basic notation

The following symbols are used without any further explicit definition.

Symbol	Description
$\mathbb{N}$	the natural numbers $\{0, 1, 2, 3, \dots\}$ including 0
$\mathbb{R}$	the real numbers, also further constrained as $\mathbb{R}_{>0}$ or $\mathbb{R}_{\geq 0}$
$a, b, x, y, \varepsilon, \dots$	$\in \mathbb{R}$, scalar variables are typeset in italic lower case
$\mathbf{v}, \mathbf{f}, \boldsymbol{\mu}, x, y, \dots$	$\in \mathbb{R}^n$, vector variables are typeset in bold lower case if their entries $\mathbf{v} = (v_i)$ are accessed, or in italic lower case if not
(v_i)	entry-wise representation of vector $\mathbf{v}$, short for $\mathbf{v} = (v_i)_{i=1,\dots,n}$
$(a_1, \dots, a_\ell)$	a tuple (row vector) of the given ℓ entries
(s_k)	infinite sequence of entries s_k, short for $(s_k)_{k \geq a}$ for some self-evident $a \in \mathbb{N}$
$\mathbf{0}$	the all-zero-vector $(0, \dots, 0)^T$
$\mathbf{1}$	the all-one-vector $(1, \dots, 1)^T$
$\mathbf{e}_k$	the vector $(0, \dots, 0, 1, 0, \dots, 0)^T$ with a single 1-entry at position k
$\sqrt{\mathbf{f}}$, $\mathbf{f}^q$, $\mathbf{x}/\mathbf{y}$	entry-wise vector operations, here $(\sqrt{f_i})$, (f_i^q) and (x_i/y_i), respectively
$A, M, \mathcal{L}, \dots$	$\in \mathbb{R}^{m \times n}$, matrix variables are typeset in italic upper case
M^T, $\mathbf{v}^T$	the transpose of matrix M and vector $\mathbf{v}$, respectively
M^{-1}	the inverse of an invertible square matrix M
M^+	the Moore-Penrose pseudoinverse of any matrix M
$[m_{ij}]$	entry-wise representation of matrix M, short for $M = [m_{ij}]_{i=1,\dots,m,\, j=1,\dots,n}$
$\operatorname{diag}(d_1, \dots, d_n)$	the $n \times n$ diagonal matrix with d_i at index position (i, i)
$\operatorname{diag}(\mathbf{d})$	the diagonal matrix with the entries of vector $\mathbf{d}$ along its main diagonal
$V, \mathcal{G}, \mathbb{G}$	set names are typed in various styles, depending on their context
$\overline{V}$	the complement $\overline{V} = \Omega \setminus V$ of the set V in an obvious superset Ω
$\sum_i$	sum over all possible values for i, short for $\sum_{i \in S}$ if set S is self-evident
$[pred]_0^1$	Iverson bracket, takes value 1 if *pred* is true, and value 0 if *pred* is false
$\mathbf{1}_A$	indicator vector of $A \subset \{1, \dots, n\}$, that is $([i \in A]_0^1) \in \mathbb{R}^n$
$x \to \cdot$	convergence of variable x
$x \xrightarrow{a.s.} \cdot$	almost sure convergence of random variable x
$\|\mathbf{x}\| = \|\mathbf{x}\|_2$	Euclidean norm of vector $\mathbf{x}$, that is $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}} = \sqrt{\sum_i x_i^2}$
$\|\mathbf{x}\|_1$	L_1-norm of vector $\mathbf{x}$, that is $\|\mathbf{x}\|_1 = \sum_i \lvert x_i \rvert$
$B_A(c, r)$	ball of radius r at $c \in \mathbb{R}^n$ in $A \subseteq \mathbb{R}^n$, that is $B_A(c, r) = \{x \in A \mid \|x - c\| \leq r\}$

Specific notation

These symbols are defined later. They are listed here as a reference for a fast lookup.

Symbol	Description	Defined on page ...
$\mathbb{W}$	set of weighted graph matrices	23, Equation (1.1)
$\mathbb{W}_{\ominus}$	set of weak graph matrices, superset of $\mathbb{W}$	59, Equation (2.5)
W	weight matrix of some graph	22
$G = (V, E, W)$	graph of vertices V and edges E of weights W	22
$E(W)$	set of indices of the non-zero entries in W	23
$\mathcal{G}(W)$	graph implied by weight matrix W	22
$D = D_W$	degree matrix $D = \operatorname{diag}(W\mathbf{1})$	22
P	random walk transition matrix $P = D^{-1}W$	27
$\operatorname{vol}(S)$	volume of a set of vertices $S \subseteq V$	22
$\operatorname{cut}(S)$	cut weight of cutting between $S \subseteq V$ and $\overline{S}$	23
p_+, p_-	intra-/inter-edge probability of a graph model	24
μ_+, μ_-	Gaussian intra-/inter-mean of the GCG model	147
$W^{\circ}_{\mathbf{f}}$	$\in \mathbb{W}_{\ominus}$, $\mathbf{f}$-selflooped matrix of $W \in \mathbb{W}_{\ominus}$	60, Definition 2.3.4
$\tilde{W}_{\mathbf{f}}$	$\in \mathbb{W}$, $\mathbf{f}$-scaled matrix of $W \in \mathbb{W}$	62, Definition 2.3.5
$\overline{W}_{\mathbf{f}}$	$\in \mathbb{W}_{\ominus}$, $\mathbf{f}$-adjusted matrix of $W \in \mathbb{W}$	63, Definition 2.3.6
$\overline{W}_{\mathbf{f},c}$	$\in \mathbb{W}_{\ominus}$, $(\mathbf{f}, c)$-adjusted matrix of $W \in \mathbb{W}$	63, Definition 2.3.6
$c^+_{W,\mathbf{f}}$	$c\mathbf{f}$-selflooping has no negative edge for $c \geq c^+_{W,\mathbf{f}}$	72, Proposition 2.5.3
$c^+ = c^+_{\tilde{W},\mathbf{f}}$	$c\mathbf{f}$-adjusting has no negative edge for $c \geq c^+$	64
$r_k(i)$	k-nearest neighbor radius at sample point $\mathbf{x}_i$	25
L	(unnormalized) Laplacian matrix	31, Definition 1.3.2
$\mathcal{L}$	(symmetric) normalized Laplacian matrix	31, Definition 1.3.3
$\mathcal{L}_{rw}$	random-walk normalized Laplacian matrix	31, Definition 1.3.4
$\mathcal{L}_{\mathbf{f}}$	$\mathbf{f}$-adjusted Laplacian matrix, $\mathcal{L}_{\mathbf{f}}(W) = \mathcal{L}(\overline{W}_{\mathbf{f}})$	65, Definition 2.3.8
$D_h(X\|Y)$	Bregman divergence implied by function h	100, Definition 3.3.10
$RE(X\|Y)$	relative entropy error	101, Definition 3.3.11
$\mathcal{P}_{\mathcal{M}}(Q)$	RE-projection of $Q \in \mathbb{R}^d$ to $\mathcal{M} \subseteq \mathbb{R}^d$	102
$\mathcal{P}^{\perp}_{\mathcal{M}}(Q)$	orthogonal projection of $Q \in \mathbb{R}^d$ to $\mathcal{M} \subseteq \mathbb{R}^d$	102
$\mathcal{P}^h_{\mathcal{M}}(Q)$	Bregman projection of Q to $\mathcal{M}$, as implied by h	102, Equation (3.8)
η_d	volume of d-dimensional unit ball	50
$\rho(M)$	spectral radius of matrix M	28, Fact 1.3.1 (a)
$\mathcal{C}_k$	set of cluster-matrices that represent a k-clustering	142
$\Psi(W)$	set of symmetric direct biproportional fits of W	98, Corollary 3.3.6
$\operatorname{aff}(A)$	affine hull of $A \subseteq \mathbb{R}^d$	98
$\operatorname{int}(A)$	interior of $A \subseteq \mathbb{R}^d$	99
$\operatorname{rel\,int}(A)$	relative interior of $A \subseteq \mathbb{R}^d$	99, Definition 3.3.8

Chapter 1

Introduction

1.1 The world of spectral graph theory

Our universe consists of all kinds of entities, such as stars, atoms, numbers, humans and ideas. These entities can be related to each other in countless ways, for example in terms of distance, weight, prime divisors, shared ancestors or even by random. Every such relation can be represented as a time-dynamic hypergraph having entities as vertices and hyperedges between related entities. If we focus only on snapshots in time and pairwise relations, the model simplifies to a static graph with edges that connect pairs of vertices. Such graphs are everywhere. Consider for example street maps, protein structures, file directories, electric circuits, or the fascinating Collatz graph that connects each natural number n by an edge to $n/2$ if n is even, or otherwise to $3n + 1$; nobody knows whether it has other cycles than the trivial cycle $1 \to 4 \to 2 \to 1$. Even if there is no explicit graph, but just a collection of entities, then it lies in our human nature to start searching for a meaningful relation. For example, if we are given a large unsorted collection of cooking recipes from a website, our impulse is to structure them in some way. We may group them into vegetarian or not, or rank them linearly by cooking time, or we may come up with a similarity relation that is the stronger the more ingredients two recipes share. So even in cases where no graph is given, we can come up with several ways for defining some. Interesting graphs provide more than a trivial structure. The graph that connects cooking recipes if they share ingredients will likely show an interesting community structure of, say, four large clusters. Recipes are intensively connected

to others of the same cluster, but less strong to recipes of other clusters. A deeper inspection might reveal that these clusters can be classified roughly as Western cuisine, Asian cuisine, African cuisine, and sweet pastries from around the world. Within each cluster, one may continue the exploratory analysis and detect more fine-granular sub-clusters. An alternative application of this graph could be to suggest new recipes to users of the website, based on their search requests and their individual click history.

The basic toolkit of graph exploration contains algorithms such as Breadth-First-Search, Dijkstra's algorithm and Kruskal's algorithm in order to explore neighborhoods, shortest paths and minimum spanning trees, respectively. Network flows and random walk diffusion are more advanced techniques. For example, an alternative distance measure to the shortest path distance between two vertices i and j is the commute time: it is defined as the expected number of steps required by a random walk started at i to visit j and finally return back to i. In contrast to the shortest path distance, the commute time is influenced by the global cluster structure of the graph. If two vertices stay close in the sense of shortest paths, but they lie in different clusters that are connected just by a thin bridge, then their commute time will be large because the random walk crosses the bridge only rarely. How can such sophisticated graph properties be studied and computed efficiently? One approach is by graph matrices, such as the adjacency matrix $W \in \{0,1\}^{n\times n}$, which has entry 1 at position (i,j) if and only if i and j are related. This definition generalizes straightforward to $W \in \mathbb{R}_{>0}^{n\times n}$ for relations that assign arbitrary positive weights. Other examples of graph matrices are the incidence matrix, the edge-adjacency matrix and the random walk transition matrix $P = D^{-1}W$ for D the diagonal matrix of the vertex degrees. The graph matrix that lies at the heart of this thesis is the Laplacian matrix $L = D - W$ of a weighted undirected graph.

Every graph matrix is like a kaleidoscope that allows to look at graphs from the viewpoint of linear algebra. Structural properties of graphs reflect in manifold ways as algebraic properties of the corresponding graph matrices. It is easy to see that the length of a shortest unweighted path from i to j is equal to the smallest exponent k for which the entry (i,j) in the matrix W^k is non-zero. It is definitely much harder to see that the commute time for all pairs of vertices can be extracted from the entries in the Moore-Penrose pseudoinverse of the Laplacian matrix (Fouss et al., 2007).

Within the field of graph theory, the subfield of **spectral graph theory** focuses on connecting graph properties to the eigenvalues and eigenvectors of graph matrices. See for example Arsić et al. (2012) for an overview on applications of spectral techniques in computer science. One aspect of spectral graph theory, which received the most attention in the machine learning community, is **spectral clustering**. Early results on spectral clustering are given by Fiedler (1973) who relates several connectivity properties of an unweighted graph to the second-smallest eigenvalue of L. In the same year, Donath and Hoffman (1973) give a lower bound on the number of cut-edges for a k-partition in terms of the k largest eigenvalues of a family of matrices that contains $-L$ as a special case. Another notable work is by Hall (1970) who proved that the r smallest positive eigenvectors of L can be used to embed the vertices

of a graph in r dimensions in a way that minimizes the energy $\mathbf{f}^T L\mathbf{f} = \sum_{i<j} w_{ij}(f_i - f_j)^2$ subject to certain constraints. This technique is known as **spectral embedding**. It is still used in graph visualization in order to determine coordinates at which to plot the vertices of any given graph in the plane.

Spectral graph theory beyond spectral clustering is much older. Cvetković et al. (1980) present a comprehensive survey of results from spectral graph theory since the 1950s. However, the most groundbreaking publication in spectral graph theory, if not even its birth, dates back to Kirchhoff (1847): he proved the fascinating result that the number of different spanning trees of an unweighted graph is equal to the product of all positive eigenvalues of the Laplacian matrix L, divided by n.

Over the decades, spectral graph theory became the active field of research that it is still today. In particular several alternative variants of the Laplacian matrix L were introduced, some of which are briefly presented now. The most famous alternatives to L are the symmetric normalized Laplacian matrix $\mathcal{L}$ and the random-walk normalized Laplacian matrix $\mathcal{L}_{rw}$, both introduced in detail in Section 1.3.5. Cvetković et al. (2010) study further the spectra of the signless Laplacian, defined as $D+W$, and of the Seidel matrix, whose entries are -1 if ij is an edge, 0 if $i = j$, and 1 otherwise. Bühler and Hein (2009) introduce the p-Laplacian, a nonlinear generalization of the Laplacian that focuses on minimizing the energy $\sum_{i<j} w_{ij}|f_i - f_j|^p$. This implies L as a special case for $p = 2$. The authors show that the second-smallest eigenvalue of the p-Laplacian converges for $p \to 1$ to the optimal Cheeger cut value. Hou et al. (2003) extend the Laplacian to graphs of mixed positive and negative edge weights, denoted as the signed Laplacian. This variant is studied in detail in Chapter 4. There are multiple approaches to adapt the Laplacian to directed graphs, that is to non-symmetric W. Chung (2005) achieves this by relating W to a new variant of the Laplacian matrix that is symmetric by definition. Boley et al. (2011) study an alternative approach that is not based on a symmetrization. Their so-called Diplacian, which also comes in a normalized variant, particularly allows to express the commute time for directed graphs. Bapat et al. (2001) study what they call the perturbed Laplacian, that is $U - W$ for an arbitrary diagonal matrix U, which particularly includes L and $-W$ as special cases. They generalize several results known for L to this general case. Finally, some recent work focuses on generalizing spectral graph theory to hypergraphs, see for example Xie and Qi (2016) and the references therein.

All these approaches have one aspect in common: they study a variant of the original definition of the Laplacian matrix L. There is no single variant of a Laplacian that outperforms all others in all aspects. Some of the alternatives are motivated by the need to adapt the Laplacian to new graph models, others are motivated by exploring whether specific properties of a graph are captured better by some other graph matrix than by L. In most cases the Laplacian variant is just a tiny modification, for example $D + W$ instead of $D - W$. Even the Laplacian matrix L itself is up to a switch of sign just a diagonal modification of the adjacency matrix because $-L = W - D$. However, it turns out that diagonal modifications, and also other syntactically simple changes, can have a strong impact on the algebraic properties such

as the eigenvectors.

In this thesis, I study the following variants of Laplacian matrices. Chapter 2 introduces the $\mathbf{f}$-adjusted Laplacian, which is shown to be a specific diagonal modification of the normalized Laplacian matrix $\mathcal{L}$. I provide a geometric intuition that allows to play with the parameter vector $\mathbf{f}$ in a way that establishes new approaches to various applications in machine learning, for example to spectral image clustering of non-uniformly sampled pixel positions. Chapter 3 studies in depth the normalization step $W \mapsto D^{-1/2}WD^{-1/2}$ that underlies the symmetric normalized Laplacian, and finally introduces the $\mathbf{f}$-fitted Laplacian as an alternative strategy to remove a certain type of sampling bias. In Chapter 4, I originally discovered an interesting real-weighted variant of the Laplacian, but it turned out that this variant is already known in the literature as the signed Laplacian. However, I contribute a novel re-interpretation of its smallest eigenvector, which suggests the signed Laplacian as the perfect candidate for a spectral approach to correlation clustering.

Similar to the distinction into Theoretical Physics and Applied Physics, the field of Machine Learning can be considered as a collaboration between Theoretical Machine Learning and Applied Machine Learning. Along this axis between theory and practice, this thesis is definitely located on the side of theory. However, it provides several explicit links into concrete machine learning applications, and it always focuses on providing a plain intuition that allows to apply the theoretical results. Moreover, the last chapter includes an extensive software implementation that particularly shows that *real* machine learning applications require additional ad-hoc preprocessing steps before a machine learning algorithm can be applied. In contrast to such diffuse practical issues, the theoretical findings discover some of the underlying intrinsic structures in a locally sharp clarity that preserves the magic of looking through a real kaleidoscope.

1.2 Structural overview on this thesis

This section describes briefly the structure of the entire thesis. The follow-up section presents various techniques and formal concepts that build a solid foundation of all the following. This introductory chapter ends with a summary of my contributions and a list of my publications.

Chapter 2, Chapter 3 and Chapter 4 present my research results in detail on both an intuitive and a technical level, where each chapter focuses on an individual topic. All chapters are connected by the same leitmotif that is also the title of this thesis: they study three variants of the graph Laplacian in theory and practice, one variant per chapter. All chapters follow the same structure:

- **Chapter introduction**. Introduces the respective topic by providing an application-oriented context.

- **Informal summary of the main contributions**. Provides a high-level summary of my contributions to this topic.
- **Formal setup**. Contains details on all relevant formal definitions and concepts that are required in order to fully understand the later contributions in all aspects.
- **Main contributions**. Summary of my main contributions, stated formally precise but without proofs.
- **Proofs and technical details**. This section goes into depth. It contains mathematical proofs and additional detailed insights.
- **Applications**. Puts the theoretical findings into practice by highlighting some concrete applications that benefit from my contributions.
- **Chapter summary**. A brief summary of the chapter at an advanced level of detail.

According to the principle "you write what you read", frequent usage of the singular form "I" in scientific texts distracts my attention because most publications are written from the plural "we" perspective. Moreover, in most cases "we" refers to "you and me". For that reason, I stick to the plural form in all later technical discussions (unless it sounds strange), while I reserve the singular form only for lifting the reader up to the meta level.

1.3 Machine learning context

This section briefly introduces some of the context that is relevant for this thesis, in particular from random graph theory, linear algebra and spectral clustering. Readers that are familiar with these topics might fly over this section. However, beside introducing basic formalisms, this section focuses on motivating and connecting these techniques on a more applied, intuitive level.

1.3.1 Graphs

Data sets are typically defined as collections of entities, in which each entity refers to a combination of attribute values. Such data sets can be represented as a table: each row corresponds to an entity, each column to an attribute, and each table cell contains the respective value. Many machine learning algorithms can more or less directly work on such **tabular data**. However, the tabular form is not well-suited to represent relational information between entities, such as friendship relations in social networks. The natural choice to represent **relational data** is by a graph. Each entity corresponds to a vertex of the graph, and each relation between entities corresponds to an edge that connects vertices. For example, the world wide web can be represented as a graph that has all web documents as its vertices,

and hyperlinks between documents as its edges. As another example, consider ad-hoc sensor networks, where each sensor (vertex) is located at some position, and can communicate with other sensors (along edges) if they are located sufficiently close.

The large variety of different types of relational information implies a large variety of different types of graphs. For example, the social network graph is undirected (i is a friend of j if and only if j is a friend of i), while the hyperlink graph is directed (i may link to j but not vice versa). The sensor network graph has edges that refer to Euclidean distances, and it further provides an embedding of its vertices into the plane at the sensor positions, which is not the case for the other two examples.

Weighted graphs

An **undirected graph** $G = (V, E)$ consists of a set of vertices $V = \{1, \dots, n\}$ and a set of edges E. For $i, j \in V$, we denote by $ij \in E$, equivalently $ji \in E$, that the undirected edge $ij = ji$ exists in the graph. We particularly allow G to contain edges $ii \in E$ that are incident to a single vertex, denoted as **selfloops**. An undirected graph has at most $(n^2 + n)/2$ edges.

A **directed graph** is defined similar, but now it holds that $E \subseteq V \times V$. The directed edges ij and ji exist independently of each other in the graph, and equality $ij = ji$ holds if and only if $i = j$. A directed graph has at most n^2 edges.

Given an undirected or directed graph $G = (V, E)$, every sequence of vertices $(v_0, v_1, \dots, v_\ell)$ with $v_i v_{i+1} \in E$ for all $0 \leq i < \ell$ is called a **walk**. It is a **closed walk** if $v_0 = v_\ell$. A walk is denoted as a **path** if $v_i \neq v_j$ whenever $i \neq j$.

A **(positive-)weighted graph** $G = (V, E, W)$ is an undirected or directed graph together with positive edge weights $w_{ij} > 0$ for all $ij \in E$. These edge weights can be arranged as the entries of a non-negative matrix $W := [w_{ij}] \in \mathbb{R}^{n \times n}_{\geq 0}$ with the additional convention that $w_{ij} := 0$ whenever $ij \notin E$. The matrix W is called the **(weighted) adjacency matrix** of G, or simply its **weight matrix**. An **unweighted graph** is just a special case of a weighted graph by the convention that $w_{ij} := 1$ for all $ij \in E$.

Any chosen graph G uniquely determines the weight matrix W, and any chosen non-negative square matrix W uniquely determines G. This gives a bijection between the set of all directed weighted graphs and the set of all non-negative matrices. In the same way, we can identify undirected weighted graphs with symmetric non-negative matrices. For that reason, we also refer by $\mathcal{G}(W)$ to the graph that corresponds to the non-negative matrix $W \in \mathbb{R}^{n \times n}_{\geq 0}$. Note that there is a slight abuse of notation for symmetric W, since $\mathcal{G}(W)$ may either refer to the undirected graph or the directed graph. Whenever this distinction makes a difference (as for the number of edges), it is always clear from the context whether we consider undirected or directed graphs.

The **(out-)degree** of vertex i is $d_i := \sum_{j \in V} w_{ij}$ that is the sum along the i'th row in W. The degree vector $\mathbf{d} := (d_i)$ can also be written as $\mathbf{d} = W\mathbf{1}$ with $\mathbf{1}$ the all-one vector. This defines the **degree matrix** $D := \operatorname{diag}(\mathbf{d})$ as the diagonal matrix that has the degree vector $\mathbf{d}$ along its diagonal. For $S \subseteq V$, we denote by $\operatorname{vol}(S) := \sum_{i \in S} d_i$ the **volume of** S, and by

$\text{cut}(S) := \sum_{i \in S, j \notin S} w_{ij}$ the weight of the cut between S and its complement $\overline{S}$.

For every matrix $A \in \mathbb{R}^{m \times n}$, we denote by $E(A) := \{ij \mid a_{ij} \neq 0\}$ the set of all non-zero index positions in A. We say that A **has at least the zeros** of B if $E(A) \subseteq E(B)$ and that they **have the same zeros** if $E(A) = E(B)$.

A **(strongly) connected component** of a graph is any maximal subset of vertices $S \subseteq V$ for which it holds that every $i \in S$ can be reached by a walk from every $j \in S$.

An **isolated vertex** is not incident to any edge, not even to a selfloop. Slightly more general, a **separated vertex** is not adjacent to any *other* vertex, but it may have a selfloop. Since isolated vertices appear in the weight matrix as a zero row and a zero column, they often require a special handling without adding much value. For that reason, we usually study the set

$$\mathbb{W} \;=\; \{X \in \mathbb{R}^{n \times n}_{\geq 0} \mid X = X^T,\; X\mathbf{1} > 0\} \tag{1.1}$$

of all graph matrices that belong to a weighted undirected graph of positive degrees (that is, without any isolated vertex). For the special case of a diagonal matrix $W \in \mathbb{W}$, the graph $\mathcal{G}(W)$ is denoted as a **loops-only graph**, since all its vertices have a selfloop but are separated from all others. A graph is **simple** if it is undirected and has no selfloop.

1.3.2 Random graph models

In machine learning, tabular data is often interpreted as the result of a random sampling process: each row represents an individual sample point drawn from a probability distribution on the space of all possible combinations of attribute values (including any label attribute). This way of thinking allows to apply all kinds of statistical reasoning. In particular, one can estimate the underlying probability distribution from a given training data set, and then apply this knowledge to new incoming sample points. For example, the fundamental Bayes Classifier estimates the label of a sample point by selecting the label that has the maximum probability to be correct, given the observed values of the other attributes.

Similar to the case of tabular data, there exist different random models for relational data. A **random graph model** always refers to some probability distribution on the set of all graphs. Random graph models are often defined in a generative way that constructs instances of the model by a random process. This avoids to state the implied probability distribution explicitly, and further allows to fit the model to applications intuitively.

For example, the earlier mentioned sensor network may be modeled by a graph that connects each sensor by an edge to all others that lie within a distance of at most m meters, weighted by the respective distance. This model is hard to define precisely in terms of sets of matrices. However, one can still exploit individual properties, such as the facts that each entry in such a matrix is at most m and that all entries satisfy the triangle inequality.

Planted partition model (PP)

A simple unweighted graph model that is commonly used to study community structures in social networks is the **Planted Partition model**:

$$PP(\ (n_1, \ldots, n_k),\ p_+,\ p_-) \quad .$$

A random graph from this model has n_i vertices of community/class i. It is generated as follows: starting with the edgeless graph on $n = \sum_{i=1}^{k} n_i$ vertices, every (non-selfloop) edge is added independently of all others by random choice. Each **intra-edge** (that is an edge connecting two vertices of the same class) is added with probability p_+, and each **inter-edge** (that is an edge connecting two vertices of different classes) is added with probability p_-. For representing communities, one further assumes the model to be **assortative**, that is $p_+ > p_-$. Such a graph shows higher connectivity within each class, and lower connectivity between different classes. It is a special case of the stochastic block model by Holland et al. (1983), which allows to define an individual edge distribution for each pair of classes. If we are given a graph of which we believe that it approximately fits to the planted partition model (even for unknown parameters), then we can apply every algorithm and every theory that is developed for this model. For example, in order to detect the true communities, we can apply the clustering algorithm by Chen et al. (2012), or the approach by Abbe and Sandon (2015), who further provide an interesting survey on recent results from this field of research.

Random geometric neighborhood graphs (RGNG)

Another graph model, studied in depth in Chapter 2 of this thesis, is the **random geometric neighborhood graph (RGNG)**. An RGNG is a weighted graph $G = (V, E, W)$ plus the additional information that G is determined by neighborhoods of random sample points. To make this precise, let us have a look at each of the terms "geometric", "neighborhood" and "random" individually. The word "geometric" in RGNG indicates that each vertex $i \in V$ is identified with a position $\mathbf{x}_i$ in d-dimensional Euclidean space $\mathbb{R}^d$. That is, at the heart of each RGNG there is a fixed set of n points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \in \mathbb{R}^d$. The word "neighborhood" in RGNG emphasizes that every edge weight w_{ij} is a **similarity measure**, which takes *larger* values the *closer* i lies to j. Since this is the opposite meaning of a distance measure, a simple similarity measure is given by the inverse Euclidean distance $\|\mathbf{x}_i - \mathbf{x}_j\|^{-1}$. However, other similarity measures have more desired properties. We study three variants of RGNGs. One is unweighted, which is a special case of a similarity measure, and two have **Gaussian weights** assigned to every edge, defined as follows for **bandwidth** parameter $\sigma > 0$:

$$w_{ij} \ := \ \frac{1}{(2\pi\sigma^2)^{d/2}} \cdot \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \quad . \tag{1.2}$$

Section 2.3.2 provides details on the origins of Equation (1.2). The three variants we study are undirected graphs, defined for any set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ as follows:

- **Gaussian graph**. Takes one parameter: bandwidth $\sigma > 0$. All pairs of vertices $i \neq j$ are connected by an edge of Gaussian weight (1.2). As a consequence, all off-diagonal entries in the weight matrix are positive.

- **r-graph**. Takes one parameter: radius $r > 0$. Two vertices $i \neq j$ are connected by an unweighted edge ($w_{ij} := 1$) if and only if their Euclidean distance is $\|\mathbf{x}_i - \mathbf{x}_j\| \leq r$.

- **weighted kNN-graph**. Takes two parameters: bandwidth $\sigma > 0$ and the intended number of neighbors $k < n$. Two vertices $i \neq j$ are connected by an edge of Gaussian weight (1.2) if either $\mathbf{x}_j$ is among the k closest sample points to $\mathbf{x}_i$, or $\mathbf{x}_i$ is among the k closest sample points to $\mathbf{x}_j$. Precisely, let $(\mathbf{x}_{\sigma_i(1)}, \mathbf{x}_{\sigma_i(2)}, \ldots, \mathbf{x}_{\sigma_i(n)})$ denote all sample points sorted in increasing distance to $\mathbf{x}_i = \mathbf{x}_{\sigma_i(1)}$. Further let $r_k(i) := \|\mathbf{x}_{\sigma_i(k+1)} - \mathbf{x}_i\|$ denote the distance of $\mathbf{x}_i$ to its k-nearest neighbor. Then $ij \in E$ if $\|\mathbf{x}_i - \mathbf{x}_j\| \leq r_k(i)$ or if $\|\mathbf{x}_i - \mathbf{x}_j\| \leq r_k(j)$. Note that this definition yields a symmetric relation, hence an undirected graph. Further every vertex is adjacent to at least k others. The weighted kNN-graph equals the Gaussian graph with most of its edges omitted.

A geometric neighborhood graph can be constructed on top of any collection of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, no randomness needs to be involved. The word "random" in RGNG indicates that the points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$ are the result of a random sampling process: the sample points $\mathbf{x}_i$ are **independently and identically distributed (i.i.d.)** drawn according to an arbitrary but fixed probability distribution on $\mathbb{R}^d$. Generating an RGNG starts with a random selection of n sample points. Once these are fixed, the remaining construction is deterministic and adds edges and edge weights according to one of the above definitions of neighborhoods. Let us summarize the RGNG construction by repeating the following sentence from above:

> **“ An RGNG is a weighted graph plus the additional information that it is determined by neighborhoods of random sample points. ”**

Two RGNGs from the same model and the same underlying probability distribution look differently because they are constructed on different sets of sample points. However, as demonstrated in Figure 1.3.1, we can expect that these graphs show similar structural properties (for example degree distribution, diameter, etc.) and hence also similar algebraic properties of their graph matrices (for example in terms of eigenvalues, see Section 1.3.5). Certain properties of the underlying probability distribution are encoded in structural properties of the graph as well as in algebraic properties of graph matrices.

Several relation-based machine learning algorithms rely on this link between similarity graphs and an implicitly underlying probability distribution. Such algorithms assume that high-density areas that are separated by low-density areas provide valuable information for the application. Examples of such algorithms are the popular density-based clustering algorithms DBSCAN (Ester et al., 1996), OPTICS (Ankerst et al., 1999) and their numerous enhancements, mode-seeking algorithms like Mean Shift (Cheng, 1995), outlier detection algorithms such

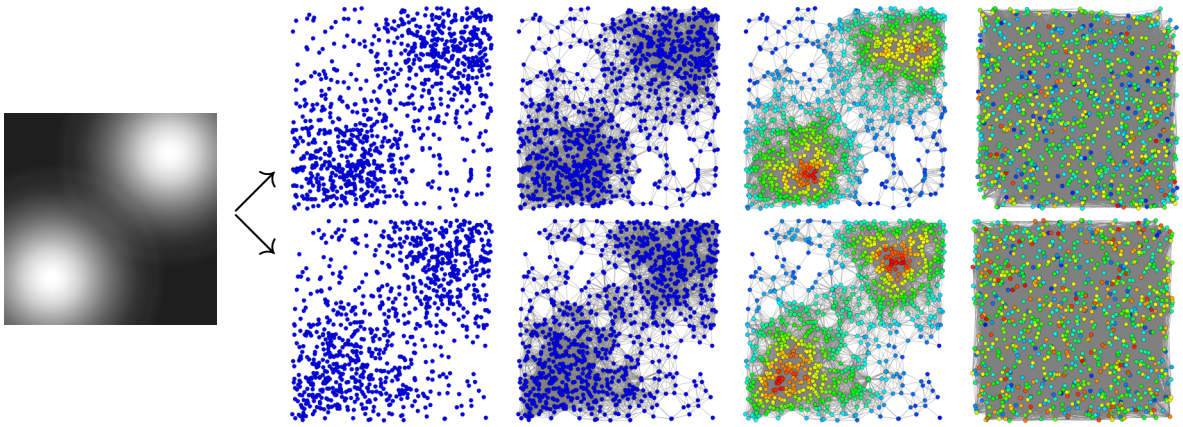


Figure 1.3.1: Generating random geometric neighborhood graphs. (a) some probability density p on $[0,1]^2$ (white = high density). p is the sum of two translated 2-dimensional Gaussian distributions clipped to the unit square. (b) two different samples (top and bottom) of 1000 sample points drawn from p. (c) vertices (sample points) connected by edges of the r-graph for $r = 0.1$. (d) the same graph, but now each vertex i is colored by a heat color that indicates its degree d_i (red = high degree). The vertex degree distribution follows roughly the profile of the underlying density p. (e) the same graph, but now each vertex is plotted at a random position rather than at its true sample point. This better visualizes the input to applications in which the true sample points are not known, but only their relations.

as Local Outlier Factor (Breunig et al., 2000), semi-supervised label propagation (Zhu and Ghahramani, 2002), geometric graph spanner construction with density-dependent slack (Chan et al., 2006), and cluster-detection by density level sets (Chaudhuri et al., 2014). These and related algorithms differ in two aspects:

- **theoretical foundation**: some of the algorithms borrow from the setting of RGNGs only the intuition that groups of strongly connected vertices correspond to high-density areas, while groups of loosely connected vertices correspond to low-density areas. Other approaches prove mathematical statements, so they particularly have to formulate precisely which type of probability distributions they consider. For example, a common mild restriction is to assume that the probability distribution has a probability density (which excludes some exotic distributions), and further that the density is continuous (at least almost everywhere, that is except for a set of zero-measure). The assumptions that are made in Chapter 2 are stated in Definition 2.3.2. The results in Chapter 2 focus on both: on a solid theoretical foundation, plus on keeping an intuition that allows to apply the algorithm creatively in practice.

- **input data**: some algorithms assume that the sample points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ are given as an input, so they can carry out any neighborhood computations directly on the sample points. Other algorithms do not require the sample points to be known, they only consider the similarity matrix W. Similar to kernel methods, such algorithms implicitly operate on an unknown coordinate embedding. This setting is also in the focus of this thesis. During the theoretical analysis we may access density p, dimensionality d, sample points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, weight matrix W, and all parameters of the RGNG construction. However, the algorithm itself does only take the weight matrix W of similarity weights as its input, and assumes that W refers (at least approximately) to some RGNG.

1.3.3 Random walks on graphs

A **random walk** $(v_0, v_1, \ldots, v_\ell)$ is a graph walk that is generated by a random process. At time $t \geq 0$ the current state is $v_t =: i$, and the follow-up vertex v_{t+1} is chosen randomly according to the **transition probabilities** $p_{ij} = w_{ij}/d_i$ for all $j \in V$. This random process satisfies the **Markovian property**: the probability for choosing the next state depends only on the current state, but is independent of the older history. Formally,

$$p_{ij} \quad = \quad P(v_{t+1} = j \mid v_t = i) \quad = \quad P(v_{t+1} = j \mid v_t = i, \ldots, v_0 = s) \; .$$

The transition probabilities define the entries of the **transition matrix** $P := [p_{ij}]$, which is constant for all random walks on this graph. P is related to W by

$$P \;=\; D^{-1}W \quad ,$$

where we assume that W does not contain an isolated vertex.

Usually one is not interested in the trajectory of a single random walk, but in the aggregated behavior of infinitely many alternative random walks simultaneously. This can be achieved by studying the **diffusion process** $(\boldsymbol{\pi}_0, \boldsymbol{\pi}_1, \ldots, \boldsymbol{\pi}_\ell)$, which no longer has a single vertex as its state, but a probability distribution $\boldsymbol{\pi}_t : V \to [0,1]$ on the vertices. We treat each $\boldsymbol{\pi}_i$ as a *row* vector. The diffusion process no longer has to make individual random decisions at each step, because each distribution $\boldsymbol{\pi}_{t+1}$ can be derived from its predecessor $\boldsymbol{\pi}_t$ deterministically by $\boldsymbol{\pi}_{t+1} = \boldsymbol{\pi}_t P$. By induction it follows that $\boldsymbol{\pi}_t = \boldsymbol{\pi}_0 P$. Thus, the whole diffusion process is determined by choosing an initial distribution $\boldsymbol{\pi}_0$. This raises the question of what happens for $t \to \infty$? Does $\boldsymbol{\pi} := \lim_{t\to\infty} \boldsymbol{\pi}_0 P$ exist? The short answer is: *it depends* on additional properties of the graph (Levin et al., 2009). However, any such limit must be a stationary point, that is $\boldsymbol{\pi} = \boldsymbol{\pi} P$, which is the motivation for denoting such $\boldsymbol{\pi}$ as a **stationary distribution**. In the typical case that $\mathcal{G}(W)$ refers to an undirected, connected, non-bipartite graph, it is guaranteed that for every initial distribution $\boldsymbol{\pi}_0$ the diffusion converges to the same unique stationary distribution $\boldsymbol{\pi}$.

1.3.4 Linear algebra background

The following fact sheet provides a brief summary of several properties of real symmetric matrices. These are applied later without further reference.

Fact 1.3.1 (Properties of real symmetric matrices)
The following statements and definitions apply to every real symmetric matrix A:

*(a) A has n real eigenvalues $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_n \in \mathbb{R}$, denoted as the **spectrum** of A, with $\rho(A) := \max\{|\alpha_1|, \ldots, |\alpha_n|\}$ the **spectral radius** of A.*

*(b) all corresponding eigenvectors are real-valued, denoted as $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in \mathbb{R}^n \setminus \{\mathbf{0}\}$. We denote $\mathbf{v}_1$ as the **smallest eigenvector**, and $\mathbf{v}_n$ as the **largest eigenvector**.*

(c) all eigenvectors are orthogonal to each other, that is $\mathbf{v}_i^T \mathbf{v}_j = 0$ whenever $i \neq j$.

(d) A can be represented as the sum $A = \sum_i \alpha_i \mathbf{v}_i \mathbf{v}_i^T$

*(e) A is **positive definite** if all eigenvalues are positive ($\alpha_1 > 0$)*

*(f) A is **positive semi-definite** if all eigenvalues are non-negative ($\alpha_1 \geq 0$)*

*(g) every matrix of the form $B = SAS^{-1}$ for an invertible matrix S is **similar** to A, which implies that B has all the same eigenvalues α_i, but to the eigenvectors $S\mathbf{v}_i$*

*(h) the **Rayleigh quotient** $R(A, \mathbf{x}) := \mathbf{x}^T A\mathbf{x} \,/\, \mathbf{x}^T\mathbf{x}$ satisfies for all vectors $\mathbf{x} \neq \mathbf{0}$ that*

$$\alpha_1 \quad \leq \quad \frac{\mathbf{x}^T A\mathbf{x}}{\mathbf{x}^T\mathbf{x}} \quad \leq \quad \alpha_n \quad .$$

As a consequence of Fact 1.3.1 (h) we get that if we are interested in the smallest or the largest eigenvalue of a given real symmetric matrix, then we can get simple upper/lower bounds on

them by evaluating $R(A, \mathbf{x})$ for arbitrary vectors $\mathbf{x} \neq \mathbf{0}$. Moreover, the extremest possible values of $R(A, \mathbf{x})$ are exactly these eigenvalues, and the respective minimizer/maximizer is a corresponding eigenvector:

$$\alpha_1 = \min_{\mathbf{x} \neq \mathbf{0}} \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = R(A, \mathbf{v}_1) \qquad \text{and} \qquad \alpha_n = \max_{\mathbf{x} \neq \mathbf{0}} \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = R(A, \mathbf{v}_n) \quad . \tag{1.3}$$

We refer to Equation (1.3) as the **Rayleigh quotient characterization** of the smallest/largest eigenvalue. A subtle but fascinating aspect of this characterization is that it bridges between two worlds: extremal eigenvalues and eigenvectors of a matrix on the one side, and an optimization problem on the other side. Section 1.3.6 shows how this bridge can be exploited by algorithms. This connection can even be extended to all other eigenvalues, by further restricting $\mathbf{x}$ to lie in the orthogonal complement of the eigenspace spanned by all smaller/larger eigenvectors. Precisely, for real symmetric matrix A let $E_{k-1} := \text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_{k-1}\}$ denote the $(k-1)$-dimensional linear subspace spanned by $k-1$ smallest eigenvectors of A. Then the k-smallest eigenvalue can be characterized together with a corresponding eigenvector as:

$$\alpha_{k+1} = \min_{\substack{\mathbf{x} \neq \mathbf{0} \\ \forall \mathbf{z} \in E_{k-1} : \mathbf{x}^T \mathbf{z} = 0}} \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = R(A, \mathbf{v}_k) \quad . \tag{1.4}$$

We refer to Equation (1.4) as the **Rayleigh quotient characterization** of the k-smallest eigenvalue. A similar result can be stated fully analogously for the k-largest eigenvalue. In particular we get that Equation (1.3) is a special case of the general characterization. The Rayleigh quotient characterization is a consequence of the **(Courant-Fisher-Weyl) minimax principle**, see for example the book by Bhatia (1997) for details.

We get from this characterization the following intuitive strategy to solve the **eigenvalue problem** $A\mathbf{x} = \alpha\mathbf{x}$ for real symmetric A: first minimize the Rayleigh quotient over all $\mathbf{x} \neq \mathbf{0}$ to get α_1 and $\mathbf{v}_1$. Then minimize the Rayleigh quotient over all $\mathbf{x} \neq \mathbf{0}$ subject to $\mathbf{x}^T \mathbf{v}_1 = 0$, which gives α_2 and $\mathbf{v}_2$. Continue in this way until α_n and $\mathbf{v}_n$. In particular, we get that the second-smallest eigenvalue is characterized by

$$\alpha_2 = \min_{\substack{\mathbf{x} \neq \mathbf{0} \\ \mathbf{x}^T \mathbf{v}_1 = 0}} \frac{\mathbf{x}^T A \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = R(A, \mathbf{v}_2) \quad . \tag{1.5}$$

Numerical eigenvalue solvers apply much more advanced strategies, particularly if they can deal with **sparse matrices** (matrices in which most entries are 0). Eigenvalue solvers work more efficiently and numerically more stable for symmetric matrices than for arbitrary matrices, and even better for positive (semi-)definite matrices. That is, whenever possible, we should prefer solving an eigenvalue problem of a positive semi-definite matrix, or at least of a symmetric matrix, rather than of an arbitrary matrix.

1.3.5 Graph matrices

The one-to-one correspondence between weighted graphs and weight matrices implies that all properties of a weighted graph are also encoded in its weight matrix, and vice versa. This is the fundamental connection between graph theory and matrix analysis. However, the complexity of formulating properties can differ extremely between the graph language and the matrix language, particularly if we focus only on linear algebra. Nevertheless, many links between graph properties and algebraic properties of the weight matrix are known:

- $\mathcal{G}(W)$ can be interpreted as an undirected graph if and only if W is symmetric
- for an unweighted graph, the number of different walks from vertex i to vertex j of length k is equal to the entry (i, j) in the matrix W^k
- $\mathcal{G}(W)$ contains no selfloop if and only if the trace of W (i.e., the sum of all main diagonal entries) is zero
- two graphs $\mathcal{G}(W)$ and $\mathcal{G}(W')$ are isomorphic if and only if W and W' are similar via some permutation matrix P, that is $W' = PWP^{-1}$
- $\mathcal{G}(W)$ does not contain any closed walk if and only if W is nilpotent (i.e., if there exists some k such that W^k is the all-zero matrix)

This list does by far not end here. Countless graph theoretical properties reflect in sometimes sophisticated properties of graph matrices, particularly in eigenvalues and eigenvectors. For example, as proven by Wilf (1967), if $\mathcal{G}(W)$ is a simple connected graph, then the minimum number c for which a c-coloring of the graph exists is bounded by $c \leq \alpha_n + 1$ for α_n the largest eigenvalue of W. While early work like this focused on the adjacency matrix, it turned out that some graph properties show up better in other graph matrices, particularly in the Laplacian matrices that are going to be introduced next. The correspondence between graphs and matrices further allows to think about modifications of matrices as of modifications of graphs. We elaborate such matrix-driven graph modifications in the later chapters.

Three standard variants of Laplacian matrices

A prominent family of alternative graph matrices, particularly in spectral graph theory, consists of the (unnormalized) Laplacian matrix L, the (symmetric) normalized Laplacian matrix $\mathcal{L}$, and the random-walk normalized Laplacian matrix $\mathcal{L}_{rw}$.

Definition 1.3.2 (unnormalized Laplacian matrix)
For an undirected graph of weight matrix W, the (unnormalized) Laplacian matrix $L = [l_{ij}]$ is defined as

$$L \;:=\; D - W \quad ,$$

with D the degree matrix. That is element-wise,

$$l_{ij} = \begin{cases} d_i - w_{ii} & i = j \\ -w_{ij} & i \neq j \end{cases} \quad .$$

L is also known in the literature as the *graph Laplacian*, the *discrete Laplacian*, the *combinatorial Laplacian*, and under several other names.

Definition 1.3.3 (symmetric normalized Laplacian matrix)
For an undirected graph of weight matrix $W \in \mathbb{W}$, the (symmetric) normalized Laplacian matrix $\mathcal{L} = [\ell_{ij}]$ is defined as

$$\mathcal{L} \;:=\; D^{-1/2} L D^{-1/2} \;=\; I - D^{-1/2} W D^{-1/2} \quad ,$$

with D the degree matrix. That is element-wise,

$$\ell_{ij} = \begin{cases} 1 - w_{ii}/d_i & i = j \\ -w_{ij}/\sqrt{d_i d_j} & i \neq j \end{cases} \quad .$$

We refer to $\mathcal{L}$ by the general term "normalized Laplacian", although there are alternative "normalized" variants. For example, the following definition introduces another variant that is closely related to $\mathcal{L}$ as well as to the transition matrix P.

Definition 1.3.4 (random-walk normalized Laplacian matrix)
For an undirected graph of weight matrix $W \in \mathbb{W}$, the random-walk normalized Laplacian matrix $\mathcal{L}_{rw} = [\ell^{rw}_{ij}]$ is defined as

$$\mathcal{L}_{rw} \;:=\; D^{-1/2} \mathcal{L} D^{1/2} \;=\; D^{-1} L \;=\; I - P \quad ,$$

with D the degree matrix and P the transition matrix. That is element-wise,

$$\ell^{rw}_{ij} = \begin{cases} 1 - w_{ii}/d_i & i = j \\ -w_{ij}/d_i & i \neq j \end{cases} \quad .$$

If we need to refer to W explicitly, we also denote these three variants of Laplacian matrices by $L(W)$, $\mathcal{L}(W)$ and $\mathcal{L}_{rw}(W)$, respectively. The following fact sheet lists some of their basic but nevertheless relevant properties.

Fact 1.3.5 (Properties of the three standard Laplacian matrices)
For $W \in \mathbb{W}$, the Laplacian matrices L, $\mathcal{L}$ and $\mathcal{L}_{rw}$ have the following properties:

(a) in L and in $\mathcal{L}_{rw}$, each row sums up to 0, that is $L\mathbf{1} = \mathcal{L}_{rw}\mathbf{1} = \mathbf{0}$. This further implies that $\mathbf{1}$ is in both cases an eigenvector to the eigenvalue 0.

(b) $\mathcal{L}$ is similar to $\mathcal{L}_{rw}$, thus it has the same eigenvalues with eigenvectors scaled by $D^{1/2}$. In particular, the vector $(\sqrt{d_i})$ is an eigenvector of $\mathcal{L}$ to eigenvalue 0.

(c) for each of L, $\mathcal{L}$ and $\mathcal{L}_{rw}$, the multiplicity of the eigenvalue zero equals the number of connected components in $\mathcal{G}(W)$. Hence the respective sorted list of eigenvalues $0 = \alpha_1 = \ldots = \alpha_k < \alpha_{k+1} \leq \ldots \leq \alpha_n$ attests k connected components.

(d) the eigenvalues of L are non-negative but unbounded. The eigenvalues of $\mathcal{L}$ and $\mathcal{L}_{rw}$ are bounded to $[0, 2]$.

(e) L and $\mathcal{L}$ are positive semi-definite, but $\mathcal{L}_{rw}$ is not (since not symmetric)

(f) $\mathcal{L}_{rw}$ has the same eigenvectors as P, with eigenvalues mapped by $\alpha \mapsto 1 - \alpha$

(g) $L(W) = L(W + X)$ for every diagonal matrix X, that is, changing the weights of selfloops does not change the unnormalized Laplacian matrix.

(h) $\mathcal{L}(W) = \mathcal{L}(c \cdot W)$ for all $c > 0$. Scaling all edge weights by the same factor does not change the normalized Laplacian matrix. The same holds true for $\mathcal{L}_{rw}$.

(i) L, $\mathcal{L}$ and $\mathcal{L}_{rw}$ are equal to the zero matrix if and only if $\mathcal{G}(W)$ is a loops-only graph

The basic results in Fact 1.3.5 cannot illustrate the rich variety of deep connections between structural graph properties and algebraic properties of graph matrices. Merris (1994) gives a comprehensive survey on graph properties that show up in the unnormalized Laplacian L. Chung (1997) is the standard reference for the normalized Laplacian $\mathcal{L}$, connecting its spectrum for example to graph cuts, flows and expansion properties. The survey by Lovász (1993) studies in particular graph connectivity in terms of eigenvalues and eigenvectors of the transition matrix P, thus of $\mathcal{L}_{rw}$. An in-depth discussion on the different characteristics of all three standard Laplacians is given by von Luxburg (2007).

1.3.6 Spectral clustering

Spectral clustering is a technique to detect clusters in a graph from the eigenvectors of its Laplacian matrices. This is achieved by describing clusters in terms of a minimization problem

that is finally relaxed to the Rayleigh quotient characterization of Laplacian eigenvalues. This section gives a brief introduction — see the tutorial by von Luxburg (2007) for details.

Balanced cuts

For a graph $G = (V, E, W)$, the **MinCut** optimization problem asks for a partition of V into two non-empty sets such that the weight of the corresponding cut is minimal. Formally,

$$\text{MinCut} \;:=\; \min_{\substack{S \subseteq V \\ S \notin \{\emptyset, V\}}} \; \text{cut}(S, \overline{S}) \quad . \tag{1.6}$$

The MinCut focuses on a cut of *absolute* smallest value, even if this is achieved by just cutting off a single vertex from the rest. Indeed, in most graphs the MinCut gives an extremely imbalanced partition. It is not suitable for the case where we want to separate two large clusters that have *relatively* few connections between them. For example, assume that we are given a connected similarity graph and we expect to find two clusters that are well-connected internally, but loosely connected to each other. The MinCut is not the right tool to find these two clusters. However, we can extend Equation (1.6) by adding a factor that penalizes strongly imbalanced clusters. This is a common strategy to prefer balanced cluster proportions, and yields the following generic objective of a **balanced cut** optimization problem:

$$\text{BalancedCut}_{\text{imbal}} \;:=\; \min_{\substack{S \subseteq V \\ S \notin \{\emptyset, V\}}} \; \text{imbal}(S, \overline{S}) \cdot \text{cut}(S, \overline{S}) \quad , \tag{1.7}$$

where imbal is an "imbalance penalty" function that gets the larger the stronger the imbalance between S and $\overline{S}$ is. This generic balanced cut criterion is also studied by Hein and Setzer (2011), who further provide a general approximation algorithm to solve it. A popular balanced cut is the CheegerCut, defined by $\text{imbal}(S, \overline{S}) := 1/\min\{|S|, |\overline{S}|\}$. Strongly related to spectral clustering is the **RatioCut**, defined as follows:

$$\text{RatioCut} \;=\; \min_{\substack{S \subseteq V \\ S \notin \{\emptyset, V\}}} \; \left(\frac{1}{|S|} + \frac{1}{|\overline{S}|} \right) \cdot \text{cut}(S, \overline{S}) \quad . \tag{1.8}$$

The RatioCut treats vertices of high degree in the same way as vertices of low degree. This can be desired or not in applications. An alternative, denoted as the **Normalized Cut (NCut)**, balances the size of clusters by their volume rather than by their cardinality:

$$\text{NCut} \;:=\; \min_{\substack{S \subseteq V \\ S \notin \{\emptyset, V\}}} \; \left(\frac{1}{\text{vol}(S)} + \frac{1}{\text{vol}(\overline{S})} \right) \cdot \text{cut}(S, \overline{S}) \quad . \tag{1.9}$$

The choice of the right balancing method for balanced cut minimization depends on the application, or more abstract, on the graph model. In any case we have to choose some balancing method, since the MinCut alone does not give meaningful results.

Spectral relaxation

As originally proven by Hagen and Kahng (1992), the RatioCut can be approximated in terms of the second-smallest eigenvector of the unnormalized Laplacian matrix L. Two steps are involved to achieve their result: "rewrite" and "relax". For the "rewrite" step one can verify that Equation (1.8) is equivalent to

$$\text{RatioCut} \;=\; \min_{\substack{\mathbf{u}\in\mathcal{U}\\ \mathbf{u}^T\mathbf{1}=0}} \frac{\mathbf{u}^T L\mathbf{u}}{2n} \quad , \tag{1.10}$$

where $\mathcal{U}$ is defined as the discrete set of all vectors $\mathbf{u} = (u_i)$ of the form

$$u_i \;=\; \begin{cases} +\sqrt{|\overline{S}|/|S|} & , \; i \in S \\ -\sqrt{|S|/|\overline{S}|} & , \; i \in \overline{S} \end{cases} \quad , \text{ for every non-empty } S \subsetneq V \quad . \tag{1.11}$$

The optimization problem (1.10) over the set $\mathcal{U}$ relates RatioCut to the unnormalized Laplacian matrix L in an exact way, so it remains to be NP-hard. Now observe that each of the $2^n - 2$ many vectors $\mathbf{u} \in \mathcal{U}$ has the squared length $n = \sum_i u_i^2 = \|\mathbf{u}\|^2 = \mathbf{u}^T\mathbf{u}$, thus we get that $\mathcal{U} \subsetneq \{\mathbf{f} \in \mathbb{R}^n \mid \mathbf{f}^T\mathbf{f} = n\} =: \mathcal{S}$. The "relax" step is a **relaxation** of the discrete optimization problem by extending the original set of feasible solutions $\mathcal{U}$ to the larger set of all vectors $\mathbf{f}$ of length $\sqrt{n}$, that is $\mathcal{S}$. The bad news is that the minima of the relaxed problem can lie outside of $\mathcal{U}$, in $\mathcal{S} \setminus \mathcal{U}$. However, the good news is that the problem is no longer NP-hard, but can be solved efficiently as an eigenvalue problem. Precisely, we see that the relaxation of (1.10) equals Equation (1.5) for $\mathbf{v}_2$ the second-smallest eigenvector of L:

$$\text{RatioCut} \;=\; \min_{\substack{\mathbf{u}\in\mathcal{U}\\ \mathbf{u}^T\mathbf{1}=0}} \frac{\mathbf{u}^T L\mathbf{u}}{2n} \;\overset{\text{relax}\,\to}{\approx}\; \min_{\substack{\mathbf{f}^T\mathbf{f}=n\\ \mathbf{f}^T\mathbf{1}=0}} \frac{\mathbf{f}^T L\mathbf{f}}{2n} \;\Leftrightarrow\; \min_{\substack{\mathbf{f}\neq\mathbf{0}\\ \mathbf{f}^T\mathbf{1}=0}} \frac{\mathbf{f}^T L\mathbf{f}}{\mathbf{f}^T\mathbf{f}} \;=\; R(L, \mathbf{v}_2) \quad .$$

The notation "$\min \cdot \;\Leftrightarrow\; \min \cdot$" refers to the relation "$\arg\min \cdot = \arg\min \cdot$", so the minimum value may change (here, by dropping the factor $1/2$), but the set of minimizers is unchanged.

Taking a similar road, it was proven by Shi and Malik (2000) that the NCut can be approximated in terms of the second-smallest eigenvector of the random-walk normalized Laplacian matrix $\mathcal{L}_{rw}$.

Spectral clustering algorithms

The generic approach to **two-class spectral clustering** of a connected graph is shown by Algorithm 1.3.2 (`2-SpectralClustering`). The basic idea is to take the second-smallest eigenvector of one of the Laplacian matrices, L, $\mathcal{L}$ or $\mathcal{L}_{rw}$, and then derive a two-partition of the graph vertices by observing whether the corresponding entry in this eigenvector is positive or not. For L, that is for approximating the RatioCut, this rounding strategy is suggested by Equation 1.11. A similar result (although not symmetric at 0) holds for NCut, that is for $\mathcal{L}_{rw}$.

Algorithm: `2-SpectralClustering`

Input:

- $G = (V, E, W)$ connected undirected graph
- selected type of Laplacian matrix: L, $\mathcal{L}$, or $\mathcal{L}_{rw}$

Output:

- $V = S \cup \overline{S}$ partition of V

Algorithm:

1. determine the second-smallest eigenvector $\mathbf{v}_2$ of the desired matrix (L, $\mathcal{L}$, or $\mathcal{L}_{rw}$)
2. define the set $S \subseteq V$ from $\mathbf{v}_2 = (v_i)$ by thresholding at 0, that is $i \in S \Leftrightarrow v_i \leq 0$

Algorithm 1.3.2: `2-SpectralClustering` algorithm

Let us put some comments on this approach:

- the algorithm's output is exactly the same for $\mathcal{L}$ and $\mathcal{L}_{rw}$, since scaling by $\sqrt{d_i}$ does not change the sign of v_i
- the second-smallest eigenvector of $\mathcal{L}_{rw}$, which is equal to the $D^{-1/2}$-multiplied second-smallest eigenvector of $\mathcal{L}$, is denoted as the **NCut score vector** of W. Its entries indicate the strength of "belief" in whether to assign the corresponding vertex to S or $\overline{S}$.
- rather than thresholding at 0, one can replace Step 2 by determining an optimal split point among several candidates (for example at every v_i), where "optimal" refers to the smallest achieved value of the corresponding objective function
- the algorithm can be generalized to unconnected graphs in various ways, for example by prepending a separate connectivity algorithm that returns any two disjoint unions of connected components if the graph is not connected

Spectral clustering can be extended to more than two clusters. The definitions of RatioCut and NCut generalize straightforward. Further, the spectral relaxation can be carried out in a similar way, incorporating the smallest k eigenvectors. This type of generalization is denoted as the **multi-vector approach**. Figure 1.3.3 shows the multi-vector approach for normalized spectral clustering for k clusters, denoted as `NormalizedSpectralClustering`.

Algorithm: `NormalizedSpectralClustering`

Input:

- $G = (V, E, W)$ undirected graph with $V = \{1, \ldots, n\}$ and weight matrix $W \in \mathbb{W}$
- k desired number of clusters

Output:

- $V = V_1 \cup \ldots \cup V_{k'}$ partition of V into $k' \leq k$ clusters

Algorithm:

1. determine the k smallest eigenvectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ of the random-walk normalized Laplacian matrix $\mathcal{L}_{rw}$
2. for every $i \in V$, define $\mathbf{a}_i \in \mathbb{R}^k$ from the i'th entry in each of $\mathbf{v}_1, \ldots, \mathbf{v}_k$. That is, with $A \in \mathbb{R}^{n \times k}$ having $\mathbf{v}_j$ as its j'th column, $\mathbf{a}_i$ equals the i'th row of A.
3. apply the k-means algorithm to the set of points $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \in \mathbb{R}^k$ in order to determine up to k clusters $V_1 \cup \ldots \cup V_{k'}$

Algorithm 1.3.3: `NormalizedSpectralClustering` algorithm for k clusters

Let us again put some comments:

- in practice it can be beneficial to avoid to compute the eigenvectors of the non-symmetric matrix $\mathcal{L}_{rw}$ directly, but to compute the eigenvectors of the positive semi-definite matrix $\mathcal{L}$ and finally scale them all by $D^{-1/2}$, or to solve the generalized eigenvalue problem $L\mathbf{x} = \alpha D\mathbf{x}$ in another way
- instead of k-means one can apply any other clustering algorithm in Step 3. A common alternative is the single linkage algorithm (Sibson, 1973).
- the same algorithm can be applied without changes to L in place of $\mathcal{L}_{rw}$. For $\mathcal{L}$, however, it is proposed by Ng et al. (2002) to additionally normalize each $\mathbf{a}_i$ to unit length before applying k-means.

An alternative strategy to generalize two-class spectral clustering to k classes is by recursive bi-partitioning, denoted as **recursive spectral clustering**. This approach generates a hierarchical cluster-tree, initialized at the root by applying the two-class spectral clustering algorithm that splits V in two partitions $V = V_1 \cup V_2$. Then, for each of V_1 and V_2 independently, the two-class spectral clustering algorithm is applied to the respective subgraph, giving the sub-partitions $V_1 = V_{1,1} \cup V_{1,2}$ and $V_2 = V_{2,1} \cup V_{2,2}$. The algorithm continues recursively until some desired pruning criterion is met (for example the desired number of clusters), or until

every sub-partition consists of only a single vertex. In the latter case this algorithm creates a full hierarchical cluster-tree with n leafs. Verma and Meila (2003) compare some multi-vector approaches and recursive approaches. They conclude that all compared approaches behave similar on "easy" instances, while on "hard" instances there is no clear winner. However, they consider the multi-vector approach to perform slightly better if there is clear structure in the data. Further, the multi-vector approach is simpler to implement and much more popular in practice.

1.4 Summary of the main results and publications

All that we get from the bare definitions of the normalized Laplacian $\mathcal{L} = [\ell_{ij}]$ and the unnormalized Laplacian $L = [l_{ij}]$ are two graph matrices that are related to each other by $\ell_{ij} = l_{ij}/\sqrt{d_i d_j}$. If we add the information that for a connected graph the second-smallest eigenvector of L is helpful for solving the RatioCut optimization problem — what can we say about $\mathcal{L}$? Since there are no general results on how eigenvectors are affected by distorting all entries of a matrix, we would probably not expect to get any similar meaningful result for $\mathcal{L}$. It would rather be surprising to find out that the second-smallest eigenvector of $\mathcal{L}$ helps on solving a similar problem, precisely on solving the NCut optimization problem.

Of course, if we further consider all results about spectral graph theory from the previous section, then the formerly hidden connection between L and $\mathcal{L}$ will become clear. However, this does not enable us to understand other types of modifications applied to a Laplacian matrix. Consider for example the following three "riddles":

1. Take the normalized Laplacian matrix $\mathcal{L}$ and set its main diagonal to arbitrary new values. Can you provide a meaningful interpretation of the result?

2. Iteratively divide by $\sqrt{d_i d_j}$ again and again, each time by the new degrees of the implicitly re-scaled weight matrix. How is the limit of this process related to $\mathcal{L}$?

3. Consider a real-weighted graph and define its degree matrix D from the *absolute* values of the edge weights. Which insights can we get from the smallest eigenvectors of L?

All three riddles describe some modification of a standard Laplacian matrix and then ask for an interpretation of doing so. We are likely not able to come up with an answer to these questions immediately. In fact, the answers turn out to show a complexity that allows to write an entire doctoral thesis about them, and to publish two papers on reputable international conferences. These three questions are precisely at the heart of Chapter 2, Chapter 3, and Chapter 4, respectively. In the remainder of this section, I provide some details on the respective context and on my main contributions in it. A technically deeper summary is given by the "Chapter summary" section to the end of each chapter.

- **Chapter 2 ("The f-adjusted Laplacian")**. Introduces two graph modifications: **f**-scaling and **f**-selflooping. Their concatenation is denoted as **f**-adjusting, which transforms a given graph into another graph of degree vector **f**. I prove that **f**-adjusting corresponds to modifying the normalized Laplacian matrix $\mathcal{L}$ only along its main diagonal. Moreover, if **f**-adjusting is applied to a random geometric neighborhood graph, then we can interpret this graph modification as a deformation of the underlying probability density. Cuts and volumes in the modified graph refer no longer to the original density, but to a different density that can be stated precisely. This interpretation of **f**-adjusting allows for a variety of applications in machine learning, for example to correct for a sampling bias. My results on **f**-adjusting are published at the International Conference on Machine Learning (Kurras et al., 2014).

- **Chapter 3 ("Symmetric iterative proportional fitting")**. Studies an alternative strategy to transform a given graph into another graph of degree vector **f**. The idea is to apply **f**-scaling iteratively until the resulting matrix converges to some limit matrix. Iterated **f**-scaling turns out to be related to a field of research that is known as matrix scaling, in particular to "Iterative Proportional Fitting (IPF)". I contribute to this field by providing a novel interpretation of IPF as a special case of a more general projection algorithm that explains simultaneously its convergence plus an interpretation of its limit matrix. I further provide a convergence proof for iterated **f**-scaling, which can be seen as a symmetrized variant of IPF that fits better to the case of symmetric matrices. Moreover, I show for the case of symmetric matrices that certain expansion properties of the corresponding graph characterize precisely the convergence behavior of iterated **f**-scaling. Again these results have an impact on different applications in machine learning. This work is published at the International Conference on Artificial Intelligence and Statistics (Kurras, 2015).

- **Chapter 4 ("Spectral correlation clustering")**. This chapter steps away from positive-weighted graphs to correlation graphs. These graphs have positive and negative edge weights that quantify the similarity or dissimilarity between vertices, respectively. Correlation clustering focuses on detecting cluster structures in correlation graphs. In contrast to similarity graphs, for correlation graphs the MinCut is highly informative, but unfortunately NP-hard even for two classes. Together with this shift of relevance of the MinCut, the relevance of the Laplacian's *smallest* eigenvector increases from trivial to highly interesting. I prove that the Rayleigh quotient characterization of the smallest eigenvector of the so-called signed Laplacian matrix is a relaxation of an "optionally" applied MinCut, denoted as OptMinCut. The same eigenvector is already derived in the literature as a relaxation from a signed variant of the balanced RatioCut, denoted as SignedRatioCut. However, I provide evidence that we shall interpret this eigenvector better as an approximation of OptMinCut rather than of SignedRatioCut. I suggest to carry out recursive spectral clustering on the smallest eigenvector of the

signed Laplacian as the canonical approach to correlation clustering by spectral methods. This approach behaves differently from the recursive approach for similarity graphs, for example, it prunes automatically whenever no negative cut exists. In contrast to the previous chapters, my work on this chapter was originally motivated by a concrete practical application: detecting cheaters in an online game. In order to finally apply my spectral correlation clustering solution to this context, an extensive software engineering overhead is required that is an additional substantial part of the chapter. The results of this youngest chapter are not yet published, but I plan to prepare a publication postdoctoral.

The two relevant publications including their abstracts are listed in Table 1.1 (overleaf).

Sven Kurras, Ulrike von Luxburg, Gilles Blanchard

The f-adjusted graph Laplacian: a diagonal modification with a geometric interpretation

31st International Conference on Machine Learning (ICML)
JMLR Workshop and Conference Proceedings, volume 32, pages 1530–1538, 2014
Editors: Eric P. Xing, Tony Jebara

Abstract: Consider a neighborhood graph, for example a k-nearest neighbor graph, that is constructed on sample points drawn according to some density p. Our goal is to re-weight the graph's edges such that all cuts and volumes behave as if the graph was built on a different sample drawn from an alternative density q. We introduce the **f**-adjusted graph and prove that it provides the correct cuts and volumes as the sample size tends to infinity. From an algebraic perspective, we show that its normalized Laplacian, denoted as the **f**-adjusted Laplacian, represents a natural family of diagonal perturbations of the original normalized Laplacian. Our technique allows to apply any cut and volume based algorithm to the **f**-adjusted graph, for example spectral clustering, in order to study the given graph as if it were built on an unaccessible sample from a different density. We point out applications in sample bias correction, data uniformization, and multi-scale analysis of graphs.

Sven Kurras

Symmetric Iterative Proportional Fitting

18th International Conference on Artificial Intelligence and Statistics (AISTATS)
JMLR Workshop and Conference Proceedings, volume 38, pages 526–534, 2015
Editors: Guy Lebanon, S.V.N. Vishwanathan

Abstract: Iterative Proportional Fitting (IPF) generates from an input matrix W a sequence of matrices that converges, under certain conditions, to a specific limit matrix W^*. This limit is the relative-entropy nearest solution to W among all matrices of prescribed row marginals $\mathbf{r}$ and column marginals $\mathbf{c}$. We prove this known fact by a novel strategy that contributes a pure algorithmic intuition. Then we focus on the symmetric setting: $W = W'$ and $\mathbf{r} = \mathbf{c}$. Since IPF inherently generates non-symmetric matrices, we introduce two symmetrized variants of IPF. We prove convergence for both of them. Further, we give a novel characterization for the existence of W^* in terms of expansion properties of the undirected weighted graph represented by W. Finally, we show how our results contribute to recent work in machine learning.

Table 1.1: List of my publications as a doctoral candidate.

Chapter 2

The f-adjusted Laplacian

2.1 Chapter introduction

Assume that we are interested in the temperature distribution over Greenland, or in the concentration of ozone in the stratosphere, or in the degree of contamination around an oil drilling in the deep sea. All these applications have in common that they study a spatial distribution of a certain intensity measure (in general, of any measure on a subset of $\mathbb{R}^d$). This setting is denoted as an **intensity landscape**. Our goal is to reveal structural properties of such an intensity landscape. The challenge is that our access is limited to a discrete partial observation, our "input data". For example, we may be given a finite collection of sample points that provide us with the intensity values at specific point-wise locations.

If the given sample points lie on a regular grid, then the input data can be interpreted as a d-dimensional rasterized image of the intensity landscape. Such data can be processed with every technique from (d-dimensional) image analysis. In particular, we can apply **spectral image clustering** (Shi and Malik, 2000) in order to determine meaningful segments of homogeneous intensities. We now describe this approach in more details: first, a neighborhood graph is constructed by connecting all pairs of locations that lie close to each other. In case of a regular grid, one typically chooses an r-graph that connects each point to its direct grid neighbors. An edge between sample points i and j is provided with the weight

$$w_{ij} \quad := \quad \mathrm{sim}_s(\mathbf{x}_i, \mathbf{x}_j) \cdot \mathrm{sim}_v(t_i, t_j) \quad , \tag{2.1}$$

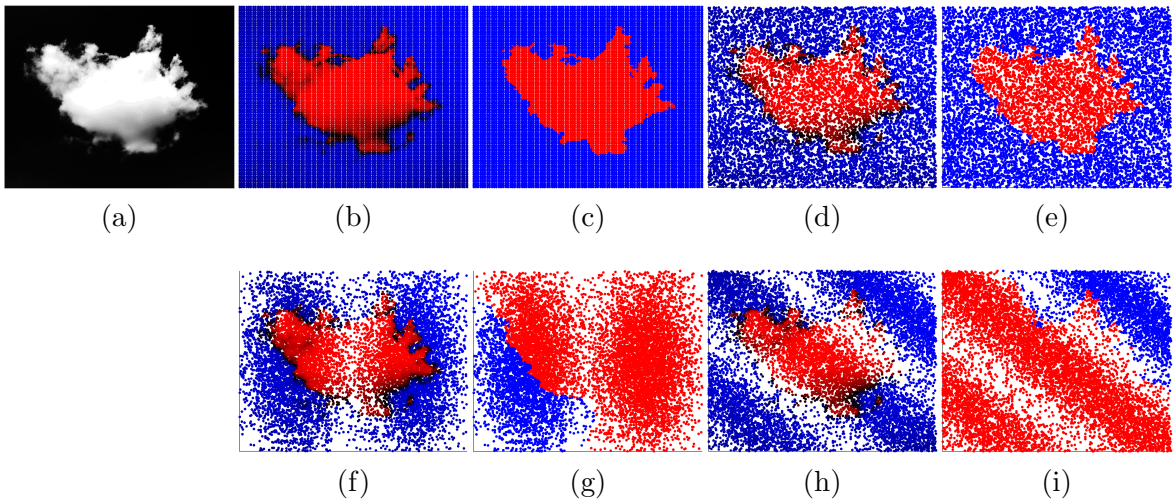


Figure 2.1.1: Spectral clustering of an intensity landscape. (a) continuous two-dimensional intensity landscape that forms the shape of a cloud. (b) discrete snapshot of the intensity landscape at the positions of a regular grid. Blue color indicates low intensities, black moderate, and red high intensities. (c) result of spectral image clustering applied to the grid sample. The true shape is successfully separated from the background. (d) the same intensity landscape accessed at uniform random positions. (e) result of spectral image clustering applied to the uniform sample. The true shape is successfully separated from the background. (f), (h) the same intensity landscape accessed at non-uniform random positions. (g), (i) results of spectral image clustering applied to the non-uniform samples. Both results are misguided by the non-uniform sample to a totally wrong segmentation.

defined as the product of the **spatial similarity** $\mathrm{sim}_s(\mathbf{x}_i, \mathbf{x}_j) := \exp(-0.5\|\mathbf{x}_i - \mathbf{x}_j\|^2/\sigma_s^2)$ of the locations $\mathbf{x}_i, \mathbf{x}_j$, and the **value similarity** $\mathrm{sim}_v(t_i, t_j) := \exp(-0.5\|t_i - t_j\|^2/\sigma_v^2)$ of the corresponding intensity values t_i, t_j. The bandwidths σ_s and σ_v are chosen individually for each domain. This approach defines a weighted graph that makes properties of the intensity landscape accessible as corresponding graph properties. Finally, spectral clustering as defined in Section 1.3.6 is applied to this graph in order to detect meaningful intensity clusters. Figure 2.1.1 (c) shows an example of this approach.

But what if we cannot access the intensity landscape in such a convenient way as provided by a regular grid? What if our access is restricted to locations that are randomly distributed according to a separate sampling distribution? The case of a uniform sampling distribution turns out to be unproblematic because the random values $\mathrm{sim}_s(\mathbf{x}_i, \mathbf{x}_j)$ behave everywhere in the same way. In particular, spectral image clustering on a uniform sample of an intensity landscape still yields meaningful intensity clusters, as shown in Figure 2.1.1 (e). However, if the sample points are distributed non-uniformly, they introduce systematic errors to the weights w_{ij} via the location-dependent expectation of $\mathrm{sim}_s(\mathbf{x}_i, \mathbf{x}_j)$. As a consequence, the above graph construction no longer gives an adequate discrete representation of the intensity landscape.

It represents a mixture of the intensity landscape and the unwanted sampling distribution instead. As shown in Figure 2.1.1 (f)-(i), applying the above approach to non-uniform sample positions can lead to undesired results.

There is a straightforward approach to circumvent this problem: *histograms*, also denoted as the **binning approach**. The binning approach divides the coordinate space into regular bins ("grid cells" of equal size). For each bin, its average intensity can be estimated by averaging over the intensities of all sample points that fall inside the bin. This approach transforms an arbitrary sample into a regular grid, defined at the bins' centers. In particular, one can now apply standard image analysis.

However, there are some issues with the binning approach:

- **Required coordinates.** Histograms, like other function approximation schemes, require access to global sample point coordinates. However, some applications do not provide such, but only pairwise distances $\|\mathbf{x}_i - \mathbf{x}_j\|$, or even a weaker notion of pairwise spatial similarities sim_s. Such a restricted scenario is common in studying sensor networks (Younis and Fahmy, 2004) and swarm robotics (Cornejo and Nagpal, 2015). Rather than global positions, the individual entities observe independent local coordinate systems, if at all. In practice, GPS is often not accessible because of physical reasons (e.g., underwater, extraterrestrial), monetary reasons or energy-constraints. Another scenario is given by social graphs, where pairwise similarity values (e.g., the number of common topic reads) are provided without an explicit embedding of the sample points into a coordinate space. A third scenario without explicit coordinates is given by iterative sampling strategies of structured objects. Here, sample points are generated by applying sequences of tiny random mutations to an initial object. This has applications for example in peer-to-peer networks (Cooper et al., 2009) and in automated software test generation (Jia and Harman, 2011). Whenever global coordinates are not accessible, binning cannot be applied.

- **Non-adaptivity.** Since algorithms from image analysis require a regular grid, all bins must have the same size. This implies a varying number of sample points per bin. Those bins that contain many sample points lead to a wastefully precise estimate on the bin's average intensity. Rather than further improving the bin's estimation quality, one would prefer to sub-divide such overstaffed bins into sub-bins of smaller size to yield more fine-granular results. Such an adaptive approach gets particularly interesting if the sampling distribution is not independent of the intensity landscape, but concentrated along the decision boundary.

To the best of my knowledge, there exists so far no approach to clustering of intensity landscapes that is coordinate-free and adaptive. In this work, I provide a first such approach. It does not require sample coordinates, and areas of higher sampling density are adaptively considered at higher resolution without biasing the clustering result. The key to this approach is a novel technique introduced in this chapter: **f**-adjusting. It can be applied to spectral

clustering of intensity landscapes in a way that solves the above introduced problem of non-uniform sampling distributions. This application is re-visited in Section 2.6.4.

In the following analysis, rather than dealing with intensity values on top of a sampling distribution, we solely focus on the sampling distribution itself. On a high level, this chapter studies how certain graph properties (such as vertex degrees, volumes, cut weights) of a random geometric neighborhood graph correspond to properties of the underlying sampling distribution. We particularly address the following four questions:

Questions motivating this chapter

(**Q1**) Can we relate vertex degrees in an RGNG to the underlying sampling distribution?

(**Q2**) Are there similar correspondences for volumes and cuts, respectively?

(**Q3**) Can we preserve such correspondences if we modify edge weights to new values?

(**Q4**) Can such edge modifications be harnessed for machine learning applications?

Questions (**Q1**) and (**Q2**) are already studied in the literature, as presented in Section 2.3.2 and Section 2.3.4. Questions (**Q3**) and (**Q4**) can be answered in the affirmative as a result of my contributions that are presented in this chapter. This chapter introduces "**f**-adjusting" as a novel tool to modify a given RGNG in a specific way.

A precise definition of **f**-adjusting is given later in Definition 2.3.6, and fundamental properties of **f**-adjusting are studied as the main contributions of this chapter.

Let us now introduce the basic idea of **f**-adjusting along with Figure 2.1.2. This perspective on **f**-adjusting is denoted as its **geometric interpretation**. We are given an RGNG G (top left in Figure 2.1.2) that is built from i.i.d. sample points of a probability density p (bottom left). p is "unknown" in the sense that no representation of p is accessible by the application. However, some structural properties of p reflect in graph properties of G as outlined before. Now assume that our main goal is to study *another* density function $\overline{p}$ that is *different* from p. For example, $\overline{p}$ may refer to the uniform distribution on the same support as p, or as in Figure 2.1.2 (bottom right), $\overline{p}$ may refer to an intensified variant of p. The ideal approach to gain insights on $\overline{p}$ is by studying another RGNG G^* (top right) that is built on top of a new sample, drawn from $\overline{p}$. The challenge in our setting is that we cannot obtain such a graph G^* based on a sample from $\overline{p}$. Further, $\overline{p}$ is unknown to us just like p. However, we assume that we *do* know how to "modify" p in order to get $\overline{p}$. The relation between p and $\overline{p}$ is in terms of the equation $\overline{p} \sim f \cdot p$ for some function f, where the proportionality operator $\sim$ means that there exists a scalar $\alpha > 0$ such that $\overline{p} = \alpha \cdot f \cdot p$. That is, $\overline{p}$ and $f \cdot p$ are "equal up to normalization". The example in Figure 2.1.2 assumes that $\overline{p} \sim p^2$, which implies that $f = p$. In this case, our knowledge is that we have to square the unknown underlying density. Another example: whenever we are interested in the uniform distribution $\overline{p} \sim 1$, then we imply that $f = p^{-1}$. In this case we know that we have to multiply the unknown underlying

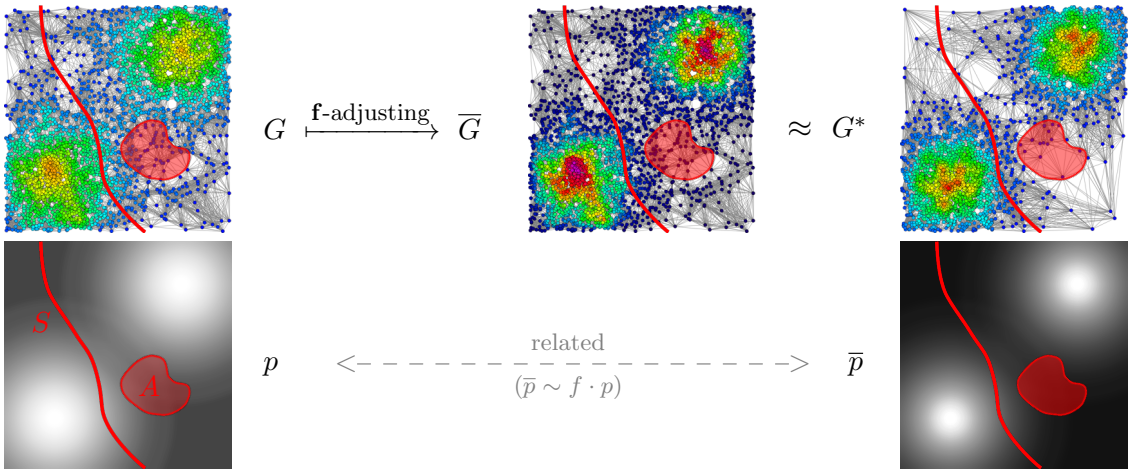


Figure 2.1.2: Geometric interpretation of **f**-adjusting. (bottom left, bottom right) two different probability densities defined on the unit square. White color indicates high probability. Here, $\overline{p}$ is an intensified variant of p. Precisely, $\overline{p} \sim p^2$, hence, $f = p$. Further, S cuts the space in two subsets, and A is another subset of the space. (top left, top right) RGNG built on top of sample points drawn from the respective underlying sampling distribution. Edges are colored in gray. Vertex heat colors correspond to the vertex degree, where red indicates high degrees. (top center) graph derived from G by **f**-adjusting. All edges are re-weighted in such a way that $\overline{G}$'s weight of the cut S and $\overline{G}$'s volume of the set A match the corresponding quantities of G^*, although $\overline{G}$ has the same sample points and graph structure as G.

density by its inverse. Precisely, we assume to have some knowledge on the transformation function $\mathcal{T}$ that transforms p to become proportional to $\overline{p}$. Formally: $\mathcal{T}(p) = f \cdot p \sim \overline{p}$.

Let us summarize the setting so far: our goal is to learn about $\overline{p}$, but we cannot get a sample or an RGNG from it. Instead, we are given an RGNG from another unknown density p, plus the knowledge on how p and $\overline{p}$ are related.

Now the basic idea addressed by **f**-adjusting is as follows: *can we directly modify the edge weights of the given graph G in a way that "corresponds" to going from p to $\overline{p}$?* Of course, we cannot avoid that the result $\overline{G}$ (top middle) of any such transformation of G will differ in many aspects from a graph G^* that is *really* built on top of $\overline{p}$. In particular, $\overline{G}$ can only refer to the same p-distributed sample points as G and to the same connectivity structure as G, while G^* is built on a different $\overline{p}$-distributed sample along with its own set of edges. However, can we at least define $\overline{G}$ in such a way that the cut weights and volumes in $\overline{G}$ are approximately the same as the cut weights and volumes in G^* ? Indeed, **f**-adjusting gives a positive answer to this question. Its parameter $\mathbf{f}$ is a vector that must be chosen according to the desired transformation function f.

This brief overview sweeps some crucial details under the rug that need to be made precise in the following. For example, in order to compare $\overline{G}$ to G^* we have to define a notion of cut weights and volumes that allows to compare geometric graphs that are built on different sample points. Ignoring such details, the overall strategy of **f**-adjusting can be memorized as follows:

> **"f-adjusting transforms a given RGNG G into another graph $\overline{G}$, whose cut weights and volumes behave as if $\overline{G}$ were an RGNG that was built on a different sampling distribution."**

2.2 Informal summary of the main contributions

The two main contributions of this chapter provide complementary mathematical foundations of the **f**-adjusting technique. Contribution 1 (Section 2.4.1) shows that the *geometric interpretation*, as presented in Figure 2.1.2, is indeed valid: we may think of **f**-adjusting as of such a density-shift. As foreshadowed above, there are some hurdles to take in order to be able to translate the informal intuition into mathematical formalisms. This particularly requires to carefully work out three different notions of volumes and cuts for geometric graphs. Once the intuition is formalized, a proof of its correctness is given by Theorem 2.4.1. Although the analytical details are quite technical, the computation of the **f**-adjusted graph itself is a simple and efficient operation. Moreover, it is the geometric intuition rather than the technical details that finally motivates a couple of applications of **f**-adjusting in machine learning.

Contribution 2 (Section 2.4.2) provides an *algebraic interpretation* of **f**-adjusting by revealing a neat equivalence between **f**-adjusting and a natural family of diagonal modifications of the normalized Laplacian matrix. This reveals that **f**-adjusting is not an artificial transformation of edge weights, but an elementary graph modification of algebraic simplicity. The algebraic results do not require that G is a geometric graph. However, for geometric graphs we get the additional fact that we can think of diagonal modifications of the normalized Laplacian matrix as of a geometrically intuitive density-shift.

2.3 Formal setup

This section introduces all concepts and formalisms that are required to understand the main contributions in the follow-up section.

2.3.1 Random geometric neighborhood graphs in the large sample limit

A general approach to the analysis of algorithms is by studying the algorithm's performance for an arbitrary large input size $n \to \infty$. This principle is standard in runtime analysis, but it also fits well to other properties of an algorithm. For example, assume that we train a classification algorithm $\mathcal{A}$ on a training sample that is arbitrary large; that is we study the classification performance of $\mathcal{A}$ in the **large sample limit** $n \to \infty$. If the feature space is bounded (which is a natural assumption in applications), then in the limit the sample points lie dense in space. Hence, intuitively, more and more training data should enable a classification algorithm to approximate the exact localization of the correct decision boundary better and better. In the large sample limit, it is reasonable to demand from $\mathcal{A}$ that it should provide the *best possible* classification performance. A classification algorithm that satisfies this optimum limit property is denoted as being *consistent* — see for example Shalev-Shwartz and Ben-David (2014) for a formal introduction.

This chapter studies algorithms on random geometric graphs in the large sample limit, where n refers to the number of sample points on which the graph is constructed. Whenever shifting n to infinity, one has to make precise how to tune all other parameters with respect to n. For example, how to choose the neighborhood radius r_n of the r_n-graph for $n \to \infty$? If $r := r_n$ is fixed to a constant, then neighborhoods cannot adapt to a more fine granular decision boundary. Therefore, we require r_n to shrink to 0 with increasing n. However, r_n must not shrink too fast if we want to provide connectivity of the r_n-graph with high probability. Similarly, the kNN-graph on a random sample will be disconnected with high probability if we fix k to stay constant while $n \to \infty$. Here, we have to increase k_n fast enough with n, but not too fast, in order to study meaningful connected graphs for all n. Identifying interesting regimes of convergence rates is subject of separate work, see for example Penrose (2003).

The geometric results in this chapter refine the work by Maier et al. (2009). For that reason we assume the same **limit scenarios** for the different types of geometric graphs as in their work.

Definition 2.3.1 (Limit scenarios)
The parameters of the studied neighborhood graphs are linked to $n \to \infty$ as follows:

- σ_n of the Gaussian graph: $\sigma_n \to 0$ at rate $n\sigma_n^{d+1}/\log n \to \infty$.
- r_n of the r-graph: $r_n \to 0$ at rate $nr_n^{d+1}/\log n \to \infty$.
- k_n and σ_n of the weighted kNN graph: $k_n \to \infty$ and $\sigma_n \to 0$ at rates $k_n/\log n \to \infty$, $k_n/n \to 0$ and $n\sigma_n^{d+1}/\log n \to \infty$, with $(k_n/n)^{1/d} \geq \sigma_n^{\alpha}$ for some $\alpha \in]0, 1[$.

The limit scenarios of Definition 2.3.1 ensure to study "meaningful" RGNGs that connect the whole sampleable space by small neighborhoods. For the r-graph, the radius r_n shrinks to zero at a rate slower than $\sqrt[d+1]{\log n/n}$. This rate is slow enough to guarantee with high probability that the r_n-graph is connected. As desired, such a graph connects the whole sampleable space by a covering of vanishing small neighborhoods. Its unweighted diameter (that is the length of a longest unweighted shortest path) goes to infinity. For the kNN graph, k_n grows faster to infinity than $\log n$, but slower than n. As for the r-graph, this regime guarantees with high probability that the graph is connected, although all neighborhoods cover vanishing small space. The weighted kNN graph additionally defines how to set the bandwidth parameter σ_n according to n and k_n. Here, the rate is chosen such that for most vertices only a negligible amount of total edge weight is truncated. The Gaussian graph is always fully connected, however, its bandwidth parameter shrinks to zero at a rate that is slow enough to "connect" the graph by a skeleton of significant edge weights.

Some of the bounds in Definition 2.3.1 are stricter than necessary for guaranteeing connectivity. This is an artifact of additional requirements in the proofs by Maier et al. (2009) for their quantitative results on volumes and cuts in RGNGs.

We further assume the following **regularity conditions** on the underlying distribution, again the same as in Maier et al. (2009).

Definition 2.3.2 (Regularity conditions)
The probability density $p : \mathbb{R}^d \to \mathbb{R}$ has support $\mathcal{X} := \{x \in \mathbb{R}^d \mid p(x) > 0\}$. Further,

- $\mathcal{X}$ is compact with smooth boundary and bounded curvature,
- p is bounded away from zero by some $p_{min} > 0$, that is $p(x) \geq p_{min}$ for $x \in \mathcal{X}$,
- p is twice differentiable with bounded gradient (on the interior of $\mathcal{X}$).

In particular it follows that p is also upper bounded by some $p_{max} \geq p(x)$.

These regularity conditions reduce the complexity of the mathematical analysis down to a still sophisticated level. Some of these constraints do not lower the applicability of the theoretical results to practical applications. For example, bounded support is a plausible assumption in practice, and a positive lower bound can easily be achieved approximately by an arbitrary tiny threshold $p_{min} > 0$. For the differentiability constraints there is strong evidence that the results can be extended to a much more general setting (for example, allowing for some discontinuities of p), at the price of introducing much more technical difficulties to the proofs and statements.

All results in this chapter that deal with the large sample limit of RGNGs hold with respect to the limit scenarios 2.3.1 and regularity conditions 2.3.2. We explicitly denote this setting as our **general assumptions**.

2.3.2 Density estimation from the vertex degrees in RGNGs

A function $\hat{p}$ is called a **density estimator** if it aims to approximate some probability density function p. Given n sample points drawn independently from p, a standard non-parametric approach to define a density estimator $\hat{p}$ is as follows (see Section 2.5.2 in Bishop (1995) for a broad introduction):

$$p(\mathbf{x}) \;\approx\; \hat{p}(\mathbf{x}) \;:=\; \frac{|B(\mathbf{x})|}{n \cdot \mathrm{vol}(B(\mathbf{x}))} \quad . \tag{2.2}$$

The measurable set $B(\mathbf{x}) \subseteq \mathbb{R}^d$ denotes a small local neighborhood around $\mathbf{x}$ of Lebesgue volume $\mathrm{vol}(B(\mathbf{x}))$, and $0 \leq |B(\mathbf{x})| \leq n$ denotes the number of sample points that fall into $B(\mathbf{x})$. For example, consider a histogram that partitions the sampleable space into a collection of finite-sized bins $B_1, B_2, B_3, \ldots$. For each $\mathbf{x} \in \mathbb{R}^d$ let $B(\mathbf{x})$ denote the bin that contains $\mathbf{x}$. Then the density estimator $\hat{p}(\mathbf{x})$ gives the fraction of all sample points that fall into bin $B(\mathbf{x})$, divided by the bin size $\mathrm{vol}(B(\mathbf{x}))$. This example illustrates further the fundamental **bias-variance trade-off**: the bin size must be as small as possible in order to localize the estimate at $\mathbf{x}$, but it must also be large enough to guarantee that each bin contains sufficiently many sample points. A larger sample size n allows for choosing smaller bins.

Depending on the precise definition of $B(\mathbf{x})$, one can derive different estimators from Equation (2.2). The classical histogram approach first fixes all bins $B(\mathbf{x})$ data-independently

and then considers the hit counts $|B(\mathbf{x})|$ as data-dependent random variables. The following alternative approach derives another density estimator from Equation 2.2 exactly the other way round. Lugosi and Nobel (1996) provide strong consistency results, precisely that $\hat{p} \xrightarrow{a.s.} p$ in L_1, under very general assumptions that particularly imply both of these approaches.

Density estimation from the distance to the k-nearest neighbor. The basic idea of this density estimator is to fix $|B(\mathbf{x})| := k$ in advance to a constant and then determine for each $\mathbf{x}$ the smallest possible ball $B(\mathbf{x})$ around $\mathbf{x}$ such that $B(\mathbf{x})$ contains exactly k sample points. This is equivalent to determining the radius of $B(\mathbf{x})$ as the distance from $\mathbf{x}$ to its k-nearest sample point, denoted as the kNN-radius $r_k(\mathbf{x})$. The volume of $B(\mathbf{x})$ can then be computed as $\mathrm{vol}(B(\mathbf{x})) = \eta_d \cdot r_k(\mathbf{x})^d$, where $\eta_d := \pi^{d/2}/(d/2)!$ denotes the volume of the d-dimensional unit ball. Thus, Equation 2.2 suggests to estimate $p(\mathbf{x})$ at $\mathbf{x} \in \mathbb{R}^d$ by computing $\hat{p}(\mathbf{x}) = r_k(\mathbf{x})^{-d} \cdot c_n$ with the constant scaling factor $c_n := k/(n\eta_d)$. Consistency of this specific instance of Equation (2.2) is already shown by Loftsgaarden and Quesenberry (1965). Note that c_n depends on n, k and d, but not on $\mathbf{x}$. In cases where c_n cannot be determined, we still get the result that $r_k(\mathbf{x})^{-d}$ is proportional to $p(\mathbf{x})$. However, the explicit knowledge of d is mandatory, since $r_k(\mathbf{x})^{-d}$ cannot be computed otherwise. This is a drawback in practical applications, where the dimension d may not be known.

We are now going to present three alternative density estimators, one for each studied RGNG. These density estimators enable us to estimate the density $p(\mathbf{x}_i)$ at the location of the sample point of vertex $i \in V$ from its degree d_i. Precisely, for each RGNG and sufficiently large n there exists a scaling factor c_n that provides $d_i \cdot c_n \xrightarrow{a.s.} p(\mathbf{x}_i)$. The factor c_n depends only on n, d and the parameters of the RGNG. In applications where c_n cannot be determined, it still holds that d_i is proportional to $p(\mathbf{x})$, without requiring any knowledge on $n, d, \mathbf{x}_i, p, \ldots$. This can be stated as follows:

> **“ The degree d_i is, up to a global scaling factor, a consistent estimate on the density $p(\mathbf{x}_i)$ at the sample point $\mathbf{x}_i$. ”**

In particular, this answers Question (Q1) from the introduction to this chapter: it is indeed possible to learn something about the sampling distribution from the vertex degrees of an RGNG. Note that Figure 1.3.1 (d) on page 26 already indicated this relationship between the degrees and the underlying density.

Density estimation from the r-graph. For the r-graph it holds that $d_i/(n \cdot \eta_d \cdot r_n^d) \xrightarrow{a.s.} p(\mathbf{x}_i)$. Note that $c_n := (n \cdot \eta_d \cdot r_n^d)^{-1}$ depends on n, d, and r_n, but not on the location of the sample point. This result can be proven directly by standard concentration inequalities, see von Luxburg et al. (2010). It can also be rendered as an instance of Equation 2.2, by defining $B(\mathbf{x}_i)$ as the ball of radius r_n around the sample point $\mathbf{x}_i$. By this, an r-graph can be seen as a data-dependent histogram. It defines a collection of overlapping bins of equal size, one placed at each sample point. Consistency of this estimator is also implied by Lugosi and Nobel (1996).

Density estimation from the Gaussian graph. For the Gaussian graph with edge weights as defined in (1.2) it holds that $d_i/n \xrightarrow{a.s.} p(\mathbf{x}_i)$, thus $c_n = 1/n$.

This can be seen by rendering this setting as an instance of multivariate kernel density estimation. In **multivariate kernel density estimation**, any non-negative integrable function $K : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ that provides unit total volume $\int K(\mathbf{x})\, d\mathbf{x} = 1$ and symmetry $K(\mathbf{x}) = K(-\mathbf{x})$ is denoted as a **kernel**. Typically studied kernels provide further properties, in particular they decrease monotonically with the distance to the origin. For example, consider the **uniform kernel** $K_1(\mathbf{x}) := \eta_d^{-1}[\|\mathbf{x}\| \leq 1]_0^1$, which is constant positive only on the unit ball, and zero everywhere outside. Another example is the **multivariate Gaussian kernel** $K_\Omega(\mathbf{x}) := (2\pi)^{-d/2} \cdot \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{x}\right)$, which concentrates its mass around the origin in a bell-shape. The **kernel density estimator** $\hat{p}(\mathbf{x})$ according to any kernel function K is defined as

$$\hat{p}(\mathbf{x}) \;:=\; \frac{1}{n}\sum_{i=1}^{n} K(\mathbf{x} - \mathbf{x}_i) \quad . \tag{2.3}$$

Illustratively, this estimator places n instances of the kernel K in space, one instance pinned at each sample point, and then averages over all their contributions to the evaluated location $\mathbf{x}$. If $\mathbf{x}$ lies in a high-density area, hence with many sample points close to it, then $\hat{p}(\mathbf{x})$ sums up to a much larger value than in low-density areas.

In order to approximate p with arbitrary precision, the kernel must further be scaled to smaller and smaller bandwidths along with $n \to \infty$. For example, the uniform kernel can be scaled by any scalar $r > 0$ to the ball of radius r, resulting in the **scaled uniform kernel** $K_r(\mathbf{x}) = (\eta_d r^d)^{-1}[\|\mathbf{x}\| \leq r]_0^1$. The multivariate Gaussian kernel can be scaled by any covariance matrix Σ, yielding the **scaled Gaussian kernel** $K_\Sigma(\mathbf{x}) := \det(\Sigma)^{-1/2} \cdot K_\Omega(\Sigma^{-1/2}\mathbf{x})$. In particular, if we choose Σ to represent independent dimensions of uniform variance, that is $\Sigma := \mathrm{diag}(\sigma^2, \ldots, \sigma^2)$, then K_Σ simplifies to

$$K_\sigma(\mathbf{x}) \;:=\; (2\pi\sigma^2)^{-d/2} \cdot \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma^2}\right) \quad , \tag{2.4}$$

the **Gaussian kernel of bandwidth** σ. Under mild assumptions on p and K, and if the bandwidth decreases at the right rate, then $\hat{p}$ is a consistent estimate on p. This is a classical result, see for example Stute (1984) and Devroye and Lugosi (2001).

Let us now see how the Gaussian graph is related to multivariate kernel density estimation: from Equations (2.3), (2.4) and (1.2) we get from $d_i = \sum_j w_{ij}$ with $w_{ii} := 0$ that

$$d_i \;=\; \sum_{j \neq i} K_\sigma(\mathbf{x}_j - \mathbf{x}_i) \;=\; n \cdot \hat{p}(\mathbf{x}_i) - K_\sigma(\mathbf{x}_i - \mathbf{x}_i) \quad .$$

Since $\alpha := K_\sigma(\mathbf{x}_i - \mathbf{x}_i) = (2\pi\sigma^2)^{-d/2}$ is constant, we get that $d_i/n = \hat{p}(\mathbf{x}_i) - \alpha/n$. The additive error α/n is asymptotically irrelevant and could even be eliminated by adding a selfloop of weight $w_{ii} := \alpha$ to every vertex of the Gaussian graph.

Density estimate from...	$\hat{p}(\mathbf{x})$	c_n
distance to k-nearest neighbor	$r_k(x)^{-d} \cdot c_n$	$k/(n\eta_d)$
r-graph	$d_i \cdot c_n$	$1/(n\eta_d r_n^d)$
Gaussian graph	$d_i \cdot c_n$	$1/n$
weighted kNN-graph	$d_i \cdot c_n$	$1/n$

Table 2.1: Overview of density estimators.. All four satisfy that $\hat{p}(\mathbf{x}) \xrightarrow{a.s.} p(\mathbf{x})$ in L_1.

The r-graph can similarly be rendered as an instance of multivariate kernel density estimation by using the uniform kernel in place of the Gaussian kernel.

Density estimation from the kNN-graph. For the weighted kNN-graph it holds that $c_n := 1/n$, just as for the Gaussian graph. This is because the limit scenario (Definition 2.3.1) implies that for large enough n it holds that $w_{ij} \approx 0$ whenever j is not among the k-nearest neighbors of i or vice versa. Precisely, the expected neighborhood radius $\mathbb{E}(r_k(i))$ of vertex i to its k-nearest neighbor can be bounded (Maier et al., 2009) for large enough n as follows: $\sqrt[d]{k_n/((n-1)\cdot\eta_d\cdot p_{max})} \leq \mathbb{E}(r_k(i)) \leq \sqrt[d]{k_n/((n-1)\cdot\eta_d\cdot p_{min})}$. Hence, because of $k_n/n \to 0$ all expected neighborhood radii $\mathbb{E}(r_k(i))$ decrease to 0, at the rate of $(k_n/n)^{1/d}$. Thus, although the number of neighbors k_n increases to infinity, k_n grows so slowly that the volumes of the corresponding k_n-neighborhoods shrink to zero. Now, the additional requirement $(k_n/n)^{1/d} \geq \sigma_n^\alpha$ for some $\alpha \in (0,1)$ comes into play: it implies that the bandwidth σ_n shrinks asymptotically even faster to zero than $(k_n/n)^{1/d}$, thus also faster than the expected neighborhood radii $\mathbb{E}(r_k(i))$. As a consequence, for sufficiently large n most edges in every k_n-neighborhood have a weight of almost zero because the bandwidth $\sigma_n \ll \mathbb{E}(r_k(i))$ puts significant weight only close to the center within each k_n-neighborhood, that is on edges to the very nearest $k_n' \ll k_n$ neighbors. For that reason, truncating all edges after the k_n-nearest neighbor removes only a negligible total weight. Thus, the weighted kNN-graph can be seen as an approximation of the Gaussian graph.

Note that the *unweighted* kNN-graph shows a totally different limit behavior than the weighted kNN-graph. In particular, the unweighted kNN-graph does *not* allow to estimate the density from its degrees because the degrees of all vertices concentrate at $\Theta(k_n)$ regardless of the underlying density. So the unweighted kNN-graph can best be described as an (almost) regular unweighted graph. Recent work by von Luxburg and Alamgir (2013) shows how to nevertheless extract a density estimate from an unweighted kNN-graph by a very different approach than from the degrees.

Table 2.1 summarizes the consistency results for the above introduced density estimators. Figure 2.3.1 visualizes them for 10000 sample points drawn from the two-dimensional density $p(x,y) := x + 0.5$ on the support $[0,1]^2$. The plots show that all approaches provide reasonable

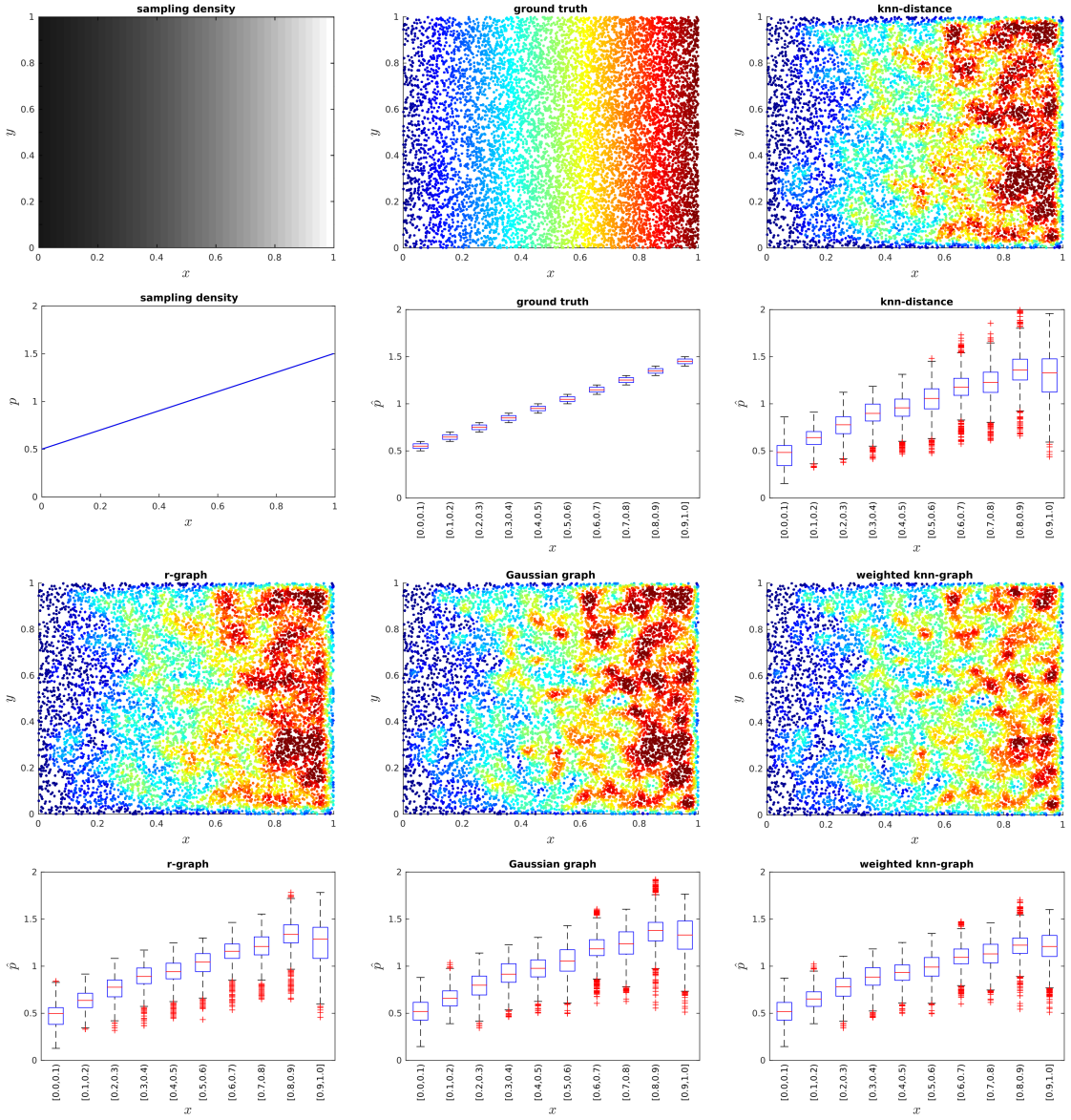


Figure 2.3.1: Proof of concept of vertex-degree-based density estimators. (top left) sampling density defined on $[0,1]^2$ by $p(x,y) = x + 0.5$, white indicates higher density. The line plot below shows the density function in x. (top middle) 10000 sample points drawn from p. The heat plot shows the ground truth at every sample point, that is $\hat{p}(\mathbf{x}_i) := p(\mathbf{x}_i)$. The box plot below shows summary statistics on the density estimates of all sample points whose x-coordinate lies in the respective interval (red line: median, box: quartile range, whiskers: up to 1.5 times the quartile range, red crosses: outliers). All boxes follow tight along the curve of $p(\mathbf{x})$. (top right) heat plot of $\hat{p}(\mathbf{x})$ on the same sample, $\hat{p}$ defined by the distance to the kNN-nearest neighbor for $k = 70$. The corresponding box plot shows that all boxes stay along the true curve of p, except close to the boundary. (bottom left) similar pair of heat plot and box plot for $\hat{p}$ defined by the Gaussian graph (49995000 edges, hidden) with $\sigma = 0.02$. (bottom middle) same for the r-graph (411464 edges) with $r = 0.05$. (bottom right) same for the weighted kNN-graph (369058 edges) with $k = 70$ and $\sigma = 0.02$.

good estimates on p. Only at those sample points $\mathbf{x}_i$ that lie close to the boundary of the support, $\hat{p}(\mathbf{x}_i)$ systematically underestimates the true density. The precise reason for this boundary effect is individual to each approach, but it always arises from the fact that the relevant local neighborhood around $\mathbf{x}_i$ overshoots the support of p, and thus contains "fewer points than it should". However, the total area that is affected by this boundary effect vanishes for $n \to \infty$.

In applications where $d, n, \mathbf{x}_i$ and all other parameters are unknown, the three degree-based estimators are still able to provide correct results except for an unknown global scaling factor on the $\hat{p}$-axis. Because of $d_i/d_j \xrightarrow{a.s.} p(\mathbf{x}_i)/p(\mathbf{x}_j)$ the degrees can still infer the relative proportions of the underlying density at the sample points. For example, if in a sufficiently large RGNG the degree of vertex i is twice as large as the degree of vertex j, then it is reasonable to assume that the density at $\mathbf{x}_i$ is twice as high as the density at $\mathbf{x}_j$, even if we we do not know any of $\mathbf{x}_i$, $\mathbf{x}_j$, n, d, or the parameters of the RGNG.

2.3.3 Three types of volumes and cuts in geometric graphs

The previous section shows how vertex degrees in RGNGs are related to point-wise estimates of the underlying density, answering Question (Q1). Question (Q2) asks whether in a similar way volumes and cuts in the graph can be related to continuous volumes and cuts of the density p in space. Before being able to answer this question in the subsequent section, we have to state the terms "volumes" and "cuts" more precisely.

Random geometric graphs are the combination of a discrete object (a weighted graph) and a continuous object (a probability distribution on $\mathbb{R}^d$). These two parts are glued together by the sampling mechanism: the vertices in the graph are identified with sample points drawn from the distribution, and pairwise distances in $\mathbb{R}^d$ determine edges and edge weights. This sketches already *three* different "views" on a geometric graph: a discrete view, and a continuous view, plus a third view that bridges between the two others. We now make this distinction precise because it is the fundamental key to understand how modifying edge weights of a geometric graph can be interpreted as implicitly modifying the underlying probability distribution. For each of the three views, we define individual notions of volumes and cut weights.

Discrete graph view. This view sees a geometric graph just as an ordinary weighted graph $G = (V, E, W)$, ignoring any information on the sample points and the underlying distribution. Volumes and cuts are defined as usual for weighted graphs (as in Section 1.3.1): the volume of any subset of vertices $U \subseteq V$ is defined as $\mathrm{vol}_G(U) := \mathrm{vol}(U) = \sum_{i \in U} d_i$, and the cut weight as $\mathrm{cut}_G(U) := \mathrm{cut}(U) = \sum_{i \in U, j \in \overline{U}} w_{ij}$. No other information such as the coordinate embeddings $\mathbf{x}_i$, or the density p, or the dimension d is given. These additional parameters can only be accessed in synthetic examples or theoretical analysis. However, depending on the type of the underlying graph model, it is still the case that the accessible quantities in the graph view, such as degrees, volumes, shortest paths, commute times, etc., reflect certain properties that can be interpreted in a meaningful way.

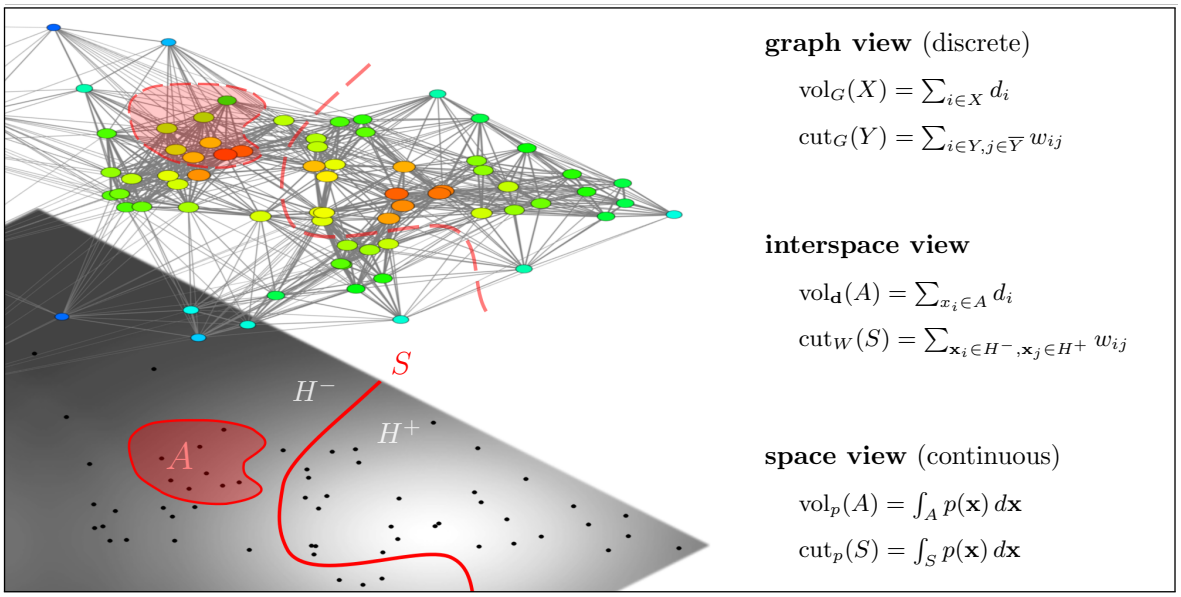


Figure 2.3.2: Graph view, space view, and interspace view of an RGNG. The space view refers to the gray-shaded plane at the bottom, where brighter color indicates higher probability density. In space view, a volume is assigned to the subset A, and a cut weight to the cutting surface S (here, more general than a hyperplane). The interspace view has access to the sample points (black dots in the plane). The neighborhood graph is built on top of this sample. The graph view refers only to the weighted graph itself, for arbitrary $X, Y \subseteq V$. It has no explicit knowledge on the sample points or the probability density. Nevertheless, its structure and edge weights contain implicit information. The set A gets a second notion of volume in the interspace, similarly the cut S another notion of cut weight. Section 2.3.4 shows how the interspace quantities are related to the space quantities in the large sample limit.

Continuous space view. This view sees only the probability distribution on $\mathbb{R}^d$, ignoring any information on the sample points and the graph structure. For the analysis, we assume that this probability distribution has a probability density $p : \mathbb{R}^d \to \mathbb{R}$ that further satisfies the regularity conditions (Definition 2.3.2). The notion of volumes and cut weights in the continuous space view is with respect to p. The **p-volume** of a measurable set $A \subseteq \mathbb{R}^d$ is defined as $\mathrm{vol}_p(A) := \int_A p(\mathbf{x})\, d\mathbf{x}$, that is the natural probability measure implied by p on $\mathbb{R}^d$. Cuts in continuous space require some concept of $(d-1)$-dimensional cutting surfaces that partition $\mathbb{R}^d$ into two measurable parts. As in the work by Maier et al. (2009), we restrict the mathematical analysis to a hyperplanes S, interpreted as a cut of the underlying space into two half-spaces denoted by H^+ and H^- (both including S itself). We define the corresponding **p-weight** by integrating p over the cutting surface, that is by $\mathrm{cut}_p(S) := \int_S p(\mathbf{x})\, d\mathbf{x}$.

Note that there is strong evidence that all following results can be generalized to more general cut surfaces, as long as they are sufficiently regular. However, this would introduce a lot of additional technical details to the proofs, without adding something new to the intuition.

Interspace view. This view "mediates" between the discrete graph view and the continuous space view by taking the sample point coordinates into account. The interspace view exploits the fact that vertex i is identified with a sample point located at $\mathbf{x}_i \in \mathbb{R}^d$. This enables us to evaluate index sets of the form $V(A) := \{i \in V \mid \mathbf{x}_i \in A\}$ where A is any subset of $\mathbb{R}^d$. These objects are not accessible to algorithms on graphs, but they serve as a theoretical construction to relate graph cuts and volumes to their continuous pendants. This is achieved in terms of a third notion of cuts and volumes: for any vector $\mathbf{d}$ and any sample drawn from p, we refer to $\mathrm{vol}_{\mathbf{d}}(A) := \mathrm{vol}_G(V(A)) = \sum_{\mathbf{x}_i \in A} d_i$ as the **interspace volume** of A with respect to vector $\mathbf{d}$. Intuitively, it sums over the degrees as provided in the discrete graph view, while the sum is indexed by the sample points as provided by the continuous space view. Moreover, let $\mathrm{cut}_W(S) := \mathrm{cut}_G(V(H^-), V(H^+)) = \sum_{\mathbf{x}_i \in H^-, \mathbf{x}_j \in H^+} w_{ij}$ denote the **interspace cut weight** of the cut surface S with respect to weight matrix W. Again, it sums over the edge weights as in the graph view, while the summands are indicated by the sample points in the space view.

Summarized, the *interspace view* connects the *graph view* to the *space view* by summing up discrete quantities from the graph view as indicated by continuous sets in space view. Figure 2.3.2 visualizes all three views.

2.3.4 The challenge: estimating volumes and cut weights of a density

We know from Section 2.3.2 that the vertex degree d_i in a random geometric graph can provide an estimate on the density p at the location of the sample point $\mathbf{x}_i$. Further, Section 2.3.3 shows how the notion of volumes and cuts in the graph can be connected to volumes and cuts in the underlying space — namely, with the help of studying volumes and cuts in the interspace view. How are the interspace quantities related to the space quantities in the large sample limit? One is easily tempted by the analogy to the density estimation to expect that the interspace volume $\mathrm{vol}_{\mathbf{d}}(A)$ should converge (up to a scaling factor) to the p-volume $\mathrm{vol}_p(A)$, and similarly the interspace cut weight $\mathrm{cut}_W(H)$ to the p-cut weight $\mathrm{cut}_p(H)$. However, it turns out that this is *not* true. Both interspace quantities converge to $\mathrm{vol}_{p^2}(A)$ and $\mathrm{cut}_{p^2}(H)$, respectively, that is to the continuous quantities of the *squared* density p^2, *not* of p itself! We refer to this fact as an **anomaly**.

This anomaly is already observed by Maier et al. (2009), but not investigated any further. The following theorem is a summary of their results on volumes and cut weights for the Gaussian graph (their Corollary 12 and Corollary 8), the r-graph (their Corollary 11 and Corollary 7) and the weighted kNN-graph (their Corollary 6 and Corollary 3).

Theorem 2.3.3 (Convergence limits of interspace quantities)
Let $p : \mathbb{R}^d \to \mathbb{R}$ denote a probability density that satisfies the regularity conditions 2.3.2. Consider a random sequence of matrices (W_n), where $W_n \in \mathbb{W}$ is the graph matrix of an RGNG that is built on n sample points drawn from p, with parameters chosen according to one of the limit scenarios 2.3.1. Let $\mathbf{d}_n$ denote the degree vector of W_n. Fix a hyperplane S in $\mathbb{R}^d$ and refer by H to either one of the two halfspaces induced by S. Then the following L_1-convergences hold true:

$$c_n \cdot \mathrm{vol}_{\mathbf{d}_n}(H) \xrightarrow{a.s.} \mathrm{vol}_{p^2}(H) \qquad \text{and} \qquad c'_n \cdot \mathrm{cut}_{W_n}(S) \xrightarrow{a.s.} \mathrm{cut}_{p^2}(S) \quad ,$$

where (c_n) and (c'_n) are sequences of scaling constants that are determined by n, d and the parameters of the RGNG.

See Maier et al. (2009) for the corresponding proofs. The remainder of this section provides a demystification of the anomaly. An intuitive reasoning is almost obvious for the interspace volumes, defined as

$$\mathrm{vol}_{\mathbf{d}}(A) \;=\; \sum_{\mathbf{x}_i \in A} d_i \qquad (\star)$$

for measurable $A \subseteq \mathbb{R}^d$. Think of the vertex degrees as a point-wise weight distribution on the underlying space: the positions of the sample points $\mathbf{x}_i$ are distributed according to density p, and each sample point is additionally weighted by d_i, which itself is known to serve as a density estimate on $p(\mathbf{x}_i)$. This perspective on $(\star)$ gives that the sum combines two different effects, each rising one factor of p. Precisely:

(i) a first factor p comes from the sampling mechanism, which distributes the sample points according to p, thus the number of sample points that hit the set A is proportional to $p(A)$. As a consequence, the number of summands in $(\star)$ is proportional to p.

(ii) a second factor p comes from the fact that we sum over the the weighted degrees d_i, each of which serves as a density estimate on $p(\mathbf{x}_i)$. As a consequence, each summand's value itself is proportional to p.

Thus, strongly simplified, the sum $(\star)$ behaves like the sum " $\sum_{i=1,\dots,P} P = P^2$ ".

A similar intuition can be derived for the interspace cut weights

$$\mathrm{cut}_W(S) \;=\; \sum_{\mathbf{x}_i \in H^-, \mathbf{x}_j \in H^+} w_{ij} \qquad (\star\star)$$

for a cutting surface S that cuts the space in two parts H^- and H^+. This sum cumulates all edge weights of edges that cross the surface S. Let us for now focus on the r-graph: all edge weights are $w_{ij} = 1$, so we have to argue why the number of summands (= number of cut

edges) behaves like p^2. In order to see this we take a closer look at the geometry: only those sample points $\mathbf{x}_i$ whose neighborhood ball $B(\mathbf{x}_i, r_n)$ of radius r_n intersects with S are able to contribute a positive number of cut edges to the sum. For large enough n, all contributing sample points lie close to the cut surface S within the thin stripe of distance r_n to S. For $H \in \{H^-, H^+\}$, a sample point $\mathbf{x}_i \in H$ is connected to all sample points in its neighborhood ball, particularly to those that lie in the "cap" $C_i := B(\mathbf{x}_i, r) \cap \overline{H}$. Each edge between $\mathbf{x}_i$ and some $\mathbf{y}_i \in C_i$ contributes a single cut edge to the sum ($\star\star$). In expectation there lie exactly $n \cdot \mathrm{vol}_p(C_i)$ many sample points in C_i, which is again proportional to p.

This identifies again two effects, this time both arise from the sampling mechanism:

(i) a first factor p comes from the fact that the number of intersecting neighborhood balls is proportional to the distribution p: in an area of t-times higher density, the cut surface S is intersected by t-times more neighborhood balls.

(ii) a second factor p comes from the fact the for each single intersecting neighborhood ball, the number of cut edges it produces is again proportional to the distribution p: the center sample point $\mathbf{x}_i$ is connected to all sample points that lie in C_i, whose number is proportional to p.

Thus, strongly simplified, the sum ($\star\star$) behaves like the sum $\sum_{i,j=1,\dots,P} 1 = P^2$.

This intuition for the interspace cut weights generalizes to Gaussian graphs and weighted kNN-graphs by focusing only on edges of significant weight. Depending on σ, there is some radius r_σ that plays the same role as r_n for the r-graph: a sample point is connected by an edge of significant weight, say $w_{ij} \geq 0.1$, to all points in distance at most r_σ, while the edge weight to sample points farer away than r_σ drops exponentially fast to 0. That is, for large enough n, one may think of the Gaussian graph and of the weighted kNN-graph, as being similar to the r-graph.

Note that, despite this plausible intuition, carrying out the formal proof details in a mathematically sound way is technically sophisticated. In particular when further dealing with effects near to the boundary of the support $\mathcal{X}$ of p. We do not dive into technical details here. The interested reader is referred to the work by Maier et al. (2009).

This anomaly implies the following notable consequence:

> **“ If applied to RGNGs, all graph algorithms that are based on volumes and cut weights are misguided by implicitly referring to the squared density p^2 rather than the density p itself. ”**

A main contribution of this chapter is a strategy to *resolve* this anomaly in order to be able to study volumes and cuts of p without squaring, and even further, a strategy to study other manipulations of p. Section 2.6.1 presents an example of this anomaly, and further shows how this anomaly can be resolved by using the technique of **f**-adjusting.

2.3.5 Negative selfloops

It turned out during my work on the algebraic properties of $\mathbf{f}$-adjusting that most statements and proofs simplify if the graph model is extended by a tiny yet powerful detail: we shall allow graphs to have negative edge weights — under two strict conditions:

- negative weights are only allowed at selfloops (that is along the main diagonal of W)
- negative weights are only allowed as long as degrees remain positive (that is all row sums stay positive)

Recall that $\mathbb{W} = \{X \in \mathbb{R}^{n\times n}_{\geq 0} \mid X = X^T, X\mathbf{1} > 0\}$ refers to the set of all graph matrices that represent a weighted undirected graph of positive degrees (that is, in which every vertex is incident to at least one edge, even if this is just a selfloop). In this chapter, we also consider the generalization of $\mathbb{W}$ to the set of **weak graph matrices**, defined as

$$\mathbb{W}_\ominus := \{X \in \mathbb{R}^{n\times n}_{\geq 0} + \operatorname{diag}(\mathbb{R}^n) \mid X = X^T, X\mathbf{1} > 0\}. \tag{2.5}$$

Its elements further allow for negative diagonal entries as long as all row sums remain positive. That is, $\mathbb{W}_\ominus$ captures all graphs that can be derived from a graph in $\mathbb{W}$ by adding negative selfloops under the above restrictions. The subscript $\ominus$ should evoke the association of a "negative selfloop". Obviously $\mathbb{W} \subset \mathbb{W}_\ominus$.

Section 2.5.1 studies properties of $\mathbb{W}_\ominus$ and $\mathbb{W}$ that are required for later proof details but that are also interesting on its own.

2.3.6 Graph modifications

By the term **graph modification** we refer to any strategy that takes a weighted graph $G = (V, E, W) = \mathcal{G}(W)$ as its input and produces another weighted graph on the same vertices $\hat{G} = (V, \hat{E}, \hat{W}) = \mathcal{G}(\hat{W})$ as its output. Further, $\hat{G}$ should stay "similar" in some sense to G. This can be made precise in different ways. Here, we focus on the constraint that the edge sets $\hat{E}$ and E coincide up to selfloops, while the new edge weights $\hat{W}$ may differ arbitrarily from W. A graph modification can be seen as a function $m : \mathbb{W}_\ominus \to \mathbb{W}_\ominus$ that maps a given matrix W to another matrix $\hat{W} := m(W)$. Depending on the graph modification, the domain and/or the image of m can either be $\mathbb{W}$ or $\mathbb{W}_\ominus$.

In this section we define the following three graph modifications, each one parameterized by an arbitrary positive vector $\mathbf{f}$:

- $\mathbf{f}$-selflooping: $\mathbb{W}_\ominus \to \mathbb{W}_\ominus,\ W \mapsto W^\circ_\mathbf{f}$
- $\mathbf{f}$-scaling: $\mathbb{W} \to \mathbb{W},\ W \mapsto \tilde{W}_\mathbf{f}$
- $\mathbf{f}$-adjusting: $\mathbb{W} \to \mathbb{W}_\ominus,\ W \mapsto \overline{W}_\mathbf{f}$

All three graph modifications are identified by one of the matrix decorations $\tilde{}$, $^\circ$ and $\bar{}$, which is considered as an operator on the matrix W with one additional parameter given as index $\mathbf{f}$. For example, for any matrix $A \in \mathbb{W}$ and any vector $\mathbf{x} \in \mathbb{R}^n_{>0}$ we refer by $\tilde{A}_\mathbf{x}$ to the $\mathbf{f}$-scaled graph matrix for $\mathbf{f} = \mathbf{x}$, also denoted as the $\mathbf{x}$-scaling of A.

Before providing the definitions for these graph modifications let us first clarify their intention: the overall goal of $\mathbf{f}$-adjusting is to modify volumes and cut weights simultaneously in a way that allows to relate the modified interspace quantities to new continuous quantities. Having that, studying volumes and cut weights of a graph that is modified by $\mathbf{f}$-adjusting corresponds to studying volumes and cut weights of a different underlying density.

Since volumes in the interspace view are affected by the vertex degrees, a first goal is to re-weight edges such that the output graph can attain any other degree vector. The parameter $\mathbf{f}$ drives all three graph modifications toward exactly this goal: each graph modification aims at providing the vector $\mathbf{f}$ as the degree vector of its output. Without any further constraints, this is a trivial task that can easily be achieved by modifying just selfloops in an appropriate way. However, the second goal is to find a re-weighting scheme that simultaneously modifies all cut weights in precisely "the right way". So we have to re-weight all edges, not just selfloops.

A main contribution of this chapter is that $\mathbf{f}$-adjusting fits volumes *and* cut weights simultaneously to the new degree vector $\mathbf{f}$ in a way that allows for a geometric interpretation: both quantities represent one and the same new underlying density. Moreover, the algebraic properties reveal that $\mathbf{f}$-adjusting is a very natural graph modification.

f-selflooping

The naive strategy to adjust the vertex degrees of a graph $\mathcal{G}(W)$ to any prescribed vector $\mathbf{f} \in \mathbb{R}^n_{>0}$ is by modifying selfloops: simply redefine the main diagonal of W such that the new row sums equal $\mathbf{f}$, leaving all off-diagonal entries unchanged. This is made precise in the following definition.

Definition 2.3.4 (f-selflooping)
For any $\mathbf{f} \in \mathbb{R}^n_{>0}$ and $W \in \mathbb{W}_\ominus$, the $\mathbf{f}$-selflooped graph matrix $W^\circ_\mathbf{f}$ is defined as:

$$W^\circ_\mathbf{f} \;:=\; W - D + F \quad \in\; \mathbb{W}_\ominus \quad ,$$

which gives for $W^\circ_\mathbf{f} = [w^\circ_{ij}]$ element-wise:

$$w^\circ_{ij} \;:=\; \begin{cases} w_{ii} - d_i + f_i & \text{for } i = j \\ w_{ij} & \text{for } i \neq j \end{cases} \quad .$$

See Figure 2.3.3 for an example. We refer to $\mathcal{G}(W^\circ_\mathbf{f})$ as the $\mathbf{f}$-selflooped graph. The circle annotation $^\circ$ should evoke the association of a "loop". This graph modification yields the

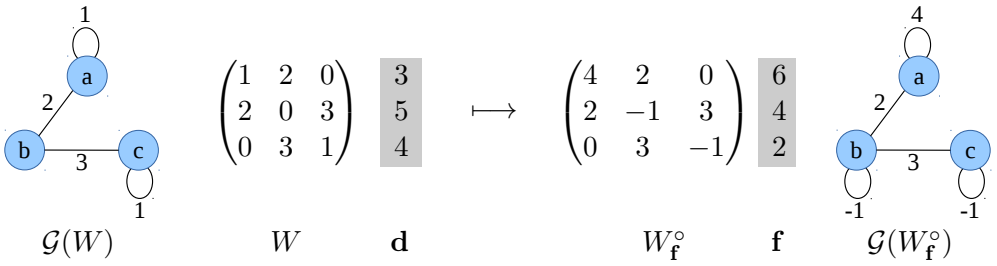


Figure 2.3.3: Example on **f**-selflooping. **f**-selflooping to $\mathbf{f} = (6, 4, 2)^T$ of the graph $\mathcal{G}(W)$. The **f**-selflooped graph $\mathcal{G}(W^\circ_{\mathbf{f}})$ has a negative selfloop at vertex b and c. This can be avoided by $c\mathbf{f}$-selflooping for $c = \max_{i\in V}\{(d_i - w_{ii})/f_i\} = 3/2$, which gives $W^\circ_{c\mathbf{f}} = [7\ 2\ 0; 2\ 1\ 3; 0\ 3\ 0]$.

intended positive vertex degrees $\mathbf{f}$ exactly, because $W^\circ_{\mathbf{f}}\mathbf{1} = (W - D + F)\mathbf{1} = \mathbf{d} - \mathbf{d} + \mathbf{f} = \mathbf{f}$. Note that $W = W^\circ_{\mathbf{d}}$.

The set $\mathbb{W}_\ominus$ is closed under **f**-selflooping, that is $W \in \mathbb{W}_\ominus \Rightarrow \tilde{W}^\circ_{\mathbf{f}} \in \mathbb{W}_\ominus$. However, the set $\mathbb{W}$ is *not* closed under **f**-selflooping: for example, $W = \left(\begin{smallmatrix}0&2\\2&0\end{smallmatrix}\right) \in \mathbb{W}$ gives that $W^\circ_{\mathbf{1}} = \left(\begin{smallmatrix}-1&2\\2&-1\end{smallmatrix}\right) \notin \mathbb{W}$. It is important to note that **f**-selflooping can force some entries on the main diagonal to take negative values, although the corresponding original entries are non-negative. It is possible to circumvent this problem of introducing negative selfloops if we accept to attain the intended degree vector $\mathbf{f}$ not exactly, but only up to a scalar multiple: for any choice of $\mathbf{f}$, there exists a sufficiently large $c > 0$ such that $c\mathbf{f}$-selflooping (that is **f**-selflooping with parameter $c\cdot\mathbf{f}$) will not cause any negative selfloop. Precisely, Proposition 2.5.3 shows that the main diagonal of $W^\circ_{c\mathbf{f}}$ is non-negative if and only if $c \geq \max_{i\in V}\{(d_i - w_{ii})/f_i\} =: c^+_{W,\mathbf{f}}$. The consequence of this avoidance strategy is that the $c\mathbf{f}$-selflooped graph has degree vector $c\mathbf{f}$ instead of $\mathbf{f}$.

Since **f**-selflooping keeps all non-selfloop edges unmodified, this graph modification can only affect volumes but not any cut weights, that is $\mathcal{G}(W)$ and $\mathcal{G}(W^\circ_{\mathbf{f}})$ always show exactly the same cut weights for all cuts. This is a drawback of **f**-selflooping for our purpose.

f-scaling

A simple strategy to adapt the row sums of a matrix $W \in \mathbb{W}$ to any prescribed vector $\mathbf{f} \in \mathbb{R}^n_{>0}$ is by proportional scaling, that is by considering $FD^{-1}W$. This matrix provides row sums $\mathbf{f}$ exactly, but it is no longer symmetric. **f**-scaling is a (multiplicative) symmetrization of the proportional scaling approach, defined as follows.

Definition 2.3.5 (f-scaling)
For any $\mathbf{f} \in \mathbb{R}^n_{>0}$ and $W \in \mathbb{W}$, the $\mathbf{f}$-scaled graph matrix $\tilde{W}_\mathbf{f}$ is defined as:

$$\tilde{W}_\mathbf{f} \;:=\; \sqrt{FD^{-1}}\,W\,\sqrt{FD^{-1}} \quad \in\; \mathbb{W} \quad ,$$

which gives for $\tilde{W}_\mathbf{f} = [\tilde{w}_{ij}]$ element-wise:

$$\tilde{w}_{ij} \;:=\; w_{ij} \cdot \sqrt{\frac{f_i\, f_j}{d_i\, d_j}} \quad .$$

Further, let $\tilde{\mathbf{d}} = (\tilde{d}_i)$ refer to the degree vector of $\tilde{W}_\mathbf{f}$, and let $\tilde{D}_\mathbf{f} := \mathrm{diag}(\tilde{\mathbf{d}})$.

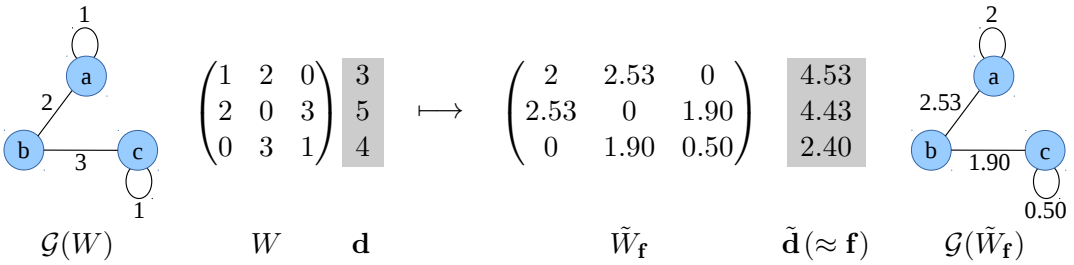


Figure 2.3.4: Example on $\mathbf{f}$-scaling. $\mathbf{f}$-scaling to $\mathbf{f} = (6, 4, 2)^T$ of the graph $\mathcal{G}(W)$. The $\mathbf{f}$-scaled graph $\mathcal{G}(\tilde{W}_\mathbf{f})$ has degree vector $\tilde{\mathbf{d}}$, which matches the intended degree vector $\mathbf{f}$ only approximately.

See Figure 2.3.4 for an example. We refer to $\mathcal{G}(\tilde{W}_\mathbf{f})$ as the $\mathbf{f}$-scaled graph. Note that $W = \tilde{W}_\mathbf{d}$.

The set $\mathbb{W}$ is closed under $\mathbf{f}$-scaling: $W \in \mathbb{W} \Rightarrow \tilde{W}_\mathbf{f} \in \mathbb{W}$. However, the set $\mathbb{W}_\ominus$ is *not* closed under $\mathbf{f}$-scaling, since for example $W = \left(\begin{smallmatrix} -5 & 6 \\ 6 & 3 \end{smallmatrix}\right) \in \mathbb{W}_\ominus$ would give that $\tilde{W}_\mathbf{1} = \left(\begin{smallmatrix} -5 & 2 \\ 2 & 1/3 \end{smallmatrix}\right) \notin \mathbb{W}_\ominus$. For that reason, we consider $\mathbf{f}$-scaling only applied to non-weak graph matrices.

The example in Figure 2.3.4 shows further that $\tilde{\mathbf{d}} \neq \mathbf{f}$ in general. $\mathbf{f}$-scaling has the drawback that it does *not* provide the intended degree vector $\mathbf{f}$ exactly, which is a consequence of the symmetrization of the proportional scaling approach. However, one can see that $\mathbf{f}$-scaling focuses on reaching the degree vector $\mathbf{f}$ approximately if we assume that $f_i/d_i \approx f_j/d_j$ whenever $ij \in E$. In this case it holds that $\tilde{d}_i = \sum_{j \in V} w_{ij}\sqrt{(f_i f_j)/(d_i d_j)} \approx \sum_{j \in V} w_{ij}(f_i/d_i) = f_i$. This assumption is particularly satisfied if $f_i \approx f_j$ and $d_i \approx d_j$ whenever $ij \in E$. The tilde annotation $\tilde{}$ of $\mathbf{f}$-scaling hints at this approximation $\tilde{\mathbf{d}} \approx \mathbf{f}$.

$\mathbf{f}$-scaling affects all entries in the matrix, so the graphs $\mathcal{G}(W)$ and $\mathcal{G}(\tilde{W}_\mathbf{f})$ differ in terms of volumes and cut weights.

f-adjusting

Both above graph modifications have their own drawback: $\mathbf{f}$-scaling does not provide the intended new degree vector $\mathbf{f}$ exactly, and $\mathbf{f}$-selflooping does not affect any cut weights. However, these drawbacks mutually compensate if both methods are combined. $\mathbf{f}$-adjusting is defined as the concatenation of first applying $\mathbf{f}$-scaling, followed by applying $\mathbf{f}$-selflooping, that is $\overline{W}_{\mathbf{f}} := (\tilde{W}_{\mathbf{f}})^{\circ}_{\mathbf{f}}$. Slightly more general, $(\mathbf{f}, c)$-adjusting is defined by first applying $\mathbf{f}$-scaling, followed by $c\mathbf{f}$-selflooping, that is $\overline{W}_{\mathbf{f},c} := (\tilde{W}_{\mathbf{f}})^{\circ}_{c\mathbf{f}}$. That is, $\mathbf{f}$-adjusting equals $(\mathbf{f}, 1)$-adjusting.

The output graph of $(\mathbf{f}, c)$-adjusting can have negative selfloops. This can be avoided by choosing a sufficiently large c.

Definition 2.3.6 ($\mathbf{f}$-adjusting and $(\mathbf{f}, c)$-adjusting)
For any $\mathbf{f} \in \mathbb{R}^n_{>0}$, $c > 0$, and $W \in \mathbb{W}$, the $(\mathbf{f}, c)$-adjusted graph matrix $\overline{W}_{\mathbf{f},c}$ is defined as the concatenation of $\mathbf{f}$-scaling followed by $c\mathbf{f}$-selflooping, that is $\overline{W}_{\mathbf{f},c} := (\tilde{W}_{\mathbf{f}})^{\circ}_{c\mathbf{f}}$. Explicitly we get that

$$\overline{W}_{\mathbf{f},c} \;:=\; \tilde{W}_{\mathbf{f}} - \tilde{D}_{\mathbf{f}} + cF \quad \in \; \mathbb{W}_{\ominus} \quad ,$$

which gives for $\overline{W}_{\mathbf{f},c} = [\overline{w}_{ij}]$ element-wise:

$$\overline{w}_{ij} \;:=\; \begin{cases} \tilde{w}_{ii} - \tilde{d}_i + cf_i \;=\; cf_i + \sum_{k \neq i} w_{ik} \cdot \sqrt{\frac{f_i f_k}{d_i d_k}} & \text{for } i = j \\ \tilde{w}_{ij} \;=\; w_{ij} \cdot \sqrt{\frac{f_i f_j}{d_i d_j}} & \text{for } i \neq j \end{cases} .$$

$\mathbf{f}$-adjusting is defined as $(\mathbf{f}, c)$-adjusting for $c = 1$, and we write $\overline{W}_{\mathbf{f}} := \overline{W}_{\mathbf{f},1}$.

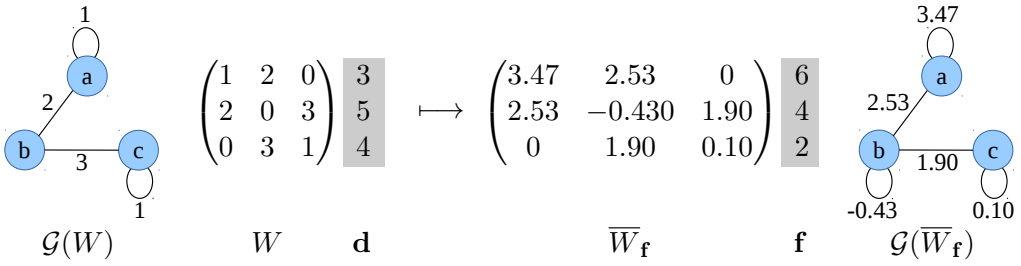


Figure 2.3.5: Example on $\mathbf{f}$-adjusting. $\mathbf{f}$-adjusting to $\mathbf{f} = (6, 4, 2)^T$ of the graph $\mathcal{G}(W)$. The $\mathbf{f}$-adjusted graph $\mathcal{G}(\overline{W}_{\mathbf{f}})$ has a negative selfloop at vertex b. This can be avoided by $(\mathbf{f}, c)$-adjusting for $c := \max_{i \in V}\{(\tilde{d}_i - \tilde{w}_{ii})/f_i\} \approx 1.99$.

See Figure 2.3.5 for an example. We refer to $\mathcal{G}(\overline{W}_{\mathbf{f}})$ as the $\mathbf{f}$-adjusted graph, and to $\mathcal{G}(\overline{W}_{\mathbf{f},c})$ as the $(\mathbf{f}, c)$-adjusted graph. Both can produce negative selfloops, but vertex degrees stay always positive because $\overline{W}_{\mathbf{f},c}\mathbf{1} = \tilde{\mathbf{d}} - \tilde{\mathbf{d}} + c\mathbf{f} = c\mathbf{f} > 0$. In particular, $(\mathbf{f}, c)$-adjusting attains the

vertex degrees $c\mathbf{f}$ exactly: first, just approximately by $\mathbf{f}$-scaling, but then exactly after adding the residuals to the selfloops by $c\mathbf{f}$-selflooping. This further implies that the $(\mathbf{f}, c)$-adjusted graph has no negative selfloop if and only if $c \geq \max_{i \in V}\{(\tilde{d}_i - \tilde{w}_{ii})/f_i\} = c^+_{\tilde{W}_\mathbf{f},\mathbf{f}}$. If $\tilde{W}_\mathbf{f}$ is clear from the context, we refer to $c^+_{\tilde{W}_\mathbf{f},\mathbf{f}}$ simply by c^+. Finally, note that $W = \overline{W}_\mathbf{d}$.

2.3.7 The $\mathbf{f}$-adjusted Laplacian

Recall the definition of the normalized Laplacian as $\mathcal{L}(W) := I - D^{-1/2}WD^{-1/2}$ for any graph matrix $W \in \mathbb{W}$. The same definition can be applied to weak graph matrices $W \in \mathbb{W}_\ominus$ and still give a valid mathematical expression. However, this does not imply that all properties of the normalized Laplacian (such as positive semi-definiteness) automatically generalize from $W \in \mathbb{W}$ to $W \in \mathbb{W}_\ominus$. Section 2.5.1 studies this generalization of the normalized Laplacian $\mathcal{L}$ to weak graph matrices in detail, and shows that the interesting properties indeed generalize. The following lemma can also be seen as a corollary of this generalization, although a direct proof is given here. It shows that all $(\mathbf{f}, c)$-adjusted graphs for all $c > 0$ are very similar to each other in terms of their normalized Laplacian matrix.

Lemma 2.3.7 (Scaling relation)
For all $W \in \mathbb{W}$, $\mathbf{f} \in \mathbb{R}^n_{>0}$ and $c > 0$ it holds that

$$\mathcal{L}(\overline{W}_\mathbf{f}) = c \cdot \mathcal{L}(\overline{W}_{\mathbf{f},c}).$$

Proof. Plugging in all definitions gives that:

$$\begin{aligned} c \cdot \mathcal{L}(\overline{W}_{\mathbf{f},c}) &= c \cdot \sqrt{(cF)^{-1}}(cF - (\tilde{W}_\mathbf{f} - \tilde{D}_\mathbf{f} + cF))\sqrt{(cF)^{-1}} \\ &= \sqrt{F^{-1}}(F - (\tilde{W}_\mathbf{f} - \tilde{D}_\mathbf{f} + F))\sqrt{F^{-1}} \\ &= \mathcal{L}(\overline{W}_{\mathbf{f},1}) \end{aligned}$$

□

Consequently, every $\mathcal{L}(\overline{W}_{\mathbf{f},c})$ has the same eigenvectors as $\mathcal{L}(\overline{W}_\mathbf{f})$, with all eigenvalues scaled by c^{-1}. In particular the order of the eigenvalues is preserved. Further, the normalized Laplacian of the generally weak $\mathbf{f}$-adjusted graph matrix $\overline{W}_\mathbf{f} \in \mathbb{W}_\ominus$ equals for any $c \geq c^+$ the normalized Laplacian of a non-weak graph matrix $\overline{W}_{\mathbf{f},c} \in \mathbb{W}$ up to a scaling factor. This implies further that every $\mathcal{L}(\overline{W}_{\mathbf{f},c})$ inherits many spectral properties from the standard normalized Laplacian $\mathcal{L}(\overline{W}_{\mathbf{f},c^+})$ of a non-weak graph such as its positive semi-definiteness and that $\sqrt{\mathbf{f}}$ is an eigenvector to eigenvalue 0 whose multiplicity matches the number of connected components in $\mathcal{G}(W)$.

As a consequence, the spectral analysis of any $(\mathbf{f}, c)$-adjusted graph can be reduced to the $\mathbf{f}$-adjusted graph. In particular, it does not matter whether or not the graph has negative selfloops. We can simply choose the $\mathbf{f}$-adjusted graph as a representative for all $(\mathbf{f}, c)$-adjusted graphs because their normalized Laplacians are the same up to scaling by c.

We refer by the term **f-adjusted Laplacian** $\mathcal{L}_{\mathbf{f}}(W)$ to the normalized Laplacian of the $\mathbf{f}$-adjusted graph $\mathcal{L}(\overline{W}_{\mathbf{f}})$.

Definition 2.3.8 (f-adjusted Laplacian)
For all $W \in \mathbb{W}$ and $\mathbf{f} \in \mathbb{R}^n_{>0}$, the $\mathbf{f}$-adjusted Laplacian of W is defined as:

$$\mathcal{L}_{\mathbf{f}}(W) \;:=\; \mathcal{L}(\overline{W}_{\mathbf{f}}) \;\in\; \mathcal{L}(\mathbb{W}_{\ominus}) \quad ,$$

that is the normalized Laplacian of the $\mathbf{f}$-adjusted graph matrix $\overline{W}_{\mathbf{f}} \in \mathbb{W}_{\ominus}$.

The $\mathbf{f}$-adjusted Laplacian shows another interesting property that is crucial for the algebraic interpretation of $\mathbf{f}$-adjusting. The following lemma states a fundamental connection between every $\mathbf{f}$-adjusted Laplacian $\mathcal{L}_{\mathbf{f}}(W)$ and the normalized Laplacian $\mathcal{L}(W) = I - \sqrt{D^{-1}}W\sqrt{D^{-1}}$.

Lemma 2.3.9 (f-adjusted Laplacian versus normalized Laplacian)
For all $W \in \mathbb{W}$ and $\mathbf{f} \in \mathbb{R}^n_{>0}$ it holds that

$$\mathcal{L}_{\mathbf{f}}(W) \;=\; \tilde{D}_{\mathbf{f}} F^{-1} - \sqrt{D^{-1}}W\sqrt{D^{-1}} \quad .$$

That is, every $\mathcal{L}_{\mathbf{f}}(W)$ is related to $\mathcal{L}(W)$ by

$$\mathcal{L}_{\mathbf{f}}(W) \;=\; \mathcal{L}(W) - \mathrm{diag}(\mathbf{h})$$

for $\mathbf{h} = (h_i) \in \mathbb{R}^n$ with

$$h_i = 1 - \sum_{j=1}^{n} w_{ij} \frac{\sqrt{f_j / f_i}}{\sqrt{d_i d_j}} \quad .$$

Proof. Right from the definitions we get that:

$$\begin{aligned}
\mathcal{L}_{\mathbf{f}}(W) &= I - \sqrt{F^{-1}}(\tilde{W} - \tilde{D}_{\mathbf{f}} + F)\sqrt{F^{-1}} \\
&= I - \sqrt{F^{-1}}\sqrt{FD^{-1}}W\sqrt{FD^{-1}}\sqrt{F^{-1}} + \tilde{D}_{\mathbf{f}} F^{-1} - I \\
&= \tilde{D}_{\mathbf{f}} F^{-1} - \sqrt{D^{-1}}W\sqrt{D^{-1}} \quad .
\end{aligned}$$

This gives with $\mathcal{L}(W) = I - \sqrt{D^{-1}}W\sqrt{D^{-1}}$ that

$$\mathcal{L}_{\mathbf{f}}(W) \;=\; \mathcal{L}(W) - \underbrace{(I - \tilde{D}_{\mathbf{f}}F^{-1})}_{=:\mathrm{diag}(\mathbf{h})} \quad ,$$

where $\mathbf{h} = (I - \tilde{D}_{\mathbf{f}}F^{-1})\mathbf{1} = (I - \sqrt{F^{-1}D^{-1}}W\sqrt{FD^{-1}})\mathbf{1} \;\in\; \mathrm{diag}(\mathbb{R}^n)$, which can be expressed element-wise as stated in the lemma. □

Lemma 2.3.9 shows that every **f**-adjustment corresponds to a specific modification of the normalized Laplacian along its main diagonal. We can think of **f**-adjusting as of replacing the identity matrix in $\mathcal{L}(W) = I - \sqrt{D^{-1}}W\sqrt{D^{-1}}$ by the new diagonal matrix $\tilde{D}_{\mathbf{f}}F^{-1}$. Its diagonal entries reflect the relative deviation $\tilde{\mathbf{d}}/\mathbf{f}$ between the intended new degrees $\mathbf{f}$ and the degrees $\tilde{\mathbf{d}}$ that are obtained by **f**-scaling without subsequent **f**-selflooping. Note that **f**-scaling alone does not reveal any clear relation to $\mathcal{L}(W)$, only its combination with subsequent **f**-selflooping collapses down to this "simple" algebraic form. We denote any modification along the main diagonal of a matrix as a **diagonal modification**. Whenever it exceeds just a tiny perturbation, it has a strong non-linear impact on the spectrum: the eigenvalues can only loosely and abstractly be bounded by Horn's inequalities (Bhatia, 2001), and nothing is known on the impact on the eigenvectors. This complexity should not be considered as a weakness, but rather as a chance: it shows that in principle diagonal modifications are able to drastically change the interpretation of eigenvalues and eigenvectors. So if we succeed in linking this graph modification to an interesting change of graph properties, then we gain a novel analysis tool from that.

A similar approach of studying diagonal modifications is taken by Bapat et al. (2001), who study the modified unnormalized Laplacian $L(W) + S$ for $S \in \mathrm{diag}(\mathbb{R}^n)$. Note that $L(W)+S \notin L(\mathbb{W}_{\ominus})$ for all $S \neq 0$, hence no diagonal modification of the unnormalized Laplacian represents the unnormalized Laplacian of another graph. Nevertheless, this approach still provides useful meta-information on the graph $\mathcal{G}(W)$. For example Wu et al. (2012) consider this diagonal modification for $S \geq 0$, and show how to interpret $(L(W)+S)^{-1}S$ as meaningful random walk absorption probabilities on $\mathcal{G}(W)$.

In contrast to the unnormalized Laplacian, the normalized Laplacian can be diagonally modified in infinitely many ways to yield the normalized Laplacian $\mathcal{L}(A)$ of another graph $A \in \mathbb{W}_{\ominus}$. The algebraic intuition, one of the main contributions in this chapter, makes this relation precise in the sense that "all valid" diagonal modifications of the normalized Laplacian $\mathcal{L}(W)$ are in a one-to-one correspondence with all possible **f**-adjustments of W.

We end this section by a corollary that collects the most important algebraic properties of **f**-adjusting.

Corollary 2.3.10 (Algebraic properties of the f-adjusted Laplacian)
For any graph matrix $W \in \mathbb{W}$, $\mathbf{f} \in \mathbb{R}^n_{>0}$ and $c > 0$, the following properties hold:

- $\mathcal{L}_{\mathbf{f}}(W) = \mathcal{L}_{c\mathbf{f}}(W)$
- $\mathcal{L}_{\mathbf{f}}(W) = c \cdot \mathcal{L}(\overline{W}_{\mathbf{f},c})$
- $\mathcal{L}_{\mathbf{f}}(W) = Z - \sqrt{D^{-1}}W\sqrt{D^{-1}}$ *where* $Z = \tilde{D}_{\mathbf{f}}/F$ *shows along its main diagonal the relative approximation error of* $\mathbf{f}$*-scaling.*
- $\mathcal{L}_{\mathbf{f}}(W)$ *is positive semi-definite, despite of any negative selfloops*
- $\mathcal{L}_{\mathbf{f}}(W)$ *allows for spectral clustering of the* $\mathbf{f}$*-adjusted graph*

2.4 Main contributions

Both contributions in this section can be summarized as follows: modifying the main diagonal of the Laplacian matrix affects volumes and cuts in a controlled way. This can be utilized for random geometric graphs in order to study a modification of the original sampling density.

2.4.1 Contribution 1: Geometric interpretation of f-adjusting

The first main contribution is a geometric interpretation of $\mathbf{f}$-adjusting if applied to an RGNG. Theorem 2.4.1 determines the continuous quantities that are represented by the volumes and cuts of an $\mathbf{f}$-adjusted graph. This justifies to think of $\mathbf{f}$-adjusting as of modifying the underlying density: studying volumes and cuts of an $\mathbf{f}$-adjusted RGNG corresponds to studying a modified variant of the original sampling density. Depending on which modification is desired in an application, one can choose a corresponding vector $\mathbf{f}$ for the $\mathbf{f}$-adjustment.

The key to this interpretation is given by the following theorem, which refines Theorem 2.3.3.

Theorem 2.4.1 (Convergence limits of interspace quantities under f-adjusting) *Let $p : \mathbb{R}^d \to \mathbb{R}$ denote a probability density that satisfies the regularity conditions 2.3.2, in particular p has compact support $\mathcal{X}$. Consider a random sequence of matrices (W_n), where $W_n \in \mathbb{W}$ is the graph matrix of an RGNG that is built on n sample points drawn from p, with parameters chosen according to one of the limit scenarios 2.3.1. Fix a hyperplane S in $\mathbb{R}^d$ and refer by H to either one of the two halfspaces induced by S. Let $f : \mathcal{X} \to \mathbb{R}_{>0}$ denote any positive-valued twice-differentiable function with bounded gradient on the interior of $\mathcal{X}$. Define $\mathbf{f}_n := (f(\mathbf{x}_i))$, that is the vector of the function evaluations of f at the sample points. Consider the sequence $\big(\overline{W}_n\big)$, where $\overline{W}_n := \overline{(W_n)}_{\mathbf{f}_n}$ denotes the $\mathbf{f}_n$-adjusted graph of W_n. Then the following L_1-convergences hold true:*

$$c_n \cdot \mathrm{vol}_{\mathbf{f}_n}(H) \xrightarrow{a.s.} \mathrm{vol}_{f \cdot p}(H) \qquad \text{and} \qquad c'_n \cdot \mathrm{cut}_{\overline{W}_n}(S) \xrightarrow{a.s.} \mathrm{cut}_{f \cdot p}(S) \quad ,$$

where (c_n) and (c'_n) are deterministic sequences of scaling constants that depend only on n, d and the parameters of the RGNG.

Theorem 2.4.1 can be summarized as follows: if the vector $\mathbf{f}$ is defined by evaluating a sufficiently smooth function f at the locations of the sample points, then volumes and cuts in the $\mathbf{f}$-adjusted graph refer, in the large sample limit, to volumes and cuts of the function $f \cdot p$ in space.

But how to define such function f in practice, particularly in the case that the sample points are not even known? The simplest strategy to start with is by choosing f as the constant 1-function on space: $f(x) = 1$. This gives $\mathbf{f} := \mathbf{1}$ the all-one-vector, and $f \cdot p = p$. Hence, volumes and cuts in the $\mathbf{1}$-adjusted graph refer to volumes and cuts of p itself, without the squaring bias that is immanent to the original graph.

In applications in which the sample points are known, one can just choose any function f and define $\mathbf{f} := (f(\mathbf{x}_i))$ as in the theorem, in order to get that volumes and cuts in the $\mathbf{f}$-adjusted graph refer to the function $f \cdot p$ in space.

But even if the sample points are not known, there are further alternatives that allow to apply $\mathbf{f}$-adjusting. The alternative studied here is to define $\mathbf{f}$ with respect to the vertex degrees in the graph, that is with respect to the consistent density estimate $d_i \cdot c_n \approx p(\mathbf{x}_i)$ for some unknown scaling factor $c_n > 0$. In particular, for any $q \in \mathbb{R}$ we get by d_i^q an estimate on the function $f = c_n^{-q} \cdot p^q$. As a consequence, for $\mathbf{f} := (d_i^q)$ the $\mathbf{f}$-adjusted graph refers to volumes and cuts of the function $p^q \cdot p = p^{q+1}$, up to a scaling factor. Consider the following three special cases:

- $q = 0$: this choice corresponds to $\mathbf{f} := \mathbf{1} = (d_i^0)$. Volumes and cuts in the $\mathbf{1}$-adjusted graph refer to $p^{0+1} = p$ itself, without squaring.
- $q = 1$: this choice corresponds to $\mathbf{f} := \mathbf{d} = (d_i)$. Since d_i is a density estimate on p, we get that the $\mathbf{d}$-adjusted graph refers to the continuous quantities p^2, just as the original

graph. Indeed the $\mathbf{d}$-adjusted graph *equals* the original graph: $W = \overline{W}_{\mathbf{d}}$.

- $q = -1$: since p is bounded away from zero, we can even choose negative values for q. For $\mathbf{f} := (d_i^{-1})$, we get that volumes and cuts of the $\mathbf{f}$-adjusted graph refer to $p^{-1}p = 1$, the uniform distribution on the support $\mathcal{X}$ of p, up to a scaling factor.

The following corollary summarizes meaningful choices of $\mathbf{f}$. It can be considered as the most relevant result of this chapter in order to make the intuition behind the geometric interpretation of $\mathbf{f}$-adjusting precise.

Corollary 2.4.2 (Choices of parameter f for f-adjusting)
Let p denote a probability density that satisfies the regularity conditions 2.3.2. Further, let G denote an RGNG that is built on top of a sample drawn from p. Volumes and cuts in G are related (in the sense of Theorem 2.4.1) to the function p^2. Volumes and cuts in the $\mathbf{f}$-adjusted graph $G_{\mathbf{f}}$ are related to modifications of p, with examples given for the following choices of $\mathbf{f} \in \mathbb{R}^n_{>0}$:

$\mathbf{f} := \ldots$	*related continuous volumes and cuts after $\mathbf{f}$-adjusting*	
$\mathbf{d}$	p^2	*biased toward squaring, as in the original graph $G = G_{\mathbf{d}}$*
$\mathbf{1}$	p	*unbiased, corrected volumes and cuts of p itself*
(d_i^{-1})	1	*uniform density on the support of p*
(d_i^q)	p^{q+1}	*any power of p*
$(f(\mathbf{x}_i))$	$f \cdot p$	*general modification of p as driven by some $f(\mathbf{x}_i)$*

The applications in Section 2.6 show how these choices of $\mathbf{f}$ can provide concrete solutions to a variety of different problems in machine learning.

2.4.2 Contribution 2: Algebraic interpretation of f-adjusting

$\mathbf{f}$-selflooping is a fairly simple graph modification: it represents every possible way of how to modify an adjacency matrix $W \in \mathbb{W}_{\ominus}$ along its main diagonal such that the result is again an adjacency matrix A of another graph. That is, $\mathbf{f}$-selflooping refers to all solutions of the equation $W + X = A$ for all $X \in \operatorname{diag}(\mathbb{R}^n)$ and $A \in \mathbb{W}_{\ominus}$.

At first glance, $\mathbf{f}$-adjusting appears to be a much more artificial graph modification. But the algebraic interpretation shows that this is not the case. $\mathbf{f}$-adjusting is equivalent to a modification of matrices that can be described in exactly the same way as $\mathbf{f}$-selflooping, but under the normalized Laplacian operator: it represents every possible way of how to modify a normalized Laplacian $\mathcal{L}(W)$ along its main diagonal such that the result is the normalized Laplacian $\mathcal{L}(A)$ of another graph. That is, $\mathbf{f}$-adjusting refers to all solutions of the equation $\mathcal{L}(W) + X = \mathcal{L}(A)$ for all $X \in \operatorname{diag}(\mathbb{R}^n)$ and $A \in \mathbb{W}_{\ominus}$.

As a consequence, both $\mathbf{f}$-selflooping and $\mathbf{f}$-adjusting can be characterized in terms of a simple matrix equation. However, $\mathbf{f}$-adjusting is more complex than $\mathbf{f}$-selflooping in the sense that it additionally modifies all non-selfloop edge weights.

We already know from Lemma 2.3.9 that every **f**-adjusting can be understood as a diagonal modification of the form $\mathcal{L}(W) + X = \mathcal{L}(A)$ for *some* $X \in \mathrm{diag}(\mathbb{R}^n)$ and $A \in \mathbb{W}_{\ominus}$. This raises the question whether also the converse is true: does *every* "valid" diagonal modification (that is any choice of $X \in \mathrm{diag}(\mathbb{R}^n)$ such that $\mathcal{L}(W) + X = \mathcal{L}(A)$ for some $A \in \mathbb{W}_{\ominus}$) imply that A must be an **f**-adjustment of W? The following theorem gives a positive answer.

Theorem 2.4.3 (Equivalence between f-adjusting and diagonal modifications)
Let $W \in \mathbb{W}$ denote any graph matrix. Fix any diagonal matrix $X \in \mathrm{diag}(\mathbb{R}^n)$ and any weak graph matrix $A \in \mathbb{W}_{\ominus}$. Then it holds that

$$\mathcal{L}(W) + X = \mathcal{L}(A)$$

if and only if $A = \overline{W}_{\mathbf{f}}$ and $X = \tilde{D}_{\mathbf{f}} F^{-1} - I$ for any $\mathbf{f} \in \mathbb{R}^n_{>0}$.

Theorem 2.4.3 gives that **f**-adjusting is *equivalent* to modifying the main diagonal of the normalized Laplacian in any valid way.

Given any **f**, one can simply determine the corresponding A and X. However, it is unclear how to achieve the converse direction: given any diagonal modification X, how to decide whether it is valid, and if yes, for which **f**. The converse direction is equivalent to deciding whether the necessary and sufficient condition $X = \tilde{D}_{\mathbf{f}} F^{-1} - I$ can be satisfied for some choice of $\mathbf{f} \in \mathbb{R}^n_{>0}$. It is not easy to find a suitable **f** because this condition refers to a system of rational equations: $x_i + 1 = \tilde{d}_i / f_i = \sum_{j \in V} w_{ij} \sqrt{f_j} / \sqrt{d_i d_j f_i}$. The following theorem provides a constructive characterization of all valid choices for X that further allows to compute the corresponding **f** explicitly.

Theorem 2.4.4 (Characterization of all valid diagonal modifications)
Let $W \in \mathbb{W}$ denote any graph matrix of a connected graph. Choose any diagonal matrix $Z \in \mathrm{diag}(\mathbb{R}^n)$ as a replacement for the identity matrix in $\mathcal{L}(W) = I - \sqrt{D^{-1}} W \sqrt{D^{-1}}$. Then it holds that

$$Z - \sqrt{D^{-1}} W \sqrt{D^{-1}} \;\in\; \mathcal{L}(\mathbb{W}_{\ominus})$$

*if and only if $Z \in \mathrm{diag}(\mathbb{R}^n_{>0})$ and Z is scaled such that $M := Z^{-1} \sqrt{D^{-1}} W \sqrt{D^{-1}}$ has spectral radius 1. This further implies that $Z - \sqrt{D^{-1}} W \sqrt{D^{-1}} = \mathcal{L}_{\mathbf{f}}(W)$ is an **f**-adjusted Laplacian of W, where $\sqrt{\mathbf{f}}$ is uniquely determined (up to scaling) as the all-positive eigenvector corresponding to the simple real eigenvalue 1 of M.*

Both theorems together form the **algebraic interpretation** of **f**-adjusting. Beside classifying **f**-adjusting as a natural graph modification with a simple representation in terms of normalized Laplacian matrices, this interpretation yields a couple of further interesting consequences:

Generating f-adjusted graphs by choosing a new diagonal. Theorem 2.4.4 allows for the following constructive way to generate other normalized Laplacians from $\mathcal{L}(W)$ by diagonal modifications: choose any candidate $\hat{Z} \in \operatorname{diag}(\mathbb{R}^n_{>0})$ for the goal of replacing the identity matrix in the normalized Laplacian $\mathcal{L}(W) = I - \sqrt{D^{-1}}W\sqrt{D^{-1}}$ toward $\hat{Z} - \sqrt{D^{-1}}W\sqrt{D^{-1}}$. This will *not* yield another normalized Laplacian in general. However, every choice of $\hat{Z}$ determines by the spectral radius $\alpha := \rho(\hat{Z}^{-1}\sqrt{D^{-1}}W\sqrt{D^{-1}})$ a unique scaling factor such that $\alpha\hat{Z} - \sqrt{D^{-1}}W\sqrt{D^{-1}}$ indeed equals the normalized Laplacian $\mathcal{L}(A)$ of some $A \in \mathbb{W}_{\ominus}$. If we think of the set $\mathbb{R}\hat{Z}$ as a "direction", then there lies in every direction exactly one other normalized Laplacian $\mathcal{L}(A)$. Further, A is an $\mathbf{f}$-adjustment of W, and all $\mathbf{f}$-adjustments of W can be generated this way. Moreover, $\mathbf{f}$ is related to $\hat{Z}$ in the way that $\sqrt{\mathbf{f}}$ is the eigenvector to the real eigenvalue α of $\hat{Z}^{-1}\sqrt{D^{-1}}W\sqrt{D^{-1}}$.

Generalization to unconnected graphs. Theorem 2.4.4 can be applied to unconnected graphs by applying it individually to each connected component. That is, for every connected component C, the spectral radius of $Z_C^{-1}\sqrt{D_C^{-1}}W_C\sqrt{D_C^{-1}}$ must equal 1, where Z_C, D_C and W_C refer to the restriction of the graph to component C. Further, the uniqueness of $\mathbf{f}$ is affected because $\mathbf{f}_C$ can be scaled individually within each connected component without changing the normalized Laplacian.

Generalization to $(\mathbf{f}, c)$-adjusting. One can generalize the original problem to the problem of determining all $X \in \operatorname{diag}(\mathbb{R}^n)$ such that $\mathcal{L}(W) + X = c \cdot \mathcal{L}(A)$ for any $c \in \mathbb{R}_{>0}$ and any $A \in \mathbb{W}_{\ominus}$. It is straightforward to show from Lemma 2.3.7 that this equation holds true if and only if A is an $(\mathbf{f}, c)$-adjustment of W.

2.5 Proofs and technical details

This section provides details on the proofs of the algebraic and of the geometric interpretation of $\mathbf{f}$-adjusting. Since this requires to deal with weak graph matrices, the first subsection collects several basic properties of this graph type.

2.5.1 Details on weak graph matrices

In the following we study various properties of weak graph matrices that are essential for the proofs of the algebraic interpretation in the next section.

Throughout this section, we define for any $W \in \mathbb{W}_{\ominus}$ and $\mathbf{f} \in \mathbb{R}^n_{>0}$ the variables $\mathbf{d} := W\mathbf{1}$, $D := \operatorname{diag}(\mathbf{d})$, $F := \operatorname{diag}(\mathbf{f})$. First, let us summarize some basic facts on the main diagonal entries of the Laplacian $L(W) = [l_{ij}]$ and the normalized Laplacian $\mathcal{L}(W) = [\ell_{ij}]$ for non-weak and weak graphs.

Fact 2.5.1 (Main diagonal entries of the Laplacian)
For $W \in \mathbb{W}$ it holds for all $i \in V$ that:

(i) $0 \leq w_{ii} \leq d_i, \quad 0 \leq w_{ii}/d_i \leq 1$

(ii) $l_{ii} = d_i - w_{ii} \in [0, d_i]$ *with* $l_{ii} = 0$ *if and only if i is a separated vertex.*

(iii) $\ell_{ii} = 1 - w_{ii}/d_i \in [0, 1]$ *with* $\ell_{ii} = 0$ *if and only if i is a separated vertex.*

For weak graph matrices (now $w_{ii} < 0$ is possible) the following properties hold.

Fact 2.5.2 (Main diagonal entries of the Laplacian of weak graphs)
For $W \in \mathbb{W}_\ominus$ it holds for all $i \in V$ that:

(i) $-\infty < w_{ii} \leq d_i, \quad -\infty < w_{ii}/d_i \leq 1$

(ii) $l_{ii} = d_i - w_{ii} \in [0, \infty)$ *with* $l_{ii} = 0$ *if and only if i is a separated vertex, and* $l_{ii} > d_i$ *if and only if i has a negative selfloop.*

(iii) $\ell_{ii} = 1 - w_{ii}/d_i \in [0, \infty)$ *with* $\ell_{ii} = 0$ *if and only if i is a separated vertex, and* $\ell_{ii} > 1$ *if and only if i has a negative selfloop.*

As already stated in Section 2.3.6, the set $\mathbb{W}$ is not closed under **f**-selflooping because $W \in \mathbb{W}$ might be mapped to some $W^\circ_{\mathbf{f}} \in \mathbb{W}_\ominus \setminus \mathbb{W}$. However, for all $W \in \mathbb{W}_\ominus$ there exists some large enough $c > 0$ such that $W^\circ_{c\mathbf{f}} \in \mathbb{W}$. The following proposition quantifies this threshold.

Proposition 2.5.3 (Non-weak output of f-selflooping)
Let $W \in \mathbb{W}_\ominus$, $\mathbf{f} \in \mathbb{R}^n_{>0}$ and $c > 0$. Define $c^+_{W,\mathbf{f}} := \max_{i \in V}\{(d_i - w_{ii})/f_i\} \geq 0$. Then it holds that $W^\circ_{c\mathbf{f}} \in \mathbb{W}$ if and only if $c \geq c^+_{W,\mathbf{f}}$.

Proof. From $W^\circ_{c\mathbf{f}} = W - D + cF$ we get that $W^\circ_{c\mathbf{f}}$ has no negative selfloop if and only if $w_{ii} - d_i + cf_i \geq 0$ for all $i \in V$. In the case $c \geq c^+_{W,\mathbf{f}}$ we get that $w_{ii} - d_i + cf_i \geq w_{ii} - d_i + \max_{i \in V}\{(d_i - w_{ii})/f_i\} \cdot f_i = \max_{i \in V}\{(d_i - w_{ii})\} - (d_i - w_{ii}) \geq 0$ for all $i \in V$. For $c < c^+_{W,\mathbf{f}}$, choose $k \in \{i \in V \mid (d_i - w_{ii})/f_i = c^+_{W,\mathbf{f}}\}$, and get that $w_{kk} - d_k + cf_k < w_{kk} - d_k + c^+_{W,\mathbf{f}} \cdot f_k = w_{kk} - d_k + (d_k - w_{kk})/f_k \cdot f_k = 0$. □

Proposition 2.5.3 directly implies the following two corollaries.

Corollary 2.5.4 (Partitioning the output of f-selflooping)
For all $W \in \mathbb{W}_{\ominus}$ and all $\mathbf{f} \in \mathbb{R}^n_{>0}$, we get the following partition:

$$\{W^{\circ}_{c\mathbf{f}} \mid c > 0\} \quad = \quad \underbrace{\{W^{\circ}_{c\mathbf{f}} \mid 0 < c < c^{+}_{W,\mathbf{f}}\}}_{\subseteq\, \mathbb{W}_{\ominus}\setminus\mathbb{W}} \;\dot\cup\; \underbrace{\{W^{\circ}_{c\mathbf{f}} \mid c \geq c^{+}_{W,\mathbf{f}}\}}_{\subseteq\, \mathbb{W}} .$$

Corollary 2.5.5 (f-selflooping of loops-only graphs)
It holds that $c^{+}_{W,\mathbf{f}} = 0$ if and only if W is loops-only ($\Leftrightarrow d_i = w_{ii}$ for all $i \in V$).

Next we study how the output of a graph modification is affected if we choose parameter $a\mathbf{f}$ instead of $\mathbf{f}$ for some $a > 0$. First, we show that $\mathbf{f}$-scaling as well as $(\mathbf{f}, c)$-adjusting are linear in their vector parameter.

Proposition 2.5.6 (Linearity of f-scaling and $(\mathbf{f}, c)$-adjusting)
For every $W \in \mathbb{W}$, $\mathbf{f} \in \mathbb{R}^n_{>0}$ and $c, a > 0$, both $\mathbf{f}$-scaling and $(\mathbf{f}, c)$-adjusting are linear:

$$\tilde{W}_{a\mathbf{f}} = a \cdot \tilde{W}_{\mathbf{f}} \qquad \text{and} \qquad \overline{W}_{a\mathbf{f},c} = a \cdot \overline{W}_{\mathbf{f},c} \quad .$$

Proof. $a\mathbf{f}$-scaling gives that $\tilde{W}_{a\mathbf{f}} = \sqrt{aFD^{-1}}W\sqrt{aFD^{-1}} = a\sqrt{FD^{-1}}W\sqrt{FD^{-1}} = a \cdot \tilde{W}_{\mathbf{f}}$, which implies $\tilde{D}_{a\mathbf{f}} = a \cdot \tilde{D}_{\mathbf{f}}$. Thus, $\overline{W}_{a\mathbf{f},c} = \tilde{W}_{a\mathbf{f}} - \tilde{D}_{a\mathbf{f}} + caF = a \cdot (\tilde{W}_{\mathbf{f}} - \tilde{D}_{\mathbf{f}} + cF) = a \cdot \overline{W}_{\mathbf{f},c}$. □

For $\mathbf{f}$-selflooping, linearity does not hold, since $W^{\circ}_{a\mathbf{f}} \neq a \cdot W^{\circ}_{\mathbf{f}}$ whenever W is non-loops-only (because the right-hand side also affects off-diagonal entries). However, the next lemma shows that at least for $a\mathbf{d}$-selflooping some other sense of linearity holds, namely an inverse-linear relation between the normalized Laplacians of $W^{\circ}_{\mathbf{d}}$ and of $W^{\circ}_{a\mathbf{d}}$.

Lemma 2.5.7 ($\mathcal{L}$-inverse-linearity of d-selflooping)
For all $W \in \mathbb{W}_{\ominus}$ it holds that $\mathcal{L}(W) = \mathcal{L}(W^{\circ}_{\mathbf{d}}) = a \cdot \mathcal{L}(W^{\circ}_{a\mathbf{d}})$ for all $a > 0$.

Proof. $W = W - D + D = W^{\circ}_{\mathbf{d}}$. Further, $\mathcal{L}(W^{\circ}_{a\mathbf{d}}) = \sqrt{(aD)^{-1}}(aD - (W - D + aD))\sqrt{(aD)^{-1}} = a^{-1}\sqrt{D^{-1}}(D - W)\sqrt{D^{-1}} = a^{-1}\mathcal{L}(W)$. □

This easy yet powerful lemma implies that all matrices in $\{\mathcal{L}(W^{\circ}_{a\mathbf{d}}) \mid a > 0\}$ share the same eigenvectors with their corresponding eigenvalues scaled according to a. This implies for example that positive semi-definiteness of $L(W)$ and $\mathcal{L}(W)$ generalizes to non-weak graph matrices, as the next lemma shows.

Lemma 2.5.8 (Generalization of $\mathcal{L}$ to weak graph matrices)
For $W \in \mathbb{W}_{\ominus}$ it holds that $\mathcal{L}(W)$ is positive semi-definite. Further, 0 is an eigenvalue of $\mathcal{L}(W)$ with multiplicity equal to the number of connected components in $\mathcal{G}(W)$, and $\sqrt{\mathbf{d}}$ is a corresponding eigenvector.

Proof. All stated properties are well-known for non-weak graph matrices (see Section 1.3.5) Adding/Removing selfloops does not affect the number of connected components. Thus, all results follow directly from Lemma 2.5.7 and Proposition 2.5.3, since there exists some $c > 0$ large enough such that $W^{\circ}_{c\mathbf{d}} \in \mathbb{W}$, thus $\mathcal{L}(W) = c \cdot \mathcal{L}(W^{\circ}_{c\mathbf{d}})$ equals the standard normalized Laplacian matrix of a non-weak graph, up to scaling by c. □

Observe that $A = cW$ implies that $\mathcal{L}(A) = \mathcal{L}(W)$. The following lemma shows that for connected graphs this implication even holds as an equivalence.

Lemma 2.5.9 (Equal normalized Laplacians)
For a connected graph $\mathcal{G}(W) \in \mathcal{G}(\mathbb{W}_{\ominus})$ and any $A \in \mathbb{W}_{\ominus}$ it holds that $\mathcal{L}(W) = \mathcal{L}(A)$ if and only if $A = c \cdot W$ for some $c > 0$.

Proof. "⇐". Assuming $A = c \cdot W$ for some $c > 0$ yields $\mathcal{L}(A) = I - \sqrt{(cD)^{-1}}cW\sqrt{(cD)^{-1}} = I - \sqrt{D^{-1}}W\sqrt{D^{-1}} = \mathcal{L}(W)$. "⇒". Let $\mathbf{t} := A\mathbf{1} > 0$. Assuming $\mathcal{L}(W) = \mathcal{L}(A) =: [\ell_{ij}]$ gives by Lemma 2.5.8 that $\sqrt{\mathbf{d}}$ is the unique (up to scaling) eigenvector of $\mathcal{L}(W)$ to the eigenvalue 0, similarly $\sqrt{\mathbf{t}}$ for $\mathcal{L}(A) = \mathcal{L}(W)$. Hence, $\mathbf{t} = \alpha\mathbf{d}$ for some $\alpha \in \mathbb{R}$, that is $t_i = \alpha \cdot d_i$ for all $i \in V$. For every $i \neq j$ we get from $\ell_{ij} = w_{ij}/\sqrt{d_i d_j} = a_{ij}/\sqrt{t_i t_j}$ that $a_{ij} = w_{ij} \cdot \sqrt{t_i t_j (d_i d_j)^{-1}} = \alpha \cdot w_{ij}$. For $i = j$ we get from $\ell_{ij} = (d_i - w_{ii})/d_i = (t_i - a_{ii})/t_i$ that $d_i - w_{ii} = d_i - a_{ii}/\alpha$, hence $a_{ii} = \alpha \cdot w_{ii}$. That is $A = \alpha \cdot W$. □

2.5.2 Proofs for the algebraic interpretation

The goal of this section is to prove Theorem 2.4.3 and Theorem 2.4.4. Their proofs rely on various aspects of the **Perron-Frobenius-Theorem (PFT)** applied to irreducible non-negative matrices. See for example Stańczak et al. (2006) for an overview on Perron-Frobenius theory. Here, we only take from PFT that the following three properties are satisfied by every irreducible non-negative matrix $A \in \mathbb{R}^{n \times n}_{\geq 0}$:

- the spectral radius $\rho(A) > 0$ is itself a real positive eigenvalue of A, denoted as the **Perron root** of A
- the Perron root is a simple eigenvalue, which implies that the corresponding left and right eigenvectors are unique (up to scaling), denoted as the **(left/right) Perron eigenvector** of A

- the Perron eigenvectors can be chosen such that all entries in it are strictly positive, and no other eigenvector can be chosen this way

For example, consider the random walk matrix $P = D^{-1}W$ of a strongly connected graph. P is irreducible because P has the same zero-entries as the irreducible matrix W. Being row-stochastic means that $P\mathbf{1} = \mathbf{1}$, thus, $\mathbf{1}$ is a right eigenvector to eigenvalue 1. PFT gives further information: since $\mathbf{1}$ is all-positive, it is the right Perron eigenvector, hence 1 is the spectral radius of P and there is no other right eigenvector to this eigenvalue. Note that the left Perron eigenvector is the stationary distribution $\boldsymbol{\pi}$, because $\boldsymbol{\pi}P = \boldsymbol{\pi}$.

We are going to prove two preparatory lemmas that finally lead to Lemma 2.5.12, which almost implies Theorem 2.4.3 and Theorem 2.4.4.

Note that in order to get more compact statements, we use everywhere throughout this section that $\tilde{W}_{\mathbf{1}} = \sqrt{D^{-1}}W\sqrt{D^{-1}}$. This gives the syntax that $\mathcal{L}(W) = I - \tilde{W}_{\mathbf{1}}$, and according to Lemma 2.3.9 that $\mathcal{L}_{\mathbf{f}}(W) = Z - \tilde{W}_{\mathbf{1}}$ for $Z = \tilde{D}_{\mathbf{f}}F^{-1}$.

Lemma 2.5.10 (Characterization of f-adjusted Laplacians)
For any connected graph $\mathcal{G}(W) \in \mathcal{G}(\mathbb{W})$ let

$$\Lambda := \{\mathcal{L}_{\mathbf{f}}(W) \mid \mathbf{f} \in \mathbb{R}^n_{>0}\}$$

*denote the **Laplacian orbit of** W **under** $\mathbf{f}$**-adjusting**. Further define*

$$\Lambda' := \{Z - \tilde{W}_{\mathbf{1}} \mid Z \in \mathrm{diag}(\mathbb{R}^n_{>0}),\ \rho(Z^{-1}\tilde{W}_{\mathbf{1}}) = 1\}$$

for $\rho(\cdot)$ the spectral radius of its argument. Then it holds that $\Lambda = \Lambda'$ with the relation $Z = \tilde{D}_{\mathbf{f}}F^{-1}$, wherein $\sqrt{\mathbf{f}} := (\sqrt{f_i})$ is the unique (up to scaling) eigenvector of $Z^{-1}\tilde{W}_{\mathbf{1}}$ for eigenvalue 1.

Proof. It is well-known that $\mathcal{G}(W)$ is connected if and only if W is irreducible. This implies, for any choice of $Z \in \mathrm{diag}(\mathbb{R}^n_{>0})$, that $Z^{-1}\tilde{W}_{\mathbf{1}}$ is irreducible (and non-negative), too, since it has the same non-zero-pattern as W.

$\Lambda \subseteq \Lambda'$: fix any $\mathcal{L}_{\mathbf{f}}(W) \in \Lambda$ for some $\mathbf{f} > 0$ and set $Z := \tilde{D}_{\mathbf{f}}F^{-1} \in \mathrm{diag}(\mathbb{R}^n_{>0})$. We prove that $\rho(Z^{-1}\tilde{W}_{\mathbf{1}}) = 1$ by finding an all-positive eigenvector $\mathbf{x}$ to the following eigenvalue problem:

$$Z^{-1}\sqrt{D^{-1}}W\sqrt{D^{-1}}\mathbf{x} = \mathbf{x}. \qquad (\star)$$

We propose that $\sqrt{\mathbf{f}}$ is such an eigenvector. Plugging $\tilde{D}_{\mathbf{f}} = \mathrm{diag}(\sqrt{FD^{-1}}W\sqrt{FD^{-1}}\mathbf{1})$ into $\tilde{D}_{\mathbf{f}}F^{-1} = Z$ element-wise, gives with $Z = \mathrm{diag}(z_1, \ldots, z_n)$ that

$$\begin{aligned} & & \tilde{D}_{\mathbf{f}} F^{-1} &= Z \\ \Leftrightarrow & \sum_{j=1}^{n} \frac{w_{ij}}{\sqrt{d_i d_j}} \cdot \sqrt{f_j} &= z_i \cdot \sqrt{f_i} \quad \text{for all } i = 1, \dots, n \\ \Leftrightarrow & \sqrt{D^{-1}} W \sqrt{D^{-1}} \sqrt{\mathbf{f}} &= Z\sqrt{\mathbf{f}} \\ \Leftrightarrow & Z^{-1}\sqrt{D^{-1}} W \sqrt{D^{-1}} \sqrt{\mathbf{f}} &= \sqrt{\mathbf{f}} \end{aligned} \tag{$\star\star$}$$

Thus $\sqrt{\mathbf{f}}$ is indeed a solution to ($\star$), hence an all-positive (right) eigenvector of $Z^{-1}\tilde{W}_{\mathbf{1}}$ to the (existing) eigenvalue 1. By PFT, there is exactly one eigenvalue providing all-positive eigenvectors. Further it determines the spectral radius and is simple. Thus, we have that $\rho(Z^{-1}\tilde{W}_{\mathbf{1}}) = 1$ is the Perron root and that the corresponding left and right eigenvectors are unique (up to scaling). With Lemma 2.3.9 we get that $Z = \tilde{D}_{\mathbf{f}} F^{-1}$ is a valid choice for Z to represent $\mathcal{L}_{\mathbf{f}}(W)$ as an element in Λ'.

$\Lambda' \subseteq \Lambda$: fix any $Z - \tilde{W}_{\mathbf{1}} \in \Lambda'$ for some $Z \in \operatorname{diag}(\mathbb{R}^n_{>0})$ with $\rho(Z^{-1}\tilde{W}_{\mathbf{1}}) = 1$. By irreducibility of $Z^{-1}\tilde{W}_{\mathbf{1}}$, there exists by PFT a unique (up to scaling) all-positive solution $\hat{\mathbf{x}}$ of ($\star$). By defining $\mathbf{f} := \hat{\mathbf{x}}^2$ we get that $\sqrt{\mathbf{f}}$ is a solution to the eigenvalue problem ($\star$). From ($\star\star$) we see that this is equivalent to $Z = \tilde{D}_{\mathbf{f}} F^{-1}$. Thus, we get that $Z - \tilde{W}_{\mathbf{1}} = \tilde{D}_{\mathbf{f}} F^{-1} - \tilde{W}_{\mathbf{1}} = \mathcal{L}_{\mathbf{f}}(W) \in \Lambda$ for this unique (up to scaling) choice of $\mathbf{f}$. □

We have already shown in Lemma 2.3.9 that $\mathbf{f}$-adjusting can be understood as a diagonal modification of the form $\mathcal{L}(W) + X = \mathcal{L}(A)$ for some $X \in \operatorname{diag}(\mathbb{R}^n)$ and $A \in \mathbb{W}_{\ominus}$. So far the question is left open if further the converse is true: does *every* such diagonal modification imply that A is an $\mathbf{f}$-adjustment of W? We now answer this in the affirmative.

Lemma 2.5.11 (Characterization of Diagonally Modified Laplacians)
Let $W \in \mathbb{W}$. Then $\mathcal{L}(W) + X = \mathcal{L}(A)$ holds true for $X \in \operatorname{diag}(\mathbb{R}^n)$ and $A \in \mathbb{W}_{\ominus}$ if and only if A is an $\mathbf{f}$-adjustment of W for some $\mathbf{f} \in \mathbb{R}^n_{>0}$. Formally, with

$$\Lambda'' := \{\mathcal{L}(W) + X \mid X \in \operatorname{diag}(\mathbb{R}^n),\ \mathcal{L}(W) + X \in \mathcal{L}(\mathbb{W}_{\ominus})\}$$

it holds that $\Lambda = \Lambda''$, where Λ as defined in Lemma 2.5.10.

Proof. $\Lambda \subseteq \Lambda''$: this direction is clear by Lemma 2.3.9. $\Lambda'' \subseteq \Lambda$: choose any $\mathcal{L}(W) + X \in \Lambda''$. With $Y := X + I$ we get that $\mathcal{L}(W) + X = Y - \tilde{W}_{\mathbf{1}} = \mathcal{L}(A) =: [\ell^A_{ij}]$ for some $A \in \mathbb{W}_{\ominus}$. First, we show that this implies that $Y = \operatorname{diag}(y_1, \dots, y_n)$ has an all-positive diagonal. Fix any $i \in V$. From $A \in \mathbb{W}_{\ominus}$ and Fact 2.5.2 it follows that $\ell^A_{ii} \geq 0$, and from $W \in \mathbb{W}$ that $w_{ii} \geq 0$. Thus, we get from $\ell^A_{ii} = y_i - w_{ii}/d_i \geq 0$ that $y_i \geq w_{ii}/d_i \geq 0$. Thus y_i is non-negative. Now assume that $y_i = 0$. This implies that $w_{ii} = 0$, hence $\ell^A_{ii} = 0$, so i must be (by Fact 2.5.2) a separated vertex in $\mathcal{G}(A)$. In $\mathcal{G}(W)$, $w_{ii} = 0$ implies by the positive degree constraint that $w_{ij} > 0$ for some $j \neq i$. For such j it holds that $-a_{ij}/\sqrt{d^A_i d^A_j} = \ell^A_{ij} = \ell_{ij} = -w_{ij}/\sqrt{d_i d_j} < 0$,

hence that $a_{ij} > 0$ in contradiction to i being a separated vertex in $\mathcal{G}(A)$. Therefore, $y_i > 0$ for all $i \in V$.

In the following we assume w.l.o.g. that W is connected, since all arguments can be applied to each connected component individually, independent of all other components.

Lemma 2.5.10 gives that for every all-positive diagonal matrix Y (in particular as chosen above) there exists some $\alpha > 0$ (to be determined later) plus some $\mathbf{h}$-adjustment $\overline{W}_{\mathbf{h}} =: M_\alpha \in \mathbb{W}_\ominus$ for some $\mathbf{h} \in \mathbb{R}^n_{>0}$ such that $\mathcal{L}(M_\alpha) = \alpha Y - \tilde{W}_{\mathbf{1}} \in \Lambda$. We now show by contradiction that for no $\beta \neq \alpha$ any other $M_\beta \in \mathbb{W}_\ominus$ of the form $\mathcal{L}(M_\beta) = \beta Y - \tilde{W}_{\mathbf{1}}$ exists: fix any $M_\beta \in \mathbb{W}_\ominus$ with $\mathcal{L}(M_\beta) = \beta Y - \tilde{W}_{\mathbf{1}}$ for some $\beta \in \mathbb{R}$. Setting $\varepsilon := \beta - \alpha$ gives that $\mathcal{L}(M_\beta) = \beta Y - \tilde{W}_{\mathbf{1}} = \mathcal{L}(M_\alpha) + \varepsilon Y$. Now assume that $\varepsilon > 0$. By Lemma 2.5.8, $\mathcal{L}(M_\beta)$ is positive semi-definite with $v := \sqrt{M_\beta \mathbf{1}}$ an all-positive eigenvector to the eigenvalue 0. We get the contradiction $0 = v^T \mathcal{L}(M_\beta) v = v^T \mathcal{L}(M_\alpha) v + \varepsilon v^T Y v > 0$, because $v^T \mathcal{L}(M_\alpha) v \geq 0$ by positive semi-definiteness of $\mathcal{L}(M_\alpha)$, and $\varepsilon v^T Y v > 0$ by all-positivity of v and Y. Now assume that $\varepsilon < 0$. Let $w := \sqrt{M_\alpha \mathbf{1}}$ denote the all-positive eigenvector of $\mathcal{L}(M_\alpha)$ to the eigenvalue 0. We get the contradiction $0 \leq w^T \mathcal{L}(M_\beta) w = w^T \mathcal{L}(M_\alpha) w + \varepsilon w^T Y w < 0$, because $w^T \mathcal{L}(M_\alpha) w = 0$, and $\varepsilon w^T Y w < 0$ by all-positivity of w and Y, and the first inequality due to positive semi-definiteness of $\mathcal{L}(M_\beta)$. Thus, no such $\beta \neq \alpha$ exists.

Now we get from $\mathcal{L}(A) = Y - \tilde{W}_{\mathbf{1}} = \mathcal{L}(M_1)$ that $\alpha = 1$ is the only possible solution of this form. This gives that $\mathcal{L}(A) = \mathcal{L}(M_1) \in \Lambda$. Further, we have from above that $M_1 = \overline{W}_{\mathbf{h}}$ is an $\mathbf{h}$-adjustment of W. From Lemma 2.5.9 it follows that $A = a \cdot M_1$ for some $a > 0$, which gives by linearity (Proposition 2.5.6) that $A = \overline{W}_{a\mathbf{h}}$ is an $\mathbf{f}$-adjustment of W for $\mathbf{f} := a\mathbf{h}$. □

Now we have all ingredients to prove our main result on the algebraic interpretation of $\mathbf{f}$-adjusting:

Lemma 2.5.12 (Complete characterization for connected graphs)
For $n > 1$ and connected $\mathcal{G}(W) \in \mathcal{G}(\mathbb{W})$ consider all solutions $(X, A, c) \in \operatorname{diag}(\mathbb{R}^n) \times \mathbb{W}_\ominus \times \mathbb{R}$ of the equation

$$\mathcal{L}(W) + X = c \cdot \mathcal{L}(A).$$

For $c \leq 0$ no solution exists. For $c > 0$, all solutions are given by $A = \overline{W}_{\mathbf{f},c}$ and $X + I = \tilde{D}_{\mathbf{f}} F^{-1} = Z$ for any choice of $\mathbf{f} \in \mathbb{R}^n_{>0}$. This is equivalent to choosing any $Z \in \operatorname{diag}(\mathbb{R}^n_{>0})$ with Perron root $\rho(Z^{-1}\tilde{W}_{\mathbf{1}}) = 1$, which determines $\sqrt{\mathbf{f}}$ as the unique (up to scaling) right Perron eigenvector.

Proof. Let $\mathcal{L}(W) =: [\ell_{ij}]$ and $\mathcal{L}(A) =: [\ell^A_{ij}]$. W is non-loops-only, so there exist $i \neq j$ with $w_{ij} > 0$, hence $\ell_{ij} < 0$. The case $c < 0$ would imply that $\ell^A_{ij} > 0$, which is impossible for all $A \in \mathbb{W}_\ominus$. The case $c = 0$ would imply that $\mathcal{L}(W) + X = 0$, hence that all off-diagonal elements in W are zero, in contradiction to being non-loops-only. Thus, no solutions for $c \leq 0$ exist.

Now consider the case $c = 1$, that is any solution of the form $\mathcal{L}(W) + X = \mathcal{L}(A)$. We get from Lemma 2.5.11 that every such solution corresponds to fixing some $\mathbf{f} \in \mathbb{R}^n_{>0}$ and setting $A := \overline{W}_{\mathbf{f},1}$. This implies by Lemma 2.3.9 that $X + I = \tilde{D}_{\mathbf{f}} F^{-1}$. It remains to show that this is equivalent to choosing $Z \in \operatorname{diag}(\mathbb{R}^n_{>0})$ with the desired properties. For fixed $\mathbf{f}$, we get from Lemma 2.5.10 that $\tilde{D}_{\mathbf{f}} F^{-1} = X + I = Z$ for some $Z \in \operatorname{diag}(\mathbb{R}^n_{>0})$ with Perron eigenvalue $\rho(Z^{-1}\tilde{W}_{\mathbf{1}}) = 1$ and $\sqrt{\mathbf{f}}$ the corresponding right Perron eigenvector. The other way round, Lemma 2.5.10 gives that choosing any $Z \in \operatorname{diag}(\mathbb{R}^n_{>0})$ with $\rho(Z^{-1}\tilde{W}_{\mathbf{1}}) = 1$ implies by setting $X := Z - I$ that $\mathcal{L}(W) + X = \mathcal{L}(A)$ for $A = \overline{W}_{\mathbf{f},1}$, and further that $Z = \tilde{D}_{\mathbf{f}} F^{-1}$ with $\sqrt{\mathbf{f}}$ being determined as the unique (up to scaling) right Perron eigenvector.

Now consider the case $c > 0$, that is $\mathcal{L}(W) + X = c \cdot \mathcal{L}(A)$ for $X \in \operatorname{diag}(\mathbb{R}^n)$ and $A \in \mathbb{W}_{\ominus}$. From $c \cdot \mathcal{L}(A) \overset{2.5.7}{=} \mathcal{L}(A^{\circ}_{c^{-1}A\mathbf{1}})$ we get that $\mathcal{L}(W) + X$ equals the normalized Laplacian of a weak graph matrix. Thus, $\mathcal{L}(W) + X \overset{2.5.11}{=} \mathcal{L}(\overline{W}_{\mathbf{g},1})$ for some $\mathbf{g} \in \mathbb{R}^n_{>0}$. This gives that $\mathcal{L}(A) = c^{-1}\mathcal{L}(\overline{W}_{\mathbf{g},1}) \overset{2.3.7}{=} \mathcal{L}(\overline{W}_{\mathbf{g},c})$. Thus we have that $A \overset{2.5.9}{=} \alpha \cdot \overline{W}_{\mathbf{g},c} \overset{2.5.6}{=} \overline{W}_{\alpha\mathbf{g},c}$ for some $\alpha > 0$. So A is the $(\mathbf{f}, c)$-adjustment of W for $\mathbf{f} := \alpha\mathbf{g}$. It follows as before that $X + I = \tilde{D}_{\mathbf{g}} \operatorname{diag}(\mathbf{g})^{-1} = \tilde{D}_{\mathbf{f}} F^{-1} = Z$ for some $Z \in \operatorname{diag}(\mathbb{R}^n_{>0})$ with Perron eigenvalue $\rho(Z^{-1}\tilde{W}_{\mathbf{1}}) = 1$ and $\sqrt{\mathbf{g}}$ a corresponding right Perron eigenvector as well as $\sqrt{\mathbf{f}} = \sqrt{\alpha\mathbf{g}}$ another one, unique up to scaling.

For the other way round, choose any $\mathbf{f} \in \mathbb{R}^n_{>0}$ (or equivalently any $Z \in \operatorname{diag}(\mathbb{R}^n_{>0})$). Setting $A := \overline{W}_{\mathbf{f},c}$ implies that $\mathcal{L}(\overline{W}_{\mathbf{f},1}) \overset{2.3.7}{=} c \cdot \mathcal{L}(A) = \mathcal{L}(W) + X$, hence by Lemma 2.3.9 that $X + I = \tilde{D}_{\mathbf{f}} F^{-1}$. □

We are now ready to prove Theorem 2.4.3 and Theorem 2.4.4.

Proof of Theorem 2.4.3. For $n = 1$ it holds that $\mathcal{L}(W) = 0$, and $\mathcal{L}(A) = 0$ for every $A \in \mathbb{W}_{\ominus}$. Further, $\mathbf{f}$-scaling yields $\tilde{D}_{\mathbf{f}} = F$ for every $\mathbf{f} \in \mathbb{R}^n_{>0}$, hence $X = 0$. Thus, the above equation is satisfied for every $\mathbf{f}$-adjustment of W, while for every choice $X \neq 0$ it holds that $\mathcal{L}(W) + X \notin \mathcal{L}(\mathbb{W}_{\ominus})$. For a connected graph $\mathcal{G}(W)$ with $n > 1$ vertices, the stated equivalence follows directly from Lemma 2.5.12 for $c = 1$. Now W consists of any number of connected components, where W_S, A_S, $\mathbf{f}_S$ and X_S refer to the respective restriction to only the vertices of component $S \subseteq V$. Note that $\overline{W}_{\mathbf{f}}$ is an $\mathbf{f}$-adjustment of W if and only if $\overline{(W_S)}_{\mathbf{f}_S}$ is an $\mathbf{f}_S$-adjustment of W_S for every component S individually. Particularly, $\mathcal{L}(W) + X = \mathcal{L}(A)$ if and only if $\mathcal{L}(W_S) + X_S = \mathcal{L}(A_S)$ individually for each component S. Thus, the stated equivalence follows by applying Lemma 2.5.12 for $c = 1$ individually to each component S. □

Proof of Theorem 2.4.4. For $n = 1$ it holds that $\mathcal{L}(W) = 0$ and further that $\rho(M) = 1$ if and only if $Z = 1$. That is, $Z = I$ is the only possible diagonal that is shared by all $\mathbf{f}$-adjustments of W, and for all $Z \neq 1$ we get that $0 \neq Z - \tilde{W}_{\mathbf{1}} \neq \mathcal{L}(\mathbb{W}_{\ominus})$. For $n > 1$, all results follow immediately by reformulating the corresponding statements in Lemma 2.5.12 for $c = 1$ and from the properties of the Perron root and the Perron eigenvectors. □

2.5.3 Details on the proof of the geometric interpretation

This section provides details on how **f**-adjusting affects the interspace quantities in terms of their continuous limit quantities, as made precise in Theorem 2.4.1. A rigorous proof of this theorem would take more than a dozen pages that are almost a copy of the work by Maier et al. (2009). For that reason, I solely present the intuitive argument rather than a formal proof here. The only difference to the original work is to consider the new degrees $f(\mathbf{x}_i)$ in place of d_i and the new edge weights $w_{ij} \cdot \sqrt{f(\mathbf{x}_i)f(\mathbf{x}_j)}/\sqrt{d_i d_j}$ in place of w_{ij} everywhere along their proofs. This introduces the function $f : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ as a new function-valued parameter that can be chosen by us in any way as long as it satisfies the same constraints that are put on p.

Let us render the overall argument slightly more precise. Recall the definitions of the interspace quantities:

$$\mathrm{vol}_{\mathbf{d}}(A) = \sum_{\mathbf{x}_i \in A} d_i \qquad \text{and} \qquad \mathrm{cut}_W(S) = \sum_{\mathbf{x}_i \in H^-, \mathbf{x}_j \in H^+} w_{ij} \quad .$$

As shown in Theorem 2.3.3, these interspace quantities converge, up to a scaling factor, to the continuous quantities $\mathrm{vol}_{p^2}(A) := \int_A p^2(\mathbf{x})\, d\mathbf{x}$ and $\mathrm{cut}_{p^2}(S) := \int_S p^2(\mathbf{x})\, d\mathbf{x}$, respectively. We refer to these integrals also by the simpler notation $\int_A p^2$ and $\int_S p^2$.

Now let $\mathbf{f} := (f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n))$ denote the vector of evaluations of some function f at all sample points. The **f**-adjusted graph provides the following interspace quantities:

$$\mathrm{vol}_{\mathbf{f}}(A) = \sum_{\mathbf{x}_i \in A} f(\mathbf{x}_i) \qquad \text{and} \qquad \mathrm{cut}_{\overline{W}_{\mathbf{f}}}(S) = \sum_{\mathbf{x}_i \in H^-, \mathbf{x}_j \in H^+} w_{ij} \cdot \frac{\sqrt{f(\mathbf{x}_i)f(\mathbf{x}_j)}}{\sqrt{d_i d_j}} \quad .$$

This holds true for any function f, but in order to achieve limit results in the following, we have to put the same regularity conditions on f as on p, see Definition 2.3.2. For the volumes, recall that one factor p in the limit quantity $\int_A p^2$ is implied by summing over the vertex degrees d_i. Since the new vertex degrees equal $f(\mathbf{x}_i)$, which corresponds to f instead of p, we achieve the limit quantity $\int_A fp$ for the interspace volumes of the **f**-adjusted graph. This explains that $c_n \cdot \mathrm{vol}_{\mathbf{f}_n}(H) \xrightarrow{a.s.} \mathrm{vol}_{f \cdot p}(H)$ in Theorem 2.4.1.

For the cut weights, recall that the original weights w_{ij} imply the limit quantity $\int_S p^2$. In the **f**-adjusted graph, every edge is re-weighted to $w_{ij} \cdot \sqrt{f(\mathbf{x}_i)f(\mathbf{x}_j)}/\sqrt{d_i d_j}$. Since the neighborhood radii and bandwidths shrink with $n \to \infty$, we get that every edge ij of significant weight implies that the distance between the corresponding sample points $\mathbf{x}_i$ and $\mathbf{x}_j$ decreases to 0. In particular, we get from the regularity conditions on p that $\sqrt{d_i d_j} \to d_i \approx d_j$, and similarly for f that $\sqrt{f(\mathbf{x}_i)f(\mathbf{x}_j)} \to f(\mathbf{x}_i) \approx f(\mathbf{x}_j)$. Thus, the additional factor on the edge weights, $\sqrt{f(\mathbf{x}_i)f(\mathbf{x}_j)}/\sqrt{d_i d_j}$, which is the geometric mean of $f(\mathbf{x}_i)/d_i$ and $f(\mathbf{x}_j)/d_j$, corresponds in the limit to f/p. Overall, this yields $\int_S p^2 \cdot f/p = \int_S fp$ as the continuous limit quantity of the cut weights after **f**-adjusting, which shows that $c'_n \cdot \mathrm{cut}_{\overline{W}_n}(S) \xrightarrow{a.s.} \mathrm{cut}_{f \cdot p}(S)$ in Theorem 2.4.1.

The way we apply Theorem 2.4.1 in the following is not by defining some function f on $\mathbb{R}^d$ explicitly, but by defining $f(\mathbf{x}_i)$ from the degrees d_i of an RGNG, in particular by $f(\mathbf{x}_i) := d_i^q$ for some $q \in \mathbb{R}$. Because of $d_i c_n \xrightarrow{a.s.} p(\mathbf{x}_i)$, this implicitly refers to the function $f = p^q$ in space. Revisit Corollary 2.4.2 for a table of meaningful choices of q.

2.6 Applications

This section highlights some applications of the **f**-adjusting technique.

2.6.1 Estimating volumes and cut weights of a density

We know from Section 2.3.2 that the vertex degrees in an RGNG provide a density estimate of the sampling density p. In contrast to that, Section 2.3.4 introduces the "anomaly" that volumes and cut weights in an RGNG do *not* refer to the sampling density p itself, but to p^2. Figure 2.6.1 visualizes this anomaly. The plots show that whenever volumes and cuts of the original graph are considered (that is for $\mathbf{f} = \mathbf{d}$), these indeed refer to p^2 rather than to p. Figure 2.6.1 further visualizes that this anomaly can be resolved by considering the $\mathbf{1}$-adjusted graph, whose volumes and cut weights indeed refer to the sampling density p. Even further, the plots show all the correspondences between $\mathbf{f} := \mathbf{d}^q$ and p^{q+1} for $q = -1, 0, 1, 2$, just as proven by the geometric interpretation for the large sample limit.

As a result, whenever we are interested in volumes and cuts of a sampling density p, we must *not* consider the original RGNG, but the $\mathbf{1}$-adjusted graph instead.

2.6.2 f-adjusted spectral clustering

Let $G = (V, E, W)$ denote an RGNG that is built on a sample drawn from density p. Recall that `2-SpectralClustering` (Algorithm 1.3.2) applied to $\mathcal{L}_{rw}(W)$ or $\mathcal{L}(W)$ focuses on minimizing the NCut criterion (1.9). Since NCut is determined by cut weights and volumes of G, it is affected by the anomaly from Section 2.3.4. As a consequence, the normalized cut that is approximated by `2-SpectralClustering` does *not* refer to the original sampling density p, but to p^2 instead! This is a fundamental but not widely known bias in spectral clustering that is inherent to all applications in the literature. The same bias affects the multi-vector approach `NormalizedSpectralClustering` (Algorithm 1.3.3) and recursive spectral clustering.

In light of the geometric interpretation of **f**-adjusting, we can expect that by spectral clustering applied to $\mathcal{L}_{\mathbf{f}}(W)$ the normalized cut will refer to another density, depending on how $\mathbf{f}$ is chosen from the table in Corollary 2.4.2. In particular, for $\mathbf{f} = \mathbf{1}$ we can expect to solve the anomaly by referring to the original sampling density p rather than to p^2.

Before we can approve this as a valid strategy, we have to take a final hurdle: the standard normalized Laplacian $\mathcal{L}(W)$ is only defined for graphs of positive edge weights, that is for $W \in \mathbb{W}$, while the **f**-adjusted Laplacian $\mathcal{L}_{\mathbf{f}}(W)$ may refer to some graph with negative

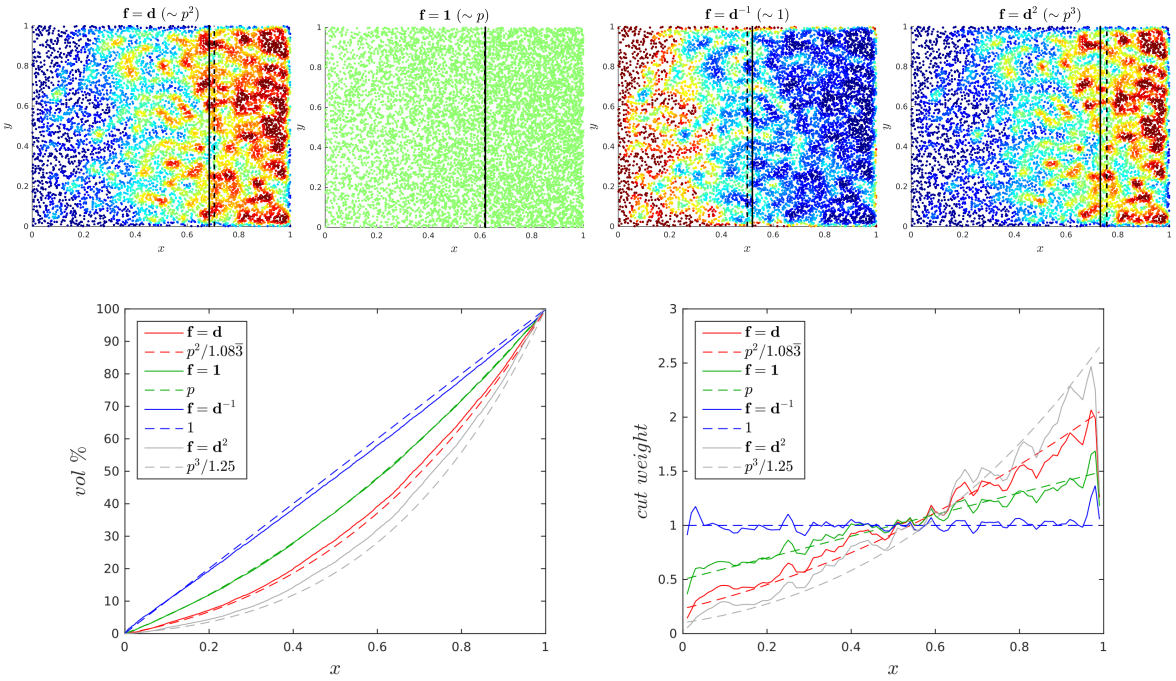


Figure 2.6.1: Resolving the anomaly via $\mathbf{f}$-adjusting. (top row) four times the same sample of size $n = 10000$ from density $p : [0,1]^2 \to \mathbb{R}$, $p(x,y) = 0.5 + x$. This is the same setting as in Figure 2.3.1, which further defines $\mathbf{d}$ as the degree vector of the weighted kNN-graph. All four heat plots show the degree vector $\mathbf{f}$ of the $\mathbf{f}$-adjusted graph for $\mathbf{f} \in \{\mathbf{d}, \mathbf{1}, \mathbf{d}^{-1}, \mathbf{d}^2\}$, respectively. Solid vertical lines mark the x-coordinates at which the total interspace volume $\mathrm{vol}_{\mathbf{f}}$ is split into halves. These empirical split points fit well to the analytical split points, which are represented by the dashed lines that halve the space volume $\mathrm{vol}_{\overline{p}}$ of the corresponding density function $\overline{p} \in \{p^2/1.08\overline{3},\ p,\ 1,\ p^3/1.25\}$. (bottom left) interspace/space volumes of the set $A(x) := [0,x] \times [0,1] \subseteq [0,1]^2$ as percentage of the total interspace/space volume. Precisely, $\mathrm{vol}_{\mathbf{f}}(A(x))/\,\mathrm{vol}_{\mathbf{f}}(A(1))$ and $\mathrm{vol}_{\overline{p}}(A(x))/\,\mathrm{vol}_{\overline{p}}(A(1))$. (bottom right) interspace/space cut weights for cutting at the line $S(x) := x \times [0,1] \subseteq [0,1]^2$. Precisely, $c \cdot \mathrm{vol}_{W_{\mathbf{f}}}(S(x))$ and $\mathrm{vol}_{\overline{p}}(S(x))$, where each scaling constant c is determined by least-squares minimization. For both, volumes and cuts, the empirical curves as determined by the $\mathbf{f}$-adjusted graph follow their analytical counterparts as determined by the corresponding density function $\overline{p}$. The slight non-optimal biases arise mainly from the boundary effect at $x = 1$, which abruptly cuts off the density landscape at its maximum.

selfloops, $\overline{W}_{\mathbf{f}} \in \mathbb{W}_{\ominus} \setminus \mathbb{W}$. We have to ensure that the interpretation of a normalized cut minimization in terms of the Rayleigh quotient characterization is preserved when generalizing from $\mathcal{L}(W)$ to $\mathcal{L}_{\mathbf{f}}(W)$. Fortunately, this is easy to see from the findings in Section 2.3.7, in particular from Lemma 2.3.7. The key is that for every $\overline{W}_{\mathbf{f}} \in \mathbb{W}_{\ominus}$ there exists some $c > 0$ such that $\overline{W}_{\mathbf{f},c} \in \mathbb{W}$. Both graph matrices show exactly the same cut weights for all cuts, and they show the same volumes up to the scaling factor c. So the NCut optimization problem of $\overline{W}_{\mathbf{f}}$ is the same as that of $\overline{W}_{\mathbf{f},c}$ multiplied by the constant c. They have particularly the same minimizer and refer to the same density. From $\mathcal{L}(\overline{W}_{\mathbf{f}}) = c \cdot \mathcal{L}(\overline{W}_{\mathbf{f},c})$ we further get that their Laplacian matrices have the same spectral properties up to scaling all eigenvalues by c. As a consequence, spectral clustering by $\mathcal{L}_{\mathbf{f}}(W)$ is essentially the same as spectral clustering by $\mathcal{L}(\overline{W}_{\mathbf{f},c})$ for $c \geq c^+$. The latter is standard spectral clustering, and it approximates the minimum normalized cut with respect to the modified density that is implied by $\mathbf{f}$.

After taking this hurdle, we can now introduce **f-adjusted spectral clustering** as a generalization of `2-SpectralClustering` and `NormalizedSpectralClustering`. In both cases, the only difference is to consider the $F^{-1/2}$-scaled eigenvectors of $\mathcal{L}_{\mathbf{f}}(W)$ in place of $\mathbf{v}_1, \mathbf{v}_2, \ldots$ within the algorithms. Precisely, we consider the entries in the vectors $F^{-1/2}\mathbf{v}_1^{\mathbf{f}}$, $F^{-1/2}\mathbf{v}_2^{\mathbf{f}}, \ldots$, where $\mathbf{v}_i^{\mathbf{f}}$ denotes the i'th-smallest eigenvector of $\mathcal{L}_{\mathbf{f}}(W)$. This particularly defines the **f-adjusted NCut score vector** as $F^{-1/2}\mathbf{v}_2^{\mathbf{f}}$.

In this way, we can indeed solve the anomaly in spectral clustering: replace $\mathcal{L}(W)$ by the $\mathbf{1}$-adjusted Laplacian $\mathcal{L}_{\mathbf{1}}(W)$, where the final eigenvector-scaling by $\sqrt{I^{-1}} = I$ can even be omitted. However, although $\mathbf{1}$-adjusting is "correct" in the sense that it allows to study the density p rather than p^2, it is important to note that squaring p can be *beneficial* for the goal of determining good clusters, since for all $q > 0$ the function p^{q+1} intensifies high-density areas stronger than low-density areas. This amplification is to a certain extent able to sharpen cluster boundaries. The advantage of $\mathbf{f}$-adjusting is that we are no longer restricted only to squaring: we can approximate the normalized cut of the modified density p^r for every $r \in \mathbb{R}$ by applying standard spectral clustering to the $\mathbf{d}^{r-1}$-adjusted Laplacian.

2.6.3 Correcting for a known sampling bias

Consider a data set in which some parts are known to be over- or underrepresented. For example, a survey among shop customers in the morning will particularly have the ages 20 to 30 being underrepresented. How can we correct for this in a graph-based learning scenario? Let $p = b \cdot \overline{p}$ denote the erroneous density that we can access under some bias $b : \mathbb{R}^d \to \mathbb{R}_{>0}$ and $\overline{p}$ the true density that we cannot access. By $\mathbf{f}$-adjusting, one can use any estimate of b in order to compensate for the bias. This can be achieved by the $\mathbf{f}$-adjusted graph for $\mathbf{f} = \mathbf{d}\mathbf{b}^{-2} := (d_i/b(\mathbf{x}_i)^2)_i$. It provides volumes and cut weights just as if the sample points were drawn from $\overline{p}$ instead of p. We only need to estimate the bias $b(\mathbf{x}_i)$ at the locations of the sample points without requiring to know $\mathbf{x}_i$ itself. Such an estimate can be given by external domain knowledge. In our above example, we could estimate the bias as $b(\mathbf{x}_i) := 1 - \alpha \cdot \exp(-(25 - \text{age}_i)^2/(2\sigma^2))$ for parameters $\alpha \in [0, 1)$ and $\sigma > 0$ that reflect the

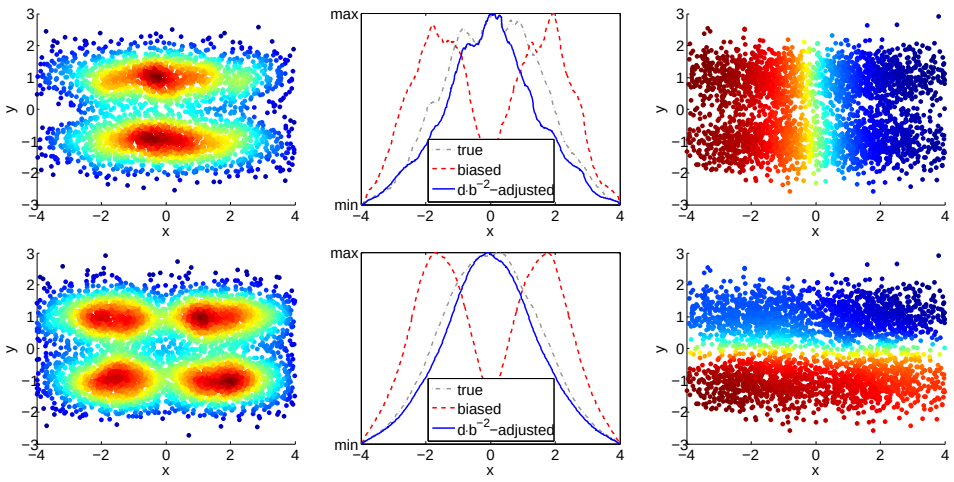


Figure 2.6.2: Correcting for a known sampling bias. (top left) exemplary sample from some true density $\overline{p}$ that we cannot access. Heat colors indicate the vertex degrees of an RGNG built from this sample. (bottom left) 3000 sample points from the biased density $p = b \cdot \overline{p}$ that underrepresents around $x \approx 0$. This is the sample we can access. (top middle) cut weights of vertical cuts at $x \in [-4, 4]$, for a single graph drawn from each the true and the biased density, and for the $\mathbf{db}^{-2}$-adjusted graph. (bottom middle) same, averaged over 20 runs. (top right) heat plot of the NCut score vector from spectral clustering of the biased graph, giving the wrong vertical cut. (bottom right) heat plot of the NCut score vector of the $\mathbf{db}^{-2}$-adjusted graph. Although it is built on the biased sample, the correct clusters are identified, since the bias is removed by $\mathbf{f}$-adjusting.

strength and the bandwidth of the underrepresentation, respectively. By $\mathbf{f}$-adjusting, we can "merge" anti-bias information given at the vertices into volumes and cut weights.

Consider the example in Figure 2.6.2. It shows two Gaussians (top left) that are underrepresented around $x \approx 0$ due to bias $b(x, y) = \min\{1, (2 + 3|x|)/8\}$. The minimum normalized cut of an RGNG built on the biased sample (bottom left) is misdirected by this bias to the wrong vertical clusters (top right). The $\mathbf{f}$-adjustment appropriately "repairs" the volumes and cut weights in the graph. Now the correct horizontal cut is revealed (bottom right).

2.6.4 Removing any non-uniform sampling bias

We now revisit the example from the introduction to this chapter, which discusses the scenario of clustering an intensity landscape. More precise, we are interested in studying the shape of a continuous intensity function that can only be accessed at the locations of a finite set of sample points. The sampling distribution itself does *not* provide information that we are interested in. Even worse, non-uniformly distributed sample points can totally misguide the output of clustering algorithms, as demonstrated again in Figure 2.6.3. However, spectral clustering works well on uniformly distributed sample points. This is exactly where $\mathbf{f}$-adjusting comes in: can we modify the graph's edge weights such that volumes and cut weights behave as if the sample points were uniformly distributed, in order to make spectral clustering of intensity

Algorithm: `UnbiasedSpectralIntensityClustering`

Input:

- $\text{sim}_s(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j = 1, \ldots, n$ the spatial similarity between i and j
- $\text{sim}_v(t_i, t_j)$ for $i, j = 1, \ldots, n$ the intensity value similarity between i and j

Output:

- $\mathbf{s} \in \mathbb{R}^n$ vector of clustering scores to be further processed

Algorithm:

1. define graph G_s by considering the spatial similarities $\text{sim}_s(\mathbf{x}_i, \mathbf{x}_j)$ as edge weights
2. with $\mathbf{d}$ the vector of vertex degrees in G_s, create the $\mathbf{d}^{-1}$-adjusted graph $\overline{G}_s$. It provides edge weights $\overline{w}_{ij}$ that remove the non-uniform sampling density information from the volumes and cut weights. $\overline{G}_s$ appears as if its volumes and cut weights were based on uniformly distributed sample points.
3. define the graph G from $\overline{G}_s$ by multiplying all edge weights with the corresponding intensity value similarities: $w_{ij} := \overline{w}_{ij} \cdot \text{sim}_v(t_i, t_j)$
4. return $\mathbf{s}$ as defined by the NCut score vector of $W = [w_{ij}]$

Algorithm 2.6.3: `UnbiasedSpectralIntensityClustering` algorithm

landscapes robust against non-uniform sampling distributions? The modified graph will still be defined on the same set of non-uniformly distributed sample points, just with different edge weights than before. Indeed this approach is possible, and it takes only the spatial similarities sim_s and the intensity value similarities sim_v as its input, without requiring access to any sampling coordinates. Precisely, we suggest to proceed as listed in Algorithm 2.6.3, denoted as **Unbiased Spectral Intensity Clustering**. The unbiased approach removes the influence of the non-uniform spatial distribution from the final edge weights, and focuses solely on the intensity values. In terms of Equation 2.1 on page 41, the algorithm replaces the spatial similarities $\text{sim}_s(\mathbf{x}_i, \mathbf{x}_j)$ with $\overline{w}_{ij}$ as given from their $\mathbf{d}^{-1}$-adjustment. Figure 2.6.4 shows that this leads to the correct segmentation, whereas the biased segmentation that is provided by the original graph is not correct.

2.6.5 Further applications

This section briefly presents some further applications of $\mathbf{f}$-adjusting. Details on these applications can be found in the supplementary material of Kurras et al. (2014).

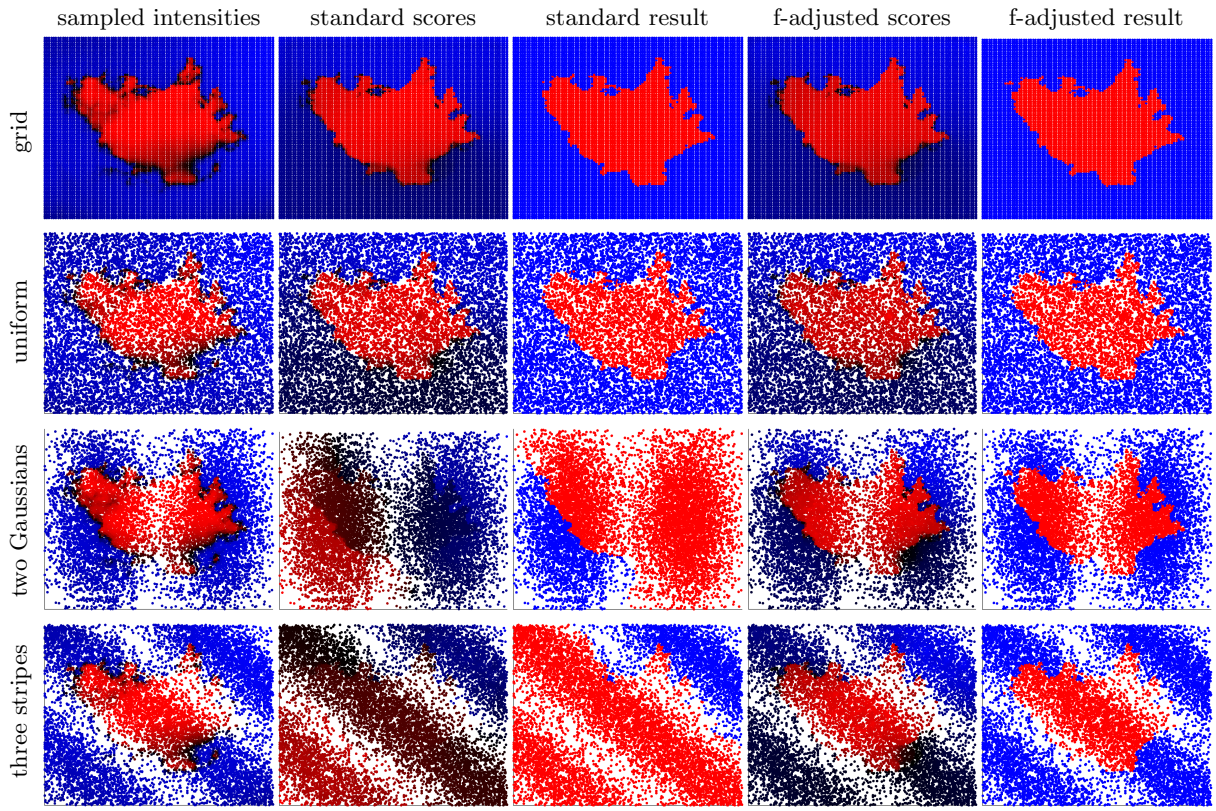


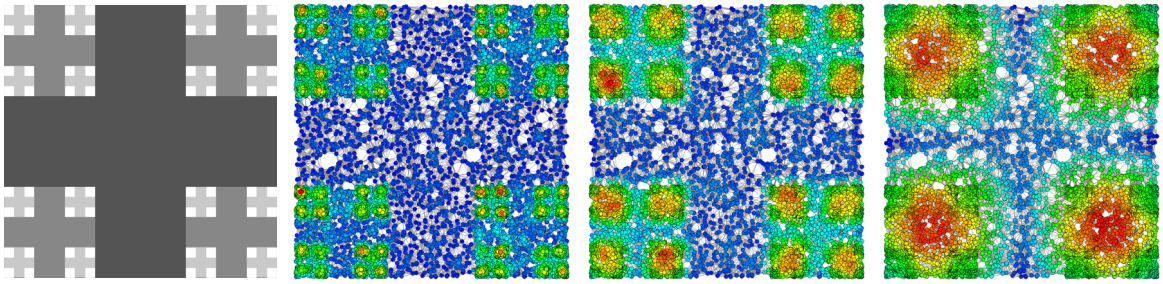
Figure 2.6.4: **f**-adjusted spectral clustering of an intensity landscape. Same setting as in Figure 2.1.1. The rows correspond to four different sampling schemes: grid sample, uniform density, two Gaussians density, stripe-shaped density. The first column shows the sample, colors indicating the intensity value. The second and the fourth column indicate by color the entries in the NCut score vector of the original graph matrix (2.1) and of the **f**-adjusted graph matrix (vector **s** in `UnbiasedSpectralIntensityClustering`), respectively. The third and the fifth column show the clustering result as derived from the respective NCut score vector via thresholding at zero. For non-uniform random samples, the clustering results derived from the original graph matrix are misguided to a wrong segmentation. The clustering results achieved by `UnbiasedSpectralIntensityClustering` successfully reconstruct the contour in all cases because the spatial bias is removed.

Biased random walks. Zlatić et al. (2010) study what they call "vertex centered biased random walks", which is a variant of the classical Markov random walk of a graph. Based on our analysis, we can give an intuitive explanation of their approach: their new random walk corresponds to **f**-adjusting for $\mathbf{f} = (f_i) := (d_i \cdot \exp(2\beta \cdot d_i/d_{max}))$ with $d_{max} := \max_{i \in V}\{d_i\}$ and strength parameter $\beta \geq 0$. According to our geometric interpretation, they implicitly study the modified density $\tilde{p}(x) = \exp(2\beta \cdot p(x)/p_{max})p(x)^2$. Hence, they amplify high-density regions in space exponentially stronger than low-density regions, which strongly intensifies the existing cluster structure, up to a certain extent.

Merging vertex weights into edge weights. Some graph-based applications provide for each vertex i a positive weight z_i that quantifies some sense of "importance" of this vertex. However, many graph algorithms ignore such external vertex weights and focus only on edge weights — so does the famous label propagation algorithm by Zhu and Ghahramani (2002). In order to make vertex weights visible to such algorithms, we have to transform the vertex weights into edge weights. Our framework suggests to achieve this via **f**-adjusting with $\mathbf{f} = (f_i) = (d_i \cdot z_i)$ in order to "merge" the additional vertex weight information into edge weights in a meaningful way.

Semi-supervised learning: label propagation limit behavior. If only very few vertices are labeled in a semi-supervised learning scenario (for example a constant number of labeled vertices while $n \to \infty$), then the soft labels provided by the label propagation algorithm tend to converge to a single global value, with thin spikes only at the fixed labeled vertices. In this situation, a meaningful class-separating threshold is increasingly difficult to find (Nadler et al., 2009). We suggest to avoid this issue by slightly "accelerating" the corresponding random walk toward labeled nodes. That is, we inform all vertices about their distance to the nearest labeled vertex, for example by assigning the vertex weight $z_i := \exp(-hd(i, V_\ell))$ with $hd(i, V_\ell)$ the shortest hop-distance from i to any labeled vertex (other distance measures between vertices can capture other structural information). These vertex weights are then represented as edge weights via **f**-adjusting with $\mathbf{f} = (f_i) = (d_i \cdot z_i)$. Experiments show that this approach can keep a strong class separation in the soft labels intact.

Multi-scale analysis of graphs. The **f**-adjustment technique can also be used to study a geometric graph G at multiple scales of granularity. Let $h > 0$ denote a scaling parameter and k_h some convolution kernel of bandwidth h that is applied to the density by convolution $p * k_h$. For each vertex i an estimate $\hat{q}_i$ of $(p * k_h)(\mathbf{x}_i)$ can be extracted from the graph by averaging over the vertex degrees in a neighborhood around i. Thus, the **f**-adjusted graph for $\mathbf{f} = (f_i) := (\hat{q}_i/d_i)$ achieves that its cuts and volumes represent $p * k_h$. This approach is visualized in Figure 2.6.5.



(a) (b) (c) (d)

Figure 2.6.5: Multi-scale analysis of graphs via **f**-adjusting. (a) a two-dimensional density p that shows a hierarchical structure. (b) heat plot of the vertex degrees of an RGNG built on a sample from p. Degrees, volumes and cut weights refer to a small scale. (c), (d) **f**-adjusting allows to study the density at larger scales by incorporating estimates of $p * k_h$ for a scalable kernel k_h of different bandwidths $h > 0$.

2.7 Chapter summary

This chapter introduces **f**-adjusting as a transformation of the edge weights of a given graph to new edge weights. At the first glance the definition may appear artificial, but I prove an *algebraic interpretation* of this graph modification that shows that **f**-adjusting is a "natural" transformation of a graph matrix: **f**-adjusting acts on the normalized Laplacian in just the same way as **f**-selflooping acts on the adjacency matrix. Precisely, I show that **f**-adjusting represents all diagonal modifications of $\mathcal{L}(W)$ of the form $\mathcal{L}(W) + X \in \mathcal{L}(\mathbb{W}_{\ominus})$, which is the normalized Laplacian's counterpart to the elementary fact that **f**-selflooping represents all diagonal modifications of W of the form $W + X \in \mathbb{W}_{\ominus}$. This is in line with the leitmotif of this thesis, since **f**-adjusting is a novel variant of the graph Laplacian.

I particularly contribute a *geometric interpretation*, which provides a plain intuition for the case that **f**-adjusting is applied to a random geometric neighborhood graph. Volumes and cut weights in the **f**-adjusted RGNG refer to a well-specified alternative sampling density $\overline{p}$ rather than to the original sampling density. The table in Corollary 2.4.2 serves as the main reference for how to interpret $\overline{p}$ depending on certain choices of **f**. The technical details of the proofs are mainly a modification of existing work, but I contribute the explicit distinction into three different views on a graph that help to interpret the results geometrically. The interspace view is a helpful bridge between the discrete graph view and the continuous view of density functions beneath.

I sketch various applications that show the variability and general applicability of this technique. In particular, I introduce **f**-adjusted spectral clustering as a generalization of spectral clustering. **f**-adjusted spectral clustering implies a couple of interesting special cases. For example, $\mathbf{f} = \mathbf{1}$ removes the inherent squaring bias that always underlies spectral clustering, $\mathbf{f} = \mathbf{d}^{-1}$ simulates a uniform sampling density on top of a non-uniform sample,

and $\mathbf{f} = \mathbf{d}\mathbf{b}^{-1}$ removes an explicitly given sampling bias $\mathbf{b}$ from the graph. Other choices of $\mathbf{f}$ allow for even more applications.

After all, it remains surprising that all the interesting consequences of $\mathbf{f}$-adjusting derive just from a certain modification of a matrix along its main diagonal.

Chapter 3

Symmetric iterative proportional fitting

3.1 Chapter introduction

Recall from the previous chapter that $\mathbf{f}$-scaling aims at transforming a given graph matrix $W \in \mathbb{W}$ of degree vector $\mathbf{d}$ into some other graph matrix of degree vector $\mathbf{f}$. However, the degree vector $\tilde{\mathbf{d}}$ that is actually attained by the $\mathbf{f}$-scaled graph matrix $\tilde{W}_{\mathbf{f}}$ does *not* equal $\mathbf{f}$ exactly, but only approximately: $\tilde{\mathbf{d}} \approx \mathbf{f}$. How to correct for the residuals $\mathbf{f} - \tilde{\mathbf{d}}$? The strategy taken by $\mathbf{f}$-adjusting is to subsequently apply $\mathbf{f}$-selflooping. This two-step strategy guarantees to attain degree vector $\mathbf{f}$ exactly by adding the residuals to the matrix $\tilde{W}_{\mathbf{f}}$ as additional selfloop weights. Since the resulting selfloop weights are potentially negative, the $\mathbf{f}$-adjusting matrix $\overline{W}_{\mathbf{f}}$ is from the set of weak graph matrices $\mathbb{W}_{\ominus} \supseteq \mathbb{W}$.

In this chapter we study another strategy to correct for the residuals: **iterated f-scaling**. As indicated by its name, the idea is to iteratively apply $\mathbf{f}$-scaling again and again in order to approximate the intended degree vector $\mathbf{f}$ better and better. We refer to $\mathbf{f}$-scaling as the function $s_{\mathbf{f}} : \mathbb{W} \to \mathbb{W}$, which is defined in accordance with Definition 2.3.5 as

$$s_{\mathbf{f}}(W) \;:=\; \tilde{W}_{\mathbf{f}} \;=\; \sqrt{F \cdot \operatorname{diag}(W\mathbf{1})^{-1}} \cdot W \cdot \sqrt{F \cdot \operatorname{diag}(W\mathbf{1})^{-1}} \quad .$$

Iterated $\mathbf{f}$-scaling yields the following recursively defined sequence $(W^{(k)})$ of matrices:

$$W^{(0)} \;:=\; W, \qquad W^{(k+1)} \;:=\; s_{\mathbf{f}}(W^{(k)}) \quad . \tag{3.1}$$

The expectation behind this approach is that the sequence (3.1) eventually converges to some limit matrix $W^{(\infty)} := \lim_{k\to\infty} W^{(k)}$ of degree vector $\mathbf{f} = W^{(\infty)}\mathbf{1}$. Moreover, since $\mathbf{f}$-scaling does neither add nor remove zero-entries, that is $E(W^{(k)}) = E(W)$ for all $k \geq 0$, we may further hope that the limit graph has the same set of edges $E(W^{(\infty)}) = E(W)$ as the original graph. Unfortunately, all these expectations are disproved. The sequence (3.1) can run into two subtle problems:

- in some cases $W^{(\infty)}$ does *not* provide degree vector $\mathbf{f}$, that is $W^{(\infty)}\mathbf{1} \neq \mathbf{f}$
- in some cases $W^{(\infty)}$ removes edges of G, that is $E(W^{(\infty)}) \subsetneq E(W)$

See Figure 3.1.1 for an example on both issues. Nevertheless, whenever the iterated $\mathbf{f}$-scaling approach is successful, it has an interesting advantage over $\mathbf{f}$-adjusting: the resulting matrix is from the set of graph matrices $\mathbb{W}$ rather than from its extension to weak graph matrices $\mathbb{W}_{\ominus}$, and no additional selfloops or other edges are added.

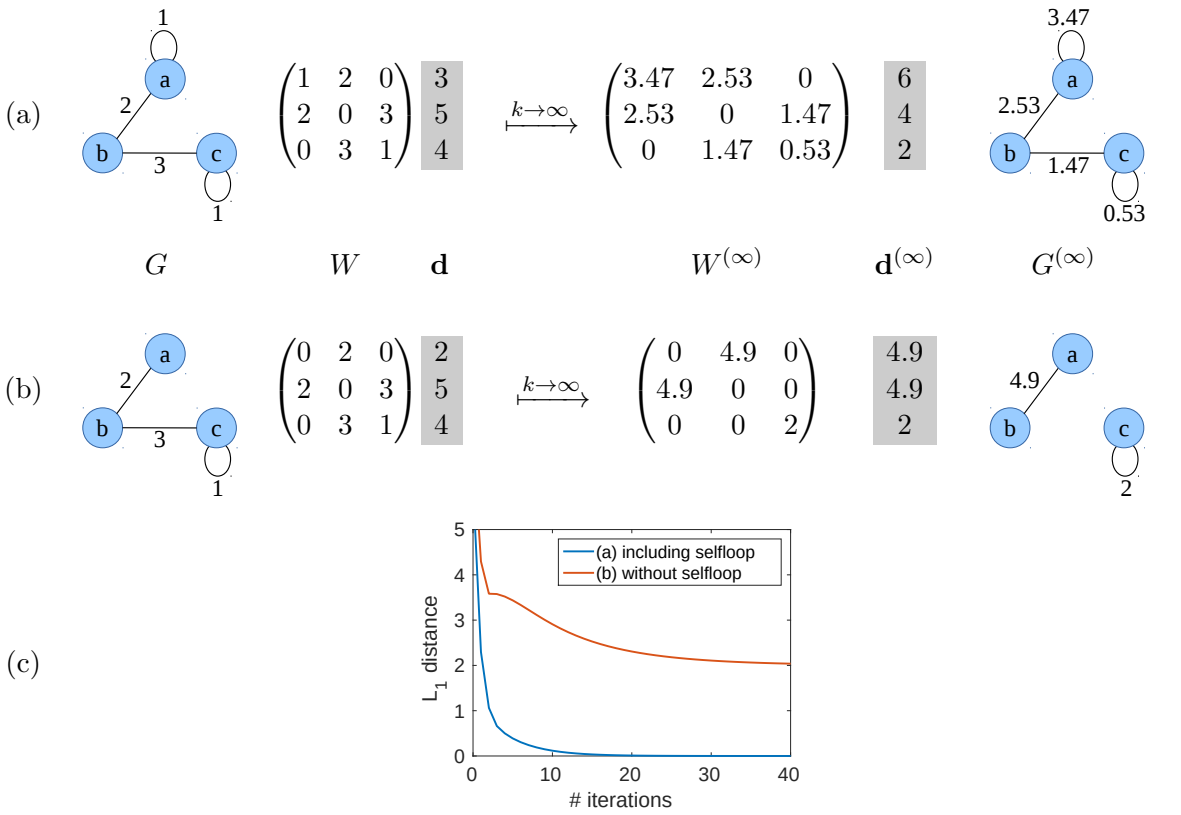


Figure 3.1.1: Example on iterated $\mathbf{f}$-scaling, for $\mathbf{f} = (6, 4, 2)^T$. (a) for this choice of G, the degree vector $\mathbf{d}^{(\infty)}$ of the limit graph matches the intended degree vector $\mathbf{f}$ exactly. (b) for the graph with the selfloop at vertex a removed, iterated $\mathbf{f}$-scaling does *not* converge to a limit graph of degree vector $\mathbf{f}$. Moreover, the limit graph has one edge less than the initial graph. (c) with $\mathbf{d}^{(k)} := W^{(k)}\mathbf{1}$, the plot shows the distance $\|\mathbf{f} - \mathbf{d}^{(k)}\|_1$ at iteration k for the examples in (a) and (b).

This chapter is concerned with the following questions:

Questions motivating this chapter

(**Q1**) What are necessary and sufficient conditions for $W^{(\infty)}\mathbf{1} = \mathbf{f}$ to hold true?

(**Q2**) What are necessary and sufficient conditions for $E(W^{(\infty)}) = E(W)$ to hold true?

(**Q3**) In which sense is $W^{(\infty)}$ related to W?

(**Q4**) Does iterated $\mathbf{f}$-scaling allow for applications in machine learning?

All four questions are answered in this chapter by contributing novel results. Studying the iterated $\mathbf{f}$-scaling matrix sequence (3.1) shows a surprisingly rich mathematical complexity and reveals deep connections to an active field of mathematical research known as **matrix scaling**. In particular, iterated $\mathbf{f}$-scaling represents a specific symmetrization of an existing approach known as **Iterative Proportional Fitting (IPF)**. For that reason, we refer to iterated $\mathbf{f}$-scaling synonymously by the term **Symmetric Iterative Proportional Fitting (SIPF)**. As another consequence, we refer to the terms row and column *sums* synonymously by the terms row and column *marginals*.

The fundamental question behind matrix scaling can be seen as a combinatorial "game" that we denote as **Fixed Marginals Matrix Game**. It has applications not only in machine learning, but also in network theory, optimal transportation, statistics, and matrix factorization. This game is defined as follows.

Fixed Marginals Matrix Game

Input: a non-negative, not necessarily symmetric matrix $W \in \mathbb{R}_{\geq 0}^{m \times n}$ together with two positive vectors $\mathbf{r} \in \mathbb{R}_{>0}^{m}$ and $\mathbf{c} \in \mathbb{R}_{>0}^{n}$
Goal: find any non-negative matrix $B \in \mathbb{R}_{\geq 0}^{m \times n}$ that provides row marginals $\mathbf{r}$ and column marginals $\mathbf{c}$, while preserving all zero-entries of W (that is $w_{ij} = 0 \Rightarrow b_{ij} = 0$).

In a variant of this game, one may further restrict B to have exactly the same zeros as W, that is $w_{ij} = 0 \Leftrightarrow b_{ij} = 0$. In case of multiple feasible solutions, one may additionally require to find a solution that is closest to W with respect to some distance function on matrices. However, it can also happen that no feasible solution exists at all, that is there exists no matrix of the desired marginals and zero-pattern. This is easy to see for $\|\mathbf{r}\|_1 \neq \|\mathbf{c}\|_1$, but this case can also be caused by more subtle combinatorial conditions. Figure 3.1.2 illustrates this game and its possible outcomes.

(a)
$$\mathrm{W} = \begin{pmatrix} 0 & 3 & 2 & 0 \\ 1 & 0 & 3 & 3 \\ 0 & 2 & 0 & 1 \end{pmatrix} \begin{matrix} 5 \\ 7 \\ 3 \end{matrix} \quad \overset{?}{- - \triangleright} \quad \mathrm{B} = \begin{matrix} \mathbf{r} \\ \\ \\ \mathbf{c} \end{matrix} \begin{pmatrix} 0 & * & * & 0 \\ * & 0 & * & * \\ 0 & * & 0 & * \\ c_1 & c_2 & c_3 & c_4 \end{pmatrix} \begin{matrix} r_1 \\ r_2 \\ r_3 \\ \\ \end{matrix}$$

(b)
$$\begin{matrix} \mathbf{r} \\ \mathbf{c} \end{matrix} \begin{matrix} = \\ = \end{matrix} \begin{matrix} (5,5,5) \\ (2,4,4,5) \end{matrix} \quad , \quad B_1 = \begin{pmatrix} 0 & 2 & 3 & 0 \\ 2 & 0 & 1 & 2 \\ 0 & 2 & 0 & 3 \end{pmatrix} \quad , \quad B_2 = \begin{pmatrix} 0 & 4 & 1 & 0 \\ 2 & 0 & 3 & 0 \\ 0 & 0 & 0 & 5 \end{pmatrix}$$

(c)
$$\begin{matrix} \mathbf{r} \\ \mathbf{c} \end{matrix} \begin{matrix} = \\ = \end{matrix} \begin{matrix} (5,5,5) \\ (4,2,7,2) \end{matrix} \quad , \quad \text{no solution exists}$$

Figure 3.1.2: Fixed marginals matrix game. (a) general goal: given W, $\mathbf{r}$ and $\mathbf{c}$, find another non-negative matrix B that provides row marginals $\mathbf{r}$ and column marginals $\mathbf{c}$, while preserving the zeros of W. (b) two of infinitely many solutions for given W, $\mathbf{r}$ and $\mathbf{c}$. B_1 provides exactly the same zeros as W. B_2 has two additional zero entries. (c) for this choice of $\mathbf{r}$ and $\mathbf{c}$ no solution exists. This can be seen from $c_2 + c_4 < r_3$.

3.2 Informal summary of the main contributions

Iterative Proportional Fitting (IPF) generates a sequence of matrices that is loosely related to iterated **f**-scaling. It is already known in the literature that IPF solves the Fixed Marginals Matrix Game whenever a feasible solution exists. Moreover, in case of multiple feasible solutions, IPF converges to the unique solution that is closest to W with respect to the so-called "relative-entropy error". These two results are proven in the literature by totally different and unrelated arguments. Section 3.3.9 covers more details on the background of IPF. My first main contribution (Section 3.4.1) is to render IPF as a special case of a general projection algorithm that brings these two separate results together. This contributes a novel single intuitive argument for the fact that IPF converges to the relative-entropy optimum.

The second main contribution (Section 3.4.2) tackles the problem that IPF does not fit well to symmetric matrices. In Section 3.3.7 I introduce Symmetric Iterative Proportional Fitting (SIPF) as an alternative to IPF for symmetric matrices. My main result is a convergence proof for the SIPF-sequence (3.1).

The third main contribution (Section 3.4.3) focuses on the case of symmetric matrices, too. It provides necessary and sufficient conditions for the existence of a feasible solution. The conditions are formulated in terms of graph properties of the graph $\mathcal{G}(W)$.

3.3 Formal setup

This section introduces all concepts and formalisms that are required to understand the main contributions in the follow-up section.

3.3.1 Six families of non-negative matrices

Throughout this chapter we deal with six different families of non-negative matrices, that is with six different subsets of $\mathbb{R}_{\geq 0}^{m \times n}$. In order to avoid some trivial cases we assume that $m, n \geq 2$. Each family is characterized by the following properties:

- either allowing for rectangular $m \times n$ matrices, or only for symmetric $n \times n$ matrices
- constraints on the row and/or column marginals
- constraints on the zero-pattern, that is on the set of positions where matrices of this family are allowed to have zero-entries.

We start by defining three families of matrices that are not restricted to be symmetric. The most general family consists of all non-negative $m \times n$ matrices that contain no zero row and no zero column, or stated slightly differently, the set of all non-negative $m \times n$ matrices that have arbitrary positive marginals:

$$\Omega \quad := \quad \{X \in \mathbb{R}_{\geq 0}^{m \times n} \mid X\mathbf{1} > 0,\ X^T\mathbf{1} > 0\} \tag{3.2}$$

The second family is parameterized by two positive vectors $\mathbf{r} \in \mathbb{R}_{>0}^m$, $\mathbf{c} \in \mathbb{R}_{>0}^n$ plus a matrix $W \in \Omega$. It is the restriction of Ω to only those matrices that exactly provide row marginals $\mathbf{r}$ and column marginals $\mathbf{c}$, while having *at least* the zeros of W.

$$\Omega(\mathbf{r}, \mathbf{c}, W) \quad := \quad \{X \in \mathbb{R}_{\geq 0}^{m \times n} \mid X\mathbf{1} = \mathbf{r},\ X^T\mathbf{1} = \mathbf{c},\ E(X) \subseteq E(W)\}, \tag{3.3}$$

The third family is similar to the second, but it is further restricted to having *exactly* the same zeros as W, that is

$$\underline{\Omega}(\mathbf{r}, \mathbf{c}, W) \quad := \quad \{X \in \mathbb{R}_{\geq 0}^{m \times n} \mid X\mathbf{1} = \mathbf{r},\ X^T\mathbf{1} = \mathbf{c},\ E(X) = E(W)\}. \tag{3.4}$$

Note that

$$\underline{\Omega}(\mathbf{r}, \mathbf{c}, W) \subseteq \Omega(\mathbf{r}, \mathbf{c}, W) \subseteq \Omega.$$

The remaining three families are strongly related to the three families already defined. They are derived from them simply by additionally restricting to symmetric matrices, where we assume $m = n$ and $\mathbf{r} = \mathbf{c}$, since otherwise this restriction is the empty set. Particularly, we

	rectangular $m \times n$		symmetric $n \times n$
any positive marginals	Ω	$\supseteq^{*)}$	$\mathbb{W}$
	$\cup\!\mid$		$\cup\!\mid$
fixed marginals & zeros *at least* those of W	$\Omega(\mathbf{r}, \mathbf{c}, W)$	$\supseteq^{*)}$	$\mathbb{W}(\mathbf{f}, W)$
	$\cup\!\mid$		$\cup\!\mid$
fixed marginals & zeros *exactly* those of W	$\underline{\Omega}(\mathbf{r}, \mathbf{c}, W)$	$\supseteq^{*)}$	$\underline{\mathbb{W}}(\mathbf{f}, W)$

Table 3.1: Six families of non-negative matrices for positive vectors $\mathbf{r}, \mathbf{c}, \mathbf{f}$, and $W \in \Omega$. The relations marked by *) require that $m = n$, $\mathbf{r} = \mathbf{c} = \mathbf{f}$ and $W \in \mathbb{W}$.

denote both marginals by the vector $\mathbf{f} := \mathbf{r} = \mathbf{c}$. The restriction of Ω to symmetric matrices yields precisely the set of graph matrices as already defined in Section 1.3.1,

$$\mathbb{W} \quad = \quad \{X \in \mathbb{R}^{n \times n}_{\geq 0} \mid X = X^T,\ X\mathbf{1} > 0\} \tag{3.5}$$

$\mathbb{W}$ contains all non-negative symmetric matrices that have arbitrary positive marginals. For a positive vector $\mathbf{f} \in \mathbb{R}^n_{>0}$ and any $W \in \mathbb{W}$ we define

$$\mathbb{W}(\mathbf{f}, W) \quad := \quad \{X \in \mathbb{R}^{n \times n}_{\geq 0} \mid X = X^T,\ X\mathbf{1} = \mathbf{f},\ E(X) \subseteq E(W)\}, \tag{3.6}$$

which consists of those matrices in $\mathbb{W}$ that have (row and column) marginals $\mathbf{f}$ while having *at least* the zeros of W. In graph language, the set $\mathbb{W}(\mathbf{f}, W)$ refers to all graphs that are defined on a subset of the edges of $\mathcal{G}(W)$ while their (positive) edge weights provide degree vector $\mathbf{f}$. Finally,

$$\underline{\mathbb{W}}(\mathbf{f}, W) \quad := \quad \{X \in \mathbb{W}(\mathbf{f}, W) \mid E(X) = E(W)\} \tag{3.7}$$

additionally restricts to providing exactly the zeros of W. In graph language, the set $\underline{\mathbb{W}}(\mathbf{f}, W)$ refers to all graphs that are defined on the *same* edges as $\mathcal{G}(W)$ while their (positive) edge weights provide degree vector $\mathbf{f}$. Note that

$$\underline{\mathbb{W}}(\mathbf{f}, W) \subseteq \mathbb{W}(\mathbf{f}, W) \subseteq \mathbb{W}$$

Sometimes we drop individual constraints by a dot, for example

$$\Omega(\mathbf{r}, \cdot, \cdot) = \{X \in \mathbb{R}^{m \times n}_{\geq 0} \mid X\mathbf{1} = \mathbf{r}\} \ (\not\subseteq \Omega) \quad .$$

Table 3.1 shows all six families together with their subset relations. The following lemma shows that either none or both of $\Omega(\mathbf{f}, \mathbf{f}, W)$ and $\mathbb{W}(\mathbf{f}, W)$ are empty.

Lemma 3.3.1 (Emptiness relations)
For $W \in \mathbb{W}$ and $\mathbf{f} \in \mathbb{R}^n_{>0}$ all following implications hold true:

$$\begin{array}{ccc} \Omega(\mathbf{f},\mathbf{f},W) = \emptyset & \Leftrightarrow & \mathbb{W}(\mathbf{f},W) = \emptyset \\ \Downarrow & & \Downarrow \\ \underline{\Omega}(\mathbf{f},\mathbf{f},W) = \emptyset & \Leftrightarrow & \underline{\mathbb{W}}(\mathbf{f},W) = \emptyset \end{array}$$

Proof. We first prove that $\Omega(\mathbf{f},\mathbf{f},W)$ and $\mathbb{W}(\mathbf{f},W)$ are either none or both empty. From the definitions we get that $\mathbb{W}(\mathbf{f},W) \subseteq \Omega(\mathbf{f},\mathbf{f},W)$, which shows $\mathbb{W}(\mathbf{f},W) \neq \emptyset \Rightarrow \Omega(\mathbf{f},\mathbf{f},W) \neq \emptyset$. For the other direction, choose any $A \in \Omega(\mathbf{f},\mathbf{f},W)$. This implies $(A + A^T)/2 \in \mathbb{W}(\mathbf{f},W)$. The same argument shows that $\underline{\Omega}(\mathbf{f},\mathbf{f},W)$ and $\underline{\mathbb{W}}(\mathbf{f},W)$ are either none or both empty. Finally, both vertical implications follow right from the respective subset relation. □

3.3.2 Matrix scaling

We call $B \in \Omega$ a **biproportional scaling** of $W \in \Omega$ if there exist any two sequences $(Y^{(k)})$ and $(Z^{(k)})$ of positive diagonal matrices $Y^{(k)}, Z^{(k)} \in \mathrm{diag}(\mathbb{R}^n_{>0})$ such that $B = \lim_{k\to\infty} Y^{(k)} W Z^{(k)}$. For example, $B = \left(\begin{smallmatrix} 2 & 0 \\ 0 & 3 \end{smallmatrix}\right)$ is a biproportional scaling of $W = \left(\begin{smallmatrix} 2 & 4 \\ 0 & 1 \end{smallmatrix}\right)$, because B can be expressed as the limit

$$\lim_{k\to\infty} \underbrace{\begin{pmatrix} 0.5^k & 0 \\ 0 & 1 \end{pmatrix}}_{Y^{(k)}} \cdot \underbrace{\begin{pmatrix} 2 & 4 \\ 0 & 1 \end{pmatrix}}_{W} \cdot \underbrace{\begin{pmatrix} 2^k & 0 \\ 0 & 3 \end{pmatrix}}_{Z^{(k)}} = \lim_{k\to\infty} \begin{pmatrix} 0.5^k \cdot 2 \cdot 2^k & 0.5^k \cdot 4 \cdot 3 \\ 1 \cdot 0 \cdot 2^k & 1 \cdot 1 \cdot 3 \end{pmatrix} = \underbrace{\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}}_{B} .$$

Often the goal of a biproportional scaling is to take the original matrix W and to fit it successively to some intended new row marginals $\mathbf{r} \in \mathbb{R}^m_{>0}$ and column marginals $\mathbf{c} \in \mathbb{R}^n_{>0}$. For that reason we synonymously refer to any biproportional scaling B of W with $\mathbf{r} = B\mathbf{1}$ and $\mathbf{c} = B^T\mathbf{1}$ as a **biproportional fit** of W to $\mathbf{r}$ and $\mathbf{c}$. In the above example, B is a biproportional fit of W to the new row marginals $\mathbf{r} = \left(\begin{smallmatrix} 2 \\ 3 \end{smallmatrix}\right)$ and column marginals $\mathbf{c} = \left(\begin{smallmatrix} 2 \\ 3 \end{smallmatrix}\right)$. The corresponding scaling sequence $(Y^{(k)} W Z^{(k)})$ can be seen as an iterative transformation of W along the matrices $W^{(k)} := Y^{(k)} W Z^{(k)}$, whose limit B achieves the desired marginals.

Uniqueness of biproportional fits

We cannot expect that the choice of W, $\mathbf{r}$ and $\mathbf{c}$ determines all diagonal scaling matrices $Y^{(k)}$ and $Z^{(k)}$ in a biproportional fit uniquely. However, it turns out that at least B is uniquely determined! The following fundamental result (Pukelsheim, 2014, Theorem 1) shows that any choice of W, $\mathbf{r}$ and $\mathbf{c}$ uniquely determines the biproportional fit B, if existing at all.

Lemma 3.3.2 (Biproportional fits are unique)
If B_1, B_2 are two biproportional fits of $W \in \Omega$ to row marginals $\mathbf{r} \in \mathbb{R}^m_{>0}$ and column marginals $\mathbf{c} \in \mathbb{R}^n_{>0}$, then it holds that $B_1 = B_2$.

This justifies to state that B is *the* biproportional fit of W to the marginals $\mathbf{r}$ and $\mathbf{c}$. Note that for some combinations of W, $\mathbf{r}$ and $\mathbf{c}$ no biproportional fit might exist at all, for example for $W = \left(\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right)$, $\mathbf{r} = \left(\begin{smallmatrix} 2 \\ 1 \end{smallmatrix}\right)$, $\mathbf{c} = \left(\begin{smallmatrix} 1 \\ 2 \end{smallmatrix}\right)$. The techniques for determining whether or not a biproportional fit exists are non-trivial, see the details in Section 3.4.3.

Directness of biproportional fits

Although the sequences $(Y^{(k)})$ and $(Z^{(k)})$ are not uniquely determined, they have an interesting property: sometimes they can be chosen to be constant, that is $Y = Y^{(k)}$ and $Z = Z^{(k)}$ for some diagonal matrices $Y, Z \in \operatorname{diag}(\mathbb{R}^n_{>0})$ and all k. In this case W and B are simply related by the factorized form $B = YWZ$, or stated differently, W factorizes as $W = Y^{-1}BZ^{-1}$. If a biproportional scaling can be written this way, then it is denoted as being **direct**. Directness is illustrated by the following example.

Example 3.3.3 (Direct and non-direct biproportional fits)
$B = \left(\begin{smallmatrix} 6 & 1 & 0 \\ 0 & 6 & 4 \end{smallmatrix}\right)$ is the direct biproportional fit of $W = \left(\begin{smallmatrix} 2 & 1 & 0 \\ 0 & 3 & 1 \end{smallmatrix}\right)$ to $\mathbf{r} = (7, 10)^T$ and $\mathbf{c} = (6, 7, 4)^T$, because it can be written, for example, as $B = \operatorname{diag}(1, 2) \cdot W \cdot \operatorname{diag}(3, 1, 2)$. $B = \left(\begin{smallmatrix} 2 & 0 \\ 0 & 3 \end{smallmatrix}\right)$ is the biproportional fit of $W = \left(\begin{smallmatrix} 2 & 4 \\ 0 & 1 \end{smallmatrix}\right)$ to $\mathbf{r} = \mathbf{c} = (2, 3)^T$, because it can be written, for example, as $B = \lim_{k\to\infty} \operatorname{diag}(0.2^k, 3) \cdot W \cdot \operatorname{diag}(5^k, 1)$. It is non-direct, because no factorized form is able to let the top right entry in W "vanish" (i.e., switch from its positive value to an additional zero-entry in the unique biproportional fit B).

Example 3.3.3 indicates that directness is related to the appearance of additional zero-entries in B that are not present in W. Indeed, the following lemma characterizes directness precisely as those biproportional scalings that have exactly the same zeros as W (i.e., $E(B) = E(W)$). All biproportional scalings with additional zeros (i.e., $E(B) \subsetneq E(W)$) are non-direct.

Lemma 3.3.4 (Directness)
Let B denote any biproportional scaling of W. Then it holds that $E(B) \subseteq E(W)$. Further, $E(B) = E(W)$ if and only if B is direct.

Lemma 3.3.4 can be derived from results by Menon (1968). It is intuitive to see why non-direct biproportional fits cannot be expressed in direct/factorized form: since $Y^{(k)}$ and $Z^{(k)}$ must have positive diagonals due to the positive marginals constraint, the only way to let entries in W vanish is to let some entries in $Y^{(k)}$ and $Z^{(k)}$ converge to zero, which implies that

some other entries have to diverge to infinity. This interplay between diverging and vanishing entries cannot be provided by a single fixed choice of positive diagonal matrices Y and Z.

Symmetric biproportional fits

In contrast to previous work on general matrix scaling of $W \in \Omega$, we are particularly interested in symmetric matrices $W \in \mathbb{W}$. We work out specific results that hold only in this special case. The following lemma shows that assuming $W \in \mathbb{W}$ and $\mathbf{r} = \mathbf{c}$ already *implies* that every biproportional fit B must be symmetric as well. The term **symmetric biproportional scaling** refers to any biproportional scaling B that satisfies $B = B^T$.

Lemma 3.3.5 (Symmetric biproportional fits)
Let B denote the biproportional fit of $W \in \mathbb{W}$ to row and column marginals $\mathbf{f} \in \mathbb{R}^n_{>0}$. Then it holds that

(i) $B = B^T$ *is symmetric*

(ii) $B = \lim_{k\to\infty} W_k$ *for a sequence of* $W_k \in \{XWX \mid X \in \operatorname{diag}(\mathbb{R}^n_{>0})\} =: \Psi(W)$

(iii) $B \in \Psi(W)$ *if and only if B is direct*

Proof. (i) Let $B = \lim_{k\to\infty} R^{(k)} W S^{(k)}$ denote the biproportional fit of W to $\mathbf{f}$. From $W = W^T$ we get that $B = \lim_{k\to\infty} R^{(k)} W^T S^{(k)}$. Transposing both sides gives $B^T = \lim_{k\to\infty} S^{(k)} W R^{(k)}$ as an alternative biproportional fit of W to $\mathbf{f}$, so by uniqueness (Lemma 3.3.2), $B^T = B$.

(ii) $T^{(k)} := \sqrt{R^{(k)} S^{(k)}}$ has diagonal entries $t_i := (r_i^{(k)} s_i^{(k)})^{1/2}$. For all i, j with $ij \in E(W)$, we get from $\lim_{k\to\infty} w_{ij}\ r_i^{(k)} s_j^{(k)} = b_{ij} = b_{ji} = \lim_{k\to\infty} w_{ij}\ r_j^{(k)} s_i^{(k)}$ and $w_{ij} = w_{ji} \neq 0$ that $\lim_{k\to\infty}(r_i^{(k)} s_j^{(k)})^{1/2} = \lim_{k\to\infty}(r_j^{(k)} s_i^{(k)})^{1/2}$. Thus, for $ij \in E(W)$,

$$\lim_{k\to\infty} w_{ij}\ t_i^{(k)} t_j^{(k)} = \lim_{k\to\infty} w_{ij} \sqrt{r_i^{(k)} s_j^{(k)} r_j^{(k)} s_i^{(k)}} = w_{ij} \cdot (\lim_{k\to\infty} \sqrt{r_i^{(k)} s_j^{(k)}})^2 = \lim_{k\to\infty} w_{ij}\ r_i^{(k)} s_j^{(k)} = b_{ij}.$$

The case $b_{ij} \neq 0$ implies that $ij \in E(W)$, hence, $\lim_{k\to\infty} w_{ij}\ t_i^{(k)} t_j^{(k)} = b_{ij}$. If $b_{ij} = 0$, then either $ij \in E(W)$, thus again $\lim_{k\to\infty} w_{ij}\ t_i^{(k)} t_j^{(k)} = b_{ij}$, or $ij \notin E(W)$, in which case $\lim_{k\to\infty} w_{ij}\ t_i^{(k)} t_j^{(k)} = \lim_{k\to\infty} 0 t_i^{(k)} t_j^{(k)} = 0 = b_{ij}$.

Thus, $\lim_{k\to\infty} T^{(k)} W T^{(k)} = B$, which proves (ii) for $W_k := T^{(k)} W T^{(k)} \in \Psi(W)$.

(iii) The implication "$\Rightarrow$" follows from the definitions, and "$\Leftarrow$" follows by applying (ii) to the direct biproportional scaling $B = RWS$. □

Since $B = B^T$ implies that $\mathbf{r} = \mathbf{c} =: \mathbf{f}$, we get from Lemma 3.3.5 the following corollary on how to interpret the set $\Psi(W)$.

Corollary 3.3.6 (Symmetric direct biproportional fits)
For $W \in \mathbb{W}$, let $B = YWZ$ denote any symmetric direct biproportional scaling of W. Then it holds that $B = XWX$ for $X = \sqrt{YZ}$. Further, the set

$$\Psi(W) \;:=\; \{XWX \mid X \in \operatorname{diag}(\mathbb{R}^n_{>0})\}$$

consists of all *symmetric direct biproportional scalings of W.*

That is, in contrast to the non-symmetric case, where a direct biproportional fit is of the form $B = YWZ$, symmetric matrices allow even for a factorization of the form $B = XWX$.

3.3.3 Affine hull and relative interior

A linear combination $\sum_{i=1}^k \alpha_i \mathbf{x}_i$ of elements $\mathbf{x}_i \in \mathbb{R}^d$ with coefficients $\alpha_i \in \mathbb{R}$ is denoted as an **affine combination** if further $\sum_{i=1}^k \alpha_i = 1$ holds. For every set $S \subseteq \mathbb{R}^d$, the **affine hull** $\operatorname{aff}(S)$ is defined as the set of all elements that can be represented as affine combinations of the elements in S, that is

$$\operatorname{aff}(S) \;:=\; \left\{ \sum_{i=1}^k \alpha_i \mathbf{x}_i \mid k \in \mathbb{N},\; \mathbf{x}_i \in S,\; \alpha_i \in \mathbb{R},\; \sum_{i=1}^k \alpha_i = 1 \right\} \;\supseteq\; S \quad .$$

The set A is affine if it contains all affine combinations of its elements, that is if and only if $A = \operatorname{aff}(A)$. In this case A is denoted as an **affine space** because all affine sets in $\mathbb{R}^d$ are translated linear subspaces. $\operatorname{aff}(S) \supseteq S$ is the smallest affine space that contains the set S.

Lemma 3.3.7 (Affine hull of sets of matrices)
For every $W \in \Omega$, $\mathbf{r} \in \mathbb{R}^m_{>0}$ and $\mathbf{c} \in \mathbb{R}^n_{>0}$ it holds that

$$(i) \qquad \operatorname{aff}(\Omega(\mathbf{r}, \cdot, W)) \;=\; \{X \in \mathbb{R}^{m \times n} \mid X\mathbf{1} = \mathbf{r},\; E(X) \subseteq E(W)\}$$

$$(ii) \qquad \operatorname{aff}(\Omega(\cdot, \mathbf{c}, W)) \;=\; \{X \in \mathbb{R}^{m \times n} \mid X^T\mathbf{1} = \mathbf{c},\; E(X) \subseteq E(W)\}$$

Proof. (i) "$\subseteq$" follows from the fact that every affine combination of elements from $\Omega(\mathbf{r}, \cdot, W)$ lies in $\Omega^{\pm}(\mathbf{r}, \cdot, W) := \{X \in \mathbb{R}^{m \times n} \mid X\mathbf{1} = \mathbf{r},\; E(X) \subseteq E(W)\}$. In order to show "$\supseteq$", fix any $A \in \Omega^{\pm}(\mathbf{r}, \cdot, W)$ and let $N(A) \subseteq E(A)$ denote the subset of indices at which $a_{ij} < 0$. It holds that $\Omega(\mathbf{r}, \cdot, A) \neq \emptyset$ because it contains the matrix that has a single entry r_i per row i at the smallest column index j with $a_{ij} \neq 0$. Let $Z \in \Omega(\mathbf{r}, \cdot, A)$. For each $kl \in N(A)$ define the matrix B^{kl} as the matrix that equals Z everywhere except for row k; this row is set to zero

except for a single entry r_k at position kl. Precisely:

$$b_{ij}^{kl} \;=\; \begin{cases} z_{ij} & \text{if } i \neq k \\ 0 & \text{if } i = k \text{ and } j \neq l \\ r_k & \text{if } i = k \text{ and } j = l \end{cases} \quad .$$

It holds that $B^{kl} \in \Omega(\mathbf{r}, \cdot, A)$. Further define

$$\alpha_{kl} \;:=\; \frac{|a_{kl}|}{r_k} \quad , \qquad \alpha_C \;:=\; 1 + \sum_{kl \in N(A)} \frac{|a_{kl}|}{r_k} \quad , \qquad C \;:=\; \alpha_C^{-1} \cdot \left(A + \sum_{kl \in N(A)} \alpha_{kl} B^{kl} \right) \quad .$$

Observe that $E(C) \subseteq E(A)$, $C \geq 0$ and $C\mathbf{1} = \mathbf{r}$, thus $C \in \Omega(\mathbf{r}, \cdot, A)$. This gives that that A can be represented as

$$A \;=\; \alpha_C \cdot C + \sum_{kl \in N(A)} (-\alpha_{kl}) \cdot B^{kl} \quad ,$$

an affine combination of $|N(A)|+1$ matrices from $\Omega(\mathbf{r}, \cdot, A)$. The proof of (ii) is analogous. □

The **interior** of a set $S \subseteq \mathbb{R}^d$ is defined as $\operatorname{int}(S) := \{x \in S \mid \exists \varepsilon_x > 0,\ B_{\mathbb{R}^d}(x, \varepsilon_x) \subseteq S\}$. This standard notion does not fit well to the case of lower-dimensional sets that are embedded into higher-dimensional spaces, because it provides $\operatorname{int}(S) = \emptyset$ whenever the set S has not full dimensionality. For example, the 2-dimensional unit-cube $H := [0,1]^2 \subseteq \mathbb{R}^2$ gives that $\operatorname{int}(H) = (0,1)^2$ as intended, but if H is embedded into $\mathbb{R}^3$ as $H' := \{(x, y, 0) \in \mathbb{R}^3 \mid x, y \in [0,1]\}$, we get that $\operatorname{int}(H') = \emptyset$, which is often not intended.

A common alternative notion of "interior" is provided by the **relative interior**, see for example Boyd and Vandenberghe (2004) for background information. The relative interior automatically adapts to lower-dimensional sets $S \subseteq \mathbb{R}^d$. It is defined relatively to the topology of the smallest affine space that contains S.

Definition 3.3.8 (Relative interior)
For all $C \subseteq \mathbb{R}^d$ the relative interior is defined as

$$\operatorname{rel\,int}(C) := \{x \in C \mid \exists \varepsilon_x > 0 : B_{\mathbb{R}^d}(x, \varepsilon_x) \cap \operatorname{aff}(C) \subseteq C\} \quad .$$

In other words, the relative interior of S contains all points $x \in S$ around which there exists a small neighborhood in $\mathbb{R}^d$ that is fully contained in C with respect to the subspace topology implied by $\operatorname{aff}(C)$, that is, after intersecting the neighborhood with $\operatorname{aff}(C)$. For convex $C \subseteq \mathbb{R}^d$ the relative interior can equivalently be defined as follows:

$$\operatorname{rel\,int}(C) := \{x \in C \mid \forall y \in C : \exists \varepsilon_y > 0 : y + (1 + \varepsilon_y)(x - y) \in C\} \quad ,$$

which shows that the relative interior contains all points that do not touch any boundary in the sense that we can step truly "behind" x from the viewpoint of any other point $y \in C$, without leaving the set C.

We end this section by providing an example that shows how $\Omega(\mathbf{r}, \cdot, W)$ is related to its subset $\underline{\Omega}(\mathbf{r}, \cdot, W)$ in terms of the relative interior.

Example 3.3.9 (Relative interior of sets of matrices)
Let $W \in \Omega$ contain at least one zero-entry, that is $|E(W)| < m \cdot n$. Then it holds that

$$\operatorname{int}(\Omega(\mathbf{r}, \cdot, W)) \;=\; \emptyset \quad ,$$

since for all $X \in \Omega(\mathbf{r}, \cdot, W)$ and all $\varepsilon > 0$ the ball $B_{\mathbb{R}^{m\times n}}(X, \varepsilon)$ contains matrices with negative entries at the zero-positions of X, thus $B_{\mathbb{R}^{m\times n}}(X, \varepsilon) \subsetneq \Omega(\mathbf{r}, \cdot, W)$. For the relative interior we get that

$$\operatorname{rel\,int}(\Omega(\mathbf{r}, \cdot, W)) \;=\; \underline{\Omega}(\mathbf{r}, \cdot, W) \ ,$$

since now $B_{\mathbb{R}^{m\times n}}(X, \varepsilon)$ is intersected with $\operatorname{aff}(\Omega(\mathbf{r}, \cdot, W))$ before requiring to be a subset of $\Omega(\mathbf{r}, \cdot, W)$. This rules out only those elements $X \in \Omega(\mathbf{r}, \cdot, W)$ that have $E(X) \subsetneq E(W)$.

3.3.4 Bregman divergences and relative entropy

For any set $\mathcal{Z}$, the function $d : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is a **metric** if it satisfies the following properties:

- non-negativity: $d(a, b) \geq 0$
- coincidence: $d(a, b) = 0 \;\Leftrightarrow\; a = b$
- symmetry: $d(a, b) = d(b, a)$
- triangle inequality: $d(a, b) \leq d(a, c) + d(c, b)$ for all $c \in \mathcal{Z}$

A metric provides a strong sense of "distance" between all elements in $\mathcal{Z}$. Many applications require only a weaker sense, for example the well-known *cosine distance* provides neither the triangle inequality nor coincidence.

Another weak sense of "distance" is provided by the family of **Bregman divergences**, whose following definition goes back to the original work by Bregman (1967a).

Definition 3.3.10 (Bregman divergence)
For convex $\mathcal{Z} \subseteq \mathbb{R}^d$ let $h : \mathcal{Z} \to \mathbb{R}$ be a strictly convex differentiable function. The Bregman divergence implied by h on $\mathcal{Z}$ is defined as the function $D_h : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ with

$$D_h(x \| y) \;:=\; h(x) - h(y) - \langle \nabla h(y), x - y \rangle \quad .$$

The standard example for a Bregman divergence is the squared Euclidean distance $\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2$, which is implied by $h(\mathbf{x}) := \frac{1}{2}\|\mathbf{x}\|^2$ on $\mathcal{Z} := \mathbb{R}^d$. However, other choices for h provide a rich variety of different Bregman divergences.

Every Bregman divergence provides the following geometric interpretation: $D_h(x\|y)$ is the amount of underestimating $h(x)$ from the supporting hyperplane tangential to h at y, evaluated at x. This can be seen as follows: the gradient $\nabla h(y)$ gives the direction and the amount of steepest increase of h at location y. Interpreting $\langle \nabla h(y), x-y\rangle$ as the projection of $x-y$ to $\nabla h(y)$ shows that it quantifies the linearly extrapolated total amount of increase/decrease when stepping from y to x. As a consequence, $\hat{h}_y(x) := h(y) + \langle \nabla h(y), x-y\rangle$ is the estimate on $h(x)$ that is given by the supporting hyperplane tangential to h at location y, evaluated at x. By convexity of h, the tangential plane is always below the epigraph of h, hence $\hat{h}_y(x) \leq h(x)$ is always an underestimate. By strict convexity of h, equality $\hat{h}_y(x) = h(x)$ holds if and only if $x = y$. In particular, every Bregman divergence satisfies non-negativity and coincidence.

Depending on additional assumptions on $\mathcal{Z}$ and h, stronger properties of D_h can be guaranteed. Many theoretical studies assume that $\mathcal{Z}$ is closed and that h is a function of *Legendre type* (see Bauschke and Lewis (2000) for a definition). These additional assumptions imply slightly stronger properties of D_h, although they still do not guarantee symmetry or the triangle inequality.

One such function of Legendre type is $h(\mathbf{x}) := \sum_i x_i \log(x_i)$, defined for all $\mathbf{x} \in \mathbb{R}^d_{\geq 0}$ with the continuous extension $0 \cdot \log(0) := 0$. By choosing $\mathcal{Z}$ to be the d-dimensional Simplex, this function h implies the **Kullback-Leibler divergence** $D_{KL}(x\|y) = \sum_i x_i \log(x_i/y_i)$, also denoted as **relative entropy**. This "distance" measure between probability distributions is well-known in information theory and Bayesian statistics. If $\mathcal{Z}$ is extended to non-negative matrices, the same function h implies the *generalized Kullback-Leibler divergence* or *generalized relative entropy*, defined as follows.

Definition 3.3.11 (Generalized relative entropy)
For matrices $X, W \in \mathbb{R}^{m\times n}_{\geq 0}$ the **generalized relative entropy** is defined as

$$RE(X\|W) \;:=\; \sum_{i,j} x_{ij}\log(x_{ij}/w_{ij}) - x_{ij} + w_{ij} \quad ,$$

with the conventions that $0 \cdot \log(0/w) := 0$ and $x \cdot \log(x/0) := \infty$ for all $w \geq 0, x > 0$. In particular, $RE(X\|W) < \infty$ if and only if $E(X) \subseteq E(W)$.

Here and later we identify $\mathbb{R}^{m\times n}$ with $\mathbb{R}^{m\cdot n}$. Note that $RE(X\|W)$ can only be represented as a Bregman divergence if $E(X) \subseteq E(W)$, but we use the additional convention that $RE(X\|W) := \infty$ for $E(X) \not\subseteq E(W)$, which is convenient for some later formal statements.

3.3.5 Bregman projections

Like other distance measures, Bregman divergences can be used to define a projection in the sense of mapping to the target element of some set that is closest to the source element. Precisely, for all closed convex $\mathcal{M} \subseteq \mathbb{R}^d$ the corresponding **Bregman projection** $\mathcal{P}^h_{\mathcal{M}}(Q)$ of the element $Q \in \mathbb{R}^d$ to the set $\mathcal{M}$ is defined as

$$\mathcal{P}^h_{\mathcal{M}}(Q) \;:=\; \arg\min_{M \in \mathcal{M}} \; D_h(M\|Q) \quad . \tag{3.8}$$

The projection target $\mathcal{P}^h_{\mathcal{M}}(Q) \in \mathcal{M}$ is unique whenever existing, that is whenever $\mathcal{M} \neq \emptyset$.

We are particularly concerned with the following two Bregman projections:

- for $h := \frac{1}{2}\|\mathbf{x}\|^2$ the corresponding Bregman divergence is the halved squared Euclidean distance $\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|^2$. Minimizing this distance is equivalent to minimizing the Euclidean distance without squaring. As a consequence, the projection $\mathcal{P}^h_{\mathcal{M}}$ equals the standard **orthogonal projection** in $\mathbb{R}^d$, denoted as $\mathcal{P}^{\perp}_{\mathcal{M}}$. This projection maps to the element in $\mathcal{M}$ that is closest to Q with respect to Euclidean distance.
- for the generalized relative entropy $D_h = RE$, we refer to (3.8) as the RE**-projection**, denoted by $\mathcal{P}_{\mathcal{M}}$ without any superscript. The extension to the case $RE(X\|W) = \infty$ in Definition 3.3.11 implies that $\mathcal{P}_{\mathcal{M}}(Q)$ is existing and unique if and only if there exists any element $A \in \mathcal{M}$ with $E(A) \subseteq E(W)$.

The following lemma shows the fundamental connection between Bregman projections and proportional row/column scaling of matrices, which allows to think of matrix scaling as of projecting to certain convex sets of matrices.

Lemma 3.3.12 (Proportional scaling is the RE-projection)
Let $W \in \Omega$ provide row marginals $\mathbf{d}_r := W\mathbf{1}$ and column marginals $\mathbf{d}_c := W^T\mathbf{1}$. Fix any $\mathbf{r} \in \mathbb{R}^m_{>0}$ and $\mathbf{c} \in \mathbb{R}^n_{>0}$. Then the RE-projections to $\mathcal{R} := \Omega(\mathbf{r}, \cdot, \cdot)$ and $\mathcal{C} := \Omega(\cdot, \mathbf{c}, \cdot)$, respectively, are given by proportional row/column scaling, that is

$$\begin{aligned} \mathcal{P}_{\mathcal{R}}(W) &= \operatorname{diag}(\mathbf{r}) \cdot \operatorname{diag}(\mathbf{d}_r)^{-1} \cdot W \quad , \\ \mathcal{P}_{\mathcal{C}}(W) &= W \cdot \operatorname{diag}(\mathbf{d}_c)^{-1} \cdot \operatorname{diag}(\mathbf{c}) \quad . \end{aligned} \tag{3.9}$$

In particular it holds that $E(\mathcal{P}_{\mathcal{R}}(W)) = E(\mathcal{P}_{\mathcal{C}}(W)) = E(W)$.

Proof. We prove that $\mathcal{P}_{\mathcal{R}}(W) \overset{!}{=} \operatorname{diag}(\mathbf{r}) \cdot \operatorname{diag}(\mathbf{d}_r)^{-1} \cdot W =: W_{\mathbf{r}}$. The proof for $\mathcal{P}_{\mathcal{C}}(W)$ is analogous. For $X \in \Omega$ define $\varrho(X) := RE(X\|W)$. From $W_{\mathbf{r}} \in \mathcal{R} \neq \emptyset$ and $\varrho(W_{\mathbf{r}}) < \infty$ it follows that $\mathcal{P}_{\mathcal{R}}(W) \in \mathcal{R}$ exists and is the unique global minimizer of ϱ over $\mathcal{R}$. It *probably*

equals $W_{\mathbf{r}}$. We are going to show that $\varrho(\mathcal{P}_{\mathcal{R}}(W)) = \varrho(W_{\mathbf{r}})$, which implies that indeed $W_{\mathbf{r}} = \mathcal{P}_{\mathcal{R}}(W)$. Right from the definitions we get that

$$\begin{aligned}
\mathcal{P}_{\mathcal{R}}(W) &= \operatorname*{arg\,min}_{X \in \mathbb{R}^{m\times n}_{\geq 0}} \{\ \varrho(X) \mid X\mathbf{1} = \mathbf{r}\ \} \\
&\overset{(1)}{=} \operatorname*{arg\,min}_{X \in \mathbb{R}^{m\times n}_{\geq 0}} \{\ \varrho(X) \mid X\mathbf{1} = \mathbf{r},\ E(X) \subseteq E(W)\ \} \\
&\overset{(2)}{=} \operatorname*{arg\,min}_{X \in \mathbb{R}^{m\times n}_{\geq 0}} \left\{ \sum_{ij \in E(W)} x_{ij}\log(x_{ij}/w_{ij}) - x_{ij} \mid X\mathbf{1} = \mathbf{r},\ E(X) \subseteq E(W) \right\} \quad ,
\end{aligned}$$

where Equality (1) comes from the fact that $RE(X\|W) = \infty$ whenever $E(X) \not\subseteq E(W)$, and Equality (2) applies that $0 \cdot \log(0/w_{ij}) := 0$.

At first glance, this may look like a standard convex optimization problem that can be solved by the Lagrangian approach. But a second view reveals that we must be careful because the partial derivative $\nabla_{x_{ij}}\varrho(X)$ "$= \log(x_{ij}/w_{ij})$" does not exist for $ij \in E(W) \setminus E(X)$, that is for $x_{ij} = 0 < w_{ij}$. In particular, it is invalid to restrict the summands to $ij \in E(X) \subseteq E(W)$ before taking the derivatives. Hence, the above describes a non-smooth optimization problem, and we cannot simply apply the standard Lagrangian approach without further considerations.

Here, we solve this issue by studying the limit of the RE-projections to the closed convex sets

$$\mathcal{R}_\varepsilon := \{X \in \mathbb{R}^{m\times n}_{\geq 0} \mid X\mathbf{1} = \mathbf{r},\ E(X) \subseteq E(W),\ w_{ij} > 0 \Rightarrow x_{ij} \geq \varepsilon\}.$$

for $\varepsilon \to 0$. Note that $\mathcal{R}_0 = \mathcal{R}$, and there exists some $\varepsilon_{\mathbf{r}}$ such that for all $\varepsilon < \varepsilon_{\mathbf{r}}$ it holds that $W_{\mathbf{r}} \in \mathcal{R}_\varepsilon$. Similar to the above we get for $\varepsilon \geq 0$ that $\mathcal{P}_{\mathcal{R}_\varepsilon}(W)$ equals

$$\operatorname*{arg\,min}_{X \in \mathbb{R}^{m\times n}_{\geq 0}} \left\{ \sum_{ij \in E(W)} x_{ij}\log(x_{ij}/w_{ij}) - x_{ij} \mid X\mathbf{1} = \mathbf{r},\ E(X) \subseteq E(W),\ w_{ij} > 0 \Rightarrow x_{ij} \geq \varepsilon \right\} \quad .$$

For $\varepsilon > 0$, all partial derivatives $\nabla_{x_{ij}}\rho(X) = \log(x_{ij}/w_{ij})$ exist. So we can take the standard Lagrangian approach, that is we aim at finding a global minimum of the Lagrangian function

$$\Lambda(X, \boldsymbol{\mu}) = \varrho(X) - \boldsymbol{\mu}^T(X\mathbf{1} - \mathbf{r}) = \sum_{ij \in E(W)} x_{ij}\log(x_{ij}/w_{ij}) - x_{ij} - \sum_{i=1}^{m} \mu_i \left(\left(\sum_{j\,:\,ij \in E(W)} x_{ij} \right) - r_i \right)$$

over $X \in \mathbb{R}^{E(W)}_{\geq \varepsilon}$, by which we treat $x_{ij} := 0$ for all $ij \notin E(W)$ as zero constants. From $w_{ij} \neq 0 \Leftrightarrow x_{ij} \neq 0$ we get that $\nabla_{x_{ij}}\Lambda(X, \boldsymbol{\mu}) = \log(x_{ij}/w_{ij}) - \mu_i$ for all $ij \in E(W)$. Setting all

derivatives to zero gives after entry-wise exponentiation that

$$x_{ij} \;=\; \begin{cases} \exp(\mu_i) \cdot w_{ij} & \text{for } ij \in E(W) \\ 0 & \text{for } ij \notin E(W) \end{cases},$$

that is in matrix notation $X = \operatorname{diag}(\exp(\boldsymbol{\mu})) \cdot W$. Multiplying $\mathbf{1}$ from the right gives that $\mathbf{r} = X\mathbf{1} = \operatorname{diag}(\exp(\boldsymbol{\mu})) \cdot W\mathbf{1}$, hence $\operatorname{diag}(\exp(\boldsymbol{\mu})) = \operatorname{diag}(\mathbf{r}) \operatorname{diag}(W\mathbf{1})^{-1}$. Thus $X = W_{\mathbf{r}}$ is indeed the (only) critical point, and $W_{\mathbf{r}} \in \mathcal{R}_\varepsilon$ for all $\varepsilon < \varepsilon_{\mathbf{r}}$. Further, the Hessian matrix is the diagonal matrix with the all-positive diagonal $\nabla_{x_{ij}} \nabla_{x_{ij}} \Lambda(X, \boldsymbol{\mu}) = 1/x_{ij}$, which is positive definite. Thus $W_{\mathbf{r}}$ is the (unique global) minimizer whenever $\varepsilon < \varepsilon_{\mathbf{r}}$, taking the (unique global) minimum $\varrho(W_{\mathbf{r}})$.

By continuity of ϱ on $\mathcal{R} = \mathcal{R}_0$, we get that $\varrho(\mathcal{P}_{\mathcal{R}}(W)) = \lim_{\varepsilon \to 0} \varrho(\mathcal{P}_{\mathcal{R}_\varepsilon}(W)) = \varrho(W_{\mathbf{r}})$. □

An example application of Lemma 3.3.12 is that the transition matrix $P = D^{-1}W$ of a graph matrix W can be interpreted as the RE-projection of W to the set of matrices of unit row sums.

Another consequence of $E(\mathcal{P}_{\mathcal{R}}(W)) = E(\mathcal{P}_{\mathcal{C}}(W)) = E(W)$ is the following sense of maximality:

$$A \in \Omega(\mathbf{r}, \cdot, W) \Rightarrow E(A) \subseteq E(\mathcal{P}_{\Omega(\mathbf{r},\cdot,\cdot)}(W)) \quad \text{and} \quad A \in \Omega(\cdot, \mathbf{c}, W) \Rightarrow E(A) \subseteq E(\mathcal{P}_{\Omega(\cdot,\mathbf{c},\cdot)}(W)) \;.$$

A similar sense of maximality also holds for $\Omega(\mathbf{r}, \mathbf{c}, W)$, as made precise by the following lemma.

Lemma 3.3.13 (Maximality of RE-projections)
Let $W \in \Omega, \mathbf{r} \in \mathbb{R}^m_{>0}$ and $\mathbf{c} \in \mathbb{R}^n_{>0}$. Then it holds that

$$A \;\in\; \Omega(\mathbf{r}, \mathbf{c}, W) \quad \Rightarrow \quad E(A) \;\subseteq\; E(\mathcal{P}_{\Omega(\mathbf{r},\mathbf{c},\cdot)}(W))$$

A proof of Lemma 3.3.13 is given by Pretzel (1980, Theorem 1).

3.3.6 Mean functions

In this thesis, we refer by the term **mean function** to the following definition.

Definition 3.3.14 (Mean function)
A function $m : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a (homogeneous) mean function on $\mathbb{R}_{\geq 0}$ if it satisfies the following properties:

- symmetry: $m(x, y) = m(y, x)$
- minmax-bounded: $m(x, y) \in [\min(x, y), \max(x, y)]$
- homogeneity: $c\, m(x, y) = m(c\, x, c\, y)$ for all $c > 0$

For example, the convex combination $c_\alpha(x, y) := (1-\alpha) \cdot \min(x, y) + \alpha \cdot \max(x, y)$ for parameter $\alpha \in [0, 1]$ provides a family of mean functions, which includes the minimum function for $\alpha = 0$, the maximum function for $\alpha = 1$, and the arithmetic mean for $\alpha = 0.5$.

Another example is the family of **Hölder p-means**, defined as

$$m_p(x, y) \;:=\; \sqrt[p]{(x^p + y^p)/2}$$

for parameter $p \in \mathbb{R} \cup \{\pm\infty\}$. This family contains the following special cases:

- for $p = -\infty$ the minimum function: $\min(x, y)$
- for $p = \infty$ the maximum function: $\max(x, y)$
- for $p = 1$ the arithmetic mean: $m_A(x, y) := (x + y)/2$
- for $p = 0$ the geometric mean: $m_G(x, y) := \sqrt{xy}$
- for $p = -1$ the harmonic mean: $m_H(x, y) := \begin{cases} 2/(x^{-1} + y^{-1}) & x, y > 0 \\ 0 & \text{else} \end{cases}$

One aspect in which mean functions differ is in whether they map closer to the smaller or closer to the larger value. For example, the arithmetic mean $m_A(a, b)$ is always "exactly in the middle" of a and b, while $c_\alpha(a, b)$ for $\alpha < 0.5$ is closer to $\min(a, b)$. The geometric mean is also biased toward $\min(a, b)$, but in contrast to c_α this bias gets stronger the larger the ratio $\max(a, b)/\min(a, b)$ is.

We denote a mean function m as **sub-arithmetic** if $m(x, y) \leq m_A(x, y)$ for all $x, y \in \mathbb{R}_{\geq 0}$, and as **super-arithmetic** if $m(x, y) \geq m_A(x, y)$. It is **strictly** sub-/super-arithmetic if $x \neq y$ implies strict inequality. Every Hölder p-mean for $p < 1$ is strictly sub-arithmetic.

The geometric mean has a fundamental property that makes it unique among all mean functions (and even beyond, as the next lemma shows): it is the only mean function that can be factorized as $m(a, b) = f(a) \cdot f(b)$ for a function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$. A similar result holds for the arithmetic mean and $m(a, b) = f(a) + f(b)$.

Lemma 3.3.15 (Characterization of arithmetic mean and geometric mean)
Let $m : \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denote any function that satisfies that $m(a, a) = a$ for all $a \geq 0$. Then the following holds:

(i) if m can be represented as $m(a, b) = f(a) + f(b)$ for some function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$, then $m = m_A$ is the arithmetic mean (and $f(x) = x/2$).

(ii) if m can be represented as $m(a, b) = f(a) \cdot f(b)$ for some function $f : \mathbb{R}_{\geq 0} \to \mathbb{R}$, then $m = m_G$ is the geometric mean (and $f(x) = \sqrt{x}$ or $f(x) = -\sqrt{x}$).

Proof. (i) $a = m(a, a) = f(a) + f(a) = 2f(a)$ implies immediately that $f(a) = a/2$ for all $a \geq 0$, hence, $m = m_A$. (ii) Assume there exist $a, b \in \mathbb{R}_{\geq 0}$ such that $f(a) > 0$ and $f(b) < 0$. Then it follows $m(a, b) = f(a)f(b) < 0$ in contradiction to m being non-negative. Hence, either $f(a) \geq 0$ for all $a \geq 0$ or $f(a) \leq 0$ for all $a \geq 0$. Now we get from $a = m(a, a) = f(a)f(a) = f(a)^2$ that either $f(a) = \sqrt{a}$ for all $a \geq 0$ or $f(a) = -\sqrt{a}$ for all $a \geq 0$. In both cases we get that $m = m_G$. □

3.3.7 Symmetrizations of iterative proportional fitting

In this section we define the following three sequences of matrices, whose elements are distinguished by the style of brackets used for their superscript:

IPF: traditional IPF-sequence ($W^{\langle k \rangle}$)

PSIPF: Pseudo-Symmetric IPF-sequence ($W^{\langle\langle k \rangle\rangle}$)

SIPF: Symmetric IPF-sequence ($W^{(k)}$)

Like iterated **f**-scaling, all three sequences follow the goal to iteratively transform their initial matrix $W^{\langle 0 \rangle}$, $W^{\langle\langle 0 \rangle\rangle}$ and $W^{(0)}$, respectively, toward a limit matrix that attains some prescribed row and column marginals. Each sequence is fully determined by choosing the initial matrix plus the intended marginals that shall be attained in the limit. All further elements of the sequence are generated deterministically according to its definition.

We study two different settings: the general setting and the symmetric setting.

- In the **general setting**, the sequence is initialized by any matrix $W \in \Omega$ plus two vectors $\mathbf{r} \in \mathbb{R}^m_{>0}$ and $\mathbf{c} \in \mathbb{R}^n_{>0}$ with $\|\mathbf{r}\|_1 = \|\mathbf{c}\|_1$. The sequence is expected to converge to a limit matrix that provides row marginals $\mathbf{r}$ and column marginals $\mathbf{c}$.

- In the **symmetric setting**, the sequence is initialized by a symmetric matrix $W \in \mathbb{W}$ plus a vector $\mathbf{f} \in \mathbb{R}^n_{>0}$. The sequence is expected to converge to a limit matrix that provides row marginals $\mathbf{f}$ and column marginals $\mathbf{f}$.

PSIPF and SIPF are restricted to the symmetric setting, only IPF can also be applied to the general setting. Further, SIPF equals iterated **f**-scaling.

Each of the three sequences is defined in two equivalent ways: a *recursive* definition of the form $W_{k+1} = g(W_k)$ for some generator function g, and an alternative representation in *factorized* form $W_{k+1} = Y_{k+1} \cdot W_0 \cdot Z_{k+1}$, which represents every element of the sequence as the product of the initial matrix with two diagonal matrices. Although both representations are mathematically equivalent, they show different advantages and drawbacks in practical applications, as studied in Section 3.3.8.

Iterative Proportional Fitting (IPF)

The idea of IPF is to scale rows and columns alternately. That is, $W^{\langle 1 \rangle}$ is derived from $W^{\langle 0 \rangle}$ by row-scaling to achieve row marginals $\mathbf{r}$ exactly (for any column marginals). Then, $W^{\langle 2 \rangle}$ is derived from $W^{\langle 1 \rangle}$ by column-scaling to achieve column marginals $\mathbf{c}$ exactly (for any row marginals). Hence, at odd steps the row marginals equal $\mathbf{r}$ (for any column marginals), and at even steps the column marginals equal $\mathbf{c}$ (for any row marginals). The intention is that eventually both marginals converge to $\mathbf{r}$ and $\mathbf{c}$ simultaneously.

Recall from Lemma 3.3.12 that proportional row-scaling and column-scaling can be interpreted as RE-projections to the sets $\mathcal{R}$ and $\mathcal{C}$, respectively. This notation is used in the following definition of the IPF-sequence.

Definition 3.3.16 (IPF-sequence)
For $W \in \Omega$, $\mathbf{r} \in \mathbb{R}^m_{>0}$, and $\mathbf{c} \in \mathbb{R}^n_{>0}$, the IPF-sequence is recursively defined with $W^{\langle 0 \rangle} := W$ as follows:

$$W^{\langle k+1 \rangle} := \begin{cases} \mathcal{P}_{\mathcal{R}}(W^{\langle k \rangle}) & \text{if } k \text{ even} , \\ \mathcal{P}_{\mathcal{C}}(W^{\langle k \rangle}) & \text{if } k \text{ odd} . \end{cases}$$

This implies that $W^{\langle k \rangle} = Y^{\langle k \rangle} W Z^{\langle k \rangle}$ for some positive diagonal matrices $Y^{\langle k \rangle} = \operatorname{diag}(\mathbf{y}^{\langle k \rangle})$ and $Z^{\langle k \rangle} = \operatorname{diag}(\mathbf{z}^{\langle k \rangle})$ with $\mathbf{y}^{\langle 0 \rangle} = \mathbf{z}^{\langle 0 \rangle} = \mathbf{1}$. This factorization can be computed explicitly by directly updating the diagonals $\mathbf{y}^{\langle k \rangle}$ and $\mathbf{z}^{\langle k \rangle}$ in each step as follows:

$$\mathbf{y}^{\langle k+1 \rangle} := \begin{cases} \mathbf{r}/(W\mathbf{z}^{\langle k \rangle}) & \text{if } k \text{ even} \\ \mathbf{y}^{\langle k \rangle} & \text{if } k \text{ odd} \end{cases} , \text{ and } \mathbf{z}^{\langle k+1 \rangle} := \begin{cases} \mathbf{z}^{\langle k \rangle} & \text{if } k \text{ even} \\ \mathbf{c}/(W^T\mathbf{y}^{\langle k \rangle}) & \text{if } k \text{ odd} \end{cases} ,$$

where the division of vectors is meant to be entry-wise.

The factorized approach is also denoted as the **RAS-method** as studied by Bacharach (1965).

The following theorem shows that IPF indeed converges to some element $W^{\langle \infty \rangle} \in \Omega(\mathbf{r}, \mathbf{c}, W)$ whenever $\Omega(\mathbf{r}, \mathbf{c}, W) \neq \emptyset$. Even stronger, $W^{\langle \infty \rangle}$ equals the projection $\mathcal{P}_{\Omega(\mathbf{r},\mathbf{c},W)}(W)$ of W to the set of feasible solutions $\Omega(\mathbf{r}, \mathbf{c}, W)$.

Theorem 3.3.17 (Convergence of IPF in the general setting)
Let $W \in \Omega, \mathbf{r} \in \mathbb{R}^m_{>0}, \mathbf{c} \in \mathbb{R}^n_{>0}$ such that $\Omega(\mathbf{r}, \mathbf{c}, W) \neq \emptyset$. Then the IPF-sequence converges to $\mathcal{P}_{\Omega(\mathbf{r},\mathbf{c},W)}(W)$.

All parts of this theorem are already proven in the literature in various ways. More information on these proofs and on my contributions to this field is provided in Section 3.3.9.

Theorem 3.3.17 implies a couple of results that are collected in the following corollary. This collection provides a detailed understanding on the limit behavior of the IPF-sequence, and is required for later arguments.

Corollary 3.3.18 (Additional limit properties of IPF)
Consider the IPF-sequence in the general setting for $\Omega(\mathbf{r}, \mathbf{c}, W) \neq \emptyset$ as in Theorem 3.3.17. Then all following results hold true:

(i) $W^{\langle\infty\rangle} = \mathcal{P}_{\Omega(\mathbf{r},\mathbf{c},W)}(W)$ *equals the biproportional fit of* W *to* $\mathbf{r}$ *and* $\mathbf{c}$

(ii) $E(W) = E(W^{\langle k\rangle}) \supseteq E(W^{\langle\infty\rangle})$ *for all* $k \geq 0$

(iii) $E(W) = E(W^{\langle\infty\rangle})$ *if and only if the biproportional fit* $W^{\langle\infty\rangle}$ *is direct*

(iv) if $E(W) \neq E(W^{\langle\infty\rangle})$ *then some entries in* $\mathbf{y}^{\langle k\rangle}$ *and* $\mathbf{z}^{\langle k\rangle}$ *diverge to infinity*

Proof. (i) follows from the factorized representation and convergence of the IPF-sequence. (ii) the first equality follows from the definition of proportional scaling. The remaining statements follow from Lemma 3.3.4. □

Of course Theorem 3.3.17 implies convergence in the symmetric setting, but the following corollary makes explicit how the convergence results for $\Omega(\mathbf{f}, \mathbf{f}, W)$ carry over to $\mathbb{W}(\mathbf{f}, W)$ plus additional properties that only hold in the symmetric setting.

Corollary 3.3.19 (Convergence of IPF in the symmetric setting)
Let $W \in \mathbb{W}$ and $\mathbf{f} \in \mathbb{R}^n_{>0}$ such that $\mathbb{W}(\mathbf{f}, W) \neq \emptyset$. Then the IPF-sequence converges to $\mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W) \in \Psi(W)$.

Proof. Lemma 3.3.1 gives that $\mathbb{W}(\mathbf{f}, W) \neq \emptyset$ if and only if $\Omega(\mathbf{f}, \mathbf{f}, W) \neq \emptyset$. Together with Theorem 3.3.17 this gives that IPF converges to $W^{\langle\infty\rangle} = \mathcal{P}_{\Omega(\mathbf{r},\mathbf{c},W)}(W)$, which is the biproportional fit of W to row and column marginals $\mathbf{f}$. Lemma 3.3.5 implies that $W^{\langle\infty\rangle} \in \Psi(W)$, particularly symmetry, hence, $W^{\langle\infty\rangle} = \mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W)$. □

Corollary 3.3.19 gives that $W \in \Psi(W)$ as well as $W^{\langle\infty\rangle} \in \Psi(W)$. However, for $k > 0$ all elements $W^{\langle k\rangle}$ of the IPF-sequence lie outside of $\Psi(W)$, they are not even symmetric. The IPF-sequence is inherently non-symmetric, even in the symmetric setting. Recall from Lemma 3.3.5 that in the symmetric setting there exist sequences of matrices in $\Psi(W)$ that converge to the biproportional fit. The IPF-sequence is not of this type. As we will see next, the PSIPF-sequence is derived from the IPF-sequence such that $W^{\langle\langle k\rangle\rangle} \in \Psi(W)$. Further, the SIPF-sequence satisfies that $W^{(k)} \in \Psi(W)$ even without being based on the IPF-sequence.

Pseudo-Symmetric Iterative Proportional Fitting (PSIPF)

We define the PSIPF-sequence only in the symmetric setting. The basic idea is that for every square matrix $A \in \Omega(\mathbf{f}, \mathbf{f}, \cdot)$ and any mean function m it follows by the symmetry of m that $m(A, A^T) := [m(a_{ij}, a_{ji})] \in \mathbb{W}(\mathbf{f}, \cdot)$. PSIPF applies this idea to every element of the IPF-sequence, that is PSIPF first determines the whole non-symmetric IPF-sequence, and afterwards symmetrizes each element individually by $W^{\langle\langle k\rangle\rangle} := m(W^{\langle k\rangle}, (W^{\langle k\rangle})^T)$. We choose the geometric mean function m_G for this symmetrization because its factorizing property (Lemma 3.3.15) achieves that each element of the PSIPF-sequence is not just symmetric, but even lies in the set $\Psi(W)$. This idea is made precise in the following definition.

Definition 3.3.20 (PSIPF-sequence)
For $W \in \mathbb{W}$ and $\mathbf{f} \in \mathbb{R}^n$, the PSIPF-sequence $(W^{\langle\langle k\rangle\rangle})$ is defined from the IPF-sequence $(W^{\langle k\rangle})$ as follows for $k \geq 0$:

$$W^{\langle\langle k\rangle\rangle} := m_G(W^{\langle k\rangle}, (W^{\langle k\rangle})^T) = \left[\sqrt{w_{ij}^{\langle k\rangle} w_{ji}^{\langle k\rangle}}\right],$$

which particularly implies that $W^{\langle\langle 0\rangle\rangle} = W$. With $S^{\langle\langle k\rangle\rangle} := \sqrt{Y^{\langle k\rangle} Z^{\langle k\rangle}}$ this is equivalent to the following factorized representation:

$$W^{\langle\langle k\rangle\rangle} = S^{\langle\langle k\rangle\rangle}\ W\ S^{\langle\langle k\rangle\rangle} = \left[\sqrt{y_i^{\langle k\rangle} z_i^{\langle k\rangle}}\ w_{ij}\ \sqrt{y_j^{\langle k\rangle} z_j^{\langle k\rangle}}\right].$$

In particular it holds that $W^{\langle\langle k\rangle\rangle} \in \Psi(W)$.

Note that $W^{\langle\langle k\rangle\rangle}$ is solely defined from the corresponding element of the IPF-sequence $W^{\langle k\rangle}$. In particular, the convergence of PSIPF is implied by the convergence of IPF, as the following theorem shows.

Theorem 3.3.21 (Convergence of PSIPF)
Let $W \in \mathbb{W}$ and $\mathbf{f} \in \mathbb{R}^n_{>0}$ such that $\mathbb{W}(\mathbf{f}, W) \neq \emptyset$. Then the PSIPF-sequence $(W^{\langle\langle k\rangle\rangle})$ converges to $\mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W)$.

Proof. Corollary 3.3.19 gives that $W^{\langle\infty\rangle} = \mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W)$, in particular $\|W^{\langle k\rangle} - W^{\langle\infty\rangle}\|_1 \to 0$. For $i = j$ it holds that $|w_{ij}^{\langle\langle k\rangle\rangle} - w_{ij}^{\langle\infty\rangle}| = |w_{ij}^{\langle k\rangle} - w_{ij}^{\langle\infty\rangle}|$, and for $i \neq j$ that

$$\begin{aligned} |w_{ij}^{\langle\langle k\rangle\rangle} - w_{ij}^{\langle\infty\rangle}| + |w_{ji}^{\langle\langle k\rangle\rangle} - w_{ji}^{\langle\infty\rangle}| &\leq 2 \cdot \max\{|w_{ij}^{\langle k\rangle} - w_{ij}^{\langle\infty\rangle}|, |w_{ji}^{\langle k\rangle} - w_{ji}^{\langle\infty\rangle}|\} \\ &\leq 2 \cdot (|w_{ij}^{\langle k\rangle} - w_{ij}^{\langle\infty\rangle}| + |w_{ji}^{\langle k\rangle} - w_{ji}^{\langle\infty\rangle}|) \end{aligned} \quad ,$$

where the first inequality uses that $W^{\langle\infty\rangle}$ is symmetric, that is $w_{ij}^{\langle\infty\rangle} = w_{ji}^{\langle\infty\rangle}$. Hence,

$$\|W^{\langle\langle k\rangle\rangle} - W^{\langle\infty\rangle}\|_1 = \sum_{i,j} |w_{ij}^{\langle\langle k\rangle\rangle} - w_{ij}^{\langle\infty\rangle}| \leq 2 \cdot \sum_{i,j} |w_{ij}^{\langle k\rangle} - w_{ij}^{\langle\infty\rangle}| = 2\|W^{\langle k\rangle} - W^{\langle\infty\rangle}\|_1 \to 0 \quad ,$$

which proves the convergence of $(W^{\langle\langle k\rangle\rangle})$ to $\mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W)$. □

The convergence result even holds in much more generality: the symmetrized sequence $(m(W^{\langle k\rangle}, (W^{\langle k\rangle})^T))$ converges to $\mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W)$ for *any* mean function m, and along symmetric matrices. However, only for $m = m_G$, that is for the PSIPF-sequence, all matrices lie in $\Psi(W)$.

Symmetric Iterative Proportional Fitting (SIPF)

We define the SIPF-sequence only in the symmetric setting. In contrast to IPF and PSIPF, which both carry out the *RE*-projections (3.9) alternately one after the other, the approach taken by SIPF is to aggregate the two projections to $\mathcal{R}$ and $\mathcal{C}$ simultaneously at each step by taking their entry-wise geometric means.

More general, for any mean function m, we define the **m-sequence** as follows.

Definition 3.3.22 (m-sequence)
For $W \in \mathbb{W}$, $\mathbf{f} \in \mathbb{R}^n$ and any mean function m (applied to matrices entry-wise), the m-sequence is defined by $W^{(0)} := W$ and

$$\begin{aligned} W^{(k+1)} &:= m(\mathcal{P}_{\mathcal{R}}(W^{(k)}), \mathcal{P}_{\mathcal{C}}(W^{(k)})) \\ &= \left[m\left(\tfrac{f_i}{d_i^{(k)}} \cdot w_{ij}^{(k)}, \tfrac{f_j}{d_j^{(k)}} \cdot w_{ij}^{(k)} \right) \right] \\ &= \left[w_{ij}^{(k)} \cdot m\left(\tfrac{f_i}{d_i^{(k)}}, \tfrac{f_j}{d_j^{(k)}} \right) \right] \\ &= \left[w_{ij} \cdot \textstyle\prod_{\ell=0}^{k} m\left(\tfrac{f_i}{d_i^{(\ell)}}, \tfrac{f_j}{d_j^{(\ell)}} \right) \right] \quad , \end{aligned} \tag{3.10}$$

where $\mathbf{d}^{(k)} = (d_i^{(k)}) := W^{(k)}\mathbf{1}$ denotes the positive row (and column) marginals of the symmetric matrix $W^{(k)} = [w_{ij}^{(k)}]$.

For example, the arithmetic mean gives the sequence $W^{(k+1)} := ((\mathcal{P}_{\mathcal{R}}(W^{(k)}) + \mathcal{P}_{\mathcal{C}}(W^{(k)}))/2)$ denoted as the m_A-sequence. In contrast to the PSIPF-sequence, which always converges to the same limit $W^{(\infty)}$ for any choice of m, the limit $W_m^{(\infty)}$ of the m-sequence can differ depending on the choice of m. If $m = \min$ or $m = \max$, then the m-sequence can even converge to an infeasible limit $W_m^{(\infty)} \notin \mathbb{W}(\mathbf{f}, W) \neq \emptyset$, although other choices of m converge to a feasible limit, see Figure 3.3.1 for an example.

The SIPF-sequence is defined by the geometric mean, that is the m_G-sequence. It allows for the following recursive and factorized representations.

Definition 3.3.23 (SIPF-sequence)
For $W \in \mathbb{W}$ and $\mathbf{f} \in \mathbb{R}^n$, the SIPF-sequence $(W^{(k)})$ is defined by $W^{(0)} := W$ and

$$W^{(k+1)} := m_G(\mathcal{P}_{\mathcal{R}}(W^{(k)}), \mathcal{P}_{\mathcal{C}}(W^{(k)})) = \sqrt{\frac{F}{D^{(k)}}} \cdot W^{(k)} \cdot \sqrt{\frac{F}{D^{(k)}}}$$

Note that $W^{(k+1)}$ equals the $\mathbf{f}$-scaling of $W^{(k)}$. The definition implies the following factorization:

$$W^{(k+1)} = S^{(k)} \cdot W \cdot S^{(k)} \qquad \text{for} \qquad S^{(k)} := \sqrt{\prod_{\ell=0}^{k} \frac{F}{D^{(\ell)}}} \quad ,$$

which particularly shows that $W^{(k)} \in \Psi(W)$.

SIPF has already been studied implicitly in the literature for the special case of $\mathbf{f} = \mathbf{1}$, that is for a doubly stochastic limit. One aspect that makes the case $\mathbf{f} = \mathbf{1}$ special is that it represents the projection to the Birkhoff polytope, which allows for a variety of specialized arguments that do not generalize to the case $\mathbf{f} \neq \mathbf{1}$. Zass and Shashua (2005) sketch a proof idea based on the monotony of the matrix permanent. Recently, Knight et al. (2014) provide a rigorous convergence proof based on the monotony of $\prod_i d_i^{(k)} (= \prod_i d_i^{(k)}/f_i)$. However, in the general case of $\mathbf{f} \neq \mathbf{1}$ these and other monotony properties do no longer hold. As a consequence, the proofs for the case $\mathbf{f} = \mathbf{1}$ cannot simply be adapted to the case $\mathbf{f} \neq \mathbf{1}$. Further, in contrast to PSIPF, the convergence of the SIPF-sequence is not implied by the convergence of the IPF-sequence, since there is no specific relation between $W^{(k)}$ and $W^{\langle k \rangle}$. The convergence proof for the SIPF-sequence requires individual arguments, and is a main contribution of this chapter.

3.3.8 Factorized versus non-factorized approaches

All three sequences, here exemplary denoted as (W_k), can be represented in factorized form as $W_k = Y_k \cdot W \cdot Z_k$ by unrolling the recursive definition back to the initial matrix $W = W_0$. This allows for two different strategies how to compute each sequence in practice:

$$
\begin{array}{cccc}
W & W_{m_G}^{(\infty)} & W_{\min}^{(\infty)} & W_{\max}^{(\infty)} \\
\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} \begin{matrix} 1 \\ 2 \\ 2 \end{matrix} &
\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{matrix} 1 \\ 1 \\ 1 \end{matrix} &
\begin{pmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix} \begin{matrix} 0.5 \\ 1 \\ 1 \end{matrix} &
\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0.5 \\ 0 & 0.5 & 0.5 \end{pmatrix} \begin{matrix} 1 \\ 1.5 \\ 1 \end{matrix} \\
\text{(a)} & \text{(b)} & \text{(c)} & \text{(d)}
\end{array}
$$

Figure 3.3.1: Convergence behavior of the m-sequence toward $\mathbf{f} = (1, 1, 1)^T$ for different choices of mean functions m. (a) initial matrix W. (b) the m_G-sequence (SIPF-sequence) provides a feasible solution with marginals $\mathbf{f}$. (c), (d) the min-sequence and the max-sequence converge to incorrect marginals.

- **non-factorized**: just compute W_k from W_{k-1} according to the recursive definition, without storing the factors Y_k and Z_k.
- **factorized**: directly update Y_k and Z_k in each step (with $Y_k = Z_k$ for SIPF). If one is interested in W_k, then W_k can directly be computed as $Y_k W Z_k$.

In the following, we focus on the SIPF-sequence $(W^{(k)})$ and its factorization $W^{(k)} = S^{(k)} W S^{(k)}$, but all arguments hold similarly for the other sequences. The factorized representation has significant advantages over the recursive definition:

- *Efficiency*: only a single vector with n elements (the diagonal of $S^{(k)}$) needs to be stored and updated at each iteration, rather than the full matrix $W^{(k)}$ with n^2 entries.
- *Self-stabilization*: assume we would not choose W as the initial matrix, but *any* other matrix from the set $\Psi(W)$. Then the factorized SIPF-sequence would still converge to the correct limit $W^{(\infty)} \in \Psi(W)$. More precise, any errors $\xi^{(k)} \in \mathbb{R}^n$ at step k affect the computed matrix $\tilde{W}^{(k)}$ in the form of $(S^{(k)} + \text{diag}(\sum_{i=1}^{k} \xi^{(k)})) \cdot W \cdot (S^{(k)} + \text{diag}(\sum_{i=1}^{k} \xi^{(k)}))$, which is still an element of the set $\Psi(W)$. Even in case of arbitrary large sporadic errors, the factorized representation always brings the sequence back toward the correct *unique* biproportional fit $W^{(\infty)}$ because it never deviates from the set $\Psi(W)$ by more than machine precision. Thus, whenever the numerically computed matrix $\tilde{W}^{(k)}$ shows marginals $\mathbf{f}$ it is guaranteed to equal $W^{(\infty)}$ up to machine precision.

  This sense of robustness is *not* provided by the non-factorized approach: here, numerical errors $\Delta^{(k)} \in \mathbb{R}^{n \times n}$ at step k affect the computed matrix by $\tilde{W}^{(k)} = W^{(k)} + \sum_{i=1}^{k} \Delta^{(i)}$. Thus $\tilde{W}^{(k)}$ might be no longer close to $\Psi(W)$, but to $\Psi(W + \Phi^{(k)})$ for some unknown implicit drift $\Phi^{(k)} \in \mathbb{R}^{n \times n}$. This can finally lead to a computed limit matrix $\tilde{W}^{(\infty)} \in \Psi(W + \Phi^{(\infty)})$ that is very different from the correct limit $W^{(\infty)} \in \Psi(W)$, although both provide the correct marginals! There is no way to check for this without determining the factors. Thus, whenever the numerically computed matrix $\tilde{W}^{(k)}$ shows marginals $\mathbf{f}$ it is guaranteed to be the biproportional fit of *some* matrix $W + \Phi^{(k)}$, but there is no way to check whether it is close to $\Psi(W)$, hence whether it is close to $W^{(\infty)}$.

- *Explicit factors*: if our goal is explicitly to determine the factorization $W = S^{-1}XS^{-1}$ for some matrix X of prescribed marginals and a positive diagonal matrix S, then there is obviously no alternative to the factorized approach.

So why not always choose the factorized approach? Unfortunately, there is a fundamental problem with the factorized representation: whenever the limit matrix $W^{(\infty)}$ turns out to be a non-direct biproportional fit $W^{(\infty)} = \lim_{k\to\infty} T^{(k)}WT^{(k)}$, then some entries in $T^{(k)}$ tend to infinity. This makes any factorized approach numerically infeasible. The recommendation is to prefer the factorized approach whenever it is applicable, that is whenever the limit matrix is a direct biproportional fit, $W^{(\infty)} = TWT$. A practical strategy would be to always start with the factorized approach and fall back to the non-factorized approach if some entries in $T^{(k)}$ exceed a certain threshold. A less pragmatic and more interesting alternative is given in Section 3.4.3, which provides a characterization of all cases that satisfy directness.

3.3.9 Background on IPF and iterative projection methods

The origins of the IPF-sequence date back almost one century: according to Bregman (1967b), IPF was proposed by G. V. Sheleikhovskii in the 1930s for calculating passenger flows, although he did not publish it. Kruithof (1937) published IPF with an application to telephone traffic networks. IPF is still known under the names "Kruithoff method" and "Sheleikhovskii method". It was re-discovered in other fields, in several variants, and with different names such as "Furness method", "Evans-Kirby method", "Murchland method", "Osborne method", "Grad method", "Jefferson-Scott method", "Macgill method", and this list is still far from being complete. Lamond and Stuart (1981) show that IPF and all above mentioned variants are special instances of the general framework given by Bregman (1967a). In his work, Bregman studies the convergence of **iterative projection methods**. Such methods aim at finding an element from a non-empty set of the form $\mathcal{F} = \mathcal{C}_1 \cap \ldots \cap \mathcal{C}_\ell$ by repeatedly projecting to $\mathcal{C}_i$, while i cycles through $1, \ldots, \ell$. This method possibly converges for every initial element x_0 to some element $x^* \in \mathcal{F}$. For the special case of orthogonal projections and all $\mathcal{C}_i$'s being linear subspaces, convergence of this method is already proven by von Neumann (1950). This result is generalized from linear subspaces to general closed convex sets by Bregman (1965), while still being limited to orthogonal projections. This setting is known as **Projection Onto Convex Sets (POCS)**. Two years later, Bregman (1967a) further generalizes POCS to non-orthogonal projections, in particular to the family of projections introduced in Section 3.3.5, which were later named after him. His generalization to non-orthogonal projections, here denoted as **Bregman-POCS**, is still relevant for recent work, for example for graphical models (Ghahramani, 2002) and compressed sensing (Yin et al., 2008).

The key how IPF fits to the method of iterative projections is given by Lemma 3.3.12: proportional row and column scaling of a matrix W corresponds to the RE-projection of W to $\mathcal{C}_1 := \Omega(\mathbf{r}, \cdot, W)$ and $\mathcal{C}_2 := \Omega(\cdot, \mathbf{c}, W)$, respectively. This identifies IPF as being an instance of Bregman-POCS with RE-projections.

> **“ IPF equals Bregman-POCS with *RE*-projections. ”**

Thus, Bregman's generalized POCS framework implies convergence of the IPF-sequence to some element $W^{\langle\infty\rangle}$ from the set $\Omega(\mathbf{r}, \mathbf{c}, W) = \mathcal{C}_1 \cap \mathcal{C}_2$ whenever it is non-empty. But why does additionally hold that $W^{\langle\infty\rangle}$ is not just *some* element from $\Omega(\mathbf{r}, \mathbf{c}, W)$, but precisely the RE-projection $\mathcal{P}_{\Omega(\mathbf{r},\mathbf{c},W)}(W)$?

On the RE-optimality of IPF

The IPF-sequence is studied by Sinkhorn (1967) for the case that all entries in W are positive: Sinkhorn proves that for *any* choice of $W \in \mathbb{R}^{m\times n}_{>0}$, $\mathbf{r} \in \mathbb{R}^m_{>0}$, $\mathbf{c} \in \mathbb{R}^n_{>0}$, the IPF-sequence converges to a limit matrix $W^{\langle\infty\rangle} \in \mathbb{R}^{m\times n}_{>0}$ of the form $W^{\langle\infty\rangle} = YWZ$ for positive diagonal matrices Y, Z that are unique up to a scaling factor. Stated differently, $W^{\langle\infty\rangle}$ is the (direct) biproportional fit of W to $\mathbf{r}$ and $\mathbf{c}$. Moreover, it is straightforward to derive from the Lagrangian that $W^{\langle\infty\rangle}$ is closest to W with respect to relative entropy. Hence, $W^{\langle\infty\rangle}$ is not just *some* matrix of marginals $\mathbf{r}$ and $\mathbf{c}$, but precisely the RE-optimal solution $\mathcal{P}_{\Omega(\mathbf{r},\mathbf{c},W)}(W)$.

However, as soon as one allows for zero-entries in W, the problem becomes much harder. For the special case of a doubly stochastic limit ($\mathbf{r} = \mathbf{c} = \mathbf{1}$), Sinkhorn and Knopp (1967) show that convergence only holds if the set of non-zeros $E(W)$ satisfies specific combinatorial properties. Even in case of convergence, the limit $W^{\langle\infty\rangle}$ can show zeros where W has non-zeros, which turns the relative entropy objective into a non-smooth optimization problem. As a consequence, the Lagrangian approach can no longer be applied without further considerations. Some publications overlook this pitfall (see Section 3.4.3). They “simply” apply the Lagrangian approach to the non-negative setting and finally obtain invalid conclusions. Hence, assuming positivity is not just about “simplifying some arguments”, as sloppily stated by Ireland and Kullback (1968), but a totally different scenario. Also the proof by Bregman (1967b) claims to imply the non-negative case, but it deals only superficially with the extension to zero-entries. The proof for the RE-optimality of IPF was rigorously carried over from the positive to the non-negative case by Csiszar (1975), who totally avoids any Lagrangian-type arguments by a measure theoretical approach. Zeros in $W^{\langle\infty\rangle}$ that are not present in W are handled technically sound by considering the Radon-Nikodym derivatives, which are able to deal with absolute continuous measures, that is with $E(W^{\langle\infty\rangle}) \subsetneq E(W)$. The only drawback of this approach is that it does not provide an algorithmic intuition on why optimality holds. What is so special about the IPF-sequence that guarantees that the limit is not just *some* element from $\Omega(\mathbf{r}, \mathbf{c}, W)$, but precisely the unique element from $\Omega(\mathbf{r}, \mathbf{c}, W)$ that is closest to W with respect to relative-entropy-error?

It is a main contribution of this chapter to provide a novel explanation for the optimality of IPF. This is achieved by identifying IPF as a special instance of a more general iterative projection scheme that further guarantees optimality.

Iterative projection method with reflections

Assume that some non-empty set $\mathcal{F} \subseteq \mathbb{R}^N$ can be written as the intersection of finitely many closed convex sets, that is $\mathcal{F} = \mathcal{C}_1 \cap \ldots \cap \mathcal{C}_\ell$. Let $\mathcal{P}_i^h(z)$ denote a Bregman projection of z to $\mathcal{C}_i$. Further we are given some $x_0 \in \mathbb{R}^N$. We define the following two consecutive goals:

- **feasibility goal**: our minimum goal is to determine an arbitrary element $x^* \in \mathcal{F}$. This is also known as the **convex feasibility problem**, as studied for example by Youla (1987) with applications in image restoration.
- **optimality goal**: this goal is stronger than the first. We want to guarantee that x^* is not just *some* element from $\mathcal{F}$, but precisely the projection $x^* = \mathcal{P}_{\mathcal{F}}^h(x_0) \in \mathcal{F}$ for some Bregman projection $\mathcal{P}^h$. This is also known as the **best approximation problem**.

As stated above, the feasibility goal can be approximated by **POCS with Bregman projections (Bregman-POCS)**, which gives the sequence (x_k) defined for $k \geq 1$ by

$$x_k := \mathcal{P}_{[k]}^h(x_{k-1}) \quad , \tag{3.11}$$

where $[k] := 1 + (k - 1 \bmod \ell)$. Observe that (3.11) equals the IPF-sequence for $\mathcal{C}_1 := \mathcal{R}$, $\mathcal{C}_2 := \mathcal{C}$, and $\mathcal{F} := \Omega(\mathbf{r}, \mathbf{c}, W) = \mathcal{R} \cap \mathcal{C}$, which aims at approximating $P_{\mathcal{F}}(W)$ for some given $x_0 := W \in \mathbb{R}^{m \cdot n}$. Bregman (1967a) proves that (3.11) converges to *some* solution $x^* \in \mathcal{F}$, but in general with $x^* \neq \mathcal{P}_{\mathcal{F}}(x_0)$. Note that non-optimality is not just an artifact of non-orthogonality. See Figure 3.3.2 for a simple example in which iterated orthogonal projections do not converge to the optimal solution. Moreover, Figure 3.3.2 shows how this problem can be solved in case of orthogonal projections: at each step, we have to shift the pre-image by adding a specific **reflection term** r_k before applying the projection. This strategy is denoted as **Dykstra's projection algorithm (DPA)**:

$$\begin{aligned} x_k &:= \mathcal{P}_{[k]}^{\perp}(x_{k-1} + r_{k-\ell}) \\ \text{and} \quad r_k &:= x_{k-1} + r_{k-\ell} - x_k \end{aligned} \tag{3.12}$$

with $r_i := 0$ whenever $i \leq 0$. The idea goes back to Dykstra (1983), and was generalized from convex cones to general closed convex sets by Boyle and Dykstra (1985), who prove that the DPA-sequence (x_k) as given by (3.12) converges to the optimal solution $x^* = \mathcal{P}_{\mathcal{F}}^{\perp}(x_0)$. Note that this algorithm has nothing to do with Dijkstra's algorithm from graph theory.

IPF does not fit into this framework, since IPF applies non-orthogonal projections. However, Bauschke and Lewis (2000) generalize Dykstra's idea to a large family of Bregman projections that particularly includes orthogonal projections and *RE*-projections. They introduce a similar reflection term that depends on the function h that induces the Bregman divergence. This defines **Dykstra's projection algorithm with Bregman projections (Bregman-DPA)**

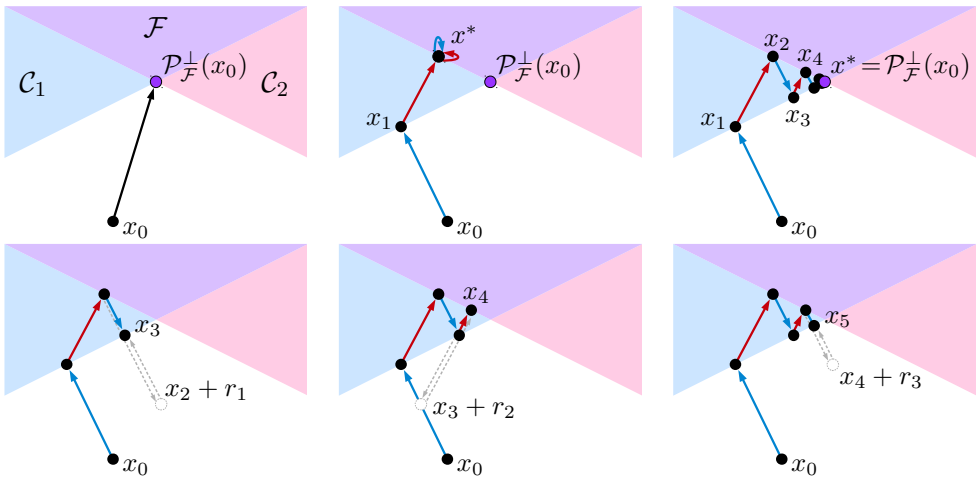


Figure 3.3.2: Iterative orthogonal projections with and without Dykstra's reflection terms. (top left) orthogonal projection of x_0 to $\mathcal{F} = \mathcal{C}_1 \cap \mathcal{C}_2$. This is the preferable approach whenever this projection is directly computable. (top middle) iterative projection method (3.11) with orthogonal projections (POCS). Reaches the feasible solution $x^* \in \mathcal{F}$ after two iterations, but x^* does *not* equal $\mathcal{P}_{\mathcal{F}}^{\perp}(x_0)$. (top right) Dykstra's projection algorithm (3.12) including reflection. Converges to the limit $x^* = \mathcal{P}_{\mathcal{F}}^{\perp}(x_0)$. (bottom row from left to right) details on Dykstra's projection algorithm for $k = 3, 4, 5$, respectively. By definition, $r_1 := x_0 - x_1$, $r_2 := x_1 - x_2$, and $r_3 = x_2 - x_3 + x_0 - x_1$. Projecting $x_{k-1} + r_{k-2}$ instead of x_{k-1} guarantees convergence toward $x^* = \mathcal{P}_{\mathcal{F}}^{\perp}(x_0)$.

by:

$$\begin{aligned} x_k &:= (\mathcal{P}^h_{[k]} \circ \nabla h^*)(\nabla h(x_{k-1}) + r_{k-\ell}) \\ \text{and} \quad r_k &:= \nabla h(x_{k-1}) + r_{k-\ell} - \nabla h(x_k) \end{aligned} \tag{3.13}$$

with $r_i := 0$ whenever $i \leq 0$, and h^* the convex conjugate of h. Bauschke and Lewis (2000) prove convergence of (3.13) to the limit $x^* = \mathcal{P}^h_{\mathcal{F}}(x_0)$. Observe the following two special cases:

- for orthogonal projections, that is for $h = \frac{1}{2}\|\mathbf{x}\|^2$, we get from $\nabla h = \text{id}$ and $\nabla h^* = \text{id}$ that Bregman-DPA (3.13) includes DPA (3.12) as a special case.
- if we **drop** all reflection terms, that is if we re-define $r_k := 0$ for all k, then we get from $\nabla h^* \circ \nabla h = \text{id}$ that Bregman-DPA (3.13) equals Bregman-POCS (3.11). In particular, this shows that IPF equals Bregman-DPA with *RE*-projections, but *with all reflection terms dropped.*

> **“ IPF equals Bregman-DPA with *RE*-projections, but with all reflection terms dropped. ”**

Bauschke and Lewis (2000, Theorem 4.3) show that if all $\mathcal{C}_i$'s are affine, then dropping all reflection terms does not affect the limit. Even stronger, their proof reveals that the sequences (3.13) and (3.11) coincide: Bregman-DPA gives the *same* sequence as Bregman-POCS in case of affine $\mathcal{C}_i$'s. As a consequence, whenever Bregman-POCS is carried out on affine sets, then it is guaranteed to converge not only to *some* element from $\mathcal{F}$, but to $\mathcal{P}^h_{\mathcal{F}}(x_0)$ because the sequence given by Bregman-POCS is exactly the same as the sequence given by Bregman-DPA in this case! However, neither $\mathcal{R}$ nor $\mathcal{C}$ are affine, so this result cannot be applied to IPF.

3.4 Main contributions

This section presents my three main contributions that are already sketched informally in Section 3.2.

3.4.1 Contribution 1: Novel proof of the IPF limit properties

We have just argued that the sequences given by Bregman-DPA and Bregman-POCS coincide if all sets $\mathcal{C}_i$ are affine. My first main contribution in this chapter is a generalization of this result, showing that both sequences coincide even under a weaker notion of affinity, which is satisfied by IPF. As a result, IPF can not only be seen as Bregman-POCS with *RE*-projections, but even stronger as Bregman-DPA with *RE*-projections. The reflection terms can be ignored in case of this weaker notion of affinity, since they do not affect the sequence at all. This

guarantees not only convergence to some feasible solution but even convergence to the optimal solution $W^{\langle\infty\rangle} = \mathcal{P}_{\Omega(\mathbf{r},\mathbf{c},W)}(W)$. This re-interpretation of IPF provides a novel and clear algorithmic intuition for the convergence *and* optimality of IPF.

> **“ IPF equals Bregman-DPA with *RE*-projections. ”**

Let us now make precise what is meant by “weaker notion of affinity”: we do no longer require the sets $\mathcal{C}_i$ to be affine as a whole because it already suffices that every element x_k for $k \geq 1$ of the non-reflected sequence (3.11) lies in the relative interior of $\mathcal{C}_{[k]}$, that is $x_k \in \operatorname{rel\,int}(\mathcal{C}_{[k]})$. This property may be called “local affinity”, since it implies the existence of a small affine neighborhood within $\mathcal{C}_{[k]}$ around each x_k. Local affinity is trivially satisfied if the sets $\mathcal{C}_i$ are affine.

This result is formulated in the following theorem.

Theorem 3.4.1 (Coincidence of Bregman-POCS and Bregman-DPA)
Let $\mathcal{F} = \mathcal{C}_1 \cap \ldots \cap \mathcal{C}_\ell$ be the non-empty intersection of ℓ closed convex sets $\mathcal{C}_i \subseteq \mathbb{R}^d$. For any $\tilde{x}_0 \in \mathbb{R}^d$ let $(\tilde{x}_k)$ denote the sequence generated by Bregman-POCS (3.11) with respect to the Bregman projection $\mathcal{P}^h$ for any suitable (see Section 3.5.1) function h. Further, for $x_0 := \tilde{x}_0$ let (x_k) denote the sequence generated by Bregman-DPA (3.13) with respect to $\mathcal{P}^h$. If it holds that $\tilde{x}_k \in \operatorname{rel\,int}(C_{[k]})$ for all $k \geq 1$, then both sequences coincide, that is $\tilde{x}_k = x_k$ for all $k \geq 0$.

The proof is carried out in Section 3.5.1. Theorem 3.4.1 implies the following corollary.

Corollary 3.4.2 (IPF equals Bregman-DPA)
The IPF-sequence is equal to the sequence (3.13) of Bregman-DPA with RE-projections.

Proof. In Section 3.3.9 we show that the IPF-sequence equals Bregman-POCS with *RE*-projections. All that remains to show in order to apply Theorem 3.4.1 is that the setting of IPF further satisfies the local affinity property. First, recall from the proof of Lemma 3.3.12 that $\mathcal{P}_{\mathcal{R}}(W^{\langle k\rangle}) = \mathcal{P}_{\Omega(\mathbf{r},\cdot,W)}(W^{\langle k\rangle})$ and $\mathcal{P}_{\mathcal{C}}(W^{\langle k\rangle}) = \mathcal{P}_{\Omega(\cdot,\mathbf{c},W)}(W^{\langle k\rangle})$. Rather than thinking of IPF as the alternating projection to $\mathcal{R} = \Omega(\mathbf{r},\cdot,\cdot)$ and $\mathcal{C} = \Omega(\cdot,\mathbf{c},\cdot)$ we can equivalently see it as the alternating projection to $\mathcal{R}_W := \Omega(\mathbf{r},\cdot,W)$ and $\mathcal{C}_W := \Omega(\cdot,\mathbf{c},W)$. Now it follows from the fact that $E(W^{\langle k\rangle}) = E(W)$ and from Example 3.3.9 that

$$W^{\langle k\rangle} \in \begin{cases} \underline{\Omega}(\mathbf{r},\cdot,W) = \operatorname{rel\,int}(\mathcal{R}_W) & \text{, if } k \text{ odd} \\ \underline{\Omega}(\cdot,\mathbf{c},W) = \operatorname{rel\,int}(\mathcal{C}_W) & \text{, if } k \text{ even} \end{cases}.$$

Thus, every element $W^{\langle k\rangle}$ lies in the relative interior of the image domain of the projection of the previous element. This allows to apply Theorem 3.4.1, which gives that dropping the reflection terms does not affect the sequence. □

The most important consequence of Corollary 3.4.2 is that the IPF-sequence converges to the RE-optimal limit whenever $\Omega(\mathbf{r}, \mathbf{c}, W) \neq \emptyset$, just like Bregman-DPA does. This is a novel and intuitive one-line-proof for Theorem 3.3.17.

3.4.2 Contribution 2: Convergence of SIPF

The second main contribution in this chapter is the proof of convergence of the SIPF-sequence, as stated in the following theorem.

Theorem 3.4.3 (Convergence of SIPF)
Let $W \in \mathbb{W}$ and $\mathbf{f} \in \mathbb{R}^n_{>0}$ such that $\mathbb{W}(\mathbf{f}, W) \neq \emptyset$. Then the SIPF-sequence $(W^{(k)})$ converges to $\mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W)$. Further, $\|\mathbf{f} - W^{(k)}\mathbf{1}\|_1 \to 0$ monotonously decreasing.

The proof of Theorem 3.4.3 is carried out in Section 3.5.2. Recall that SIPF is synonymous for iterated $\mathbf{f}$-scaling. Thus, Theorem 3.4.3 allows us particularly to think about $\mathbf{f}$-scaling of some graph $G = (V, E, W)$ as follows: the $\mathbf{f}$-scaled graph matrix $\tilde{W}_\mathbf{f}$ is a first-step approximation of re-weighting all edges such that they yield degree vector $\mathbf{f}$. If $\mathbf{f}$-scaling is iteratively repeated, then this process is now proven to converge to a graph $G^{(\infty)} = (V, E^{(\infty)}, W^{(\infty)})$ of degree vector $\mathbf{f}$, whenever any such graph exists. Precisely, the limit graph is the unique graph that achieves degree vector $\mathbf{f}$ on a subset of edges of E, while being closest to W with respect to relative-entropy. Section 3.6 presents applications of this result.

3.4.3 Contribution 3: Feasibility of SIPF in terms of graph properties

Recall the three convergence results for the symmetric setting: Corollary 3.3.19 for the IPF-sequence, Theorem 3.3.21 for the PSIPF-sequence and Theorem 3.4.3 for the SIPF-sequence. The third main contribution in this chapter studies the applicability of these three convergence results to undirected weighted graphs of weight matrix W. All three results prove convergence of the respective sequence to the same limit $W^{\langle\infty\rangle} = \mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W)$. Further, they all base on the same fundamental **feasibility assumption**: they require the existence of an arbitrary feasible solution $A \in \mathbb{W}(\mathbf{f}, W) \neq \emptyset$. We further distinguish between the case that there exists a **total solution**, that is a feasible solution with $E(A) = E(W)$, and the case that no feasible solution is a total solution, that is $E(A) \subsetneq E(W)$ for all $A \in \mathbb{W}(\mathbf{f}, W)$. Summarized, we study the following three scenarios:

- **total feasibility**: there exists a total solution, that is $A \in \mathbb{W}(\mathbf{f}, W)$ with $E(A) = E(W)$. The maximality result of Lemma 3.5.6 implies that the limit matrix $W^{\langle\infty\rangle}$ is another total solution: $E(W^{\langle\infty\rangle}) = E(W)$. Further, Lemma 3.3.4 shows that the corresponding biproportional fit is direct, that is $W^{\langle\infty\rangle} = SWS$ for some positive diagonal matrix S. This allows to choose the preferable factorized variant for each of the three algorithms.

- **non-total feasibility**: there is no total solution, but there exists a non-total solution, that is some $A \in \mathbb{W}(\mathbf{f}, W)$ with $E(A) \subsetneq E(W)$. This particularly implies that $W^{\langle\infty\rangle}$

cannot be a total solution neither. So the corresponding biproportional fit is not direct, thus some entries in the corresponding sequence of diagonal matrices tend to zero and others to infinity. As a consequence, the factorized approaches cannot be applied, but one has to choose the non-factorized, more error-prone algorithms.

- **infeasibility**: there exists no feasible solution at all, that is $\mathbb{W}(\mathbf{f}, W) = \emptyset$. In this case the sequences may or may not converge, and any limit matrix is guaranteed to *not* provide the desired marginals $\mathbf{f}$.

Which of the three scenarios applies is fully determined just by the existence of a (total or non-total) feasible solution. Can we re-formulate this abstract existence criterion in terms of a more intuitive criterion?

Intuitively, feasibility should be related to some degree of sparsity of the non-zero-entries in W. As stated in Section 3.3.9, if all entries in W are non-zero, then there always exists a feasible (even total) solution for every choice of $\mathbf{f}$. The more entries in W are zero in an inappropriate way, the less likely there exists a feasible solution. Sparsity decreases the degree of freedom for the existence of a proper edge re-weighting. Can we precisely identify this type of sparsity and derive an intuitive criterion for the existence of a solution?

Indeed this is exactly the novel contribution of this section. The type of sparsity is adequately captured by introducing the concept of "weak $\mathbf{f}$-expansion". Weak $\mathbf{f}$-expansion is a notion of weighted vertex expansion that is complementary to existent results from expander graph theory.

Definition 3.4.4 (Weak f-expansion)
Let $G = (V, E, W)$ for $W \in \mathbb{W}$ denote an undirected weighted graph, and $N(S) := \{j \in V \mid \exists i \in S : ij \in E\}$ the vertex neighborhood of $S \subseteq V$. G is a **weak f-expander** for a positive vector $\mathbf{f}$ it it holds for all subsets $S \subseteq V$ that

$$\sum_{i \in N(S)} f_i \quad \geq \quad \sum_{i \in S} f_i \quad . \tag{3.14}$$

A weak $\mathbf{f}$-expander is denoted as being **strict for** S if the inequality in (3.14) is strict for this choice of S.

A trivial example for a weak $\mathbf{f}$-expander is every graph that has self-loops at all vertices. Such a graph is a weak $\mathbf{f}$-expander for every choice of $\mathbf{f} \in \mathbb{R}^n_{>0}$ because $N(S) \supseteq S$ for all S.

The term "weak" in Definition 3.4.4 indicates that weak $\mathbf{f}$-expansion refers to a rather weak sense of weighted graph expansion. To see this, let $\mathbf{f} = \mathbf{1}$. In this case, weak $\mathbf{f}$-expansion claims that $|N(S)| \geq |S|$ for every subset $S \subseteq V$, thus the *neighborhood* of a set is never smaller than the size of the set itself. In contrast to that, in expander graph theory the term "expansion" typically refers to the much stronger assumption that inequality (3.14) holds true

for the **boundary** $\delta(S) := N(S) \setminus S$ in place of the neighborhood $N(S)$. Since the boundary is usually much smaller than the whole neighborhood, the "classical" notion of expander graphs is much more restrictive.

We contribute two results in terms of two propositions (Proposition 3.4.5 and Proposition 3.4.8): the first addresses the existence of any feasible solution at all, the second addresses the existence of a total solution. The reason to call them "Proposition" rather than "Theorem" is that both can be derived from existing results in the literature (although this requires a considerable amount of work).

The first proposition shows that weak $\mathbf{f}$-expansion is exactly the property that a graph has to satisfy in order to provide feasibility $\mathbb{W}(\mathbf{f}, W) \neq \emptyset$, thus in order to guarantee that IPF/PSIPF/SIPF converge to a feasible solution.

Proposition 3.4.5 (Feasibility)
Let $W \in \mathbb{W}$ and $\mathbf{f} \in \mathbb{R}^n_{>0}$. Then $\mathbb{W}(\mathbf{f}, W) \neq \emptyset$ if and only if $\mathcal{G}(W)$ is a weak $\mathbf{f}$-expander.

While attempting to prove Proposition 3.4.5 it turned out that it can be derived from existing results in the literature. For example, it can be derived from a similar result for non-symmetric matrices by Pukelsheim (2014, $(c) \Leftrightarrow (d)$ in Theorem 3), and also from a result for multi-graphs by Behrend (2013, Theorem 6). In network theory a similar result for flows in directed graphs is known as the Gale-Hoffman theorem (Gale, 1957, Hoffman, 1960), which is a weighted variant of Philip Hall's famous Marriage Theorem. However, our restriction to the symmetric and undirected setting allows to re-formulate these abstract results in terms of weak $\mathbf{f}$-expansion, which is much more crisp and intuitive than all other formulations.

Proposition 3.4.5 characterizes whether or not any (total or non-total) solution exists. We now refine this result in order to identify whether even a total solution exists, that is $A \in \mathbb{W}(\mathbf{f}, W)$ with $E(A) = E(W)$. Again this can be rendered in terms of weak $\mathbf{f}$-expansion. The key to the existence of a total solution is that the weak $\mathbf{f}$-expansion of $\mathcal{G}(W)$ must be *strict* for all subsets $S \subseteq V$ whenever strictness is "possible in principle", that is up to "trivial" cases that obviously enforce Equation (3.14) to hold at most with equality. Such trivial cases include $S = \emptyset$ and $S = V$. Another trivial case can be identified for a connected graph that is bipartite with respect to $V = V_1 \cup V_2$. If S is chosen to equal V_1 or V_2, then strictness in (3.14) is impossible, which is easy to see from the fact that mutually $N(V_1) = V_2$ and $N(V_2) = V_1$. For connected graphs, this describes already all cases that enforce equality: two exceptions ($S \in \{\emptyset, V\}$) for non-bipartite graphs, and four exceptions ($S \in \{\emptyset, V, V_1, V_2\}$) for bipartite graphs. For general unconnected graphs, we must further consider combinations of these "trivial" cases across all components: whenever $S \subseteq V$ is purely a combination of such "trivial" cases, one for each connected component, then we cannot achieve strictness for S in (3.14), but only equality. The following definition helps to formulate all exceptions precisely.

Definition 3.4.6 (Non-bipartite decomposition)
Let $G = (V, E)$ denote some graph. Let $V = D_1 \cup \ldots \cup D_K$ be a partition of V into $K \geq 1$ subsets such that for all $\ell = 1, \ldots, K$ it holds that $D_\ell \neq \emptyset$ plus one of the following two statements:

1. D_ℓ refers a non-bipartite connected component of the graph
2. D_ℓ forms together with $D_{\ell'}$ for some $\ell' \in \{\ell - 1, \ell + 1\}$ the unique bipartition $D_\ell \cup D_{\ell'}$ of a bipartite connected component of the graph.

Such a partition $\bigcup_{\ell=1}^{K} D_\ell$ is denoted as a **non-bipartite decomposition** of the graph.

For example, a graph is connected and non-bipartite if and only if $K = 1$, that is $V = D_1$. A connected bipartite graph has the bipartite decomposition $V = D_1 \cup D_2$. However, a similar bipartite decomposition for $K = 2$ can also characterize another graph that consists of two separate components, both of which are non-bipartite.

The following facts on the non-bipartite decomposition follow right from the definitions:

Fact 3.4.7 (Properties of the non-bipartite decomposition)
Let $\bigcup_{\ell=1}^{K} D_\ell$ denote a non-bipartite decomposition of a graph $G = (V, E)$. Then all the following statements hold true:

(i) K equals the number of non-bipartite connected components plus twice the number of bipartite connected components.

(ii) the non-bipartite decomposition is unique (up to re-indexing the sets D_ℓ).

(iii) if $S \subseteq V$ is a combination of the D_ℓ's, that is $S = \bigcup_{i \in I} D_i$ for some $I \subseteq \{1, \ldots, K\}$, then Equation (3.14) is forced to hold with equality for S.

Finally, our second proposition shows that a total solution exists if and only if all choices of $S \subseteq V$ that are not a "trivial" combination in the sense of Fact 3.4.7 (iii) provide strict expansion.

Proposition 3.4.8 (Total feasibility)
Let $\mathcal{G}(W) = (V, E, W)$ denote a graph and $\bigcup_{i=\ell}^{K} D_\ell$ its non-bipartite decomposition. Further let $\mathbf{f} \in \mathbb{R}^n_{>0}$. Then there exists $A \in \mathbb{W}(\mathbf{f}, W)$ with $E(A) = E(W)$ if and only if $\mathcal{G}(W)$ is a weak $\mathbf{f}$-expander that is strict for all $S \notin \{\bigcup_{\ell \in I} D_\ell \mid I \subseteq \{1, \ldots, K\}\}$.

As before, Proposition 3.4.8 can be seen as a re-formulation of more general existing results. It can be derived from results on general rectangular matrices by Brualdi (1968), formulated

in terms of constraints on sub-matrices. It can also be derived from results on multi-graphs by Behrend (2013, Theorem 7), formulated in terms of tri-partitions of the vertex set. The special case of connected non-bipartite graphs, that is $K = 1$, is further equivalent to considering symmetric "connected matrices" as by Pukelsheim (2014, Theorem 2). However, it takes a considerable effort to transform any of these results to the plain formulation presented in Proposition 3.4.8.

To give an example on how to apply Proposition 3.4.8, we show that there always exists a total solution if all entries in W are positive. Such W characterizes a complete graph with arbitrary positive weights on all its edges, including self-loops. As argued above, such a graph is a weak $\mathbf{f}$-expander for every choice of $\mathbf{f}$. In order to further derive the existence of a total solution, it remains to show that the expansion is strict whenever possible in principle. Since the complete graph is connected and non-bipartite, exceptions from requiring strictness are only allowed for $S \in \{\emptyset, V\}$. For all $S \notin \{\emptyset, V\}$, we get that $N(S) \supsetneq S$, which immediately implies that Equation 3.14 holds with strict inequality. This gives a short and intuitive proof for why a total solution must exist in this case. Stated differently, this result also implies that every symmetric matrix $W \in \mathbb{R}^{n \times n}_{>0}$ factorizes as $W = SBS$, where S is some positive diagonal matrix, and B can be constrained to any positive marginals $\mathbf{f} \in \mathbb{R}^n_{>0}$.

3.5 Proofs and technical details

In this section we prove convergence and *RE*-optimality of the IPF sequence and of the SIPF sequence, respectively. The last subsection outlines the proof for the total feasibility result.

3.5.1 IPF convergence

We now provide the proof of Theorem 3.4.1. This theorem is a generalization of a result by Bauschke and Lewis (2000, Theorem 4.3) from sequences in affine spaces to sequences that always project to the relative interior. We say that function $h : \mathbb{R}^d \to \mathbb{R}$ is **suitable** if it satisfies the assumptions in Bauschke and Lewis (2000) of being *Legendre*, *co-finite*, and *very strictly convex*. The precise definitions of these terms are not further needed in the following, but it is worth mentioning that they are satisfied by several different Bregman projections, in particular by orthogonal projections and *RE*-projections. Each of these two is induced by a suitable function h as a Bregman divergence. Let $(\tilde{x}_k)$ denote the sequence generated by Bregman-POCS (3.11) for some $\tilde{x}_0 \in \mathbb{R}^d$, that is

$$\tilde{x}_k \;:=\; \mathcal{P}^h_{[k]}(\tilde{x}_{k-1}) \quad .$$

Further, let (x_k) denote the sequence generated by Bregman-DPA (3.13) for $x_0 = \tilde{x}_0$, that is

$$\begin{aligned} x_k &:= (\mathcal{P}^h_{[k]} \circ \nabla h^*)(\nabla h(x_{k-1}) + r_{k-\ell}) \\ \text{and} \quad r_k &:= \nabla h(x_{k-1}) + r_{k-\ell} - \nabla h(x_k) \quad . \end{aligned}$$

The key assumption from which it finally follows that $(\tilde{x}_k)$ and (x_k) coincide is that "*for every $k \geq 1$ there exists a ball around $\tilde{x}_k$ in $\mathcal{C}_{[k]}$ that contains an affine basis of* $\operatorname{aff}(\mathcal{C}_{[k]})$ *together with its negates*".

Let us give some intuition on how this property can be helpful: if for a linear subspace $\mathbb{L} \subseteq \mathbb{R}^d$ and some $q \in \mathbb{R}^d$ it holds that $\langle x, q\rangle \leq 0$ for all $x \in \mathbb{L}$, then it follows that $\langle x, q\rangle = 0$ for all $x \in \mathbb{L}$. This can be seen from the fact that particularly every basis vector b of $\mathbb{L}$ satisfies that $\langle b, q\rangle \leq 0$ and $\langle -b, q\rangle \leq 0 \Leftrightarrow \langle b, q\rangle \geq 0$, which gives together that $\langle b, q\rangle = 0$. Now the important thing to note is that it is already sufficient if this inequality holds just within an arbitrary small ball around $\mathbf{0}$, say, for all $x \in B(\mathbf{0}, \varepsilon)$, since this ball already contains a smaller-scaled basis of $\mathbb{L}$ plus its negates. So we get that the local information of $\langle x, q\rangle \leq 0$ for all $x \in B(\mathbf{0}, \varepsilon)$ implies the global information that $\langle x, q\rangle = 0$ for all $x \in \mathbb{L}$.

This insight is re-formulated in the following lemma for $\mathbb{L}$ defined as the difference space of an affine subspace $\mathcal{A} \subseteq \mathbb{R}^d$.

Lemma 3.5.1 (Implied global orthogonality)
Let $\mathcal{A} \subseteq \mathbb{R}^N$ denote an affine subspace. Then it holds for all $q \in \mathbb{R}^d$ and $a \in \mathcal{A}$:

$$\forall x \in \mathcal{A} \ : \ \langle x - a, q\rangle = 0 \qquad \Leftrightarrow \qquad \exists \varepsilon > 0 : \forall x \in \mathcal{B}_{\mathcal{A}}(a, \varepsilon) \ : \ \langle x - a, q\rangle \leq 0.$$

The statement holds true if $\leq$ is replaced by $\geq$.

Proof. The direction "$\Rightarrow$" is trivial, so we only have to prove "$\Leftarrow$": let $\mathbb{L}$ denote the linear subspace that corresponds to $\mathcal{A}$ with the origin translated to $a \in \mathcal{A}$, thus, $x \in \mathcal{A} \Leftrightarrow x - a \in \mathbb{L}$. Let $b_1, \ldots, b_c$ denote a basis of $\mathbb{L}$. With $b'_i := \varepsilon \cdot b_i / (\max_i\{\|b_i\|\})$ for $i = 1, \ldots, c$ we get that $b'_1, \ldots, b'_c$ is also a basis of $\mathbb{L}$, and that $a \pm b'_i \in \mathcal{B}_{\mathcal{A}}(a, \varepsilon)$. Thus, $\langle b'_i, q\rangle \leq 0$ and $\langle -b'_i, q\rangle \leq 0$, which implies that $\langle b'_i, q\rangle = 0$ for all i. Thus, we get for all $x \in \mathcal{A}$ from the representation $x = a + \sum_{i=1}^c \alpha_i b'_i$ for some $\alpha_1, \ldots, \alpha_c \in \mathbb{R}$ that $\langle x - a, q\rangle = \sum_{i=1}^c \alpha_i \langle b'_i, q\rangle = 0$. □

Lemma 3.5.1 is the core tool that we need to generalize (Bauschke and Lewis, 2000, Theorem 4.3) from affine spaces to locally affine sequences.

Proof (of Theorem 3.4.1). The proof shows by induction that $x_k = \tilde{x}_k$ for all $k \geq 0$. By definition, $x_0 = \tilde{x}_0$. Now fix k with $1 \leq k \leq \ell$. By induction, $x_{k-1} = \tilde{x}_{k-1}$, and by definition, $r_{k-\ell} = 0$. This gives with $\nabla h^* \circ \nabla h = \mathrm{id}$ that

$$x_k = (\mathcal{P}^h_{[k]} \circ \nabla h^*)(\nabla h(x_{k-1}) + 0) = (\mathcal{P}^h_{[k]} \circ \mathrm{id})(x_{k-1}) = \mathcal{P}^h_{[k]}(\tilde{x}_{k-1}) = \tilde{x}_k \quad .$$

Now fix $k > \ell$ with by induction $x_i = \tilde{x}_i$ for all $i < k$. Let $\mathcal{A}_i := \operatorname{aff}(\mathcal{C}_i)$ denote the affine hull of $\mathcal{C}_i$ for $i = 1, \dots, \ell$. Note that $\mathcal{C}_{[k]} = \mathcal{C}_{[k-\ell]}$ and $\mathcal{A}_{[k]} = \mathcal{A}_{[k-\ell]}$. From $\tilde{x}_{k-\ell} \in \operatorname{rel\,int}(\mathcal{C}_{[k]})$ we get that there exists $\varepsilon_{k-\ell} > 0$ such that $\mathcal{B}_{\mathcal{A}_{[k]}}(\tilde{x}_{k-\ell}, \varepsilon_{k-\ell}) \subseteq \mathcal{C}_{[k]}$. Because of $x_{k-\ell} = \tilde{x}_{k-\ell}$ we have that $\mathcal{B}_{\mathcal{A}_{[k]}}(x_{k-\ell}, \varepsilon_{k-\ell}) \subseteq \mathcal{C}_{[k]}$. From Bauschke and Lewis (2000, Equation 1 in Theorem 3.2) we get that for all $c \in \mathcal{C}_{[k]}$ it holds that $\langle c - x_{k-\ell}, r_{k-\ell} \rangle \leq 0$. So we can apply Lemma 3.5.1 in order to get that $\langle a - x_{k-\ell}, r_k \rangle = 0$ for all $a \in \mathcal{A}_{[k]}$. For the underlying difference space $\mathbb{L}_{[k]} = \mathcal{A}_{[k]} - x_{k-\ell}$ we are free to choose any other element $z \in \mathcal{A}_{[k]}$ for its representation, that is $\mathcal{A}_{[k]} - x_{k-\ell} = \mathcal{A}_{[k]} - z$. Choosing $z := x_k$ gives that $\langle a - x_k, r_k \rangle = 0$ for all $a \in \mathcal{A}_{[k]} \supseteq \mathcal{C}_{[k]}$. Now we have that $x_{k-1} = \tilde{x}_{k-1}$, and we claim that

$$\tilde{x}_k = \mathcal{P}^h_{[k]}(\tilde{x}_{k-1}) \quad \overset{!}{=} \quad (\mathcal{P}^h_{[k]} \circ \nabla h^*)(\nabla h(x_{k-1}) + r_{k-\ell}) = x_k. \tag{$\star$}$$

Applying the characterization

$$a' = \mathcal{P}^h_{[k]}(a) \quad \Leftrightarrow \quad \forall c \in \mathcal{C}_{[k]} \ : \ \big\langle c - a', \nabla h(a) - \nabla h(a') \big\rangle \leq 0 \tag{$\divideontimes$}$$

in the direction of "$\Rightarrow$" with $a := \nabla h^*(\nabla h(x_{k-1}) + r_{k-\ell})$ and corresponding $a' := x_k$ to the right side in ($\star$) gives:

$$\begin{array}{rcll} 0 & \geq & \langle c - x_k, \nabla h(\nabla h^*(\nabla h(x_{k-1}) + r_{k-\ell})) - \nabla h(x_k) \rangle & \\ & = & \langle c - x_k, \nabla h(x_{k-1}) + r_{k-\ell} - \nabla h(x_k) \rangle & \nabla h \circ \nabla h^* = \mathrm{id} \\ & = & \langle c - x_k, \nabla h(\tilde{x}_{k-1}) - \nabla h(x_k) \rangle + \langle c - x_k, r_{k-\ell} \rangle & x_{k-1} = \tilde{x}_{k-1} \\ & = & \langle c - x_k, \nabla h(\tilde{x}_{k-1}) - \nabla h(x_k) \rangle & \langle c - x_k, r_{k-\ell} \rangle = 0 \end{array}$$

From ($\divideontimes$) again, now in direction of "$\Leftarrow$" with $a := \tilde{x}_{k-1}$ and $a' = x_k$, we get the result $x_k = \mathcal{P}^h_{[k]}(\tilde{x}_{k-1})$, hence $x_k = \tilde{x}_k$.

By induction, the sequences (x_k) and $(\tilde{x}_k)$ coincide. □

3.5.2 SIPF convergence

In this section we prove convergence of the SIPF-sequence, that is the m-sequence for $m = m_G$. The proof is based on four lemmas that are of their own interest because they even hold for $m \neq m_G$. Throughout this section, $(W^{(k)})$ denotes the m-sequence of W, where m is always clear from the context. Further, let $\mathbf{d}^{(k)} := W^{(k)}\mathbf{1}$.

The first lemma guarantees L_1-monotony for every m-sequence.

Lemma 3.5.2 (L_1-monotony)
For all $W \in \mathbb{W}$ and mean function m, the m-sequence of W implies that $\|\mathbf{f} - \mathbf{d}^{(k)}\|_1$ is monotonously decreasing.

Proof. For $k \geq 0$ we get with $s_i := f_i/d_i^{(k)}$ that

$$\begin{aligned}
\|\mathbf{f} - \mathbf{d}^{(k+1)}\|_1 &= \textstyle\sum_i |f_i - d_i^{(k+1)}| \\
&= \textstyle\sum_i |\sum_j w_{ij}^{(k)} s_i - \sum_j w_{ij}^{(k)} m(s_i, s_j)| \\
&\leq \textstyle\sum_i \sum_j w_{ij}^{(k)} |s_i - m(s_i, s_j)| \\
&= \textstyle\sum_i \sum_{j>i} w_{ij}^{(k)} \left(|s_i - m(s_i, s_j)| + |s_j - m(s_j, s_i)|\right) \\
&\overset{(\star)}{=} \textstyle\sum_i \sum_{j>i} w_{ij}^{(k)} |s_i - s_j| \\
&\leq \textstyle\sum_i \sum_{j>i} w_{ij}^{(k)} \left(|s_i - 1| + |s_j - 1|\right) \\
&\leq \textstyle\sum_i \sum_j w_{ij}^{(k)} |s_i - 1| \\
&= \textstyle\sum_i |s_i - 1| \cdot d_i^{(k)} = \sum_i |f_i - d_i^{(k)}| = \|\mathbf{f} - \mathbf{d}^{(k)}\|_1 \quad ,
\end{aligned}$$

where equality $(\star)$ holds true because of $m(s_i, s_j) = m(s_j, s_i) \in [\min(s_i, s_j), \max(s_i, s_j)]$. □

The second lemma bounds the "volume" $\sum_{ij} w_{ij}^{(k)} = \|\mathbf{d}^{(k)}\|_1$ at the k'th iteration from above or below by $\|\mathbf{f}\|_1$ if the mean function is sub-arithmetic or super-arithmetic.

Lemma 3.5.3 (Volume bounds)
For all $W \in \mathbb{W}$ and mean function m, the m-sequence of W satisfies for all $k \geq 1$ that

$$\begin{array}{lrcll}
\textit{(i)} & \|\mathbf{d}^{(k)}\|_1 & \leq & \|\mathbf{f}\|_1 & \textit{if } m \textit{ is sub-arithmetic} \\
\textit{(ii)} & \|\mathbf{d}^{(k)}\|_1 & = & \|\mathbf{f}\|_1 & \textit{if } m = m_A \\
\textit{(iii)} & \|\mathbf{d}^{(k)}\|_1 & \geq & \|\mathbf{f}\|_1 & \textit{if } m \textit{ is super-arithmetic}
\end{array}$$

If m is strict in (i) or (iii), then equality holds if and only if $f_i/d_i^{(k)} = f_j/d_j^{(k)}$ for all $w_{ij} \neq 0$.

Proof. (i) For $k \geq 0$ we get with $s_i := f_i/d_i^{(k)}$ that

$$\begin{aligned}
\|\mathbf{f}\|_1 - \|\mathbf{d}^{(k+1)}\|_1 &= \textstyle\sum_i (\sum_j w_{ij}^{(k)} s_i - \sum_j w_{ij}^{(k)} m(s_i, s_j)) \\
&= \textstyle\sum_i \sum_{j>i} w_{ij}^{(k)} \underbrace{(s_i - m(s_i, s_j) + s_j - m(s_j, s_i))}_{\geq 0} \quad ,
\end{aligned}$$

where non-negativity of each summand follows from the fact that for $x, y \in \mathbb{R}_{\geq 0}$ and sub-

arithmetic m it holds that

$$x + y - 2\,m(x,y) \quad \geq \quad x + y - 2\,(x+y)/2 \quad = \quad 0 \quad . \qquad (\star)$$

If m is strictly sub-arithmetic, then the inequality in $(\star)$ holds with equality if and only if $x = y$. This implies that $\|\mathbf{f}\|_1 - \|\mathbf{d}^{(k+1)}\|_1 \geq 0$ holds with equality if and only if it $s_i = s_j$ for all i, j such that $w_{ij}^{(k)} \neq 0$, that is $w_{ij} \neq 0$. (ii) follows from (i) and the fact that $(\star)$ always holds with equality for $m = m_A$. (iii) equals (i) with all inequalities flipped. □

Every sequence in a bounded subset $S \subseteq \mathbb{R}^d$ has at least one limit point. The third lemma characterizes all limit points of m-sequences.

Lemma 3.5.4 (Limit points)
Every m-sequence is bounded, so it has at least one limit point W^*. If $\|\mathbf{f} - \mathbf{d}^{(k)}\|_1 \to 0$, then every limit point W^* satisfies $W^*\mathbf{1} = \mathbf{f}$. If further $m = m_G$, then W^* is the (unique) biproportional fit of W to row and column marginals $\mathbf{f}$, and it holds that $\lim_{k\to\infty} W^{(k)} = W^*$.

Proof. Monotony (Lemma 3.5.2) implies that there exists $L > 0$ such that $w_{ij}^{(k)} \leq L$ for all i, j, k. Hence $(W^{(k)})$ is bounded and thus has at least one limit point W^* in compact $[0, L]^{n\times n}$. We now show that if $\|\mathbf{f} - \mathbf{d}^{(k)}\|_1 \to 0$, then every limit point W^* of the m-sequence satisfies $W^*\mathbf{1} = \mathbf{f}$. Let $(W^{(k_i)})$ denote any subsequence that converges to W^*. Then

$$\|\mathbf{f} - W^*\mathbf{1}\|_1 \;\leq\; \|\mathbf{f} - \mathbf{d}^{(k_i)}\|_1 + \|\mathbf{d}^{(k_i)} - W^*\mathbf{1}\|_1 \;=\; \underbrace{\|\mathbf{f} - \mathbf{d}^{(k_i)}\|_1}_{\to\,0} + \underbrace{\|(W^{(k_i)} - W^*)\mathbf{1}\|_1}_{\to\,0} \;\to\; 0 \quad .$$

Now assume that $m = m_G$. Then $W^{(k)} = T^{(k)}WT^{(k)}$ for some $T^{(k)} \in \mathrm{diag}(\mathbb{R}^n_{>0})$, which gives that any limit point W^* is a biproportional scaling of W, that is $W^* = \lim_{i\to\infty} T^{(k_i)}WT^{(k_i)}$ for a subsequence $(W^{(k_i)})$. Because of $W^*\mathbf{1} = \mathbf{f}$ we get that W^* is unique by the uniqueness of biproportional fits (Lemma 3.3.2). So W^* is the only limit point, and any bounded sequence with a single limit point must converge to it, hence $W^{(k)} \to W^*$. □

The fourth lemma proves strong convergence under relative-entropy error if the volumes bound each other.

Lemma 3.5.5 (Strong convergence)
For any $\mathbf{x} := (x_1, \ldots, x_n) \in \mathbb{R}^n_{>0}$ *and* $\mathbf{a} := (a_1, \ldots, a_n) \in \mathbb{R}^n_{>0}$ *with* $\sum_i x_i \leq \sum_i a_i$ *let* $f(\mathbf{x}) := \sum_i a_i \log \frac{a_i}{x_i}$. *Then*

$$f(\mathbf{x}) \geq 0 \text{ with equality if and only if } \mathbf{x} = \mathbf{a}. \tag{3.15}$$

Further, for any sequence $(\mathbf{x}^{(k)})_{k\geq 0}$ *in* $\mathbb{R}^n_{>0}$ *with* $\sum_i x_i^{(k)} \leq \sum_i a_i$ *it holds that*

$$\lim_{k\to\infty} f(\mathbf{x}^{(k)}) = 0 \quad \Leftrightarrow \quad \lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{a}. \tag{3.16}$$

Proof. (3.15) follows from non-negativity and coincidence (see page 100) of the Bregman divergence $RE(\mathbf{a}\|\mathbf{x}) = \sum_i a_i \log(a_i/x_i) - a_i + x_i \geq 0$, thus $\sum_i a_i \log(a_i/x_i) \geq \sum_i a_i - \sum_i x_i \geq 0$ with equality if and only if $\mathbf{a} = \mathbf{x}$. We now prove (3.16). "$\Leftarrow$" follows from continuity of f, so it remains to show "$\Rightarrow$". From $\sum_i x_i^{(k)} \leq \sum_i a_i =: a$ we get that $x_i^{(k)} \in (0, a]$ for all $i \in \{1, \ldots, n\}$ and $k \geq 0$. Compactness of $[0, a]^n$ implies that all limit points of $S := (\mathbf{x}^{(k)})$ lie within $[0, a]^n$, and that there exists at least one. Let $\mathbf{c}$ denote a limit point of S and $(\mathbf{y}^{(k)})$ a subsequence that converges to $\mathbf{c}$. Note that $\lim_{\varepsilon\to 0} a_i \log(a_i/\varepsilon) = \infty$. Hence we get by $f(\mathbf{y}^{(k)}) \to 0$ and by continuity of f that $\mathbf{c} \in (0, a]^n$ and $f(\mathbf{c}) = 0$. Now (3.15) implies that $\mathbf{c} = \mathbf{a}$ is the unique limit point of S, thus $\mathbf{x}^{(k)} \to \mathbf{a}$. □

Now we are ready to prove Theorem 3.4.3. The proof is inspired by ideas of Pretzel (1980).

Proof (of Theorem 3.4.3). The only missing ingredient is to prove that $\|\mathbf{f} - \mathbf{d}^{(k)}\|_1 \to 0$, which is now our goal. By assumption there exists some $A = [a_{ij}] \in \mathbb{W}(\mathbf{f}, W)$. Equation (3.10) gives that $w_{ij}^{(k+1)} = w_{ij} \cdot u_{ij}^{(k)}$ with $u_{ij}^{(k)} = \prod_{\ell=0}^{k} m\left(s_i^{(\ell)}, s_j^{(\ell)}\right) \neq 0$ and $s_i^{(\ell)} := f_i / d_i^{(\ell)}$ for all $i, j \in \{1, \ldots, n\}$ and $k \geq 0$. From $w_{ij} = 0 \Leftrightarrow w_{ij}^{(k)} = 0 \Rightarrow a_{ij} = 0$ we get that

$$\begin{aligned} v^{(k+1)} &:= \textstyle\sum_{i,j} a_{ij} \log(w_{ij}^{(k+1)}/w_{ij}) = \sum_{i,j} a_{ij} \log u_{ij}^{(k)} \\ &= \textstyle\sum_{i,j} a_{ij} \sum_{\ell=0}^{k} \log m\left(s_i^{(\ell)}, s_j^{(\ell)}\right) \quad . \end{aligned}$$

For $k \geq 1$ this gives for $m = m_G$ that

$$\begin{aligned} v^{(k+1)} - v^{(k)} &= \textstyle\sum_{i,j} a_{ij} \log m\left(s_i^{(k)}, s_j^{(k)}\right) \\ &= \textstyle\frac{1}{2} \sum_{i,j} a_{ij} (\log s_i^{(k)} + \log s_j^{(k)}) \\ &= \textstyle\frac{1}{2} \sum_i f_i \log s_i^{(k)} + \frac{1}{2} \sum_j f_j \log s_j^{(k)} \\ &= \textstyle\sum_i f_i \log f_i / d_i^{(k)} \quad . \end{aligned}$$

Since m_G is sub-arithmetic, we get from Lemma 3.5.3 that $\sum_i d_i^{(k)} \leq \sum_i f_i$. This allows to apply Lemma 3.5.5 which gives that $v^{(k+1)} \geq v^{(k)}$, thus, $(v^{(k)})_{k\geq 0}$ is monotonously increasing. Further $v^{(k)}$ is bounded from above because $w_{ij}^{(k)} \leq \|\mathbf{d}^{(k)}\|_1 \leq \|\mathbf{f}\|_1 = \sum_{i,j} a_{ij}$ implies together with $w_{min} := \min\{w_{ij} \mid w_{ij} > 0\}$ that

$$\begin{aligned} v^{(k)} &= \textstyle\sum_{i,j} a_{ij} \log(w_{ij}^{(k)}/w_{ij}) \leq \sum_{i,j} a_{ij} \log(\|\mathbf{f}\|_1/w_{min}) \\ &= \|\mathbf{f}\|_1 \cdot \log(\|\mathbf{f}\|_1/w_{min}) \ < \ \infty. \quad (\divideontimes) \end{aligned}$$

It follows that $\lim_{k\to\infty} v^{(k)}$ exists, which implies that $\sum_i f_i \log f_i/d_i^{(k)} = v^{(k+1)} - v^{(k)} \to 0$ and hence with Lemma 3.5.5 that $d_i^{(k)} \to f_i$ for all $i \in \{1, \ldots, n\}$. Thus $\|\mathbf{f} - \mathbf{d}^{(k)}\|_1 \to 0$ with monotony given by Lemma 3.5.2. This proves that $\mathbb{W}(\mathbf{f}, W) \neq \emptyset$ is sufficient to let the m_G-sequence converge to some $W^{(\infty)} \in \mathbb{W}(\mathbf{f}, W)$. By Lemma 3.5.4 we get that $W^{(\infty)}$ is the unique biproportional fit of W to $\mathbf{f}$, hence the same limit as for the IPF-sequence. In particular this implies that $W^{(\infty)}$ is RE-optimal, thus $W^{(\infty)} = \mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W)$. □

Corollary 3.5.6 (Maximality)
For all $A \in \mathbb{W}(\mathbf{f}, W)$ it holds that $E(A) \subseteq E(\mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W))$.

Proof. For all $A \in \mathbb{W}(\mathbf{f}, W)$, equation $(\divideontimes)$ implies that

$$a_{ij} \log(w_{ij}^{(k)}/w_{ij}) + (\|\mathbf{f}\|_1 - a_{ij}) \log(\|\mathbf{f}\|_1/w_{min}) \geq v^{(k)} \geq v^{(1)} > -\infty \quad ,$$

thus $a_{ij} \neq 0 \Rightarrow \lim_{k\to\infty} w_{ij}^{(k)} \neq 0$. □

We denote this property of $W^{(\infty)} = \mathcal{P}_{\mathbb{W}(\mathbf{f},W)}(W)$ as having **maximum possible support**. In particular we get with Lemma 3.3.4 that the biproportional fit $W^{(\infty)}$ is direct if and only if there exists some $A \in \mathbb{W}(\mathbf{f}, W)$ with $E(A) = E(W)$.

3.5.3 Feasibility results

This section presents details on the proof of Proposition 3.4.8. It is straightforward to derive Proposition 3.4.8 from the following Proposition 3.4.8* by applying the latter individually to each connected component.

Proposition 3.4.8* (Total feasibility of a connected graph)
Let $G = (V, E, W)$ denote a connected graph. Further let $\mathbf{f} \in \mathbb{R}^n_{>0}$. Then there exists $A \in \mathbb{W}(\mathbf{f}, W)$ with $E(A) = E(W)$ if and only if G is a weak $\mathbf{f}$-expander that is strict for all $S \notin \{\emptyset, V\}$ if G is non-bipartite, or strict for all $S \notin \{\emptyset, V_1, V_2, V\}$ if G is bipartite with respect to $V = V_1 \cup V_2$.

We prove Proposition 3.4.8* by a reduction to Theorem 7 of Behrend (2013), who formulates a related result in terms of tri-partitions of multi-graphs. We need some additional notation for sketching how to carry out this reduction: for $i \in V$ and $X \subseteq V$ we write $i \sim X$ if i is adjacent to a vertex from X, that is if $ix \in E$ for some $x \in X$. Otherwise we write $i \not\sim X$. For $X, Y \subseteq V$ let $G[X,Y] := \{ij = ji \in E \mid (i,j) \in X \times Y\}$ denote the set of edges connecting some vertex in X to some vertex in Y. $B \subseteq V$ is an **independent set** if and only if $G[B,B] = \emptyset$. Observe that for all $X, Y \subseteq V$ it holds that $G[X \cup Y, Y] = \emptyset \Leftrightarrow G[X,Y] = G[Y,Y] = \emptyset$. For any set $B \subseteq V$, we call any disjoint $\tilde{B} \subseteq V \setminus B$ satisfying $G[\tilde{B}, B] = \emptyset$ a **non-adjacent opponent of** B. This also defines their **rest** $R(B, \tilde{B}) := V \setminus (B \cup \tilde{B})$, which partitions V into $V = B \cup \tilde{B} \cup R(B, \tilde{B})$. The set $B^* := \{i \in V \setminus B \mid i \not\sim B\}$ is the (unique) **maximum non-adjacent opponent** of B. Thus, all vertices from the rest $R(B, B^*)$ are adjacent to B. Any other non-adjacent opponent $\tilde{B}$ of B satisfies that $\tilde{B} \subset B^*$. Let $\delta B := \{i \in V \setminus B \mid \exists j \in B : ij \in E\}$ denote the **vertex boundary** of B. It holds that $\delta B = N(B) \setminus B$. Further $\delta B = N(B)$ if and only if B is an independent set. For any set B and its maximum non-adjacent opponent B^* their rest is $R(B, B^*) = \delta B$.

Proof (sketch) of Proposition 3.4.8*. We start with Theorem 7 of Behrend (2013), restricted to a connected graph with $n \geq 2$ vertices. This is equivalent to the tri-partition statement (i) below. We then transform (i) along a sequence of equivalent statements (i) $\Leftrightarrow$ (ii) $\Leftrightarrow$ (iii) $\Leftrightarrow$ (iv) $\Leftrightarrow$ (v) to the more intuitive statement (v) in terms of $\mathbf{f}$-expansion, which is equivalent to Proposition 3.4.8*.

Let $G = (V, E, W)$ for $W \in \Omega$ be connected, and $\mathbf{f} \in \mathbb{R}^n_{>0}$. Then there exists a total solution in $\mathbb{W}(\mathbf{f}, W)$ if and only if any of the following equivalent statements holds:

(i) $\sum_{i \in A} f_i \geq \sum_{j \in B} f_j$ for all $A, \tilde{A}, B$ with $V = A \cup \tilde{A} \cup B$ and $G[\tilde{A} \cup B, B] = \emptyset$, where equality holds if and only if additionally $G[A, A \cup \tilde{A}] = \emptyset$.

(ii) $\sum_{i \in A} f_i \geq \sum_{j \in B} f_j$ for all $A, \tilde{A}, B$ with $V = A \cup \tilde{A} \cup B$ and $G[\tilde{A} \cup B, B] = \emptyset$, where equality holds if and only if additionally either $A = B = \emptyset$ or ($V = A \cup B$ and $G[A,A] = G[B,B] = \emptyset$).

(iii) $\sum_{i \in A} f_i \geq \sum_{j \in B} f_j$ for all independent sets B, non-adjacent opponents $\tilde{B}$ and $A = R(B, \tilde{B})$, where equality holds if and only if additionally either $A = B = \emptyset$ or ($V = A \cup B$ and $G[A,A] = G[B,B] = \emptyset$).

(iv) $\sum_{i \in A} f_i \geq \sum_{j \in B} f_j$ for all independent sets B, and $A = N(B)$, where equality holds if and only if additionally either $A = B = \emptyset$ or ($V = A \cup B$ and $G[A,A] = G[B,B] = \emptyset$).

(v) $\sum_{i \in A} f_i \geq \sum_{j \in B} f_j$ for all subsets $B \subseteq V$, and $A = N(B)$, where equality holds if and only if additionally either $B = \emptyset$, or $B = V$, or ($V = A \cup B$ and $G[A,A] = G[B,B] = \emptyset$).

Each intermediate step follows more or less straightforward by carefully applying all definitions. □

3.6 Applications

IPF is widely used in theory and practice across several different disciplines, so this chapter has a potential impact on a variety of applications. Let us point out some examples: in quadratic non-negative matrix factorization (Yang and Oja, 2012), SIPF is the canonical candidate for determining the factorization of $W \in \mathbb{W}$ into the form $W = YXY$ for a positive diagonal matrix Y and with X being further constrained to prescribed marginals $\mathbf{f}$. In numerical analysis of matrices, similar to Knight et al. (2014) for the doubly stochastic case ($\mathbf{f} = \mathbf{1}$), SIPF can serve as a matrix preconditioner while providing the flexibility of the case $\mathbf{f} \neq \mathbf{1}$. As another example, Corollary 3.5.6 corresponds to the combinatorial problem of identifying the unique minimum set of edges in a graph that must be removed (set to weight 0) in order to be able to achieve degree vector $\mathbf{f}$ on the remaining edges.

Before we highlight further applications in more detail, let us focus again on the relevance of the feasibility studies: it is important to remember that feasibility of IPF/PSIPF/SIPF is not guaranteed in general, but only for a specific family of graph matrices. The graph G must be a weak $\mathbf{f}$-expander, and whenever a factorized approach is used, G must additionally satisfy the strictness assumptions in Proposition 3.4.8. This fact is overseen by Zass and Shashua (2005), who misleadingly state their Proposition 1 (convergence to a doubly stochastic limit) to hold for every non-negative symmetric matrix, without the necessary feasibility conditions (a counterexample that does not converge to a doubly stochastic limit is the simple path graph on 3 vertices). In particular the factorization stated in Proposition 2 by Zass and Shashua (2006) does *not* apply in general. The authors take the tempting Lagrangian approach for their proof, which is invalid whenever $E(W^{(\infty)}) \neq E(W)$. Rendering feasibility in terms of weak $\mathbf{f}$-expansion makes this requirement intuitively accessible.

3.6.1 Restricted earth mover distance

Let the vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^d_{\geq 0}$ represent two distributions of some kind of mass weights on d dimensions. Assume further that the d dimensions themselves are arranged in some distance relation to each other as specified by a matrix $M \in \mathbb{R}^{d \times d}_{\geq 0}$, denoted as the **ground metric**. For example, we can think of an image with d pixels arranged in r rows and c columns as of a single point in d-dimensional space, whose dimension axes (pixel positions) can further be related to each other by the Euclidean distance between the corresponding pixel coordinates. The ground metric $M = [m_{ij}]$ specifies the costs for moving one unit of mass from one dimension to another. For example, the cost of moving mass $x \geq 0$ from dimension i to dimension j is $x \cdot m_{ij}$. Based on this intuition, the **Earth Mover Distance (EMD)** between $\mathbf{a}$ and $\mathbf{b}$ is defined as the minimum cost of a transportation plan for moving all masses (initially distributed according to $\mathbf{a}$) between the dimensions in a single step such that the result is $\mathbf{b}$.

Known algorithms for computing EMD take at least time super-cubic in d. Cuturi (2013) introduces the **Sinkhorn distance** by adding an entropic regularization to EMD that avoids over-complex transportation plans in a precise sense. This solution comes at the price of having

higher than optimal overall costs. However, this approach reduces not only the solution's structural complexity, but also the computational complexity: the dual of the Sinkhorn distance can be approximated efficiently by performing IPF of the matrix $K := \exp(-M)$ toward row marginals $\mathbf{a}$ and column marginals $\mathbf{b}$. Since all entries in K are positive, convergence is guaranteed for all positive vectors $\mathbf{a}$, $\mathbf{b}$.

In the light of the results in this chapter, we suggest to relax from positive to non-negative IPF, that is to allow for 0-entries in K. Such entries in K correspond to ∞-entries in M, thus to pairs of dimensions between which mass transport is impossible. For example, such M can be chosen to restrict the mass transport being possible only between nearby dimensions. This approach allows to yield sparse matrices K, which can be a crucial improvement: a three-dimensional CRT-image of size $100 \times 100 \times 100$ pixels (i.e., $d = 1000000$) requires $8 \cdot d^2 \approx 8000$ GB of data storage for the $d \times d$-matrix K at double precision. This lets the Sinkhorn distance become computationally infeasible on standard hardware. If we instead restrict the mass transportation to only pixels in Euclidean distance at most 10, then each dimension is connected to at most $21^3 \cdot \pi/4$ other dimensions, yielding a storage size for K of less than 60 GB, that is 0.75%, which even fits into RAM of a standard server hardware. The drawback of the sparsity of K is that for some choices of $\mathbf{a}$ and $\mathbf{b}$ there might be no solution, that is $\Omega(\mathbf{a}, \mathbf{b}, K) = \emptyset$. For such "incompatible" pairs of $\mathbf{a}$ and $\mathbf{b}$ the Sinkhorn distance becomes infinity, indicating that there exists no suitable transportation plan. Depending on the application, this issue can be addressed as follows:

1. in classification problems, data is often normalized before being processed further (e.g., images by centering, scaling, and rotation). After normalization, the mass transport can likely be restricted to smaller radii without leading to incompatible pairs of elements, at least if these are from the same or similar classes. This allows for meaningful (finite) Sinkhorn distances whenever $\mathbf{a}$ and $\mathbf{b}$ belong to the same class or to similar classes, while an infinite distance is tolerable if $\mathbf{a}$ and $\mathbf{b}$ belong to very different classes.

2. instead of a single sparse K, we might consider a sequence $K_1, K_2, ..., K_R$ of matrices with decreasing sparsity (e.g., by doubling the radii of possible mass transport when going from K_i to K_{i+1}). Then one may first compute the sparse Sinkhorn distance on K_1 efficiently, and only in case of an infinite distance continue with K_2, K_3, ..., until a finite distance is attained. This can drastically speed up the average computation time, while still yielding a finite Sinkhorn distance for all pairs of elements.

A similar generalization from positive to non-negative IPF applies to the work of Cuturi and Doucet (2014) on computing the Barycenter of multiple points $\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_n \in \mathbb{R}^d_{\geq 0}$ with respect to a regularized 2-Wasserstein distance.

3.6.2 Iterative correction of Hi-C data

Imakaev et al. (2012) introduce a method called "Iterative Correction and Eigenvector decomposition" (ICE) for studying genomes by observing the interactions between different

locations on it. Experimental measurements provide a histogram of pairwise interactions between the locations (denoted as *double-sided reads*). Optionally, additional oriented actions (so-called *single-sided reads*) are considered, which we do not consider here. The double-sided reads define a graph $\mathcal{G}(W)$ with the locations as vertices and with the empirical amounts of pairwise interaction as edge weights. Degrees in this graph refer to the observed overall "visibility" of that location. These empirical visibilities are non-uniform due to several biases in the experimental setup. However, deeper domain knowledge suggests that the "true" visibilities should be uniform. This motivates to find a degree-balanced matrix that "best approximates" the empirical data matrix W. The authors consider the relative-entropy nearest doubly stochastic matrix $W^{(\infty)} := \mathcal{P}_{\mathbb{W}(\mathbf{1},W)}(W)$ as the most interesting approximation. We have shown in this chapter that it can be approximated by IPF or any of its symmetrized variants. However, the authors suggest another iterative algorithm, denoted as Iterative Correction (IC). They do not provide a convergence proof, and indeed the convergence behavior is different: IC may diverge for unconnected graphs in cases where IPF and its variants still converge. And even for connected graphs, IC can run into serious numerical problems, producing extremely large numbers (outside the range of double precision) for representing intermediate steps of the iteration. So we suggest to replace IC by SIPF whenever one-sided reads are skipped. SIPF does not only fix all above mentioned issues, but further enables to deal with non-uniform visibilities $\mathbf{f} \neq \mathbf{1}$. This allows for related applications where the uniform visibility assumption does not hold, or where some of the biases are known and can be corrected separately, so we only need to correct for the remaining (non-uniform) bias. Another improvement can be achieved for the comparison of two different genome matrices W_1 and W_2. The authors suggest to $\mathbf{1}$-balance W_1 and W_2 individually before comparing them. SIPF alternatively allows to directly compare the $\mathbf{d}_2$-fitted W_1 to W_2 and vice versa.

The ICE method ends with an analysis of the largest eigenvectors of the "corrected" graph matrix $W^{(\infty)}$. For doubly stochastic $W^{(\infty)}$ this is equal to classical spectral analysis, that is to explore structural properties of $W^{(\infty)}$ by the smallest eigenvectors of its normalized Laplacian matrix. Hence, in this case the ICE method can compactly be summarized as classical spectral analysis of the normalized Laplacian of the "$\mathbf{1}$-fitted" graph. This motivates the following application, which generalizes this approach to $\mathbf{f} \neq \mathbf{1}$.

3.6.3 Inference using the f-fitted Laplacian

Recall that the normalized graph Laplacian $\mathcal{L}(W)$ of a matrix $W \in \mathbb{W}$ has the same eigenvectors (with eigenvalue λ_i mapped to $1-\lambda_i$) as the matrix $D^{-1/2}WD^{-1/2} = W^{(1)}$, the first element of the SIPF-sequence for $\mathbf{f} = \mathbf{1}$. This allows for a novel interpretation of the type of normalization that is considered by $\mathcal{L}(W)$:

> **"Eigenvalues and eigenvectors of $\mathcal{L}(W)$ refer to the first step of SIPF-scaling W toward the uniform degree vector $\mathbf{1}$."**

Thus, $\mathcal{L}(W)$ captures properties that show up when transforming W into a matrix of approximately degree vector $\mathbf{1}$, precisely of degree vector $\mathbf{d}^{(1)} := W^{(1)}\mathbf{1} \approx \mathbf{1}$. However, in general it holds that $\mathbf{d}^{(1)} \neq \mathbf{1}$ and that $\mathbf{d}^{(1)}$ is still correlated to the original degree vector $\mathbf{d}$. The previous chapter presented an approach to correct for the residual errors $\mathbf{d}^{(1)} - \mathbf{1}$: by subsequent $\mathbf{1}$-selflooping of $W^{(1)}$ we get the $\mathbf{1}$-adjusted graph matrix $\overline{W}_{\mathbf{1}}$, which achieves degree vector $\mathbf{1}$ exactly. This also led to the definition of the $\mathbf{f}$-adjusted Laplacian as $\mathcal{L}_{\mathbf{f}}(W) := \mathcal{L}(\overline{W}_{\mathbf{f}})$. In this chapter, we suggest to repeatedly iterate $\mathbf{1}$-scaling instead, eventually converging to a well-understood limit matrix $W^{(\infty)}$ of degree vector $\mathbf{1}$. Complementary to the definition of the $\mathbf{f}$-adjusted Laplacian, this suggests to define the **$\mathbf{f}$-fitted Laplacian** by $\widehat{\mathcal{L}}_{\mathbf{f}}(W) := \mathcal{L}(\mathcal{P}_{\mathbb{W}(\mathbf{f},W)})$, where SIPF is the natural candidate to approximate $\mathcal{P}_{\mathbb{W}(\mathbf{f},W)} = W^{(\infty)}$.

Whenever $W^{(1)} \approx W^{(\infty)}$, the geometric interpretation of $\overline{W}_{\mathbf{f}}$ for geometric graphs as a density shift also applies to $W^{(\infty)}$. In this case the $\mathbf{f}$-adjusted Laplacian and the $\mathbf{f}$-fitted Laplacian capture similar properties. Whenever $W^{(1)}$ and $W^{(\infty)}$ differ substantially, $\mathcal{L}_{\mathbf{f}}(W)$ and $\widehat{\mathcal{L}}_{\mathbf{f}}(W)$ focus on different aspects. The spectral analysis of the $\mathbf{f}$-fitted Laplacian focuses on inferring about W after "correcting" its degrees to $\mathbf{f}$ by replacing W with the relative-entropy nearest re-weighting that provides degree vector $\mathbf{f}$. For $\|\mathbf{f}\|_1 = 1$ this allows for a statistical intuition: the entries of $W^{(\infty)}$ can be interpreted as joint probabilities that achieve the marginal probability distribution $\mathbf{f}$ on the support of W (i.e., restricted to non-zero entries of W). Depending on the application, this statistical interpretation of the $\mathbf{f}$-fitted Laplacian can be better suited for a problem than the geometric intuition of the $\mathbf{f}$-adjusted Laplacian. In particular, the above presented ICE method refers to this statistical interpretation and provides a prominent application of the spectral analysis of the $\mathbf{1}$-fitted Laplacian.

3.7 Chapter summary

This chapter is complementary to the previous chapter: both chapters study a method to transform a symmetric matrix W of unspecified degree vector into another symmetric matrix of a prescribed degree vector $\mathbf{f}$. The solution studied in the previous chapter is based on concatenating a single execution of $\mathbf{f}$-scaling followed by $\mathbf{f}$-selflooping. The solution in this chapter is based on the repeated application of $\mathbf{f}$-scaling until convergence. I relate this iteration to known iterative projection methods from the literature, in particular to IPF. I contribute to this field by providing a novel algorithmic argument for the fact that IPF converges to the relative-entropy optimum. Precisely, I show that local affinity is already sufficient for the limit to be the Bregman projection of the initial element. Further, I address the issue that IPF does not fit well to symmetric matrices by introducing two symmetric alternatives: PSIPF is a straightforward symmetrization of IPF, and SIPF is equal to iterated $\mathbf{f}$-scaling. Both variants allow for a factorized approach that is preferable over the recursive approach whenever applicable. I give convergence proofs for both approaches. Further, I

study the feasibility problem of whether the limit in the symmetric setting is indeed a feasible solution or not. I contribute a novel characterization of all feasible scenarios in terms of intuitive graph expansion properties, precisely by introducing the concept of weak **f**-expansion. This type of expansion is not yet studied in depth in graph theory, although it provides an interesting complexity. The feasibility study particularly separates between "total feasibility" and "non-total feasibility", which is shown to determine precisely the applicability of the factorized algorithm variant.

Finally, practical relevance is emphasized by connecting the contributions of this chapter to various applications.

Chapter 4

Spectral correlation clustering

4.1 Chapter introduction

This chapter is motivated by an application from practice that was driven by me from its theoretical analysis through to a full-fledged software implementation. The term **Massively Multiplayer Online Role Playing Game (MMORPG)** refers to a computer game that allows thousands of people across the Internet to interact with each other in a virtual world. Every human **user** controls a virtual character that is denoted as a **player** in the sequel. A big aspect of most MMORPGs is to allow players to fight against each other in a **player versus player combat (PVP combat)**. Most long-term-users ally their players with others in large groups such as **guilds**, which may consist of hundreds of players. Large PVP combats between two or more guilds are often planned events that require their participants to arrange to be online at the same time around the world. These users are often enthusiasts that enjoy to dive deep into the virtual world, while investing a lot of time and money, for example in order to buy virtual goods. Games are designed to provide an addictive experience for its faithful users.

The impression of virtual reality is intensified by providing the feeling of an open world: everybody is free to do what she wants, for example talk to each other, share goods, etc. In particular, everybody can attack everybody else at any time. This "freedom" comes at a price. MMORPGs often face a problem known as "Zerging". A **Zerg** is a larger group of players who simply search for much smaller groups in order to kill and plunder them. Often

the Zerg's users are not interested in celebrating the virtual reality, and often they even intentionally spoil the game experience of other players (this behavior is denoted as **griefing**). While this kind of player behavior is generally permitted, it is considered as being unfair, and repeated victims of Zergs will loose their joy about the game. Game developers are serious about balancing the game experience between competition and fun, so they are particularly interested in strategies to discourage people from Zerging.

Albion Online[1] is an MMORPG developed at Sandbox Interactive GmbH in Berlin, Germany. This company came up with a novel idea that is so far not implemented in any MMORPG: detect and penalize Zergs automatically by a suitable algorithm. Once a Zerg is identified, the game physics can then assign any type of **debuff** (temporary disadvantage) to these players in order to antagonize Zerging. An appropriate algorithm has the following input and output:

- **input**: potentially the complete world information, particularly a record of all interactions between all players. There are three basic types of interactions: adversarial activities (damage points by weapons or magic attacks), friendly activities (positive effects such as health points by magic spells, or staying close to each other for a long time without fighting), and neutral activities. Adversarial and friendly activities can be quantified as dissimilarity values and similarity values, respectively.

- **output**: There are two outputs. The first output is, separately for each PVP combat in the virtual world, a meaningful assignment of all participating players to an unspecified number of teams (= clusters). The second output is derived from this clustering: each player has attached a time-smoothed **debuff score** from the interval $[0, 1]$, which gets the larger the stronger the team of this player is.

The output of this algorithm is then taken to identify and penalize Zergs. In the prototype, the game physics simply applies debuffs that are directly determined by the players' debuff scores, hence, proportional to their team strengths. By this, all teams are encouraged to fight against teams of a comparable overall strength. A final version may focus on an additional classification of whether a team forms a Zerg or not, for example by taking the team strengths in relation to each other into account or by analyzing behavioral patterns of the players in a team.

As soon as any debuff strategy is in action, users will try to avoid getting a debuff. That is, they want to hide their true membership in a clearly superior team. Many examples from practice show that there are always some users that act out an enormous creativity in finding ways to "cheat" computer algorithms. For example, in virtual trading systems there might be complex interactions of buying, producing and selling goods that finally lead to a universal recipe of generating virtual money. Similar problems arise in complex customizable equipment systems, where some specific combinations might turn out to provide overreaching advantages.

[1] `www.albiononline.com`

Thus, the clustering algorithm must be as robust as possible against the most creative cheaters. Because of this, one may also denote the overall goal as "clustering against adversaries".

There are basically two strategies for a player to avoid being recognized as an actor of a superior team A:

- **Friendly fire**. Attack some team colleague from team A in order to hinder the algorithm to detect the affiliation to them. If the cheater is a successful fake-enemy of team A then team A appears weaker than it is, receiving a decreased debuff.
- **Unfriendly help**. Help adversaries from another team B in order to hide the true opposite affiliation. If the cheater is a successful fake-ally of team B, then team B appears stronger than it is, receiving an increased debuff.

The clustering strategy has to be aware of these ways of cheating. However, note that players can always cheat so strongly, for example by vehement friendly fire, that it is impossible for any algorithm to detect the *true* team affiliation (where "true" refers to the secretly shared knowledge of the users that form a team). Hence, it is impossible to detect the true teams in general. Fortunately, both cheating strategies require a behavior that increasingly weakens the true team. This has the following important consequence: we do not have to focus on the (impossible) goal of detecting the *true* team affiliation. It is sufficient to focus on detecting *effective* teams. The difference is that we do not care about whether a player *truly* belongs to team A, we only care about whether the player *effectively* acts for or against team A. If a player contributes in total significantly more damage than heal to its true team A, then it is alright to misclassify this player to another team.

> **“ Strong cheaters are debuff enough for their true team. ”**

So the output team assignment is expected to identify the *effective* teams, separately for each combat. The corresponding clustering technique has to respect the following considerations:

- the number of effective teams in a PVP combat is not known in advance, in particular it is not always 2. However, we may assume that it is a rather small number, say, up to 5.
- since our goal is to detect teams especially in the case of a strong team that attacks a weak team, the clustering algorithm should not assume that the clusters are roughly balanced
- the clustering strategy should support adding must-have-edges because additional world information can allow to definitely assign some players explicitly to the same team
- the world's clustering will be repeated over and over again. Each repetition should finish its computation within just a few seconds in order to react fast enough to Zergs.

Sandbox Interactive GmbH entrusted me with developing a prototype of the "Anti-Zerging Clustering Software" in order to solve the above described goals. This challenge stimulated my research toward developing a novel spectral approach to correlation clustering. My approach consists of a theoretical contribution (Section 4.4.1) that yields a novel algorithm (Section 4.4.2). Finally this approach is put into practice, which requires to solve additional practical integration issues and an extensive preprocessing, as presented in Section 4.6.

4.2 Informal summary of the main contributions

Our goal is to cluster the players of a PVP combat into an unspecified number of effective teams. This goal is formalized in the subsequent formal setup as an instance of correlation clustering. The canonical objective in correlation clustering is to minimize the multicut of a graph that has both positive (similarity) edge weights and negative (dissimilarity) edge weights. The signed Laplacian $\overline{L}$, as introduced in Section 4.3.6, captures interesting properties of real-weighted graphs. $\overline{L} = \overline{D} - W$ is defined in almost the same way as the Laplacian $L = D - W$ with the only difference that its degree matrix $\overline{D}$ is defined by summing over the *absolute* values of the edge weights. For positive-weighted graphs both matrices $\overline{L} = L$ coincide. My first contribution in this chapter is to show that the smallest eigenvector of $\overline{L}$, which is the "trivial" eigenvector in case of pure similarity-based clustering, becomes highly interesting as soon as we mix positive and negative edge weights. Precisely, I prove that the smallest eigenvector solves a relaxation of the correlation clustering problem for at most two classes, denoted as OptMinCut This is in contrast to existing literature, which considers the smallest eigenvector as the relaxation of a signed variant of RatioCut. I argue that my interpretation is more appropriate. These findings are the first main contribution of this chapter (Section 4.4.1).

The second main contribution (Section 4.4.2) applies these insights by introducing a novel spectral approach to correlation clustering. The suggested algorithm is recursive spectral clustering with respect to the smallest eigenvector of the signed Laplacian. It shows some interesting properties that make it very different from recursive spectral clustering of similarity-graphs. The later experiments study the performance of this approach in comparison to existing spectral and non-spectral approaches to correlation clustering. It turns out that my approach can outperform or at least measure up to all other alternatives.

4.3 Formal setup

This section introduces all the concepts and formalisms that are required in order to understand the main contributions.

4.3.1 Correlation measures

Many algorithms from machine learning require access to either a similarity measure or a dissimilarity measure. Applications often provide one of these measures, so this requirement fits well to practice. However, some applications provide a combination of both measures, denoted as a **correlation measure**. For any set V of objects, a correlation measure $\gamma : V \times V \to \mathbb{R}$ quantifies the relation of two objects in terms of a real value c that is interpreted as follows:

- ($c \gg 0$) high positive values indicate high confidence about a strong similarity
- ($c \ll 0$) low negative values indicate high confidence about a strong dissimilarity
- ($c \approx 0$) the closer the absolute value is to zero, the lower is the confidence on a significant similarity ($c > 0$) or dissimilarity ($c < 0$)
- ($c = 0$, equivalent to a missing edge) value exactly zero indicates zero-confidence on any attitude, that is effectively the same as having no explicit information given
- (c vs. $-c$) positive/negative values are scaled such that a positive correlation $c > 0$ is of roughly the same relevance as a negative correlation $-c$

Summarized, a correlation measure is the combination of a similarity measure (positive values) and a dissimilarity measure (negative values) together with an intermediate area of high uncertainty (values around zero). Clustering with respect to a correlation measure is denoted as **correlation clustering (CC)**.

The various types of correlation coefficients, such as Pearson's, Spearman's and Kendall's correlation coefficients, fit to the above informal description on the interval $[-1, 1]$. However, a correlation measure is not required to base on a mathematical precise definition. For example, pairs of books may be labeled by humans either as being similar (value 1) or dissimilar (value -1), which provides an informally defined correlation measure. As another example, consider an Internet platform like reddit on which users frequently up-vote or down-vote individual items (posts or comments) of others. A simple measure of correlation between two users i and j can then be defined by $c_{ij} = \sum_{k \in V_{ij}} a_k$, where V_{ij} is the set of all items for which both i and j gave any vote, and $a_k = 1$ if their votes agree, and $a_k = -1$ if they gave opposite votes.

4.3.2 Cluster Matrices

Any partition of $V = \{1, \ldots, n\}$ into k non-empty disjoint subsets $C_1 \dot\cup \ldots \dot\cup C_k$ is denoted as a **k-clustering**. Each k-clustering can be represented by a vector $\mathbf{y} \in \{1, \ldots, k\}^n$ that

has its i'th entry equal to $\ell \in \{1, \dots, k\}$ if and only if $i \in C_\ell$. An alternative representation is by a matrix $Y \in \{0,1\}^{n \times n}$ that has entry $Y_{ij} = 1$ if and only if i and j belong to the same cluster. Such a matrix is denoted as a **cluster-matrix**. $Y = Y^T$ is symmetric and has all main diagonal entries equal to 1. We denote by $\mathcal{C}_k$ the set of all cluster-matrices that indicate any k-clustering. For example, the partition $\{1,3\} \cup \{2,4,5\}$ can be represented by the following cluster matrix $Y \in \mathcal{C}_2$:

$$Y = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}}_{=\mathbf{1}_{C_1}\mathbf{1}_{C_1}^T} + \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}}_{=\mathbf{1}_{C_2}\mathbf{1}_{C_2}^T} \quad .$$

Note that the i'th row/column equals the characteristic vector $\mathbf{1}_{C_j} = ([i \in C_j]_0^1)$ of the corresponding cluster, which implies the representation

$$Y = \sum_{i=1}^{k} \mathbf{1}_{C_i} \mathbf{1}_{C_i}^T \quad .$$

The following proposition lists some basic properties of cluster matrices.

Proposition 4.3.1 (Properties of cluster matrices)
For a cluster matrix $Y \in \mathcal{C}_k$ all the following statements hold true:

(i) there exists a permutation P such that PYP^{-1} has block-diagonal form of k blocks

(ii) Y has rank k

(iii) Y is positive semi-definite

*(iv) the **nuclear norm** $\|Y\|_\infty$, that is the sum of all singular values, equals n*

(v) the function $\varphi : \{-1,1\}^n \to \mathcal{C}_1 \cup \mathcal{C}_2$, $\mathbf{f} \mapsto (\mathbf{f}\mathbf{f}^T + \mathbf{1}\mathbf{1}^T)/2$ is well-defined and surjective. Further, it holds with $\mathbf{f} = (f_i)$ and $Y := \varphi(\mathbf{f})$ that

a) $f_i = f_j \Leftrightarrow Y_{ij} = 1$, and equivalently $f_i \neq f_j \Leftrightarrow Y_{ij} = 0$

b) $(1 - Y_{ij}) = (f_i - f_j)^2/4$ and $Y_{ij} = (f_i + f_j)^2/4$ for all i, j

c) every $Y \in \mathcal{C}_1 \cup \mathcal{C}_2$ has exactly two pre-images in $\{-1,1\}^n$

d) $Y \in \mathcal{C}_1 \Leftrightarrow \mathbf{f} \in \{-\mathbf{1}, \mathbf{1}\} \Leftrightarrow f_i = f_j$ for all i, j

e) for $Y \in \mathcal{C}_2$ the two pre-images are given by $\mathbf{1}_{C_1} - \mathbf{1}_{C_2}$ and $\mathbf{1}_{C_2} - \mathbf{1}_{C_1}$

Proof. (i) and (ii) are straightforward to see. For (iii), note that the spectrum of a block-diagonal matrix is the union of the spectra of all blocks. The block that corresponds to cluster C_i equals $\mathbf{1}\mathbf{1}^T \in \mathbb{R}^{|C_i|\times|C_i|}$, which has $|C_i|$ as a simple eigenvalue, plus the eigenvalue 0 of algebraic multiplicity $|C_i| - 1$. For (iv), note that for a positive semi-definite matrix its singular values and eigenvalues coincide. Thus, we get from the proof of (iii), or simply from the trace of Y, that they sum up to n. For (v), observe that for all $\mathbf{f} \in \{-1, 1\}^n$ and $Y := (\mathbf{f}\mathbf{f}^T + \mathbf{1}\mathbf{1}^T)/2$ it holds that $Y_{ij} = (f_i f_j + 1)/2$, thus, $f_i = f_j \Rightarrow Y_{ij} = 1$ and $f_i \neq f_j \Rightarrow Y_{ij} = 0$. This proves a) and the fact that φ is well-defined. b) follows from a). c) follows from the fact that $\varphi(\mathbf{f}) = \varphi(-\mathbf{f})$ and that for $\mathbf{f}' \notin \{-\mathbf{f}, \mathbf{f}\}$ there exists some i, j such that $f_i f_j \neq f'_i f'_j$, thus $\varphi(\mathbf{f}) \neq \varphi(\mathbf{f}')$. d) follows from a) and c). For e), observe that $\mathbf{f} := \mathbf{1}_{C_1} - \mathbf{1}_{C_2}$ satisfies $((\mathbf{1}_{C_1} - \mathbf{1}_{C_2})(\mathbf{1}_{C_1} - \mathbf{1}_{C_2})^T + \mathbf{1}\mathbf{1}^T)/2 = (\mathbf{1}_{C_1}\mathbf{1}_{C_1}^T - \mathbf{1}_{C_2}\mathbf{1}_{C_1}^T - \mathbf{1}_{C_1}\mathbf{1}_{C_2}^T + \mathbf{1}_{C_2}\mathbf{1}_{C_2}^T + \mathbf{1}\mathbf{1}^T)/2 = (2 \cdot \mathbf{1}_{C_1}\mathbf{1}_{C_1}^T + 2 \cdot \mathbf{1}_{C_2}\mathbf{1}_{C_2}^T)/2 = \sum_{k=1,2} \mathbf{1}_{C_k}\mathbf{1}_{C_k}^T = Y$, where we use that $\mathbf{1}\mathbf{1}^T = \mathbf{1}_{C_1}\mathbf{1}_{C_1}^T + \mathbf{1}_{C_2}\mathbf{1}_{C_1}^T + \mathbf{1}_{C_1}\mathbf{1}_{C_2}^T + \mathbf{1}_{C_2}\mathbf{1}_{C_2}^T$. Further, $-(\mathbf{1}_{C_1} - \mathbf{1}_{C_2}) = \mathbf{1}_{C_2} - \mathbf{1}_{C_1}$.

□

4.3.3 Complexity of the minimum cut problem

For graphs with non-negative weights, the minimum k-cut for fixed k can be computed in polynomial time $\mathcal{O}(|V|^{k^2})$, as shown by Goldschmidt and Hochbaum (1994). In contrast to that, the maximum cut problem is NP-hard for every $k \geq 2$. Even worse, maximum cuts are even hard to approximate ("APX-hard"). Trevisan et al. (2000) show that there exists no polynomial time approximation algorithm that achieves an approximation ratio better than $16/17 \approx 0.941$. Goemans and Williamson (1995) give a randomized polynomial time approximation algorithm (by a semi-definite program plus randomized rounding) that achieves the approximation ratio ≈ 0.879. The remaining gap is subject of active research, where it is believed that the latter bound is already sharp.

However, all these results do only consider graphs of non-negative weights. If we allow for negative edges, the complexity of the minimum cut also increases to be NP-hard, which can be seen as follows: any algorithm that can find the minimum k-cut in a real-weighted graph is particularly able to solve the maximum k-cut on a non-negative graph, simply by determining the minimum k-cut after negating all edge weights. Indeed, for real-weighted graphs, the minimum cut problem and the maximum cut problem coincide. Hence, the minimum k-cut problem on real-weighted graphs is NP-hard for every $k \geq 2$, and even hard to approximate.

4.3.4 Correlation clustering

This section introduces correlation clustering, including a formal definition and related work.

Overview on correlation clustering

The problem of correlation clustering was introduced by Bansal et al. (2004). They study complete graphs with every edge weighted by either $+1$ or -1. Their goal is to find a k-clustering that maximizes the number of $+1$ edges within clusters plus the number of -1

edges between different clusters. In contrast to most other clustering problems, k is not given as an input, but is determined by the algorithm. In particular, the output of the trivial clustering $k = 1$ is possible. This makes correlation clustering substantially different from other clustering problems that force $k > 1$. Bansal et al. (2004) show that correlation clustering of the complete $\{-1, +1\}$-weighted graph is NP-hard. They present an approximation algorithm that is based on combinatorial arguments and that provides a constant-factor approximation of the optimum.

A **perfect correlation clustering** is a partitioning of V into clusters for which every positive edge is an intra-edge and every negative edge is an inter-edge. Such a perfect clustering exists if and only if every cycle traverses only an even number of negative edges. This can easily be checked algorithmically by a Depth-First-Search, which can also be modified to construct the corresponding perfect clustering (if existing). For that reason, the only interesting case for correlation clustering is that the graph contains erroneous edges ($+1$ edges between different clusters or -1 edges within a cluster), for example triangles of edge weights $(+1, +1, -1)$.

Charikar et al. (2003) provide an $\mathcal{O}(\log n)$ approximation algorithm that can deal with general $\{-1, +1\}$-weighted graphs. For the same type of graphs, Berg and Jarvisalo (2013) provide an exact algorithm that determines the optimal solution (with exponential worst-case running time) by a reduction to MaxSAT. An $\mathcal{O}(\log n)$ approximation algorithm for general real-weighted graphs is given by (Demaine and Immorlica, 2003), and independently by (Emanuel and Fiat, 2003), later published by all four authors together in Demaine et al. (2006). Similar to Charikar et al. (2003), all their solutions base on a formulation of correlation clustering as a linear program, followed by the rounding technique of "region growing" as introduced by Leighton and Rao (1999) for multicommodity max-flow min-cut problems. However, their linear programs base on $\mathcal{O}(n^3)$ inequality constraints, which makes this approach infeasible for large-scale applications. Charikar and Wirth (2004) reformulate correlation clustering as a quadratic program that is maximized by semi-definite programming plus randomized rounding, similar to the approach by Goemans and Williamson (1995) for the maximum cut for non-negative weights. They achieve an $\mathcal{O}(\log n)$ approximation to the optimum cut weight. However, they achieve this bound by a clustering that consists of either 2 or n clusters, which is often not sufficient for applications. The above approaches do not scale well beyond some hundreds of vertices. Their focus lies on proving best-possible approximation guarantees. In contrast to that, Bagon and Galun (2011) introduce three heuristics without proving any bounds: **Expand**, **Swap** and **AICM**. The first two heuristics are greedy iterative strategies that exploit a concept known as "roof duality" (Rother et al., 2007) to solve binary subproblems in a way that allows partial access to their global optimum. The third heuristic is another greedy iterative strategy, based on ICM (iterated conditional modes) by Besag (1986). In contrast to the original ICM method, AICM varies k over the iterations in order to adapt to the unknown number of clusters. Expand and Swap do not scale well, but AICM can be applied to graphs with hundreds of thousands of vertices on standard computer hardware.

There further exist spectral approaches to correlation clustering, presented in Section 4.3.7.

The CC functional

In its most general form, correlation clustering is defined as the following non-convex optimization problem for any given matrix $W \in \mathbb{R}^{n \times n}$:

$$\underset{\substack{k=1,\dots,n \\ Y \in \mathcal{C}_k}}{\arg\max} \quad \sum_{i,j} w_{ij} Y_{ij} \quad . \tag{4.1}$$

Without loss of generality, we can assume that W is symmetric because by symmetry of Y the edges ij and ji are either both inter-cluster edges or both intra-cluster edges, so the objective value for any W is the same as for $(W + W^T)/2$. We can further assume, without loss of generality, that the main diagonal of W is zero, since $Y_{ii} = 1$ for all i implies that $\sum_i w_{ii} Y_{ii}$ is just a constant additive term to the objective function. The objective function $\sum_{i,j} w_{ij} Y_{ij}$ is denoted as the **CC functional**. It sums over all intra-cluster edges. Since $\sum_{i,j} w_{ij}$ is constant, we get for all constants $a > 0$, $b < 0$, and $A, B \in \mathbb{R}^{n \times n}$ that

$$\underset{\substack{k=1,\dots,n \\ Y \in \mathcal{C}_k}}{\arg\max} \sum_{i,j} w_{ij} Y_{ij} \quad = \quad \underset{\substack{k=1,\dots,n \\ Y \in \mathcal{C}_k}}{\arg\max} \sum_{i,j} w_{ij} (a Y_{ij} + A_{ij}) \quad = \quad \underset{\substack{k=1,\dots,n \\ Y \in \mathcal{C}_k}}{\arg\min} \sum_{i,j} w_{ij} (b Y_{ij} + B_{ij})$$

In particular, we get for $b = -1$ and $B = \mathbf{1}\mathbf{1}^T$ that maximizing the CC functional is equivalent to finding the minimum real-weighted multi-cut:

$$\underset{\substack{k=1,\dots,n \\ Y \in \mathcal{C}_k}}{\arg\min} \sum_{i,j} w_{ij} (1 - Y_{ij}) \quad . \tag{4.2}$$

Similar, for $Z := 2Y - \mathbf{1}\mathbf{1}^T$ and $\mathcal{C}_k^{\pm} := 2\,\mathcal{C}_k - \mathbf{1}\mathbf{1}^T$ we get that maximizing the CC functional is equivalent to

$$\underset{\substack{k=1,\dots,n \\ Z \in \mathcal{C}_k^{\pm}}}{\arg\max} \sum_{i,j} w_{ij} Z_{ij} \quad , \tag{4.3}$$

that is maximizing the total weight of intra-edges minus the total weight of inter-edges. This matches the original formulation by Bansal et al. (2004).

Some publications study the matrix W by its AR-decomposition, which is defined as follows.

Definition 4.3.2 (AR-decomposition)
For $W \in \mathbb{R}^{m\times n}$, let $A \in \mathbb{R}^{m\times n}_{\geq 0}$ be defined by the positive entries in W, and $R \in \mathbb{R}^{m\times n}_{\geq 0}$ by the absolute values of the negative entries in W. Then

$$W \;=\; A - R$$

is the unique decomposition of W into the difference of two non-negative matrices $A, R \in \mathbb{R}^{m\times n}_{\geq 0}$ such that for every ij it holds that at most one of a_{ij} and r_{ij} is non-zero. We refer to this decomposition as the **AR-decomposition**. A is denoted as the **attraction matrix**, and R as the **rejection matrix**. Further, we set $D_A := \mathrm{diag}(A\mathbf{1})$ and $D_R := \mathrm{diag}(R\mathbf{1})$.

Constrained correlation clustering

Some applications provide additional information on the correlations between objects. For example, a domain expert may identify some pairs of vertices that definitely not belong to the same class, or in semi-supervised learning some true labels may be known. Such additional information can be encoded as "must-links" and "cannot-links". A **must-link** (ML) refers to a pair of vertices that is forced to stay in the same cluster. A **cannot-link** (CL) forces two vertices to stay in different clusters. This line of research is denoted as **constrained clustering**, and is typically studied in similarity-based clustering. However, the same constraints can be applied on top of correlation clustering.

The literature is ambiguous on how to interpret "to force" the vertices to stay in the same or different clusters. This term refers either to hard constraints or soft constraints:

- One can think of hard constraints as of adding special edges of weight ∞ and $-\infty$, respectively, to the graph. Hard constraints are required to be conflict-free because otherwise no feasible solution exists. Hard ML constraints are transitive: all vertices that are connected by a sequence of must-links will definitely end in the same cluster. Hard CL constraints are not transitive, instead, any sequence of CL constraints forces its vertices to hop conflict-free between the clusters. Note that not all algorithms allow to implement hard constraints directly by adding $\pm\infty$ edges. For example, spectral clustering cannot consider hard constraints in this way because there is no generalization of eigenvectors to deal with $\pm\infty$ entries. However, there exist spectral approaches to constrained clustering, see Wang et al. (2014) for an overview.

- Soft constraints are allowed to be violated if absolutely necessary. One way to think about soft constraints is to fix an arbitrarily large constant $C > \sum_{i,j} |w_{ij}|$ and then connect any two vertices of a must-link by an edge of weight $C + w_{ij}$, and any two vertices of a cannot-link by an edge of weight $-C + w_{ij}$. The result minimizes the number of non-satisfied constraints as a primary goal, since reducing the objective

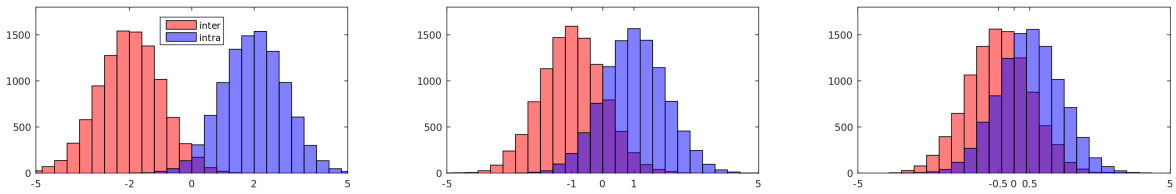


Figure 4.3.1: Edge weight distribution for a graph sampled from the Gaussian correlation graph model $GCG(\,(100, 100),\, 1,\, \mu,\, 1,\, -\mu)$. Weights of intra-edges are sampled from $\mathcal{N}(\mu, 1)$, and weights of inter-edges from $\mathcal{N}(-\mu, 1)$. (left) histograms for the case of $\mu = 2$, which represents a correlation measure that separates classes strongly. In particular, most inter-edges have positive weights and most inter-edges negative weights, since the probability p_{err} of an error (having a misguiding sign) is only $p_{err} = 2.3\%$. (center) similar for $\mu = 1$. Now $p_{err} = 15.9\%$. (right) now $\mu = 0.5$. This correlation measure separates only weakly, since $p_{err} = 30.9\%$. For $\mu = 0$ we would reach $p_{err} = 50\%$, hence, the weights would not contain any class-separating information.

function by C is more important than reducing it by any combination of cut edges.

Constrained clustering with soft constraints shows parallels to correlation clustering: every instance of correlation clustering on $\{-1, 1\}$-weighted graphs can equivalently be interpreted as constrained clustering with $+1/-1$ edges that represent soft ML/CL constraints on an originally empty graph. Further, some publications (Cucuringu et al., 2016) consider soft constraints just as an additional measure of weighted similarity/dissimilarity, which effectively makes their setting equivalent to correlation clustering.

4.3.5 Gaussian correlation graph model

In order to study properties of correlation clustering algorithms, we need a graph model that generates the graph structure plus a correlation measure on its edges. Such a graph model should provide some parameters that allow for interesting and intuitive variations. Here, I introduce the **Gaussian Correlation Graph model**,

$$GCG(\,(n_1, \ldots, n_k),\; p_+,\; \mu_+,\; p_-,\; \mu_-) \quad ,$$

which takes four parameters in addition to the vector of k cluster sizes $(n_1, \ldots, n_k)$: two probabilities $p_+, p_- \in [0, 1]$ and two Gaussian means $\mu_+, \mu_- \in \mathbb{R}$. The edge generation strategy of GCG equals that of the planted partition model, but it additionally assigns Gaussian distributed edge weights. A random graph is created according to this model by sampling each intra-cluster edge independently with probability p_+, plus assigning an edge weight from the normal distribution $\mathcal{N}(\mu_+, 1)$ with mean μ_+ and standard deviation 1. The parameters p_- and μ_- apply analogously to inter-edges.

For $p_+ = p_-$ and $\mu_+ = \mu_-$, the graph does not contain any meaningful correlation information. No clusters can be detected by any strategy. If $p_+ > p_-$ and $\mu_+ = \mu_-$, then

some clustering algorithms may start to work, although the edge weights still contain no information. The intended parameter setting is $\mu_+ > 0$ and $\mu_- < 0$. The larger the absolute values of μ_+ and μ_-, the stronger is the separation, which helps in detecting clusters. The stronger the separation is, the more it even allows for an arbitrary relation between p_+ and p_-, that is even the case $p_+ < p_-$ is allowed in this graph model. Figure 4.3.1 visualizes the separation quality by showing histograms of the edge weights of intra-edges and inter-edges.

The parameters of the GCG model allow for example to study a clustering algorithm in the following scenarios:

- **minimal separation**. $p := p_+ = p_-$ and $\mu := \mu_+ = -\mu_- > 0$. In this setting we let $\mu \to 0$ in order to see how well (for different fixed choices of p) the clustering algorithm is able to find the correct clusters if the separation strength decreases, that is if all edge weights become noisier.

- **dis-/similarity information boost**. For any p_+, p_- and $\mu := \mu_+ = -\mu_- > 0$. This setting studies how the clustering algorithm considers similarity information and dissimilarity information. In particular, a correlation clustering algorithm should *benefit* from increasing p_- for any fixed p_+.

4.3.6 Signed Laplacian matrices

This section presents a generalization of graph Laplacians from positive edge weights to any real-valued edge weights. The **unoriented incidence matrix** $U \in \{0,1\}^{|V|\times|E|}$ of a graph $G = (V, E, W)$ has its entry u_{ie} set to 1 if and only if e is a non-selfloop edge that is incident to vertex i. Each column in U has either none or two 1-entries. The **oriented incidence matrix** $O \in \{-1, 0, 1\}^{|V|\times|E|}$ equals U but flips the sign of any one of the two non-zero entries in a column. The unnormalized Laplacian of a positive-weighted graph can be represented as $L = O \cdot M \cdot O^T$, where $M \in \operatorname{diag}(\mathbb{R}_{>0}^{|E|})$ is the diagonal matrix that lists the weights of all edges along its diagonal. A straightforward approach to generalize L to negative edge weights is to simply allow for negative edge weights along M. This approach preserves the original representation as $L = D - W$ for D being defined by the row sums of the real symmetric matrix W. However, such L is no longer positive semi-definite in general, and it is not clear how its applicability to minimizing balanced cuts is affected.

Another approach to generalizing Laplacians to real symmetric matrices $W \in \mathbb{R}^{n\times n}$ is already studied in the field of graph theory on **signed graphs** by Zaslavsky (1982). A signed graph is an ordinary graph that labels every edge to be of one the following two types: positive or negative. This edge label is independent of any edge weights, but one can interpret every real-weighted graph as a signed graph with positive edge weights, where the edge label represents the sign of the original edge weight. For signed graphs, an individual type of an incidence matrix is used, denoted as the **signed incidence matrix**. It combines the oriented incidence matrix with the unoriented incidence matrix: for every positive edge, the corresponding column equals that of the oriented incidence matrix, and for every negative

edge, the column equals that of the unoriented incidence matrix. The signed incidence matrix implies the following definition of the signed Laplacian $\overline{L}$.

Definition 4.3.3 (Signed Laplacian)
For a real-weighted undirected graph $G = (V, E, W)$, the **signed Laplacian** $\overline{L}$ is defined as

$$\overline{L} := S \cdot M \cdot S^T \quad ,$$

where S is the signed incidence matrix, and M the corresponding diagonal matrix that lists all absolute edge weights along its main diagonal. $\overline{L}$ has the following entry-wise representation:

$$\overline{L} \quad := \quad \begin{cases} \sum_{k \neq i} |w_{ik}| & i = j \\ -w_{ij} & i \neq j \end{cases} \quad .$$

Note that $\overline{L}$ is "just" a diagonal modification of the matrix $D - W$. However, it turns out that $\overline{L}$ has several interesting properties: it is easy to see by Gershgorin's circle theorem, or by representing $\overline{L}$ as a Gramian matrix $\overline{L} = (S\sqrt{M})(S\sqrt{M})^T$, that $\overline{L}$ is positive semi-definite. Moreover, Kunegis et al. (2010) prove that $\overline{L}$ is even positive definite in all interesting cases: $\overline{L}$ has a strictly positive smallest eigenvalue if and only if it is "unbalanced", which is here equivalent to the case that no perfect correlation clustering exists (this gives together with the results by Germina et al. (2011) that the multiplicity of the eigenvalue 0 equals the number of connected components for which a perfect clustering exists). The authors study extensions of balanced cuts to negative weights as well as normalized variants of $\overline{L}$. In particular, they generalize some graph kernels to the real-weighted case, including a signed variant of the resistance distance (Kunegis et al., 2008).

Definition 4.3.3 implies the following representation of $\overline{L}$ as

$$\overline{L} \;=\; \overline{D} - W - 2 \cdot R_{diag} \quad ,$$

where $\overline{D} := \mathrm{diag}(\overline{d}_1, \ldots, \overline{d}_n)$ with $\overline{d}_i := \sum_k |w_{ik}|$, and $R_{diag} := \mathrm{diag}(r_{11}, \ldots, r_{nn})$ with $W = A - R$ the AR-decomposition. Note that $\overline{L}$ is not affected by any selfloops at all, only its representation in terms of the adjacency matrix. If we restrict any existing selfloops to have positive weights, then we get the simple representation as

$$\overline{L} \;=\; \overline{D} - W \quad ,$$

and further the interesting fact that

$$\overline{L} \;=\; L(A) + L_{signless}(R) \quad .$$

Since I could not find any reference that explicitly states this connection to the unnormalized

Laplacian and the signless Laplacian, I emphasize this relation again (for graphs with no negative-weighted selfloops):

> **“ The signed Laplacian is the sum of the unnormalized Laplacian of the attraction plus the signless Laplacian of the rejection. ”**

Fully analogous to the non-negative case, normalized variants of $\overline{L}$ can be defined.

Definition 4.3.4 (Signed symmetric normalized Laplacian)
For symmetric $W \in \mathbb{R}^{n \times n}$ with non-negative main diagonal and no zero-row, the **signed normalized Laplacian** $\overline{\mathcal{L}}$ is defined as

$$\overline{\mathcal{L}} \;:=\; \overline{D}^{-1/2} \cdot \overline{L} \cdot \overline{D}^{-1/2} \;=\; I - \overline{D}^{-1/2} W \overline{D}^{-1/2} \quad .$$

It has the following entry-wise representation:

$$\overline{\mathcal{L}} \;:=\; \begin{cases} 1 - \frac{w_{ii}}{\overline{d}_i} & i = j \\ \frac{-w_{ij}}{\sqrt{\overline{d}_i}\sqrt{\overline{d}_j}} & i \neq j \end{cases} \quad .$$

Definition 4.3.5 (Signed random walk normalized Laplacian)
For symmetric $W \in \mathbb{R}^{n \times n}$ with non-negative main diagonal and no zero-row, let $\overline{P} := \overline{D}^{-1} W$ denote the **signed random walk transition matrix**, and

$$\overline{\mathcal{L}}_{rw} \;:=\; \overline{D}^{-1/2} \cdot \overline{\mathcal{L}} \cdot \overline{D}^{1/2} \;=\; I - \overline{P}$$

the **signed random walk normalized Laplacian**.

The reason to exclude zero-rows in Definition 4.3.4 and Definition 4.3.5 is just to avoid division by zero due to $\overline{d}_i = 0$. However, one can easily generalize to this case by considering the pseudoinverse $(\overline{D}^{+})^{1/2}$ instead of $\overline{D}^{-1/2}$.

In contrast to the unnormalized Laplacian, the normalized variants are affected by selfloops. Allowing for positive selfloops is possible without any changes to the notation, and already considered in both definitions. However, negative selfloops are problematic: in order to achieve a meaningful Rayleigh quotient characterization of the normalized Laplacians, we should multiply by $(\overline{D} - 2 \cdot R_{diag})^{-1/2}$ rather than by $\overline{D}^{-1/2}$, but this raises the question how to deal with non-positive entries in $\overline{D} - 2 \cdot R_{diag}$. For that reason, the definitions of the normalized Laplacians are restricted to positive-weighted selfloops.

Finally, note that for all positive-weighted graphs, the signed Laplacian $\overline{L}$ coincides with the standard unnormalized Laplacian L, and similarly $\overline{\mathcal{L}} = \mathcal{L}$ and $\overline{\mathcal{L}}_{rw} = \mathcal{L}_{rw}$. That is, the signed Laplacian (and its variants) can be interpreted as a straightforward generalization of the standard unnormalized Laplacian (and its variants) from positive-weighted graphs to real-weighted graphs.

4.3.7 Spectral approaches to correlation clustering

This section gives an overview on existing spectral approaches to correlation clustering and finally motivates my alternative approach.

The workaround approach: transform dissimilarities into similarities

Like many other clustering algorithms, spectral clustering is based on a similarity measure. Now consider an application that provides a correlation measure, or equivalently, both a similarity measure s and a dissimilarity measure d. A common approach to such problems in practice is to first combine s and d into a single new similarity measure $\hat{s}$, and then provide $\hat{s}$ as the input to a similarity-based clustering algorithm. This approach has two issues that one should be aware of:

- there is usually no canonical choice for how to convert a dissimilarity measure d into a similarity measure s_d and vice versa, but this choice can have a strong impact on the result. Some typical choices are $s_d := \exp(-d)$ and $s_d := (1+d)^{-1/2}$, transforming large dissimilarity values into small positive similarity values in the unit interval. There are even more alternatives to finally combine s_d and s into a single similarity measure $\hat{s}$.

- the second issue is more fundamental: the transformed dissimilarities might affect the result not as intended! We expect that strongly similar objects get assigned to the same cluster, while strongly dissimilar objects get assigned to different clusters. However, treating strong dissimilarities as very low similarity values in order to apply similarity-based clustering is often *not* the right strategy. This is because similarity-based clustering algorithms put their focus only on the influence of high similarity values, which are interpreted as high similarities plus high confidence, while low values indicate either low similarity or low confidence. In particular, very low similarity values ($w_{ij} \approx 0$) are treated as if no information were given ($w_{ij} = 0$), rather than having a strong opposite influence on the result. For spectral clustering, this property can be seen from its implicit objective function. For example, the spectral relaxation of minimizing the RatioCut (1.8) is equivalent to minimizing the following objective function over certain choices of $\mathbf{f} \in \mathbb{R}^n$ with $\|\mathbf{f}\| = 1$:

$$\sum_{i,j} w_{ij}(f_i - f_j)^2 \quad .$$

  Minimizing this term implies that high similarity weights $w_{ij} \gg 0$ enforce that $f_i \approx f_j$

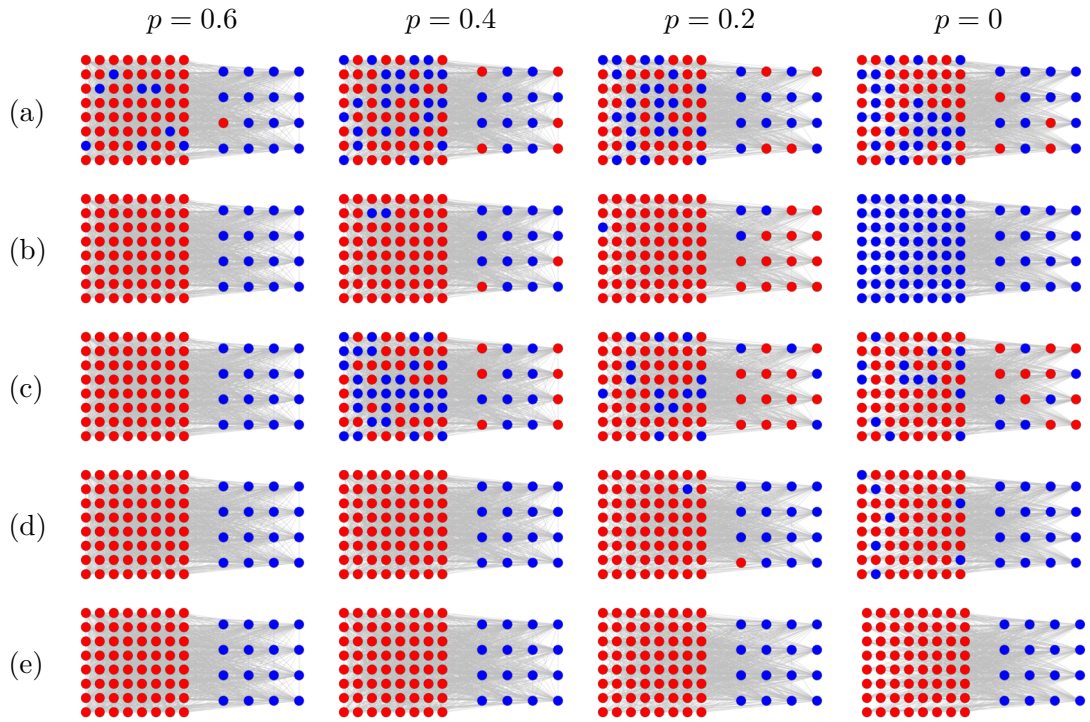


Figure 4.3.2: Spectral clustering with embedded dissimilarities. Each column shows five times the same two-cluster graph that is sampled from $GCG(\,(64, 16),\, p,\, 0.8,\, 0.6,\, -0.8)$. The smaller p the fewer intra-edges/similarities are contained in the graph, thus any clustering algorithm has to exploit the information given by the inter-edges/dissimilarities. The vertex colorings are the result of five different clustering strategies, one per row. Each strategy computes a vector $\mathbf{f} = (f_i)$ and assigns vertex i to the red cluster if $f_i > 0$, or to the blue cluster if $f_i \leq 0$. Along row (a), $\mathbf{f}$ is the second smallest eigenvector of the unnormalized Laplacian $L(W) = D - W$ directly applied to the real-valued matrix W. (b) $\mathbf{f}$ is the second smallest eigenvector of $L(A)$ for $W = A - R$ the AR-decomposition, hence, all dissimilarity information is simply ignored. (c) $\mathbf{f}$ is the second smallest eigenvector of $L(\tilde{W})$, where $\tilde{w}_{ij} := w_{ij} + 1$ if $w_{ij} > 0$ and $\tilde{w}_{ij} := \exp(w_{ij})$ otherwise. This way of embedding dissimilarities gives even worse results than ignoring them as in (b). (d) $\mathbf{f}$ is the second smallest eigenvector of $L(\tilde{W})$, where $\tilde{w}_{ij} := w_{ij} + 1$ if $w_{ij} \geq 0$ and $\tilde{w}_{ij} := \exp(w_{ij})$ otherwise. In contrast to all other approaches, this matrix does not preserve sparsity, however, it outperforms all previous approaches. (e) $\mathbf{f}$ is the *smallest* eigenvector of the signed Laplacian matrix $\overline{L}(W)$. It yields the correct result in all examples. This shows that even in the case of just a few or no similarities, the information given by the dissimilarities can still be sufficient to find the true clusters.

just as intended. However, formerly strong dissimilarities now refer to small weights $w_{ij} \approx 0$ that indeed *allow* for a large difference between f_i and f_j, but there is no penalty term that explicitly *enforces* f_i and f_j to differ. In case $w_{ij} \approx 0$ it does simply not matter whether $f_i \approx f_j$ or $f_i \not\approx f_j$. In this case, any larger difference between f_i and f_j in the result is caused only implicitly due to structural dependencies on some other high similarities. Spectral clustering treats low values and missing values in the same way: by mostly ignoring both. Figure 4.3.2 visualizes this issue.

So, how to solve this problem? The answer is pragmatic: rather than combining similarity and dissimilarity measures into a single similarity measure, keep them separated and apply another strategy that exploits both aspects. There already exist such algorithms, some of which are presented in Section 4.3.4, and some others will be presented now.

Direct spectral approaches to correlation clustering

There already exist spectral approaches to correlation clustering that can directly deal with correlation weights.

- Yu and Shi (2001) define a generalization of the NCut objective to real weights, and show how a relaxed version of this objective can be solved by the largest K eigenvectors of the following generalized eigenvalue problem:

$$(D_R + \delta I - W)\mathbf{v} = \lambda(D_A + D_R + 2\delta I)\mathbf{v} \quad ,$$

  where $\delta \geq 0$ acts as a regularization parameter. We refer to their approach as **AR**.

- Kunegis et al. (2010) study the signed variants of the Laplacian matrices as presented in Section 4.3.6, including generalizations of the balanced cut objectives to real weights. They suggest to consider the smallest K eigenvectors of one of the signed Laplacian matrices, that is they study particularly the eigenvalue problems

$$\text{unnormalized:} \quad (\overline{D} - W)\mathbf{v} = \lambda\mathbf{v} \quad , \quad \text{random walk normalized:} \quad (\overline{D} - W)\mathbf{v} = \lambda\overline{D}\mathbf{v} \quad .$$

  We only consider the random walk normalized variant, denoted as **SL**, since it outperforms the unnormalized variant in the experiments.

- Cucuringu et al. (2016) formulate their approach as soft constrained clustering, which can also be understood as correlation clustering. They take the AR-decomposition of W and then consider the smallest K eigenvectors of the eigenvalue problem

$$L(A)\mathbf{v} = \lambda L(R)\mathbf{v} \quad ,$$

for $L(A) = D_A - A$ and $L(R) = D_R - R$ the corresponding unnormalized Laplacians. We refer to their approach as **LL**.

All these spectral approaches to correlation clustering agree on two aspects:

1. they all focus on minimizing a balanced cut by adapting classical spectral clustering to the non-negative setting.

2. they all focus on solving multi-class instances by running k-means (or a similar algorithm) on the spectral embedding of the vertices into $\mathbb{R}^K$ as given by the entries of the K smallest/largest eigenvectors.

Regarding the second point, Lee et al. (2014) show that considering the spectral embedding is near-optimal for the goal of minimizing a specific balanced k-cut, namely the Cheeger cut generalized to higher orders $k \geq 2$. As a consequence, considering the spectral embedding is a reasonable choice for the goal of the first aspect: minimizing balanced cuts.

However, my alternative spectral approach, as presented in Section 4.4.2, differs from the other spectral approaches in both aspects. In particular, it does *not* focus on balanced cuts.

4.3.8 Correlation clustering without balance constraints

As already discussed in Section 1.3.6, the minimum cut of a similarity graph prefers to cut off individual vertices. This issue is the original motivation for studying the minimum cut under balance constraints, such as RatioCut and NCut. However, it is a fallacy to believe that this issue applies to correlation graphs, too. Even quite the contrary is true: the minimum cut of a correlation graph is no longer biased toward meaningless imbalanced partitions, but toward the correct clusters! The reason for this fact is that correlation weights provide a stronger separation than similarity weights: intra-edges tend to be positive, but inter-cluster edges tend to be negative. This is far more informative than in similarity-based clustering, where all edge weights are positive.

This simple, yet fundamental insight can be illustrated in terms of the following graph model for $\{-1, 1\}$-weighted graphs, denoted as the **Signed Planted Partition model**:

$$SPP(\ (n_1, \ldots, n_k),\ p_+,\ \xi_+,\ p_-,\ \xi_-) \quad .$$

This model takes four parameters in addition to the vector of the k cluster sizes: $p_+ \in [0, 1]$ is the sampling probability for each intra-edge. The weight of an intra-edge is either 1 ("correct") or -1 ("erroneous"), where the "erroneous" case occurs randomly according to the intra-error probability $\xi_+ \in [0, 1]$. Similarly, $p_- \in [0, 1]$ is the sampling probability for each inter-edge. Its weight is either -1 ("correct") or 1 ("erroneous"), according to the inter-error probability $\xi_- \in [0, 1]$. Graphs from $SPP(\ (n_1, \ldots, n_k),\ p_+,\ 0,\ p_-,\ 0)$ trivially provide a perfect correlation clustering that can be determined simply by cutting at all negative edges. For small error probabilities ξ_+ and ξ_-, most correlation clustering algorithms are still able

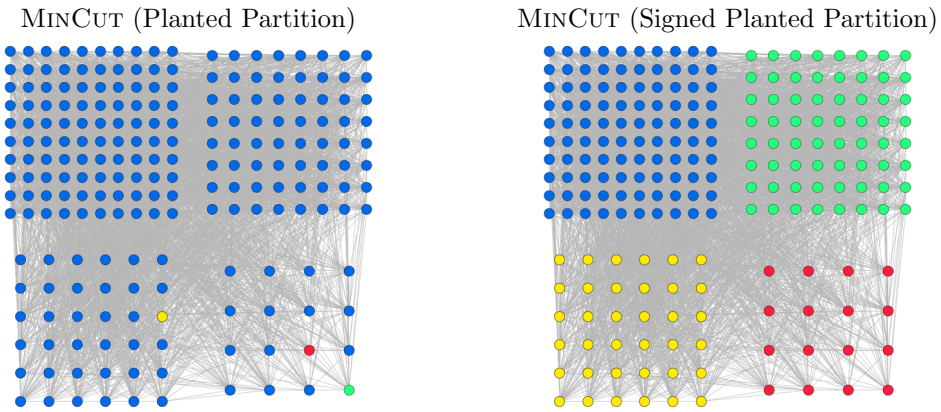


Figure 4.3.3: Unbalanced minimum cut of a correlation graph. (left) clustering result given by the minimum 4-cut of a graph from the planted partition model $PP((100, 64, 36, 16), 0.3, 0.15)$. The cut simply cuts off 3 individual vertices of minimum degree. Only under additional balance constraints (like the normalized 4-cut, not shown here) the correct 4-clustering is found. (right) the same graph after "correcting" all inter-edge weights from 1 to -1. Now the minimum cut easily finds the correct 4-clustering (even for an unspecified number of clusters $k \geq 1$). No balance constraints are required.

to find the correct clusters. The standard planted partition model can be identified as an instance of the signed planted partition model as follows:

$$PP(\ (n_1, \ldots, n_k),\ p_+,\ p_-) \quad = \quad SPP(\ (n_1, \ldots, n_k),\ p_+,\ 0,\ p_-,\ 1) \quad .$$

That is, in the standard planted partition model, *every* inter-edge is erroneous because it has a misguiding positive weight rather than a helpful negative weight. Figure 4.3.3 provides an example which shows that a hard instance of the planted partition model becomes trivial as soon as the implicit inter-error probability ξ_- switches from 1 to 0, which turns all inter-edge weights from $+1$ into -1. This insight generalizes to weighted similarity graphs: every inter-edge in a similarity graph is principally misguiding and should better not even exist rather than existing with a positive weight. Let us summarize this insight in a catchy statement:

“ Correlation clustering is *simpler* than similarity clustering. ”

Of course this statement is not meant in a precise complexity theoretical sense: both, similarity clustering (= balanced cut minimization of non-negative graphs) and correlation clustering (= minimization of the CC functional, that is unbalanced cut minimization of real-weighted graphs) are NP-hard problems. However, there is a difference: correlation clustering provides with the CC functional a canonical objective function to minimize. In contrast to that, similarity-based clustering provides no universal answer to the question which objective

function to choose — choosing the right balancing method is an inductive bias that must be made individually for each problem domain (although normalized cuts work often well).

All non-spectral correlation clustering algorithms in this chapter focus on unbalanced cut minimization. In contrast to that, all existing spectral approaches to correlation clustering focus on balanced cut minimization - motivated by its usefulness for similarity clustering. But since balance constraints are not required in correlation clustering, I suggest that spectral strategies should rather focus on *unbalanced* cut minimization, too. Additional balance constraints should be considered only if an application explicitly requires them. In this sense, unbalanced spectral correlation clustering is the "canonical" spectral approach to correlation clustering.

4.3.9 Performance metrics for evaluation

In order to compare different correlation clustering algorithms, we need to choose a performance metric that allows for a fair comparison between clusterings of a different number of clusters. A performance metric is **extrinsic** if it compares a clustering to some given true classes. A common extrinsic metric is purity, defined as follows.

Definition 4.3.6 (Purity)
For any finite set $V = C_1 \dot\cup \ldots \dot\cup C_\ell$ of n elements partitioned into ℓ classes, the **purity** of any k-clustering $V = V_1 \dot\cup \ldots \dot\cup V_k$ is defined as

$$\frac{1}{n} \sum_{i=1}^{k} \max_{j=1\ldots\ell} |V_i \cap C_j| \quad \in \quad [0,1] \quad .$$

In words, each cluster V_i is "assigned" to the class C_j to which it has the largest overlap. Summing over the sizes of these overlaps defines the purity (after finally dividing by n). Purity is the fraction of elements that is labeled correctly if each cluster is assigned to a class by majority vote. The higher the purity, the more likely it is that most or all elements in a cluster belong to the same class. While this is a simple and intuitive score, it has a caveat. Purity equals 1 if and only if every cluster V_i is a subset of some class, regardless of the total number of clusters. That is, purity focuses on achieving perfect homogeneity within a cluster, while totally ignoring the number of clusters. In particular, the trivial clustering into n singleton clusters achieves the best-possible purity score of 1, too. Thus, purity alone is not a good choice when comparing clustering results of a different number of clusters, since it prefers partitions that have more clusters. However, if we consider purity together with the number of clusters as a two-parameter-criterion, then we can achieve a fair comparison. We consider one clustering better than another, if it provides higher purity *and* a lower number of clusters at the same time.

There are several alternative metrics that deal with the trade-off between homogeneity and the number of clusters in various ways. Amigó et al. (2009) compare several extrinsic

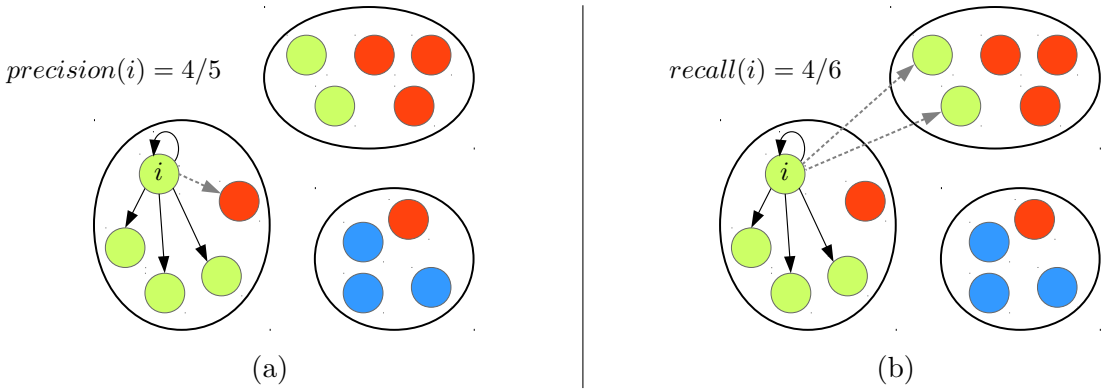


Figure 4.3.4: BCubed precision and recall. Both plots show the same 14 elements of three classes (colors) grouped into three clusters (ellipsoids). (a) the precision of element i is the fraction of elements in the same cluster that belong to the class of i. (b) the recall of element i is the fraction of all elements of the same class that stay in the same cluster with i. The overall BCubed precision/recall of the clustering is defined by averaging over all elements.

clustering metrics by an interesting axiomatic approach: first, they define and justify four properties that a good metric should provide. Informally, a metric is good if it acts according to the following four principles.

- **homogeneity**: each cluster should only contain elements of a single class
- **completeness**: each cluster should contain all elements of a class
- **rag bag**: assigning an element wrongly to a homogeneous cluster is worse than assigning it wrongly to a heterogeneous cluster
- **clusters size vs. quantity**: a small error in a single large cluster is better than several small errors in several small clusters

Then, the authors study which axioms are satisfied by which clustering metrics. For example, the purity measure does not satisfy completeness, since it does not penalize the segmentation of a class into multiple homogeneous clusters. It turns out that only the metrics "BCubed precision" and "BCubed recall" (Bagga and Baldwin, 1998) satisfy all four axioms, as well as their harmonic mean "BCubed F". The BCubed metrics are based on an element-wise definition of precision and recall, finally taking their respective average scores. Figure 4.3.4 visualizes the idea of this element-wise precision and recall, and Definition 4.3.7 states it precisely.

Definition 4.3.7 (BCubed metrics)
Let $V = C_1 \cup \ldots \cup C_\ell$ denote a finite set of n elements partitioned into ℓ classes, and $V = V_1 \cup \ldots \cup V_k$ another partition into k clusters. **BCubed precision** BCP and **BCubed recall** BCR of the clustering are defined as

$$BCP := \frac{1}{n} \sum_{i \in V} \frac{|C(i) \cap V(i)|}{|V(i)|} \quad , \qquad BCR := \frac{1}{n} \sum_{i \in V} \frac{|C(i) \cap V(i)|}{|C(i)|} \quad ,$$

where $C(i)$ denotes the class that contains i, and $V(i)$ the cluster that contains i. **BCubed F** is defined as the harmonic mean of BCP and BCR:

$$BCF := \frac{1}{0.5/BCP + 0.5/BCR} \quad .$$

We consider the following quantities for the later comparison of different clusterings:

NC The number of clusters. Should not deviate much from the number of true classes.

PUR Purity as made precise in Definition 4.3.6. The higher the better, but always considered together with the number of clusters.

BCF The BCubed F metric as defined in Definition 4.3.7. The higher the better.

CWD Cut Weight Difference. That is the difference $c - c^*$ between the cut weight c given by the clusters and the cut weight c^* given by the true classes. The closer to 0 the better. Negative values indicate the presence of strong noise on the edge weights, which misguides the cut minimization approach toward a non-intended clustering.

4.4 Main Contributions

This section presents my main contributions to spectral correlation clustering. The first main contribution is that the smallest eigenvector of the signed Laplacian matrix can be interpreted as a spectral relaxation of the CC functional constrained to at most 2-clusterings. The second contribution is an efficient algorithm that exploits this insight in order to approximately maximize the CC functional of correlation-weighted graphs.

4.4.1 Contribution 1: Unbalanced spectral relaxation of the optional MinCut

It is a well-known fact (see Fact 1.3.5) that the smallest eigenvalue of the unnormalized Laplacian equals zero, with $\mathbf{1}$ a corresponding eigenvector. The same result holds true for the random walk normalized Laplacian matrix, and a similar result holds true for the symmetric

normalized Laplacian matrix, where the corresponding eigenvector is $\sqrt{\mathbf{d}}$ instead of $\mathbf{1}$. In all cases, the multiplicity of the zero-eigenvalue equals the number of connected components in the graph. For connected graphs, the second smallest eigenvector is an interesting object for studying balanced 2-cuts, and higher eigenvectors are helpful in determining a larger number of clusters. The alternative recursive spectral clustering approach focuses solely on the second-smallest eigenvector of the subgraphs. In all these scenarios, the "trivial" smallest eigenvector of the graph Laplacians is no longer of interest.

However, the smallest eigenvector is less trivial than it seems! In order to see why, consider the Rayleigh quotient characterization (1.3) for the unnormalized Laplacian matrix $L = D - W$ of non-negative symmetric W, which can be re-formulated as follows (see Section 4.5.1 for details):

$$\begin{aligned} R(L, \mathbf{v}_1) \;=\; \min_{\mathbf{f} \neq \mathbf{0}} \frac{\mathbf{f}^T L \mathbf{f}}{\mathbf{f}^T \mathbf{f}} \quad &\Leftrightarrow \quad \min_{\mathbf{f}^T\mathbf{f} = n} \sum_{\substack{i,j \\ i \neq j}} w_{ij}(f_i - f_j)^2 \\ &\overset{\leftarrow \text{relax}}{\approx} \quad \min_{\mathbf{f} \in \{-1,1\}^n} \sum_{\substack{i,j \\ i \neq j}} w_{ij}(f_i - f_j)^2 \\ &\Leftrightarrow \quad \max_{\substack{k=1,2 \\ Y \in \mathcal{C}_k}} \sum_{i,j} w_{ij} Y_{ij} \quad . \end{aligned} \tag{4.4}$$

Equation (4.4) shows that the "trivial" smallest eigenvector of L focuses on solving all the following equivalent problems:

- maximize the CC functional (4.1), restricted to at most 2-clusterings
- find a 1-or-2-clustering of V that minimizes the sum of all cut edges. Since the term "MinCut" does not allow for the trivial 1-clustering, we denote this extended definition as the "optional MinCut", see Definition 4.4.1.
- determine the strongest negative 2-cut, or return the 1-clustering of cut weight 0 if no negative cut exists

The "optional MinCut" is made precise in the following definition.

Definition 4.4.1 (optional MinCut)
Let $G = (V, E, W)$ denote an undirected graph with real-weighted edges. The **optional MinCut**, also denoted as **OptMinCut**, is defined as

$$\text{OptMinCut}(W) \;:=\; \min_{S \subseteq V} \text{cut}(S) \quad = \quad \min_{S \subseteq V} \sum_{i \in S, j \notin S} w_{ij},$$

where the trivial cases $S = \emptyset$ and $S = V$ are particularly allowed.

The optional MinCut is always non-positive, since the trivial 1-clustering has cut weight 0. Let us again state the following important relation between the CC functional and the optional MinCut, as given by Equation 4.2:

> **“ Maximizing the CC functional restricted to at most two clusters is equivalent to determining the OptMinCut. ”**

Hence, every time we run an algorithm to compute the smallest eigenvector of L, this algorithm implicitly minimizes a rather non-trivial objective function: the optional MinCut. For positive-weighted graphs, every eigensolver successfully outputs a vector v according to the minimum eigenvalue 0 (= weight of the optional MinCut). It either holds that $v = \mathbf{1}$, which indicates the “trivial” clustering into a single cluster of cut weight 0, or v indicates a 2-cut that separates disconnected graph components from each other, which “trivially” yields cut weight 0, too. In any case, it is not the smallest eigenvector that is trivial per se, it is the context of a positive-weighted graph that only allows for trivial solutions to the OptMinCut optimization problem. When stepping from positive-weighted graphs to real-weighted graphs, the relevance of the information contained in the smallest eigenvector changes drastically, together with the complexity of computing the exact minimum cut.

The enriched interpretation of the smallest eigenvector gets apparent for the signed Laplacian $\overline{L}$, as shown by the following lemma.

Lemma 4.4.2 (Signed Laplacian relaxation)
Let $G = (V, E, W)$ denote a real-weighted graph with $W = A - R$ the AR-decomposition of its weight matrix. The smallest eigenvector of the signed Laplacian $\overline{L}$ is a minimizer of the following relaxation of OptMinCut*:*

$$\begin{aligned}
\min_{\mathbf{f}\in\mathbb{R}^n} \frac{\mathbf{f}^T\overline{L}\mathbf{f}}{\mathbf{f}^T\mathbf{f}} \quad &\Leftrightarrow \quad \min_{\mathbf{f}^T\mathbf{f}=n} \sum_{\substack{i,j\\ i\neq j}} a_{ij}(f_i - f_j)^2 + \sum_{\substack{i,j\\ i\neq j}} r_{ij}(f_i + f_j)^2 \qquad (\star) \\
&\overset{\leftarrow \textit{relax}}{\approx} \quad \min_{\mathbf{f}\in\{-1,1\}^n} \sum_{\substack{i,j\\ i\neq j}} a_{ij}(f_i - f_j)^2 + \sum_{\substack{i,j\\ i\neq j}} r_{ij}(f_i + f_j)^2 \\
&\Leftrightarrow \quad \mathrm{OptMinCut}(W) \quad .
\end{aligned}$$

Let us study the two sums in $(\star)$. They represent two contrary effects on how the entries f_i of the smallest eigenvector are located along the real line:

- $\sum a_{ij}(f_i - f_j)^2$ is affected only by positive entries in W: minimizing this sum implies that high similarity values $a_{ij} > 0$ force the entries f_i and f_j to stay close to each other, $f_i \approx f_j$. In particular, it is likely that both entries have the same sign, $\mathrm{sgn}(f_i) = \mathrm{sgn}(f_j)$.

- $\sum r_{ij}(f_i + f_j)^2$ is affected only by negative entries in W. This is an extra penalty term that is missing in purely similarity-based approaches. High dissimilarity values $r_{ij} > 0$ force the entries f_i and f_j to stay away from each other in order to provide $f_i \approx -f_j$. In particular, it is likely that both entries have opposite signs, $\operatorname{sgn}(f_i) = -\operatorname{sgn}(f_j)$.

This observation suggests to define a clustering $V = V_+ \cup V_-$ from the entries in $\mathbf{f}$ as follows: $i \in V_+$ if $f_i > 0$, and $i \in V_-$ if $f_i \leq 0$, that is to split at zero.

A similar spectral relaxation of the OptMinCut can be achieved by the largest eigenvector of the adjacency matrix, as shown in the following Lemma.

Lemma 4.4.3 (Adjacency matrix relaxation)
The largest eigenvector of the real-weighted adjacency matrix W is a maximizer of the following relaxation of OptMinCut*:*

$$\max_{\mathbf{f}\in\mathbb{R}^n} \frac{\mathbf{f}^T W \mathbf{f}}{\mathbf{f}^T \mathbf{f}} \;\Leftrightarrow\; \max_{\mathbf{f}^T\mathbf{f}=n} \sum_{i,j} w_{ij} f_i f_j \overset{\leftarrow relax}{\approx} \max_{\mathbf{f}\in\{-1,1\}^n} \sum_{i,j} w_{ij} f_i f_j \;\Leftrightarrow\; \text{OptMinCut}(W)$$

Let us study the sum $\sum w_{ij} f_i f_j$:

- similarity values $w_{ij} > 0$ force the entries f_i and f_j to tend to $\operatorname{sgn}(f_i) = \operatorname{sgn}(f_j)$
- dissimilarity values $w_{ij} < 0$ force the entries f_i and f_j to tend to $\operatorname{sgn}(f_i) = -\operatorname{sgn}(f_j)$

In both cases, a stronger confidence $|w_{ij}|$ encourages larger values of $|f_i|$ and $|f_j|$. Again this suggests to split the entries of $\mathbf{f}$ at zero in order to get a 2-cut that approximately minimizes the optional MinCut.

Both approaches perform well in the experiments, but they are sensitive to strong degree imbalances, as for example in case of a hub vertex. For that reason, a third candidate is superior to both: the relaxation by the signed random walk Laplacian.

Lemma 4.4.4 (Signed random walk Laplacian relaxation)
The smallest eigenvector of the signed random walk Laplacian $\overline{\mathcal{L}}_{rw}$ is a maximizer of the following relaxation of OptMinCut*:*

$$\max_{\mathbf{f}\in\mathbb{R}^n} \frac{\mathbf{f}^T W \mathbf{f}}{\mathbf{f}^T \overline{D} \mathbf{f}} \;\Leftrightarrow\; \max_{\mathbf{f}^T\mathbf{f}=n} \frac{\sum_{i,j} w_{ij} f_i f_j}{\sum_i \overline{d}_i f_i^2} \overset{\leftarrow relax}{\approx} \max_{\mathbf{f}\in\{-1,1\}^n} \frac{\sum_{i,j} w_{ij} f_i f_j}{\sum_i \overline{d}_i f_i^2} \;\Leftrightarrow\; \text{OptMinCut}(W)$$

Let us summarize the above: for correlation-weighted graphs, the smallest eigenvector of $\overline{L}$ (and its variants) provides a novel spectral approach to correlation clustering. This stems from the fact that the smallest eigenvector solves a relaxation of the OptMinCut optimization

problem, which is equivalent to maximizing the CC functional restricted to at most two partitions. For positive-weighted graphs OptMinCut is solved by every "trivial" cut of weight 0, which is always correctly selected by any "trivial" eigenvector of $\overline{L} = L$ and its variants.

4.4.2 Contribution 2: The SCC algorithm

In contrast to all alternative spectral approaches from Section 4.3.7, which focus on balanced clustering, my approach focuses on the minimum cut *without* any balance constraints. As suggested by the previous section, it considers only the smallest eigenvector of the signed normalized Laplacian.

Main idea

The idea of the **Spectral Correlation Clustering (SCC) algorithm** is to recursively subdivide the graph into two subgraphs by solving the spectral relaxation of the optional MinCut. If this yields a 2-cut, then we continue the recursive processing for each of the two identified subgraphs. If we get the trivial 1-clustering (which implies cut weight 0), then we apply no further subdivision. The subdivision can also be stopped earlier (optional) by any additional explicit pruning strategy. The output of SCC is a partitioning of the vertices into an unspecified number of clusters. Spectral Correlation Clustering is equal to recursive spectral clustering carried out on the smallest eigenvector of the signed normalized Laplacian matrix. Beside the fact of being defined with respect to the smallest rather than the second-smallest eigenvector, SCC shows two fundamental differences to recursive spectral clustering of similarity graphs:

- **auto-balancing/adaptivity**: for similarity graphs, every subdivision step requires explicit balance constraints (by taking second-smallest eigenvectors) in order to avoid meaningless cuts and early over-segmentation. In contrast to that, SCC does not require any additional constraints. It adapts automatically to any class balance. Moreover, the hierarchy of unconstrained minimum 2-cuts is conceptually simple and formally accessible — in contrast to a hierarchy of balanced cuts, for which it is hard to grasp the interplay of the balance constraints across different levels of the hierarchy.
- **auto-pruning/efficiency**: in contrast to similarity-based recursive clustering, which always subdivides a graph down to n single-vertex clusters at its leafs, SCC does not require an explicit pruning strategy because it automatically prunes at subgraphs that provide no negative cut. Since all higher-level subgraphs are split at their *strongest* negative cut, this implicit pruning even favors early stopping.

Because of its implicit pruning, SCC does not require the number of clusters to be given as an input parameter. For simple instances of correlation clustering, SCC finds the correct number of clusters automatically. For more complex (noisy and sparse) instances, SCC tends to identify too many clusters. However, these over-segmented results from SCC are good-natured in the sense that the clusters still respect class boundaries, so the correct class can be recovered

by a union of clusters. Although there is no formal proof for this property, it can be seen intuitively from the hierarchy of unconstrained minimum 2-cuts, and it is also justified in the later experiments by the high purity scores. Like other spectral clustering techniques, SCC can be computed efficiently and is highly scalable.

The SCC algorithm

The SCC algorithm recursively sub-divides the vertex set V of the input graph $G = (V, E, W)$ into a hierarchical structure of bi-partitions. This structure is represented as a binary tree, the **cluster-tree**. Each tree node has either none or two children. The root node corresponds to V, and any other tree node refers to a non-empty subset of V. Whenever a tree node t has children u and v, they represent a proper bi-partition of their parent, that is $V_t = V_u \dot\cup V_v$. As a consequence, the leafs of the tree (that is all tree nodes that have no children) provide a k-clustering $V = V_1 \dot\cup V_2 \dot\cup \ldots \dot\cup V_k$, where $k \in \{1, \ldots, n\}$ is defined by the number of leafs.

The cluster-tree is created in the SCC algorithm by a call to the recursive subroutine `ClusterTree` (Algorithm 4.4.1). The SCC algorithm itself is listed as Algorithm 4.4.2. Beside creating the cluster-tree, it further deals with the case of unconnected graphs by executing the SCC algorithm on each connected component separately.

Alternative spectral relaxations. In Step 4 of the `ClusterTree` subroutine, the SCC algorithm employs the signed normalized Laplacian $\overline{\mathcal{L}}$ for solving the spectral relaxation of the CC functional. The reason for picking $\overline{\mathcal{L}}$ is that it performs better than the alternative candidates $\overline{\mathcal{L}}_{rw}$, $\overline{L}$ and W. Choosing $\overline{\mathcal{L}}_{rw}$ would give exactly the same clustering results as $\overline{\mathcal{L}}$, since the smallest eigenvector v of $\overline{\mathcal{L}}$ implies that $w := \overline{D}^{-1/2} v$ is the smallest eigenvector of $\overline{\mathcal{L}}_{rw}$, yielding the same bi-partition $V_+ \dot\cup V_-$ when splitting at 0. The advantage of $\overline{\mathcal{L}}$ is rather practical, as already stated in Section 1.3.4: since $\overline{\mathcal{L}}$ is guaranteed to be symmetric and positive semi-definite, we can apply more efficient eigenvector approximation algorithms than those available for non-symmetric $\overline{\mathcal{L}}_{rw}$. Further, an empirical comparison of $\overline{\mathcal{L}}$, $\overline{L}$ and W similar to the experiments in Section 4.5.3 revealed that $\overline{L}$ is consistently outperformed by $\overline{\mathcal{L}}$ and W on instances that have just a weak class separation, even if other thresholding techniques than splitting at 0 are applied. W and $\overline{\mathcal{L}}$ perform equally well in many cases unless the graph contains hub vertices that introduce strong degree imbalances. In that case $\overline{\mathcal{L}}$ is superior over W. Thus, since $\overline{\mathcal{L}}$ is consistently superior over its alternatives, we only consider SCC defined with respect to $\overline{\mathcal{L}}$ in the following.

Pruning strategies. There is no need to define an explicit pruning strategy for the SCC algorithm. However, pruning strategies allow to incorporate additional domain knowledge into the clustering process. There exist several different pruning strategies that can be applied "online" in Step 3 of the `ClusterTree` subroutine. Moreover, one can also add an additional "offline" pruning strategy that is applied after the creation of the cluster-tree has finished. In contrast to online-pruning, such an offline-pruning approach is able to base its decision on information on the whole tree.

Algorithm: `ClusterTree`

Input:

- $G = (V, E, W)$ a real-weighted graph (constant global input)
- t tree node with no children
- $\hat{V} \subseteq V$ set of vertices attached to t

Output:

- via side-effect: either t is left unchanged, or two new tree nodes are attached to t as its children (both already processed recursively by `ClusterTree`).

Algorithm:

1. determine the subgraph $\hat{G} := (\hat{V}, \hat{E}, \hat{W})$ that is induced by $\hat{V}$ in G
2. if $\hat{E} = \emptyset$, then return
3. (optional) evaluate any additional explicit pruning strategy, and return if it applies
4. compute $\mathbf{f}$ as the smallest eigenvector of the signed normalized Laplacian $\overline{\mathcal{L}}$ of $\hat{W}$
5. if all entries in $\mathbf{f}$ are non-negative, or if all entries are non-positive, then return
6. split $\hat{V}$ into $V_+ \cup V_-$ according to the positive and non-positive entries in $\mathbf{f}$, that is $V_+ := \{i \in \hat{V} \mid f_i > 0\}$ and $V_- := \{i \in \hat{V} \mid f_i \leq 0\}$
7. create two new tree nodes: t_+ for V_+ and t_- for V_-, and attach them to t as its children. Call `ClusterTree(`t_+`)` and `ClusterTree(`t_-`)`.

Algorithm 4.4.1: `ClusterTree` algorithm

Algorithm: `SCC`

Input:

- $G = (V, E, W)$ a real-weighted graph

Output:

- $\ell : V \to \mathbb{N}$ cluster label for each vertex

Algorithm:

1. determine the connected components of G. If G consists of multiple components, then call `SCC` for each component individually, define ℓ by combining the outputs using individual cluster labels for each component, and return ℓ.
2. create a tree node t with V attached to it and call `ClusterTree(`t`)`
3. determine the partition $V = V_1 \cup V_2 \cup \ldots \cup V_k$ from the k leafs of the cluster-tree (as found by a tree-walk started at t)
4. define $\ell(i) := c$ for $i \in V_c$ and return ℓ

Algorithm 4.4.2: `SCC` algorithm

Two examples for pruning online during the creation of the cluster-tree are:

1. during the recursive calls to `ClusterTree`, keep track of the current recursion level and prune if the level exceeds some threshold d. This bounds the number of leafs in the cluster-tree to at most 2^d.

2. compute the partition $V_+ \dot\cup V_-$ and the corresponding (negative) cut weight of splitting $\hat{V}$. Prune if this cut weight is negligibly close to zero.

Two examples for offline-pruning after constructing the cluster-tree are:

1. compute the unpruned cluster-tree and the (negative) total cut weight c that is achieved by it. Merge two siblings of leafs that increase the total cut weight as least as possible if the new total cut weight is still below $0.95c$. Iteratively merge in this way until any further merge would raise the total cut weight above $0.95c$.

2. for a fixed k and a cluster-tree with more than k leafs, try to extract the best k-clustering as follows: randomly select any k vertices that form a partition of V, for example by a sequence of $k-1$ randomly selected splits started at the root. Repeat this randomization multiple times and keep the solution with the minimal total cut.

The application of any explicit pruning strategy should be motivated from additional knowledge on the respective problem domain, rather than being applied in general. For example, in the application in Section 4.6 we prune as soon as it holds for every vertex $i \in \hat{V}$ that $\sum_{j\in\hat{V}} w_{ij} - \sum_{j\notin\hat{V}} w_{ij} > \tau$ for some threshold parameter τ. This pruning criterion can be interpreted in its context as a level of user satisfaction, so we prune if all users "tolerate" to be grouped together as $\hat{V}$.

Must-links. The SCC algorithm can directly be applied to soft-constrained must-links and to soft-constrained cannot-links. It is further possible to deal with hard-constrained must-links by preprocessing the input graph to the SCC algorithm as follows: all vertices that are connected via must-links are replaced together by a single representative super-vertex. This super-vertex aggregates all edge weights of its vertices in such a way that it contributes the same amounts to volumes and cut weights as if all individual vertices were present. Precisely, grouping the vertices $A \subseteq V$ of an undirected graph into the super-vertex a defines the new set of vertices $V' = (V \setminus A) \cup \{a\}$ and the following edge weights w_{ai} for $i \in V'$:

$$w_{ai} \;=\; \begin{cases} \sum_{k\in A} w_{ki} & i \neq a \\ \sum_{k,\ell\in A} w_{k\ell} & i = a \end{cases}$$

In contrast to must-links, there is no straightforward way to deal with hard-constrained cannot-links.

Other variants of the SCC algorithm. The SCC algorithm can be varied in many ways that further improve its performance. For example, there exist several strategies to increase the robustness of the `ClusterTree` subroutine against suboptimal choices of V_+ and V_-. A simple such strategy is to finally move vertices between V_+ and V_- as long as this further decreases the cut weight. Furthermore, since each of the subgraphs induced by V_+ and V_- can be disconnected, one could further split V_+ and V_- into their connected components $V_{+,1}, \dots, V_{+,a}, V_{-,1}, \dots, V_{-,b}$ at the end of each call to `ClusterTree`. Optionally, one could then further check for every pair of connected subsets $(V_{+,x}, V_{+,y})$ for $x = 1, \dots, a$ and $y = 1, \dots, b$, whether merging these two subsets improves the cut weight. In these variants, each inner node of the resulting cluster-tree can have more than two children.

A similar merging strategy can particularly be applied to the partition $V = V_1 \dot\cup V_2 \dot\cup \dots \dot\cup V_k$ in Step 3 of `SCC` in order to reduce the number of clusters as long as this improves the cut. As another alternative, one could consider each V_i as a super-vertex and recursively apply the SCC algorithm to the graph that consists of these k super-vertices with aggregated edges between them. The general approach of these strategies is to first split in too many, but not far too many, high-purity subcomponents, and finally merge them as long as this improves the overall cut.

Indeed, although not presented here, the usefulness of the above and other strategies is confirmed by additional experiments. However, any such algorithmic fine-tuning steps away from the plain spectral approach and is about to make the comparison to non-optimized strategies less fair. For that reason, this thesis focuses on the performance of the bare SCC algorithm as such, without any additional enhancements.

4.5 Proofs and technical details

This section proves and discusses the spectral relaxation of the OptMinCut optimization problem. Further, it compares the SCC algorithm quantitatively and qualitatively to the spectral and non-spectral alternative approaches to correlation clustering.

4.5.1 Proofs of the spectral relaxations

Before we prove the relaxations, let us first motivate the definition of the signed Laplacian from a Rayleigh quotient point of view.

Motivation by the Rayleigh quotient characterization

The Rayleigh quotient characterization of the smallest eigenvector of $L = D - W$ can be related as follows to the CC functional for non-negative symmetric W:

$$\begin{aligned}
\min_{\mathbf{f}\in\mathbb{R}^n} \frac{\mathbf{f}^T L\mathbf{f}}{\mathbf{f}^T\mathbf{f}} \quad &\Leftrightarrow \quad \min_{\mathbf{f}^T\mathbf{f}=n} \sum_{\substack{i,j\\ i\neq j}} w_{ij}(f_i - f_j)^2 \\
&\overset{\text{relax}}{\approx}\rightarrow \quad \min_{\mathbf{f}\in\{-1,1\}^n} \sum_{\substack{i,j\\ i\neq j}} w_{ij}(f_i - f_j)^2 \\
&\Leftrightarrow \quad \max_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} \sum_{i,j} w_{ij} Y_{ij} \quad .
\end{aligned}$$

This can be derived as a special case from the next equation block by setting $r_{ij} := 0$. If we apply the definitions of L and D unchanged to a real-valued matrix W, then we get with $W = A - R$ the AR-decomposition (Definition 4.3.2) of W that

$$L = D - W = \begin{cases} d_i^A - d_i^R - a_{ii} + r_{ii} & i = j \\ r_{ij} - a_{ij} & i \neq j \end{cases} \quad .$$

The Rayleigh quotient characterization of this real-valued alternative can be related to the CC functional as follows:

$$\begin{aligned}
\min_{\mathbf{f}\in\mathbb{R}^n} \frac{\mathbf{f}^T L\mathbf{f}}{\mathbf{f}^T\mathbf{f}} \quad &\Leftrightarrow \quad \min_{\mathbf{f}^T\mathbf{f}=n} \sum_{\substack{i,j\\ i\neq j}} a_{ij}(f_i - f_j)^2 + \sum_{\substack{i,j\\ i\neq j}} r_{ij}(f_i + f_j)^2 - 4\cdot\sum_{\substack{i,j\\ i\neq j}} r_{ij} f_i^2 \qquad (\star) \\
&\overset{\text{relax}}{\approx}\rightarrow \quad \min_{\mathbf{f}\in\{-1,1\}^n} \sum_{\substack{i,j\\ i\neq j}} a_{ij}(f_i - f_j)^2 + \sum_{\substack{i,j\\ i\neq j}} r_{ij}(f_i + f_j)^2 - 4\cdot\sum_{\substack{i,j\\ i\neq j}} r_{ij} f_i^2 \\
&\Leftrightarrow \quad \min_{\mathbf{f}\in\{-1,1\}^n} \sum_{\substack{i,j\\ i\neq j}} a_{ij}(f_i - f_j)^2 + \sum_{\substack{i,j\\ i\neq j}} r_{ij}(f_i + f_j)^2 \\
&\Leftrightarrow \quad \min_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} \sum_{i,j} a_{ij}(1 - Y_{ij}) + \sum_{i,j} r_{ij} Y_{ij} \\
&\Leftrightarrow \quad \max_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} \sum_{i,j} w_{ij} Y_{ij} \quad .
\end{aligned} \tag{4.5}$$

Two of the three summands that appear in $(\star)$, namely $\sum a_{ij}(f_i - f_j)^2$ and $\sum r_{ij}(f_i + f_j)^2$, are already discussed in Lemma 4.4.2. The third summand in $(\star)$ is $-4\cdot\sum r_{ij} f_i^2$. In the non-relaxed objective function this summand vanishes because $-4\cdot\sum r_{ij}\cdot(\pm 1)^2$ is just an additive constant that does not affect the minimizer. However, this third summand drastically worsens the quality of the fractional relaxation because every negative entry in W distorts the

resulting eigenvector through $r_{ij} f_i^2$ in an unwanted way. Numerical experiments show that this kind of straightforward generalization of the unnormalized Laplacian to real-weighted graphs performs bad in practice because of this distortion term.

This raises the question whether we can find an alternative variant $\overline{L}$ of the unnormalized Laplacian, whose Rayleigh quotient characterization finally achieves the "perfect" relaxation for real-weighted W without any additional distortion, namely

$$\min_{\mathbf{f}\in\mathbb{R}^n} \frac{\mathbf{f}^T\overline{L}\mathbf{f}}{\mathbf{f}^T\mathbf{f}} \quad \Leftrightarrow \quad \min_{\mathbf{f}^T\mathbf{f}=n} \sum_{\substack{i,j \\ i\neq j}} a_{ij}(f_i - f_j)^2 + \sum_{\substack{i,j \\ i\neq j}} r_{ij}(f_i + f_j)^2 \quad .$$

Indeed, as proven in the next section, this goal can be achieved by choosing the following modification along the main diagonal of L:

$$\overline{L} \;=\; [\overline{\ell}_{ij}] \;:=\; \begin{cases} d_i^A + d_i^R - a_{ii} - r_{ii} & i = j \\ r_{ij} - a_{ij} & i \neq j \end{cases} \tag{4.6}$$

In particular, $\overline{L}$ fits to the central theme of this thesis of studying variants of Laplacian matrices. I discovered the matrix $\overline{L}$ precisely as sketched here, by studying how to modify the unnormalized Laplacian in order to fit its Rayleigh quotient characterization to the CC functional without any additional distortion terms. It turned out that $\overline{L}$ is the correct definition to achieve this, so in this sense it is the optimal choice for linking some Rayleigh quotient characterization to the CC functional. Further, $\overline{L}$ has several other interesting properties. Later I realized that $\overline{L}$ is already known in the literature as the signed Laplacian matrix (Definition 4.3.3 matches exactly (4.6)). However, its correspondence to the CC functional by focusing on the smallest eigenvector of the AR-decomposition is a novel contribution.

Proofs

The proofs of Lemma 4.4.2, Lemma 4.4.3 and Lemma 4.4.4 apply the following lemma, which shows a general way to constrain the minimization over a generalized Rayleigh quotient without affecting the result.

Lemma 4.5.1 (Freescale)
Let $s : \mathbb{S}^{(n-1)} \to \mathbb{R}_{>0}$ denote any function on the $(n-1)$-dimensional unit sphere $\mathbb{S}^{(n-1)} := \{\mathbf{x} \in \mathbb{R}^n \mid \|\mathbf{x}\| = 1\}$. For all $A, B \in \mathbb{R}^{n\times n}$ with $\mathbf{f}^T B\mathbf{f} \neq 0$ for all $\mathbf{f} \neq \mathbf{0}$ it holds that

$$\min_{\mathbf{f}\neq\mathbf{0}} \frac{\mathbf{f}^T A\mathbf{f}}{\mathbf{f}^T B\mathbf{f}} \quad = \quad \min_{\|\mathbf{f}\|=s(\mathbf{f}/\|\mathbf{f}\|)} \frac{\mathbf{f}^T A\mathbf{f}}{\mathbf{f}^T B\mathbf{f}} \quad .$$

Proof. For all $\mathbf{f} \neq \mathbf{0}$ and all $c \neq 0$ it holds that

$$\frac{(c\mathbf{f})^T A(c\mathbf{f})}{(c\mathbf{f})^T B(c\mathbf{f})} = \frac{c^2}{c^2} \cdot \frac{\mathbf{f}^T A\mathbf{f}}{\mathbf{f}^T B\mathbf{f}} = \frac{\mathbf{f}^T A\mathbf{f}}{\mathbf{f}^T B\mathbf{f}} \quad .$$

Thus, any minimum that is achieved at some $\mathbf{f}^* \in \mathbb{R}^n$ is also achieved at $c\mathbf{f}^*$ for all $c \neq 0$, in particular for c chosen such that $\|c\mathbf{f}^*\| = s(\mathbf{f}^*/\|\mathbf{f}^*\|)$. □

Proof. (of Lemma 4.4.2) First, observe that

$$\min_{\mathbf{f}\neq\mathbf{0}} \frac{\mathbf{f}^T\overline{L}\mathbf{f}}{\mathbf{f}^T\mathbf{f}} \overset{4.5.1}{=} \min_{\|\mathbf{f}\|=\sqrt{n}} \frac{\mathbf{f}^T\overline{L}\mathbf{f}}{\mathbf{f}^T\mathbf{f}} = \min_{\mathbf{f}^T\mathbf{f}=n} \frac{\mathbf{f}^T\overline{L}\mathbf{f}}{n} \Leftrightarrow \min_{\mathbf{f}^T\mathbf{f}=n} \mathbf{f}^T\overline{L}\mathbf{f} \quad .$$

Further, $\mathbf{f}^T\overline{L}\mathbf{f}$ can be expressed with (4.6) as:

$$\begin{aligned}
\mathbf{f}^T\overline{L}\mathbf{f} &= \sum_{i,j} \overline{\ell}_{ij} f_i f_j \\
&= \sum_{\substack{i,j \\ i\neq j}} (r_{ij} - a_{ij}) f_i f_j + \sum_i (-r_{ii} - a_{ii}) f_i^2 + \sum_i (\sum_j a_{ij} + \sum_j r_{ij}) f_i^2 \\
&= \sum_{\substack{i,j \\ i\neq j}} -a_{ij} f_i f_j + \sum_i -a_{ii} f_i^2 + 2 \cdot \frac{1}{2} \sum_{i,j} a_{ij} f_i^2 \\
&\quad + \sum_{\substack{i,j \\ i\neq j}} r_{ij} f_i f_j + \sum_i -r_{ii} f_i^2 + 2 \cdot \frac{1}{2} \sum_{i,j} r_{ij} f_i^2 \\
&\overset{\substack{a_{ij}=a_{ji} \\ r_{ij}=r_{ji}}}{=} \sum_{\substack{i,j \\ i\neq j}} -a_{ij} f_i f_j + \sum_i -a_{ii} f_i^2 + \frac{1}{2} \sum_{i,j} a_{ij} f_i^2 + \frac{1}{2} \sum_{i,j} a_{ij} f_j^2 \\
&\quad + \sum_{\substack{i,j \\ i\neq j}} r_{ij} f_i f_j + \sum_i -r_{ii} f_i^2 + \frac{1}{2} \sum_{i,j} r_{ij} f_i^2 + \frac{1}{2} \sum_{i,j} r_{ij} f_j^2 \\
&= \frac{1}{2} \sum_{\substack{i,j \\ i\neq j}} a_{ij} (f_i^2 - 2 f_i f_j + f_j^2) + \frac{1}{2} \sum_{\substack{i,j \\ i\neq j}} r_{ij} (f_i^2 + 2 f_i f_j + f_j^2) \\
&= \frac{1}{2} \sum_{\substack{i,j \\ i\neq j}} a_{ij} (f_i - f_j)^2 + \frac{1}{2} \sum_{\substack{i,j \\ i\neq j}} r_{ij} (f_i + f_j)^2
\end{aligned}$$

This proves the equivalence at the relaxed side of the relaxation:

$$\min_{\mathbf{f}\in\mathbb{R}^n} \frac{\mathbf{f}^T\overline{L}\mathbf{f}}{\mathbf{f}^T\mathbf{f}} \quad \Leftrightarrow \quad \min_{\mathbf{f}^T\mathbf{f}=n} \sum_{\substack{i,j \\ i\neq j}} a_{ij} (f_i - f_j)^2 + \sum_{\substack{i,j \\ i\neq j}} r_{ij} (f_i + f_j)^2$$

.

For the non-relaxed side, observe that

$$\begin{aligned}
& \min_{\mathbf{f}\in\{-1,1\}^n} \sum_{\substack{i,j\\ i\neq j}} a_{ij}(f_i - f_j)^2 + \sum_{\substack{i,j\\ i\neq j}} r_{ij}(f_i + f_j)^2 \\
\overset{4.3.1(v)}{=} & \min_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} 4\sum_{\substack{i,j\\ i\neq j}} a_{ij}(1 - Y_{ij}) + 4\sum_{\substack{i,j\\ i\neq j}} r_{ij}Y_{ij} \\
\overset{\substack{a_{ij}=a_{ji}\\ r_{ij}=r_{ji}}}{=} & \min_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} 8\sum_{i<j}(r_{ij} - a_{ij})Y_{ij} + 4\sum_{\substack{i,j\\ i\neq j}} a_{ij} \\
\Leftrightarrow & \min_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} \sum_{i<j}(r_{ij} - a_{ij})Y_{ij} \\
\overset{w_{ij}=w_{ji}}{\Leftrightarrow} & \min_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} \sum_{i,j} w_{ij}(\mathbf{1}\mathbf{1}^T - Y_{ij}) \\
= & \min_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} -\sum_{i,j} w_{ij}Y_{ij} + \sum_{i,j} w_{ij} \\
\Leftrightarrow & \max_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} \sum_{i,j} w_{ij}Y_{ij}
\end{aligned}$$

□

Proof. (of Lemma 4.4.3) Similar to above we get that

$$\max_{\mathbf{f}\in\mathbb{R}^n} \frac{\mathbf{f}^T W\mathbf{f}}{\mathbf{f}^T\mathbf{f}} \Leftrightarrow \max_{\mathbf{f}^T\mathbf{f}=n} \sum_{i,j} w_{ij}f_if_j \overset{\leftarrow \text{relax}}{\approx} \max_{\mathbf{f}\in\{-1,1\}^n} \sum_{i,j} w_{ij}f_if_j \overset{4.3.1(v)}{\Leftrightarrow} \max_{\substack{k=1,2\\ Y\in\mathcal{C}_k}} \sum_{i,j} w_{ij}Y_{ij} \quad .$$

□

Proof. (of Lemma 4.4.4) The eigenvalues and eigenvectors of $\overline{\mathcal{L}}_{rw}$ can be related as follows to the eigenvalues and eigenvectors of $\overline{D}^{-1/2}W\overline{D}^{-1/2}$:

$$\begin{aligned}
& \overline{\mathcal{L}}_{rw}\mathbf{f} = \alpha\mathbf{f} \\
\Leftrightarrow\ & \mathbf{f} - \overline{D}^{-1}W\mathbf{f} = \alpha\mathbf{f} \\
\Leftrightarrow\ & \overline{D}^{-1}W\mathbf{f} = (1-\alpha)\mathbf{f} \\
\Leftrightarrow\ & \overline{D}^{-1/2}W\mathbf{f} = (1-\alpha)\overline{D}^{1/2}\mathbf{f} \\
\Leftrightarrow\ & \overline{D}^{-1/2}W\overline{D}^{-1/2}\mathbf{g} = \beta\mathbf{g} \quad ,
\end{aligned}$$

where $\mathbf{g} := \overline{D}^{1/2}\mathbf{f}$ and $\beta := 1 - \alpha$. In particular, the smallest eigenvalue α_{min} (with corresponding eigenvector $\mathbf{f}_{min}$) is determined by the largest eigenvalue $\beta_{max} = 1 - \alpha_{min}$ (and the corresponding eigenvector $\mathbf{g}_{max} = \overline{D}^{1/2}\mathbf{f}_{min}$). Using the Rayleigh quotient characterization, the eigenvalue maximization of $\overline{D}^{-1/2}W\overline{D}^{-1/2}$ can be written as

$$
\begin{aligned}
& \max_{\mathbf{g}\neq\mathbf{0}} \frac{\mathbf{g}^T\overline{D}^{-1/2}W\overline{D}^{-1/2}\mathbf{g}}{\mathbf{g}^T\mathbf{g}} \\
= \quad & \max_{\mathbf{g}\neq\mathbf{0}} \frac{(\overline{D}^{-1/2}\mathbf{g})^T W\overline{D}^{-1/2}\mathbf{g}}{\mathbf{g}^T\mathbf{g}} \\
\overset{\mathbf{f}=\overline{D}^{-1/2}\mathbf{g}}{=} \quad & \max_{\mathbf{f}\neq\mathbf{0}} \frac{\mathbf{f}^T W\mathbf{f}}{\mathbf{f}^T\overline{D}\mathbf{f}} \\
\overset{4.5.1}{\Leftrightarrow} \quad & \max_{\mathbf{f}^T\mathbf{f}=n} \frac{\sum_{i,j} w_{ij} f_i f_j}{\sum_i \bar{d}_i f_i^2} \\
\overset{\leftarrow \text{relax}}{\approx} \quad & \max_{\mathbf{f}\in\{-1,1\}^n} \frac{\sum_{i,j} w_{ij} f_i f_j}{\sum_i \bar{d}_i f_i^2} \\
\overset{4.3.1(v)}{\Leftrightarrow} \quad & \max_{\substack{k=1,2 \\ Y\in\mathcal{C}_k}} \sum_{i,j} w_{ij} Y_{ij} \quad .
\end{aligned}
$$

□

4.5.2 Discussion on the spectral relaxations

All proofs in the previous section reformulate the maximization of the CC functional as the problem of finding an extremal point of an objective function defined on $\{-1, 1\}^n$. The NP-hard optimization problem is then "relaxed" by extending this discrete domain to the continuous domain $\mathcal{S} := \{\mathbf{f} \in \mathbb{R}^n \mid \|\mathbf{f}\| = \sqrt{n}\} = \{\mathbf{f} \in \mathbb{R}^n \mid \mathbf{f}^T\mathbf{f} = n\}$ of all vectors of length $\sqrt{n}$. Geometrically, the set $\{-1, 1\}^n$ contains the 2^n corners of an n-dimensional hypercube of side length 2 that is centered at the origin. The relaxation extends this set of feasible solutions to the sphere of radius $\sqrt{n}$ centered at the origin, which particularly contains all hypercube corners. The optimization over the sphere $\mathcal{S}$ can finally be stated via the Rayleigh quotient characterization as an eigenvalue problem. The eigenvalue problem can be solved efficiently to get a fractional solution $\mathbf{f}^*$ that serves as an approximation of the original discrete problem. From the optimal solution $\mathbf{f}^* \in \mathcal{S}$ of the relaxed problem, we determine its closest discrete solution in $\{-1, 1\}^n$ by setting each entry to -1 or 1, depending on whether it is positive (rounding to 1) or non-positive (rounding to -1).

All other steps along the way from the CC functional to the eigenvalue problem are equivalences and equalities, so it is only the relaxation which transforms the NP-hard problem into an efficient approximation. Since the relaxed domain $\mathcal{S}$ is a superset of the original

domain $\{-1,1\}^n$, the value of an optimal solution of the relaxed problem gives a lower bound on the optimum of the non-relaxed problem. As a consequence, we can derive from the smallest eigenvalue a lower bound on the best-possible value of the CC functional for $k \leq 2$ (after carefully collecting the additive/multiplicative constants that canceled out in the equivalence steps of the above proofs). An upper bound is given by the rounded solution itself. Hence although there is no general result on the approximation quality of this relaxation, one can derive a lower bound and an upper bound on OptMinCut for each graph individually by this spectral approach.

In general, it is possible to obtain one and the same optimization problem as a relaxation of different other optimization problems. Here, we relax the NP-hard problem $\min_{\mathbf{f}\in\{-1,1\}^n} \mathbf{f}^T\overline{L}\mathbf{f}$ of finding the optional MinCut to the eigenvalue problem $\min_{\mathbf{f}\in\mathcal{S}}\ \mathbf{f}^T\overline{L}\mathbf{f}$. Interestingly, Kunegis et al. (2010) take another road to the same eigenvalue problem. They adapt the idea of the RatioCut from similarity graphs to real-weighted graphs, denoted as SignedRatioCut. The problem of finding the minimum SignedRatioCut can be expressed in almost the same way as for the MinCut, precisely as $\min_{\mathbf{u}\in\mathcal{U}}\ \mathbf{u}^T\overline{L}\mathbf{u}$, where $\mathcal{U}$ consists of all vectors $\mathbf{u} = (u_i)$ of the form

$$u_i = \begin{cases} +\frac{1}{2}\left(\sqrt{|V_1|/|V_2|} + \sqrt{|V_2|/|V_1|}\right) & i \in V_1 \\ -\frac{1}{2}\left(\sqrt{|V_1|/|V_2|} + \sqrt{|V_2|/|V_1|}\right) & i \in V_2 \end{cases}, \quad \text{for any bi-partition } V = V_1 \dot\cup V_2. \tag{4.7}$$

The authors suggest to solve the minimization of the SignedRatioCut by a relaxation to the smallest eigenvector of $\overline{L}$, that is by the same eigenvalue problem $\min_{\mathbf{f}\in\mathcal{S}}\ \mathbf{f}^T\overline{L}\mathbf{f}$ as for the optional MinCut. Thus, although OptMinCut the SignedRatioCut are different discrete optimization problems, their relaxation leads to exactly the same eigenvalue problem. This raises the question, which of the two non-relaxed problems is "better" represented by the eigenvalue problem: SignedRatioCut or OptMinCut?

Comparison of OptMinCut to SignedRatioCut. Since there exist no general approximation bounds for the relaxations, I argue informally that the smallest eigenvector of the signed Laplacian focuses stronger on minimizing the unbalanced OptMinCut than on minimizing the balanced SignedRatioCut. Or stated differently, the term "SignedRatioCut" as an adaption of the RatioCut from positive-weighted graphs to real-weighted graphs is misleading because it shows an unintuitive way of balancing. Instead, the SignedRatioCut behaves in almost the same way as OptMinCut, which provides a far simpler intuition.

1. **Explaining 1-clusterings**. The SignedRatioCut is not defined for the trivial clustering $V = V \dot\cup \emptyset$, but only for proper 2-clusterings. This comes from the RatioCut that also enforces a 2-clustering. However, the RatioCut is based on an additional orthogonality constraint that guarantees that the corresponding eigenvector has positive and negative values, thus always giving a proper 2-clustering. Here, there is no such additional constraint, and it happens frequently that all entries of the smallest eigenvector of the signed

Laplacian have the same sign. This case has no representation as a SignedRatioCut, one cannot define a value for the SignedRatioCut of a 1-clustering that is consistent with the values of 2-clusterings. From the viewpoint of the SignedRatioCut, the eigenvector approximation returns occasionally a 1-clustering without explaining why. In contrast to that, the OptMinCut includes this case consistently as the case that all cuts in the graph are non-negative, and thus the 1-clustering is simply a best cut of value zero.

2. **Relaxation accuracy**. From (4.7) we get that every vector $\mathbf{u} \in \mathcal{U}$ is a scaled version of some vector $\mathbf{f} \in \{-1,1\}^n$, precisely $\mathbf{u} = s(\mathbf{f}) \cdot \mathbf{f}$ for the scaling factor $s(\mathbf{f}) = (\sqrt{|\{f_i = 1\}|/|\{f_i = -1\}|} + \sqrt{|\{f_i = -1\}|/|\{f_i = 1\}|})/2$. If the objective function for the SignedRatioCut were $\mathbf{u}^T\overline{L}\mathbf{u}/\mathbf{u}^T\mathbf{u}$ rather than $\mathbf{u}^T\overline{L}\mathbf{u}$, then we would get by Lemma 4.5.1 that the SignedRatioCut is almost equivalent to OptMinCut (up to 1-clusterings). Precisely, for $\mathcal{F} := \{-1,1\}^n \setminus \{-\mathbf{1}, \mathbf{1}\}$ we would get that

$$\min_{\mathbf{u}\in\mathcal{U}} \frac{\mathbf{u}^T\overline{L}\mathbf{u}}{\mathbf{u}^T\mathbf{u}} \overset{4.5.1}{=} \min_{\mathbf{f}\in\mathcal{F}} \frac{\mathbf{f}^T\overline{L}\mathbf{f}}{\mathbf{f}^T\mathbf{f}} = \min_{\mathbf{f}\in\mathcal{F}} \frac{\mathbf{f}^T\overline{L}\mathbf{f}}{n} \Leftrightarrow \min_{\mathbf{f}\in\mathcal{F}} \mathbf{f}^T\overline{L}\mathbf{f} \overset{\text{relax}\rightarrow}{\approx} \min_{\mathbf{f}\in\mathcal{S}} \mathbf{f}^T\overline{L}\mathbf{f} \quad .$$

But since the SignedRatioCut relaxes from $\min_{\mathbf{u}\in\mathcal{U}} \mathbf{u}^T\overline{L}\mathbf{u}$ to $\min_{\mathbf{f}\in\mathcal{S}} \mathbf{f}^T\overline{L}\mathbf{f}$, it introduces an additional distortion because of $\mathbf{u}^T\mathbf{u}$ being non-constant. Precisely, we can see from

$$\min_{\mathbf{u}\in\mathcal{U}} \mathbf{u}^T\overline{L}\mathbf{u} = \min_{\mathbf{f}\in\mathcal{F}} s(\mathbf{f})^2\mathbf{f}^T\overline{L}\mathbf{f} \overset{\text{relax}\rightarrow}{\approx} \min_{\mathbf{f}\in\mathcal{S}} \mathbf{f}^T\overline{L}\mathbf{f}$$

that the relaxation of the SignedRatioCut is two-fold: it extends the domain from discrete points to a continuous domain, and additionally omits the individual lengths of the discrete points. It holds that $s(\mathbf{u}) = 1$ if and only if $|V_1| = |V_2|$, but s increases with stronger imbalances to values above $\sqrt{n}/2$ for $|V_1| = 1$. The relaxation of the SignedRatioCut omits the individual factors $s(\mathbf{f})^2$, which indicates that some deeper properties of the SignedRatioCut are invisible to the eigenvalue problem. In contrast to that, OptMinCut refers for all $\mathbf{f} \in \mathcal{F}$ to the perfectly regular, constant function $s(\mathbf{f}) = 1$.

3. **Intuition**. For positive-weighted graphs it is easy to give examples that demonstrate the strong difference between RatioCut and MinCut. See for example Figure 4.5.1 (a) and let $\alpha = 1$ and $\beta = 2$. The MinCut (if forced to provide a 2-clustering) would cut at α, yielding the unbalanced 2-clustering $(V_3, V_1 \cup V_2)$ of cut weight 1. The RatioCut prefers because of its balance constraints to cut at β, yielding the balanced 2-clustering $(V_2, V_1 \cup V_3)$ of non-minimal cut weight 2. Figure 4.5.1 (a) is also an example in which OptMinCut and SignedRatioCut suggest different solutions. The optional MinCut would not cut the graph at all, which coincides with the shown 1-clustering that is given by the entries of the smallest eigenvector of $\overline{L}$. In contrast to that, for positive-weighted graphs the minimum SignedRatioCut coincides with the minimum RatioCut. Thus the minimum SignedRatioCut is not represented by the smallest eigenvector, but rather by the second-smallest eigenvector that is orthogonal to it.

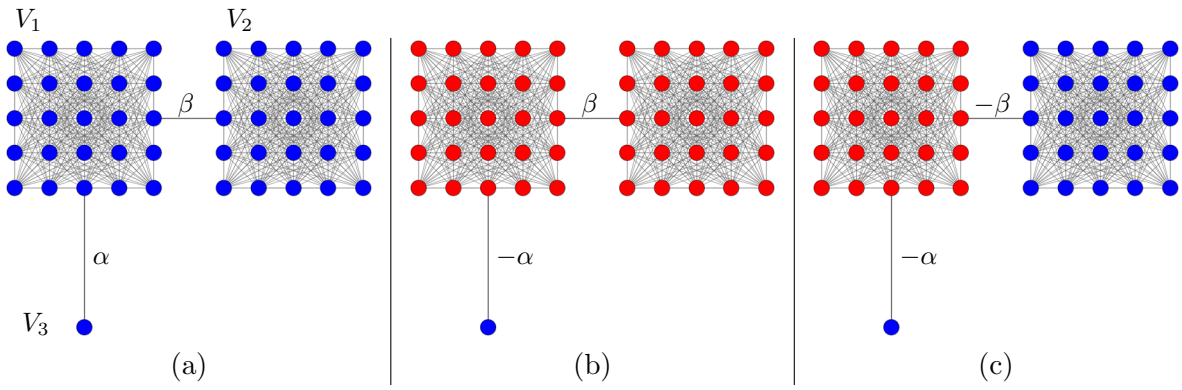


Figure 4.5.1: SignedRatioCut versus OptMinCut. Three different cuts (indicated by red/blue color), as given by the positive/negative entries of the smallest eigenvector of $\overline{L}$ for arbitrary edge weights $\alpha, \beta > 0$. All other edges have unit weight. (a) the graph does not provide a negative cut. OptMinCut agrees with the eigenvector to provide the trivial 1-clustering of cut weight zero. SignedRatioCut disagrees since it prefers the balanced cut $(V_2, V_1 \cup V_3)$, as long as β is not extremely larger than α. (b) the graph provides a negative cut $(V_3, V_1 \cup V_2)$ that is found by the eigenvector. OptMinCut and SignedRatioCut agree with the eigenvector cut. However, SignedRatioCut does no longer respect any balance constraints that would stop it from cutting the single vertex, even for arbitrarily small $\alpha, \beta > 0$. (c) both edges have a negative weight. The eigenvector suggests the cut $(V_1, V_2 \cup V_3)$. OptMinCut and SignedRatioCut agree, even without any balance constraints taking effect.

For positive-weighted graphs the RatioCut penalizes unbalanced cuts in an intuitive way. For real-weighted graphs, this intuition is not well preserved by the SignedRatioCut. The balance effect of the SignedRatioCut is much more subtle, and it extremely depends on whether cuts of positive or negative weights are considered. In Figure 4.5.1 (b) the cut given by the eigenvector coincides with optional MinCut and with the minimum SignedRatioCut. However, while the cut is easy to interpret in terms of the minimum cut, it is hard to grasp why SignedRatioCut gives this extremely unbalanced cut. The benefit of cutting at $-\alpha$ rather than at β can be arbitrary small, that is $\alpha + \beta < \varepsilon$ for any $\varepsilon > 0$. For example, $\alpha = -0.0001$ and $\beta = 0.0001$. Despite the negligible absolute improvement, the SignedRatioCut always favors the extremely unbalanced cut $(V_3, V_1 \cup V_2)$ over the balanced cut $(V_1, V_2 \cup V_3)$. It considers cutting a slightly negative edge as more important than respecting any balance constraints, which contradicts the intuition of a ratio cut.

In Figure 4.5.1 (c), the cut given by the eigenvector coincides again with the optional MinCut and with the SignedRatioCut. The cut is well-balanced. However, this is not because of any balance constraints, but only because of the fact that the unconstrained minimum cut adapts to the true clusters in case of correlation weights.

Summarized, in Figure 4.5.1 (a) the smallest eigenvector agrees with the optional MinCut, but it differs from the minimum SignedRatioCut. In (b) and (c) the eigenvector agrees with OptMinCut and with SignedRatioCut. However, in (b) the SignedRatioCut shows that its bias to balanced clusters is by far not that intuitive as for the RatioCut on positive-weighted graphs. Further (c) shows that the more complex explanation of balancing constraints is not required to explain a balanced clustering result.

If we interpret the smallest eigenvector of the signed Laplacian as an approximation of the SignedRatioCut, then we have to accept that it occasionally suggests 1-clusterings that contradict the minimum SignedRatioCut, and that its balance constraints fail to work under some circumstances. Further its spectral relaxation is more hand-wavy.

If we interpret the smallest eigenvector of the signed Laplacian as an approximation of the unconstrained, optional MinCut, then this interpretation is simple and consistent in all cases, even in the special case of positive-weighted graphs. Further the whole relaxation can be described in terms of a plain geometric intuition: the set of feasible solutions is extended from the discrete set of hypercube corners to the sphere they span.

4.5.3 Comparing SCC to other spectral approaches

This section compares the SCC algorithm (Algorithm 4.4.2) to the spectral approaches introduced in Section 4.3.7, denoted as `AR`, `SL` and `LL`. The SCC algorithm does not require to set any additional parameters. All other spectral approaches require an additional strategy to determine the intended number of clusters k and the number of relevant eigenvectors K (smallest or largest) to consider for the embedding. This is a major drawback of these methods if the number of clusters is unknown. Even worse, in contrast to the case of similarity graphs, where several heuristics for choosing k and K exist, the literature does not yet provide general heuristics for spectral k-way correlation clustering. For that reason we determine the best choice of k and K by a computationally expensive grid-search. Precisely, for all other spectral approaches we apply the following three strategies to a graph that is known to have c clusters:

- `FIXED`. Find $k := c$ clusters from the selected $K := c$ eigenvectors by one of the clustering algorithms below. This choice is suggested by Yu and Shi (2001) for the `AR` algorithm.
- `EQUAL`. For all $K = k \in \{1, \dots, 2\,c\}$ run the clustering algorithm and select from these $2\,c$ alternatives one that provides minimal cut weight.
- `GRID`. For all combinations of $1 \leq K \leq k \leq 2\,c$ run the clustering algorithm and finally select the best combination (in terms of minimal cut weight) among these $2\,c^2 + c$ candidates. Although this brute-force approach is computationally intense, its results are better than those of *any* heuristic for choosing k and K.

For every choice of k and K, we apply the following two clustering algorithms on the embedding of the vertices into $\mathbb{R}^K$ as given by the K selected eigenvectors.

- `SLINK`. Run the single linkage algorithm (Sibson, 1973) to identify up to k clusters.
- `KMEANS`. Run k-means to identify up to k clusters.

In total we compare the SCC algorithm (denoted as `SCC`) to 18 alternatives, namely to all combinations of the following name schema:

$$(\texttt{AR} \,|\, \texttt{SL} \,|\, \texttt{LL}) - (\texttt{SLINK} \mid \texttt{KMEANS}) - (\texttt{FIXED} \mid \texttt{EQUAL} \mid \texttt{GRID}) \tag{4.8}$$

`SCC` refers to the unpruned SCC algorithm. In some of the experiments we further consider a pruned variant by aborting the recursion at depth ℓ for $\ell = 1, 2, 3, 4$ (denoted as `SCC-`ℓ), which bounds the number of clusters to at most 2^ℓ.

Qualitative comparison

Figure 4.5.2 shows that `SCC` can correctly recover strongly imbalanced clusters, while all other spectral alternatives fail. Precisely, the alternatives are `AR`, `SL`, and `LL`, each in the 2 different variants `SLINK` and `KMEANS`. For all 6 alternative approaches, the number of eigenvectors K for the embedding and the number of clusters k are determined by the `GRID` approach, that is by keeping the best result among all combinations of $1 \leq K \leq k \leq 8$. The graph in Figure 4.5.2 (a) is sampled from the Gaussian Correlation Graph model $GCG(\,(64, 64, 64, 64),\, 0.5,\, 1,\, 0.5,\, -1)$. It has four clusters of size 64, and every edge is present with probability 0.5. The edge weights are sampled from $\mathcal{N}(1, 1)$ in case of an intra-edge and from $\mathcal{N}(-1, 1)$ in case of an inter-edge, which causes small errors in 15.9% of the cases (see Figure 4.3.1). The `SCC` algorithm finds the correct clusters. The alternatives find the correct clusters if they rely on `SLINK` for the cluster identification, only `LL` provides an almost correct result even for `KMEANS`.

The graph in Figure 4.5.2 (b) is sampled from $GCG(\,(144, 16, 9, 4),\, 0.5,\, 1,\, 0.5,\, -1)$. The only difference to the above is that now the clusters have strongly imbalanced sizes $144, 16, 9, 4$. The `SCC` algorithm still finds the correct clusters. All alternatives correctly identify the large cluster, but they all fail in identifying the three small clusters. Since the results derive from the `GRID` approach, this fail is not caused by choosing suboptimal parameters k and K. It is rather likely, as also demonstrated in Figure 4.5.3, that the embeddings given by the most relevant eigenvectors do not separate the small clusters well enough, so no clustering algorithm is able to detect the correct clusters by following any embedding-based approach.

Quantitative comparison

This section compares the SCC algorithm to the other spectral approaches quantitatively. The experimental setup is as follows: we create graphs of $n = 3400$ vertices from 7 different classes according to the random graph model $GCG(\,(100, 200, 300, 400, 600, 800, 1000),\, p,\, \mu,\, p,\, -\mu)$ for $p \in \{120/n, 60/n, 30/n\}$ and $\mu \in \{1.5, 1, 0.75\}$. For each of the 9 parameter combinations (p, μ), we sample 10 random instances and apply the following 22 algorithms to every instance: `SCC`, `SCC-2`, `SCC-3`, `SCC-4`, and the 18 alternative spectral approaches from (4.8). We study

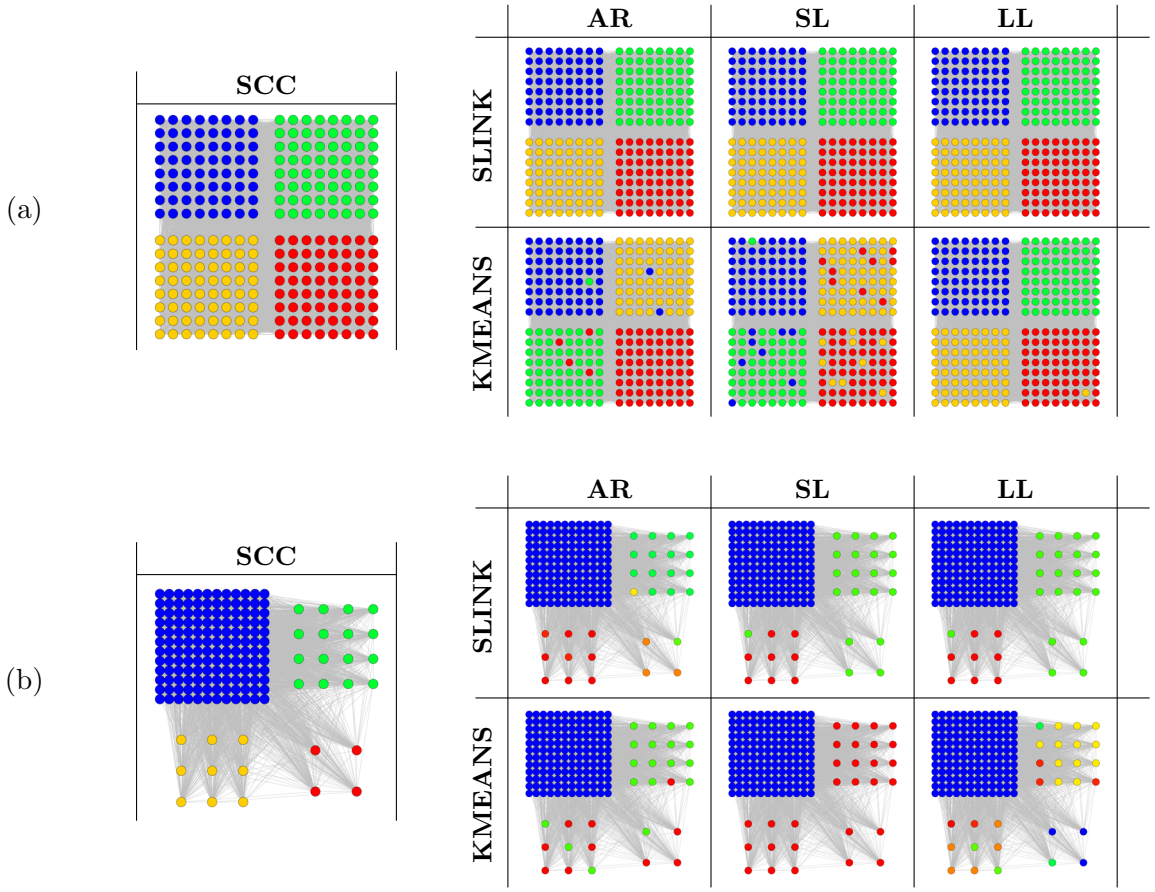


Figure 4.5.2: Balanced versus unbalanced spectral correlation clustering. Clustering results for `SCC` versus 3 spectral clustering alternatives (`AR`, `SL`, `LL`) in 2 different variants (`SLINK`, `KMEANS`). For all 6 alternative approaches, their best result via the `GRID` approach is shown. (a) clustering results for a graph with four equal-sized clusters, sampled from $GCG(\,(64, 64, 64, 64),\, 0.5,\, 1,\, 0.5,\, -1)$. `SCC` finds the correct clusters, the other approaches succeed in case of the `SLINK` approach. (b) clustering results for a graph with strongly imbalanced clusters, sampled from $GCG(\,(144, 16, 9, 4),\, 0.5,\, 1,\, 0.5,\, -1)$. `SCC` still finds the correct clusters, but all other approaches fail in identifying the three small clusters.

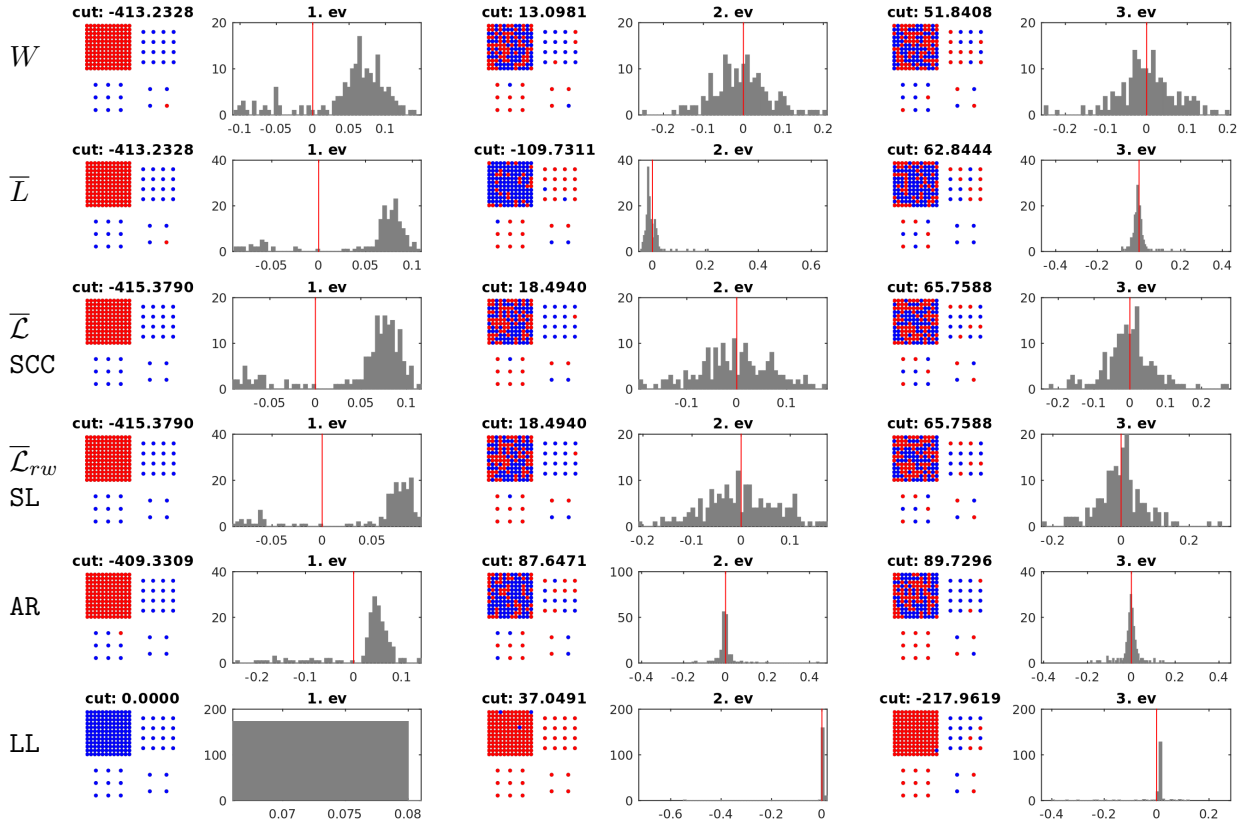


Figure 4.5.3: Eigenvectors of spectral approaches to correlation clustering. Details on the three most relevant eigenvectors for correlation clustering of a graph sampled from $GCG(\,(144, 16, 9, 4),\, 0.1,\, 1,\, 0.1,\, -1)$. From top row to bottom row: largest eigenvectors of the adjacency matrix W, smallest of the signed Laplacian $\overline{L}$, smallest of the signed normalized Laplacian $\overline{\mathcal{L}}$ (as used in `SCC`), smallest of the signed random-walk normalized Laplacian $\mathcal{L}_{rw}$ (equals those of $\overline{\mathcal{L}}$ re-scaled, and same as used in `SL`), largest of `AR`, smallest of `LL`. For each eigenvector, after normalizing to unit norm, a histogram of its entries is shown (horizontal axis ranging from 0.005 quantile to 0.995 quantile). The graph to its left has vertices colored in red/blue (edges not shown) depending on whether the corresponding entry in the eigenvector is above/below 0. `SCC` and `SL` yield a perfect 2-cut with their first eigenvector, `AR` an almost perfect cut. However, instead of continuing recursively (as `SCC`), both `SL` and `AR` consider the noisier higher eigenvectors for their embedding-based clustering strategy. The first eigenvector of `LL` is constant as expected (since the graph has a single component), but also the higher eigenvectors do not provide clearly separable entries for this graph.

the performance metrics NC (number of clusters), PUR (purity), CWD (cut weight difference) and BCF (BCubedF) as defined in Section 4.3.9. Figure 4.5.4 shows the results for NC and PUR, similar does Figure 4.5.5 for CWD and BCF of exactly the same experiments. For all four performance metrics, an individual 3×3 grid shows one box-plot for each of the 9 parameter combinations (p, μ). The top left grid cell refers to the easiest instances (largest p and largest μ). Edge weight noise increases toward the right (decreasing μ), and edge sparsity increases toward the bottom (decreasing p). The bottom right grid cell refers to the hardest instances.

The purity of all 22 algorithms worsens when moving toward harder instances. `SCC` achieves clearly the best purity in all cases, but we see from the NC results that `SCC` also generates the largest number of clusters (even exceeding 20, reaching values between 76 and 203). So the good purity performance of `SCC` could just be an effect of the large number of clusters, rather than being the result of a good clustering. However it turns out that the clustering must be reasonably good: `SCC-2`, `SCC-3` and `SCC-4` apply level-pruning to the cluster-tree of `SCC` in order to provide at most 4, 8 and 16 clusters, respectively. Although they strongly reduce the number of clusters, they reach a similar high purity as unpruned `SCC`. This shows that `SCC-2` and `SCC-3` are superior to all other approaches in most cases because they simultaneously achieve higher purity and a smaller or equal number of clusters. This further shows that `SCC` indeed provides a good-natured over-segmentation that only rarely overlaps the boundaries between true classes. A closer look reveals that several clusters determined by `SCC` consist of a single or few vertices. These mini-clusters derive from a slightly erroneous cut at a higher level in the cluster-tree, which packs some vertices wrongly together with a large group of others that refer to another class. Such vertices are finally separated from the group on a deeper level in the cluster-tree and form a mini-cluster. Such mini-clusters can likely be merged into other clusters that is not a sibling in the cluster-tree. This insight suggests to apply a post-processing strategy that takes the result of `SCC` and then merges its smallest clusters into larger clusters according to some precise criteria (for example, as long as the total cut weight decreases). However, as stated above, our focus lies on comparing the spectral techniques "as is" without applying advanced post-processing strategies to any of them.

For the non-SCC variants, there is no consistent winner among `SLINK` and `KMEANS`. Each outperforms the other in some cases and under different performance aspects. The comparison between `FIXED` ($k = K = 7$), `EQUAL` (best of $1 \leq K = k \leq 14$) and `GRID` (best of $1 \leq K \leq k \leq 14$) is less ambiguous: the `FIXED` approach is clearly outperformed by the others. It always provides the correct number of clusters by definition, but the bad scores of PUR, CWD and BCF show that the identified clusters deviate strongly from the true classes. `SCC-3` demonstrates that it is indeed possible to yield far better clustering results with the same number of clusters (± 1). The `EQUAL` approach performs often slightly better than `FIXED` in terms of purity, but it tends to over-segment by doubling the number of clusters. In all parameter settings, and for each of `SL`, `AR` and `LL`, the best result for PUR, CWD and BCF is always achieved by a `GRID` approach (either `SLINK` or `KMEANS`). Although `GRID` also tends to

over-segment, the clustering result seems to respect class boundaries well, as indicated by the good BCF and PUR scores. The GRID approach selects the best out of the 105 combinations of $1 \leq K \leq k \leq 14$. The optimal combination of K and k varies strongly over the instances, but in many cases K is significantly lower than k. In some cases even $K = 1$ and $k \approx 7$ is selected, which means for SL that the whole k-clustering result is found by running k-means on the entries of the single smallest eigenvector. Further note that the alternative spectral approaches do not benefit from allowing for much higher K and k than $2\,c$. Extending the parameter search grid, for example to $1 \leq K \leq k \leq 28$, does not further improve Purity, CWD or BCubedF, but only increases median and variance of the number of clusters, and the computation time by another factor of four.

Figure 4.5.5 shows that all variants of SCC achieve significantly smaller cut weights than all other spectral approaches. This demonstrates the focus of SCC on minimizing the cut. The other approaches seem to respect additional constraints, such as balance constraints, that prevent them from reaching smaller cut weights. This does not imply that they perform worse: it is not guaranteed that a smaller cut weight always achieves better clustering results. However, also in terms of BCubedF, which scores a combination of precision (like purity) and recall between the clustering results and the true classes, SCC and its pruned variants yield in most cases higher scores than all other spectral approaches. This gives evidence that the CC functional is a good optimization criterion for correlation clustering, and that SCC performs well in maximizing it, without requiring any additional balance constraints.

Another benefit of SCC in the current implementation is that computing the cluster-tree is much more efficient (even if unpruned) than the GRID approach. While SCC completes in 1 second on average, the GRID approach with KMEANS takes more than 1 minute. However, this is not a rigorous performance evaluation, since both approaches can further be optimized or modified. For example, both approaches could compute their subproblems in parallel, and one could also choose an *adaptive* grid approach on $1 \leq K \leq k \leq n$ rather than GRID.

These experiments show that SCC is an interesting candidate for correlation clustering, in particular on sparse, noisy and imbalanced instances that are in the focus of the clustering application in Section 4.6. SCC performs better than all embedding approaches that cluster n embedded points in $\mathbb{R}^K$ into k clusters for any K and k.

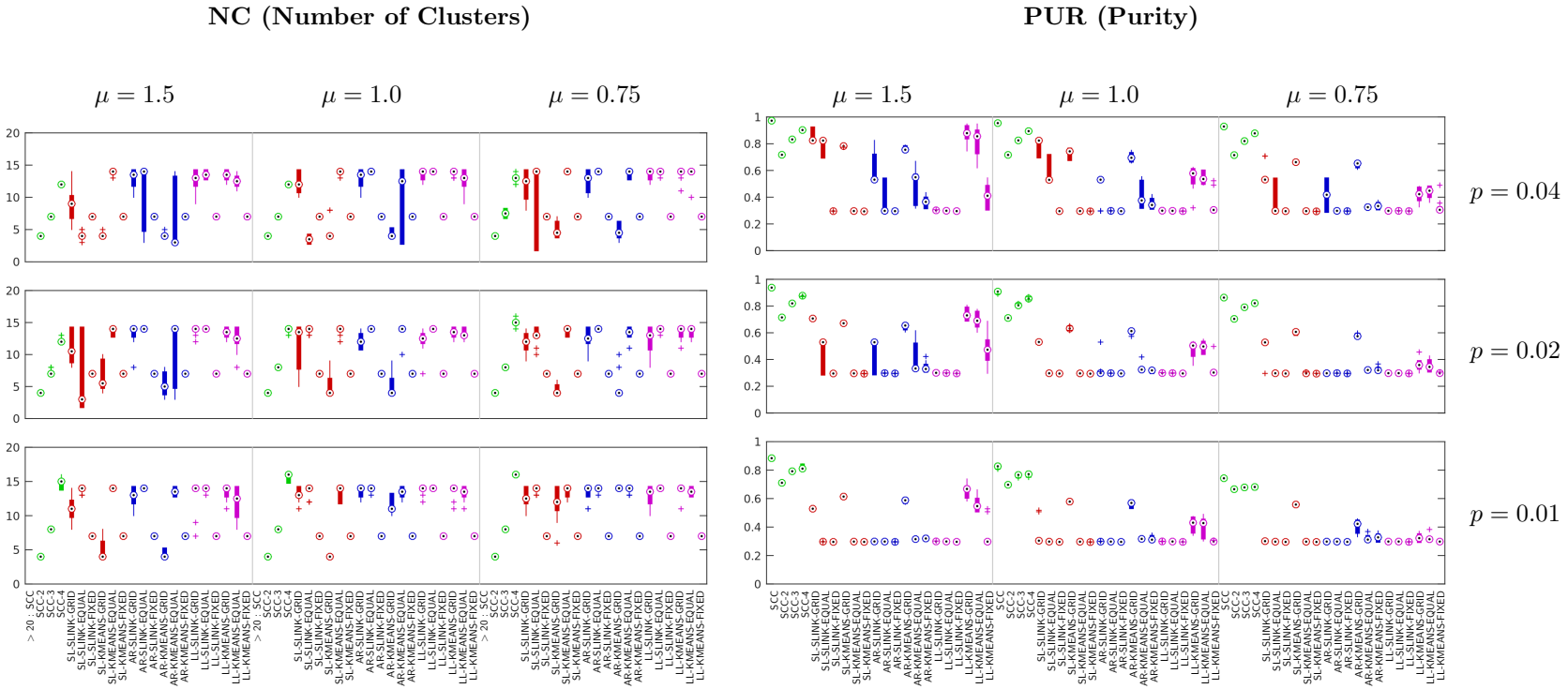


Figure 4.5.4: SCC versus other spectral approaches by number of clusters and purity. Boxes and whiskers are determined as follows from the 10 independent repetitions of each individual experiment: the median is plotted as a dot, the solid box marks the quartile range (from 0.25 to 0.75 quantile), a line extends to the farthest value that is not considered as an outlier (at most 1.5 times the length of the quartile range), and every outlier is marked by a cross. The left plot shows the number of clusters (the closer to 7 the better) as returned by the respective algorithm. The right plot shows the purity of the clustering (the larger the better). `SCC` has an excellent purity, but yields more than 20 clusters in all experiments. In most cases, `SCC-2` and `SCC-3` still outperform all other approaches in terms of purity while having the same or a fewer number of clusters.

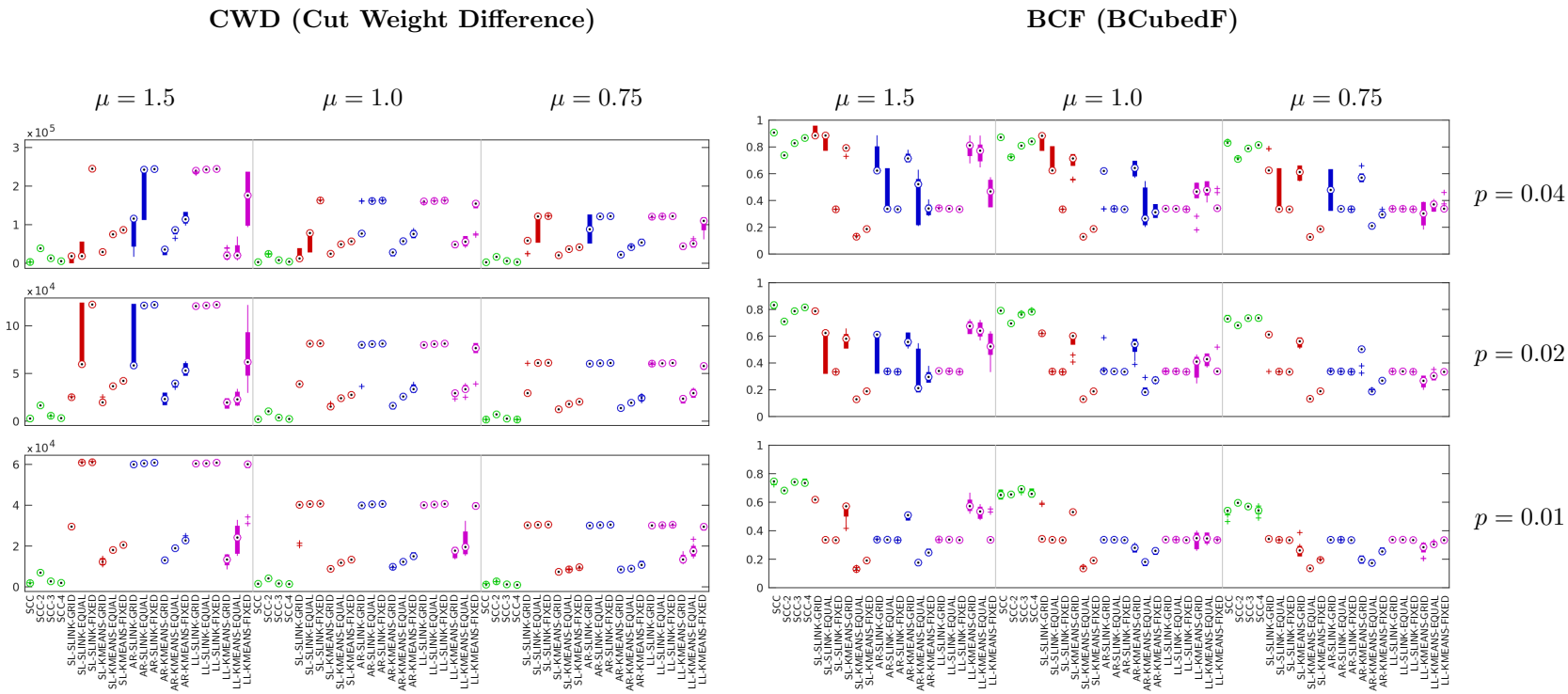


Figure 4.5.5: SCC versus other spectral approaches by cut weight and BCubedF. Boxes and whiskers are defined as in Figure 4.5.4. The left plot shows the difference of the cut weight that is achieved by the algorithm's clustering and the cut weight of the respective true class partition (the smaller the better). The right plot shows the BCubedF performance metric that quantifies the quality of the overlap of the algorithm's clustering with the true class partition (the larger the better). In almost all cases, the `SCC` variants achieve significant smaller cut weights and larger BCubedF scores than all other spectral approaches.

4.5.4 Comparing SCC to state-of-the-art

In this section, we compare SCC to the non-spectral approaches Expand, Swap and AICM of Bagon and Galun (2011), as introduced in Section 4.3.4. Expand and Swap are computationally intense and do not scale well beyond a couple of hundreds of vertices on standard computer hardware. AICM is as fast as SCC, so we can compare these approaches also on larger instances. In all experiments, we also present results for six alternative spectral approaches, namely for all six combinations of (SL | AR | LL) − (SLINK | KMEANS) − GRID.

Quantitative comparison

This experiment studies graphs from the model $GCG(\,(100, 100, 100, 100), p, \mu, p, -\mu)$ for every combination of parameters $p \in \{0.4, 0.2, 0.1\}$ and $\mu \in \{1.0, 0.8, 0.6\}$. Similar to Section 4.5.3, we compare SCC, SCC-2 and SCC-3 to the 9 alternatives in terms of the performance measures NC, PUR, CWD and BCF. Figure 4.5.6 shows the results of these experiments. On relatively dense and low-noise instances (large p and μ), some approaches achieve perfect results (Expand, AICM and all SLINK variants). The approaches Swap, SCC and SCC-3 find too many clusters, but since their purity is 1.0, the correct classes could be recovered exactly by merging appropriate clusters. As in the previous section, this suggests to apply a post-processing strategy that merges mini-clusters into larger clusters as long as the total cut weight shrinks. Including such a post-processing, Swap, SCC and SCC-3 would also achieve perfect results on these instances. The other approaches (SCC-2 and all KMEANS variants) detect the correct number of clusters, but they wrongly group some vertices from different classes together (as indicated by the non-optimal purity).

On sparser and noisier instances, SCC and AICM show a strong bias toward over-segmentation. However, both approaches still achieve high purity, which indicates that a sufficient post-processing strategy can mostly correct this over-segmentation. In particular, SCC-3 demonstrates that one can indeed reach a similar high purity by a small number of clusters, even simply by level-pruning rather than by an advanced post-processing strategy. Expand and Swap do much slower tend to over-segment, although they are also unbounded in the number of clusters to find, too. In contrast to the approaches SCC, Expand, Swap and AICM, all other approaches are restricted by definition to at most 8 or 4 clusters, respectively, so they cannot over-segment too strongly.

The computation time of Expand lies between 10 and 120 seconds, Swap between 2 and 10 seconds, and all other approaches far below 1 second.

Summarized, one can rank the algorithms by their clustering performance as follows: Expand and Swap share the first place, where Swap is slightly worse than Expand, but faster by an order of magnitude. However, none of the two approaches is able to scale-up to thousands of vertices. The second place is shared by SCC and AICM. They both tend to over-segment in a good-natured way, which can mostly be corrected by an additional post-processing strategy. While AICM is slightly better than SCC on dense graphs, SCC outperforms AICM on sparse instances, which particularly reflects in a far better BCubedF score. So as before, the

Figure 4.5.6: SCC versus state-of-the-art, quantitatively. Average statistics over 10 graphs sampled from the model $GCG(\,(100, 100, 100, 100), p, \mu, p, -\mu)$ for each parameter combination (p, μ) in terms of the number of clusters (the closer to 4 the better), purity (the larger the better), cut weight difference to the true cut weight (the smaller the better), and the BCubedF metric (the larger the better). Boxes and whiskers are defined as in Figure 4.5.5. Overall, `Expand` and `Swap` show the best cluster quality, followed by `SCC` and `AICM`, outperforming all other spectral alternatives.

performance of SCC is relatively robust against sparsity and noise. The third place is taken by all other spectral approaches, which are clearly outperformed on all harder instances, even though they base on the respective best-possible GRID result.

Qualitative comparison

In this section we compare SCC to AICM, SL-SLINK-GRID and SL-KMEANS-GRID on graphs of size $n = 3400$, for which it is no longer feasible to compute Expand or Swap. Precisely, we sample a graph from the model $GCG(\,(100, 200, 300, 400, 600, 800, 1000),\, p,\, \mu,\, p,\, -\mu)$ for the parameter combinations $(p, \mu) \in \{(2.0, 0.1), (1.4, 0.08), (1.0, 0.1), (0.8, 0.8)\}$ and apply all four clustering algorithms to the instance. Figure 4.5.7 plots the resulting cluster-matrices $Y \in \{0, 1\}^{3400\times3400}$ as a black-and-white image. Pixel (i, j) is black if and only if vertices i and j are assigned by the algorithm to the same cluster (i.e., $Y_{ij} = 1$). Because of the meaningful ordering of the generated vertices, the correct clustering corresponds to a block-diagonal form of 7 blocks. For the easiest instance (top row), AICM and SL-SLINK recover the correct clustering exactly. SCC over-segments the smallest cluster, but the correct clustering could be recovered by a sufficient post-processing strategy. SL-KMEANS is not able to distinguish the smallest four clusters well from each other. With increasing sparsity and noise (toward bottom), each approach shows an individual behavior on how it looses the ability to recover the true classes. SL-KMEANS always finds some clusters that represent true classes at least roughly, while making an increasing amount of errors across the board. SL-SLINK separates supersets of true classes from each other sharply, but it tends to put all vertices into a single large cluster, plus keeping a few singleton-clusters. Both these embedding-based approaches perform bad in recovering the small clusters. AICM looses the ability to detect clusters in increasing order of their size, while it reveals larger clusters quite reliably. This behavior is similar to that of SCC, but while SCC over-segments the smaller clusters earlier than AICM does, SCC is able to detect fuzzier clusters more robustly than AICM. For $p = 0.01$, SCC is still able to partially detect 3 classes with reasonable accuracy, while AICM cannot find more than the single largest class.

In accordance with the previous section, this experiment shows that SCC and AICM perform similarly, with SCC having a slight advantage on sparse and noisy instances. This makes SCC particularly interesting for the exploratory analysis, where SCC might point to fuzzy structures that are invisible to other approaches.

4.6 Application: Anti-zerging clustering software

This section presents the main application that motivated my research on correlation clustering. It provides several details on the software architecture (Section 4.6.1) and on the algorithmic superstructure (Section 4.6.2) that is required to finally apply the correlation clustering

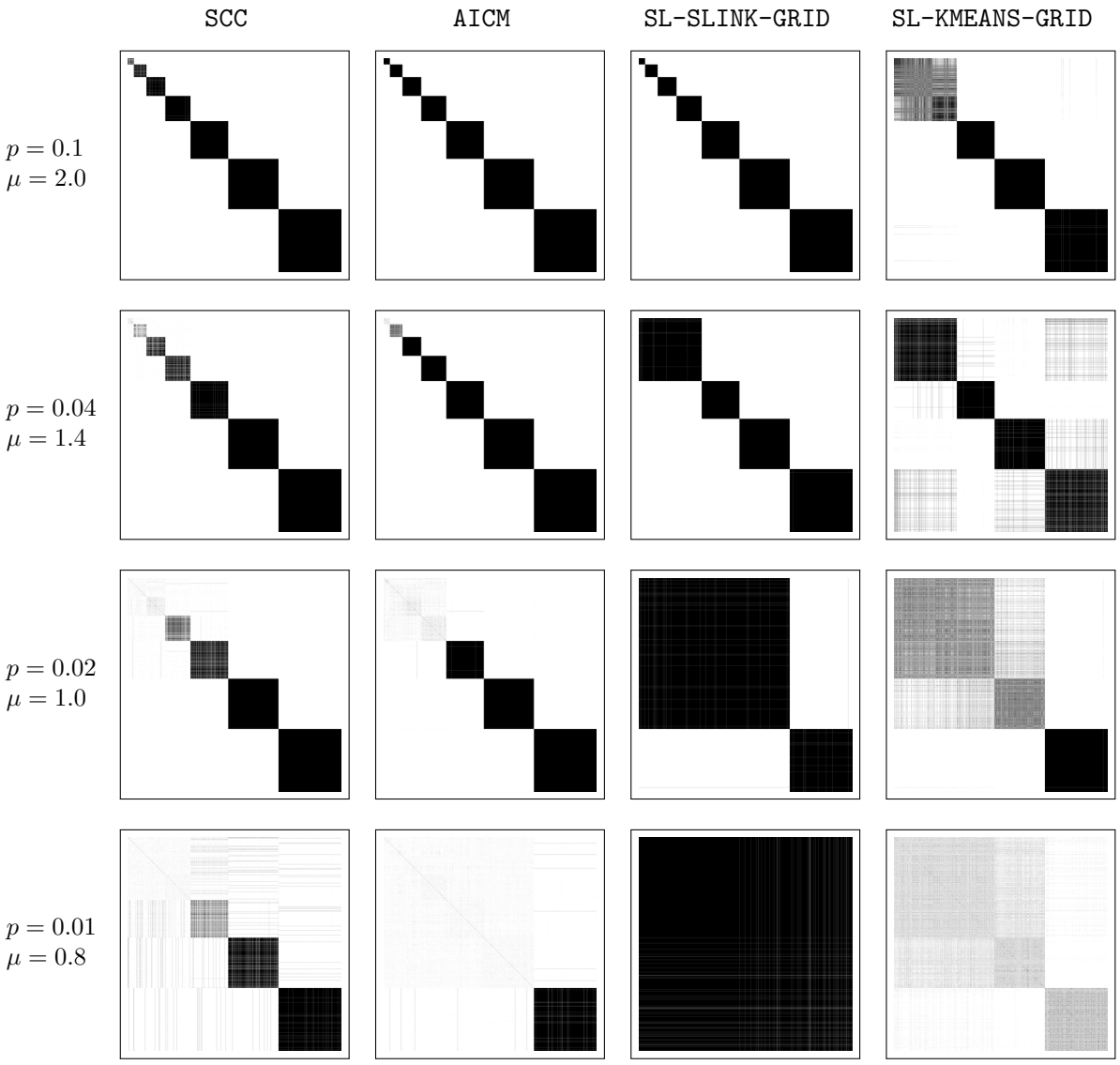


Figure 4.5.7: SCC versus state-of-the-art by exemplary cluster-matrices. Each row corresponds to a graph that is sampled from the model $GCG(\,(100, 200, 300, 400, 600, 800, 1000),\, p,\, \mu,\, p,\, -\mu)$ for the respective parameters (p, μ). Each column shows the cluster matrices as being computed by the respective algorithm. The cluster matrix is a 3400×3400 matrix with entry (i, j) colored in black if and only if the corresponding vertices i and j are grouped by the algorithm into the same cluster. The perfect block-diagonal form vanishes for sparser and noisier instances.

algorithm to individual combats. This section ends with a performance evaluation of the SCC algorithm in this context (Section 4.6.3).

4.6.1 Overview on the software architecture

The architecture of the Anti-Zerging Clustering Software follows the structure of how the virtual world is partitioned into smaller subparts: the full world map is divided into ca. 300 individual **regions**. Each region covers a virtual area of a few square kilometers. Every active player is located at a position in exactly one region. A player can potentially interact with all other players in the same region, but no interaction is possible across region boundaries. Players can move from one region to a neighbored region by crossing their common boundary. The regions are technically supplied by a few dozens of dedicated game servers. Each game server is responsible for computing the game physics of a number of regions plus the communication to all user client machines that currently act in one of the regions.

In order to avoid that players who received a debuff can get rid of it by just moving to another region, we have to keep track of all information on a player even if it moves to another region. A centralized approach for storing player information across regions is to maintain a global **world graph**: the vertices of this graph are all active players in all regions, and edges hold aggregated information on all recent interactions between two players. The world graph is dynamically updated by all current activities in all regions. Further, it slowly forgets about old activities and removes an edge if no recent interactions exist. This yields a sparse graph: there might be up to 10000 players simultaneously playing across all regions, but each player is connected to only, say, $5 - 200$ other players. The world graph is at the heart of the clustering software.

The overall structure of the Anti-Zerging Clustering Software and its environment is visualized in Figure 4.6.1, which shows three of the following four main components.

- **Game server**. Beside its other tasks, each game server collects information on all activities in its regions, and aggregates them over a time window of a few seconds into a **region snapshot**. Each snapshot gets a server timestamp attached and is then transmitted over TCP to the clustering server, serialized as a JSON string. Further, each game server expects to repeatedly receive an information update on the most recent debuff scores of all its players. The debuff scores affect the game physics in an appropriate way.

- **Clustering server**. The clustering server is implemented in Java. It consists of three components that run as separate threads.
  - **Server agent.** Manages the connections to all game servers, based on the server socket of the the standard `java.nio` package, which allows to deal asynchronously with a large number of incoming connections of TCP clients. An efficient de-/serialization between the JSON format and internal Java objects is provided

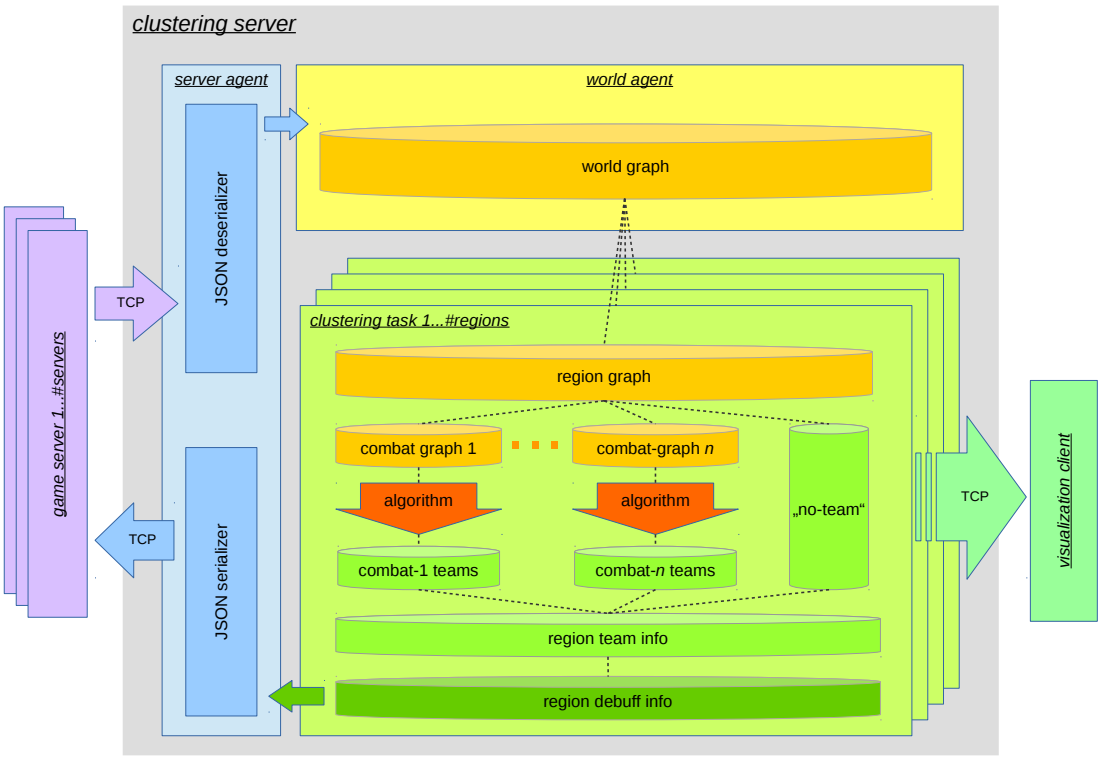


Figure 4.6.1: Environment and overall structure of the clustering software. The environment consists of the game servers, the clustering server, and the visualization client. Not shown: the test simulator that can replace the game servers in order to replay (and modify) previously recorded real data streams. See the text for details on the clustering server.

by Google's GSON-library[2], which outperformed other tested alternatives.

  - **World agent.** Maintains the world graph, which holds present and past information on all interactions of active players. Further, the world agent repeatedly prepares the next clustering phase by partitioning the world graph into separate region graphs (see the details in the following section) and finally starts a thread pool with one clustering task per region.
  - **Clustering task.** Executes the clustering process for a single region independently of all others. The clustering process starts with a heuristic that identifies all individual combats in the region. Each player is assigned to one of the combats, or to the "no-team" if it is not involved in any combat. Teams are identified separately for each combat by executing a correlation clustering algorithm on the combat graph. The resulting team information is integrated into the update strategy for the debuff scores. The updated debuff scores are finally sent to the server agent in order to forward them to the appropriate game servers.

- **Visualization client**. This component is a separate windowed application that can remotely connect via TCP to a running clustering server in order to display real-time information about the world and the clustering processes. It enables a live-visualization of every region graph (including a navigable history per region), plus detailed text output for statistical information on the clustering results. See Figure 4.6.2 for an example. Technically, this TCP-connection uses the Kryo-library[3] at both ends, which turned out to be superior over a couple of tested alternatives in terms of de-/serialization speed of Java objects as well as in simplicity of use.

- **Test simulator**. This component simulates one or more game servers for testing purposes without requiring any real game server. It has access to real data sets that were recorded during a session with 30 volunteers who played through ten different variants of PVP combats. These data feeds can be modified before transmission in order to allow for more variety. For example, test scenarios can be duplicated and translated, and the cheating strategies "friendly fire" and "unfriendly health" can be injected at configurable rates. For all combats in this test data, the ground truth of the correct team association is accessible, which allows to study the performance of the clustering algorithm quantitatively. If data with ground truth is sent to the clustering software, then the visualization shows extra information that is not available otherwise, such as the confusion matrix, classification performance, etc.

Each region can have an individual configuration for its clustering process, or it falls back to a default configuration. All parameters can be re-configured live without downtime by

[2] `https://github.com/google/gson`
[3] `https://github.com/EsotericSoftware/kryo`

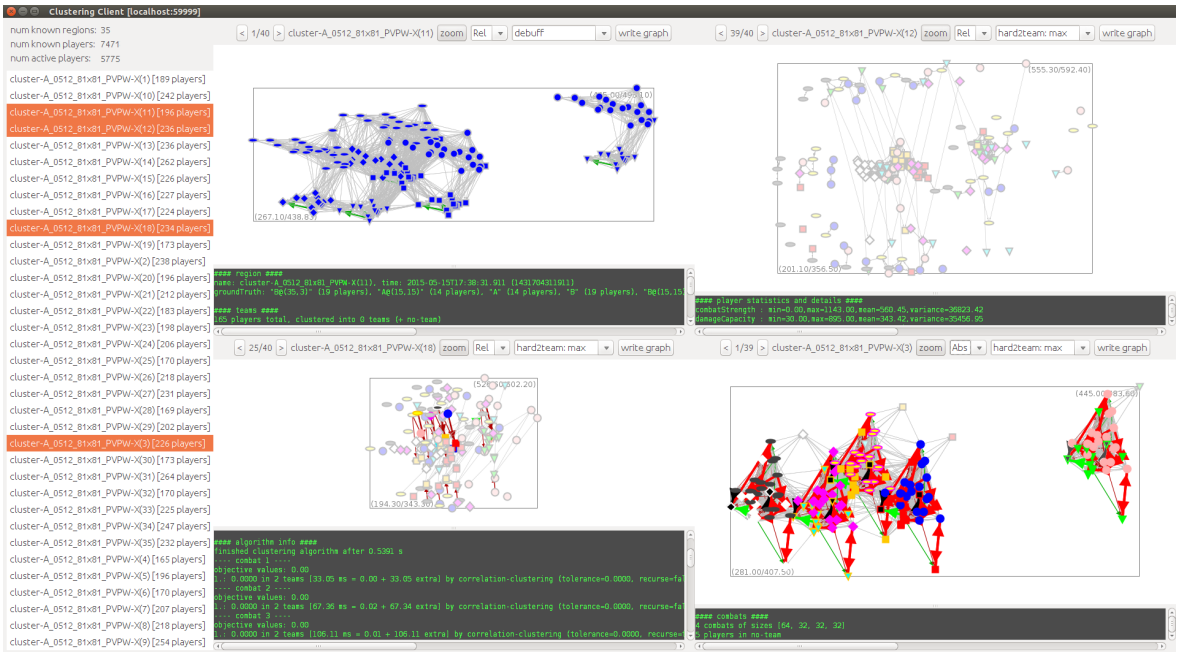


Figure 4.6.2: Screenshot of the visualization client. The left panel lists all regions that are currently known at the clustering server to which the GUI is connected. The user can select up to 16 regions and is henceforth notified by the server about all related updates. The right panel shows details for all selected regions: a plot of the region graph (including configurable color-encoded information) plus a text box that provides several details on the region's state and the clustering process. The plots are updated live as soon as new region information comes in. Alternatively, the user can browse back and forth in history in order to manually inspect details or to export the data in various formats.

modifying the configuration files (using Java's `WatchService` to get notified on file changes). A modified configuration is automatically reloaded and injected into the running system.

4.6.2 From the world graph to updated debuff scores

This section provides details on the construction of the world graph and on the chain of steps that are required to finally derive updated debuff scores from it.

The world graph

The world graph is a dynamic directed graph with properties attached to its vertices and edges. Since the graph changes over time, it can be modeled as a sequence of static graphs $G_t = (V_t, E_t)$ for some time parameter t. Every vertex corresponds to a player that is currently acting or that became inactive recently (the user just exited the game, leaving behind recent player interactions that still continue to have an effect in the world graph). Together with each vertex, information on the corresponding player is stored, for example:

- **region**: the region in which the player currently resides
- **location**: the location in the region at which the player was placed at the end of the latest region snapshot
- **health points**: the number of remaining health points of this player (a player dies if its health points fall below 0)
- **guild**: the guild, if any, to which this player belongs
- **strength**: a parameter that quantifies the overall strength of this player, that is its potential power depending on its equipment and magic capabilities

A directed edge from player i to j exists if and only if player i recently acted toward player j, for example by a weapon attack, a magic spell, or just by staying close to j. Together with each edge, additional information on all recent interactions of this oriented pair (i, j) is stored, in particular a timestamp of the last edge update together with several similarity scores and dissimilarity scores. All dis-/similarity scores take non-negative values. Each such dis-/similarity score is aggregated over the past in a way that cumulates repeated activities, but that also damps outdated activities over time back to zero. If eventually all dis-/similarity scores for the edge (i, j) fall below a certain threshold, then this edge is removed from the world graph. The following example shows such an aggregated update of a dissimilarity score.

Example 4.6.1 (Aggregation of absolute damage points)
Let $t(i, j)$ denote the timestamp (in seconds) of the latest update of edge (i, j), and $d_{abs}(i, j)$ the dissimilarity score of the aggregated absolute damage points caused by player i on player j. If a region snapshot at time t tells that player i caused 50 additional damage points on player j, then the new score $d'_{abs}(i, j)$ is computed as $d'_{abs}(i, j) := 0.9^{t-t(i,j)} \cdot d_{abs}(i, j) + 50$. That is, the old score decays exponentially at the rate -0.1 before the current damage points are added.

Note that Example 4.6.1 gives just a simplified impression of the overall world graph update mechanism, as carried out by the world agent. The world agent particularly implements strategies to handle various race conditions:

- region snapshots can be received out of time-order from different game servers due to irregular transmission latencies
- region snapshots are taken by each game server independently of the others at individual time intervals that are not in sync with the other servers
- whenever a player moves from one region to another, then it shows up in two time-overlapping snapshots of different regions

The following list shows some of the information that is stored along with every edge (i, j):

- **timestamp**: the time of the latest update of this edge, that is $t(i, j)$ in Example 4.6.1
- **absolute damage**: aggregated score of absolute damage points caused by player i on j, that is $d_{abs}(i, j)$ in Example 4.6.1. Similar aggregated scores are defined separately for healing points donated by player i to j as well as for other kinds of benevolent and malevolent magic spells. All these scores act on roughly the same scale, so that 10 adversarial (dissimilarity) points outweigh 10 friendly (similarity) points.
- **relative damage**: rather than adding the absolute damage points, this aggregation cumulates the damage points divided by the remaining health points of player j. Similar relative scores are defined for the other similarities and dissimilarities. Relative scores treat both adversarial and friendly activities more serious if the (resulting or preceding) number of health points is close to zero, while the activity is less relevant if the number of health points is large. This further limits the options of cheaters, since cheating activities have less impact as long as they are not *essentially* harmful or helpful.
- **friendly close time**: a similarity score that is updated in a very different way than the other scores. The basic idea is to increase this score over time if players i and j stay close to each other without performing any combat operations. Thus, high scores means that they acted peaceful next to each other, although no environmental reasons preclude them from attacking each other. This value decreases at individual speeds if the players are far away from each other, or if one attacks the other. The corresponding rates of increase and decrease depend further on the distance between the players (very close, close, medium, far), resulting in several individual parameters to tune.

All the aggregated scores that are stored at the world graph's edges can be combined in several different ways into a single correlation score c_{ij} for that edge, for example via

$$c_{ij} \;:=\; \alpha_h \cdot \texttt{heal}(i,j) - \alpha_d \cdot \texttt{damage}(i,j) + \alpha_f \cdot \texttt{friendlyCloseTime}(i,j) \tag{4.9}$$

for some positive scaling parameters $\alpha_h, \alpha_d, \alpha_f$. This shows that we can think of the world graph as a correlation graph. High positive scores $c_{ij} \gg 0$ indicate high confidence that both players belong to the same team, low negative scores $c_{ij} \ll 0$ give evidence that players i and j are adversaries, and scores close to zero reflect uncertainty. In particular, cheaters can provoke erroneous negative scores via friendly fire and erroneous positive scores via unfriendly help. This suggests to determine the unknown number of participating teams by maximizing the corresponding CC functional. Here it is a particularly useful property of the CC functional that the case of non-symmetric scores $c_{ij} \neq c_{ji}$ can be reduced to the case of symmetric scores, without affecting the result, by considering the symmetric weights $w_{ij} := w_{ji} := c_{ij} + c_{ji}$ for the correlation clustering algorithm.

From the world graph to region graphs

The world graph usually contains several independent PVP combats at a time, even multiple independent combats per region. Our goal is to identify the effective teams within each combat individually, so we first have to define a way to separate one combat from another combat. Since no interaction is possible across region boundaries, it is reasonable to define that one and the same combat can only include players that stay in the same region. Hence, if some players of an ongoing combat in region A move to another region B and continue fighting each other, then they will be treated as a separate combat. This is reasonable because the balance of power in these two regions can be very different. Moreover, all historical information between players that currently reside in different regions is still kept in the world graph. So as as soon as other players move between regions A and B, all their previous associations will immediately have an effect on the clustering.

Treating regions separately allows for a convenient intermediate step toward identifying individual combat graphs, as carried out by the world agent: the world graph is subdivided into **region graphs**, that is the subgraph of the world graph that is induced by all players that currently stay in the same region. This splitting particularly holds back any region-crossing edges from the further processing. There is one more step involved in defining a region graph: since all edges still refer to different individual time stamps, all information in a region graph needs to be extrapolated (damped) to the latest timestamp for which a snapshot from this region was received. After this extrapolation, all information in a region graph refers to the same time, although different regions may refer to different times.

Finally, the world agent enqueues one new clustering task per region to a processing queue. The processing queue is worked through by a thread pool of multiple threads, which allows to exploit multi-core CPU-power. In parallel to the execution of the clustering tasks, the world agent continues updating the original world graph from the newest incoming region snapshots. When all tasks in the processing queue are finished, the world agent initiates the next clustering phase as soon as possible (after a minimum total delay of at least one second from the beginning of the previous clustering phase).

From a region graph to combat graphs

For a human observer it is in most cases easy to see where in a region a PVP combat happens, and which players participate. However, algorithms need precise instructions that are applied to *all* cases. This requires to think carefully about the boundary cases, where it is even for a human no longer easy to see whether there is a single combat, or multiple combats close to each other, and which player precisely belongs to which combat. There is no canonical solution to make this sharp definition of a PVP combat precise. After carefully considering its consequences, I developed the following two-step strategy: first, the region graph is subdivided into **sub-combats** and the **no-team**. the second step is to merge sub-combats if they likely belong to a single combat.

A sub-combat is defined according to the following informal idea:

> **“ A player belongs to a sub-combat, if it is directly or indirectly associated by recent activities to any adversarial activity. ”**

The following strategy makes this idea precise:

1. an edge (i, j) is denoted as *active* if the timestamp of its last harmful or helpful activity lies back no longer than 60 seconds

2. consider the subgraph of the region graph that consists only of active edges, and find all its *weak* connected components (that is, connectivity ignores the direction of all interactions)

3. every component that contains *any* adversarial edge is defined as a sub-combat, and consists of at least two players. All other components consist of players that are not associated to any sub-combat, so they are added to the no-team.

This strategy assigns actively fighting players to a sub-combat as well as magicians that only focus on healing their team members (even if they operate just indirectly by helping others that help others). This strategy further implies that players can become part of a sub-combat against their wish by being attacked.

Particularly at the beginning of a larger combat, when harmful and helpful activities do not form a single weak graph component yet, it happens that a single large combat is subdivided into multiple sub-combats. However, in this case there are usually many non-active edges (for example due to friendly close time similarities or from outdated actions) between several players from different sub-combats. For that reason, the second part of the strategy iteratively merges two sub-combats into a single one, whenever they are connected by several non-active edges (relatively to the size of the smaller sub-combat). This merging process ends up with a number of combats, each consisting of one or more sub-combats. The individual combats in a region show a lot of internal interaction, but no or almost no interaction in between. Each combat defines its **combat graph**, that is the subgraph of the region graph that is induced by all players of the combat. All combat graphs are further processed one after the other, independently of all others.

Note that the problem of subdividing the region graph into combats and the no-team is similar to the general problem of identifying clusters (combats) on a noise floor (no-team). However, in contrast to some generic clustering algorithm, the hand-crafted clustering strategy presented here allows for a far better control of the precise details. It turns out that this specific problem is simple enough to be rigorously solved by explicit and intuitive connectivity rules.

From combat graphs to team assignments

Identifying the teams within a combat is much more difficult than identifying combats as a whole in a region. The first step is to transform the given combat graph into an undirected

simple graph $G = (V, E, W)$ with correlation weights on its edges. This is achieved as described in the text following Equation (4.9) on page 193, although in practice we consider some more edge parameters than shown there. Further, we may choose either the absolute or the relative dis-/similarity scores for defining c_{ij}. A comparison of these two approaches showed that the relative scores perform better than the absolute scores.

Once the graph G is defined, we can apply any correlation clustering algorithm in order to partition the vertices into $V = V_1 \cup \ldots \cup V_k$ for some unspecified number of clusters k. This k-clustering assigns the players of this combat to an unspecified number of teams.

The clustering algorithm that is used in the later experiments is the SCC algorithm in two variants: **unpruned**, that is the original SCC algorithm without any pruning strategy, and a **(satisfaction-)pruned** variant, which stops the further subdivision within the `ClusterTree` algorithm whenever all players of the current vertex subset are "satisfied" with being grouped together. This approach bases on the following definition of the level of satisfaction of player i about its assignment to team T.

Definition 4.6.2 (Satisfaction)
For a correlation graph $G = (V, E, W)$, any team $T \subseteq V$, and every player $i \in T$, the satisfaction $s_i(T)$ of player i about staying in team T is defined as

$$s_i(T) \;:=\; \sum_{j \in T} w_{ij} - \sum_{j \notin T} w_{ij} \quad .$$

The satisfaction about staying in team T increases with strong team-internal similarities and with strong dissimilarities to players in other teams. Satisfaction is reduced if there are strong team-internal dissimilarities or strong similarities to players that are assigned to a different team. Whenever $s_i(T)$ is above some threshold τ, then player i is "satisfied" with its team members and with the separation to other teams. Note that $s_i(T)$ does not depend on how the external players $j \notin T$ are clustered into teams exactly, so the satisfaction remains unchanged even if the external players are going to change their teams (without entering T).

In the Anti-Zerging application it makes sense to set τ to a negative value, so that players accept to stay in team T as long as their satisfaction does not fall below some negative "tolerance" τ. A negative value of τ has the effect of making the clustering more robust against cheating by friendly fire and against over-segmentation (since the hurdle for creating a new team is higher) at the price of an increased chance for under-segmentation and wrong assignments (a player can wrongly be assigned to a team as long as this is tolerated by the player and the team members).

The pruning parameter τ is a single intuitive parameter that controls the overall behavior of the satisfaction-pruned SCC algorithm.

From team assignments to debuff scores

Once the team assignments are computed, the final step is to determine new debuff scores for all players. In the current implementation this is achieved by the following straightforward solution:

1. for every team T, compute its strength by summing over all team members' individual strengths:
$$\texttt{strength}(T) \;:=\; \sum_{i \in T} \texttt{strength}(i) \quad .$$

2. map the team strengths to team debuff scores in the interval $[0, 1]$, where 0 means no debuff, 0.5 moderate debuff, and 1 the strongest possible debuff. The team debuff is computed from the team strength as follows:
$$\texttt{debuff}(T) \;:=\; \begin{cases} 0 & , \; \texttt{strength}(T) \leq \texttt{offset} \\ 1 - \exp\left(-\frac{\texttt{strength}(T) - \texttt{offset}}{\texttt{damping}}\right) & , \; \texttt{strength}(T) > \texttt{offset} \end{cases}$$
for some fixed parameters `offset` and `damping` that control the horizontal shift and the initial slope of the curve (see Figure 4.6.3).

3. for each player i, update its individual debuff score $\texttt{debuff}(i)$. This is achieved by a time-smoothed interpolation from the previous value of $\texttt{debuff}(i)$ toward i's new team debuff (or toward value 0 if player i is in the "no-team") as the new intended value. Precisely, the interpolation results from constraining $\texttt{debuff}(i)$ to increase toward the intended value by at most $\ell^+ > 0$ units per second (or to decrease by at most $\ell^- < 0$ units per second) for some parameters ℓ^+ and ℓ^-.

The updated scores $\texttt{debuff}(i)$ of all players i are finally sent back to the appropriate game servers in order to consider them in the game physics.

This strategy assigns positive debuff scores to almost all teams that participate in any combat. Zergs are penalized indirectly by receiving higher debuff scores than their strongly inferior victims. One can think of sophisticated alternative strategies to define debuff scores, for example, by first detecting which team fights against which other teams, emitting positive debuff scores only in case of strong imbalances. However, the optimal solution depends to a great extent on the way how the debuff scores affect the game physics: if a positive debuff score would directly lower the power of the players' weapons and show up a warning symbol, then the implemented strategy of assigning debuff scores to almost all teams would likely annoy users. But if the debuff scores affect players more subtle, then such a strategy might be accepted by users. For example, positive debuff scores could cause a tiny slowdown of the maximum sprinting speed. Strongly inferior teams will then have a slightly better chance to flee, which can even be justified by the psychological aspect that they run for their life, while strongly superior players cannot exploit their full potential in sprinting. At the end of the day,

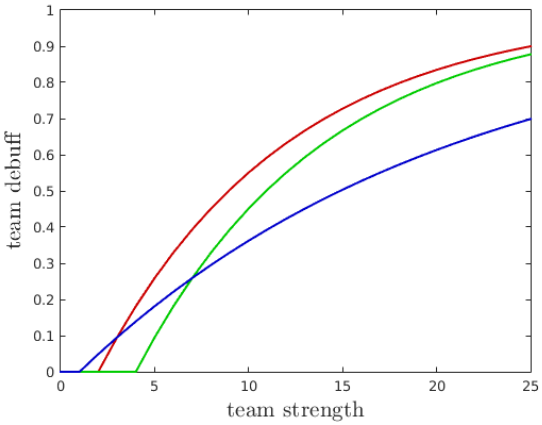


Figure 4.6.3: Mapping from team strength to team debuff. Example plot for three different combinations of the parameters (`offset`, `damping`). Red: $(2, 10)$. Green: $(4, 10)$. Blue: $(1, 20)$. The function equals zero as long as the team strength is below `offset`. For larger values it grows monotonically toward approximating 1. The initial slope depends on `damping`.

the whole approach of "clustering against Zergs" can only be successful if it is possible to include debuff scores into the gameplay in a way that is approved by the users. However, this aspect is out of the scope of this thesis.

4.6.3 Findings from the experiments

In order to have some ground truth available, Sandbox Interactive GmbH arranged a session with 30 test users that played through ten different combat scenarios. During the session all interactions including the information on which player belongs to which team were recorded as a stream of region snapshots. The combats followed coarse storyboards that I prepared in advance, including scenarios such as "*three teams against each other*", "*two teams against each other, where one team is fully surrounded by the other*" and "*a small team of strong players against a large team of weak players*". The recorded data streams build the basis for the test simulator described in Section 4.6.1. By replaying (and transforming) these streams, we can compare the clustering results against the true team assignments.

However, as already discussed in Section 4.1, sometimes it is not possible or not even desired to recover the *true* teams. We should rather focus on identifying the *effective* teams, although there is no such ground truth data available (even not easy to define). Sometimes the clustering results deviate from the true teams, but a human observer agrees with the clustering results rather than considering them as a misclassification. Let us show this issue exemplary on the "*three teams against each other*" scenario. In Figure 4.6.4 (a) the true teams are visualized as (blue) circles, (yellow) squares, and (cyan) triangles, respectively. Since no combat is going on, all players are assigned to the no-team (grayed out), connected only by

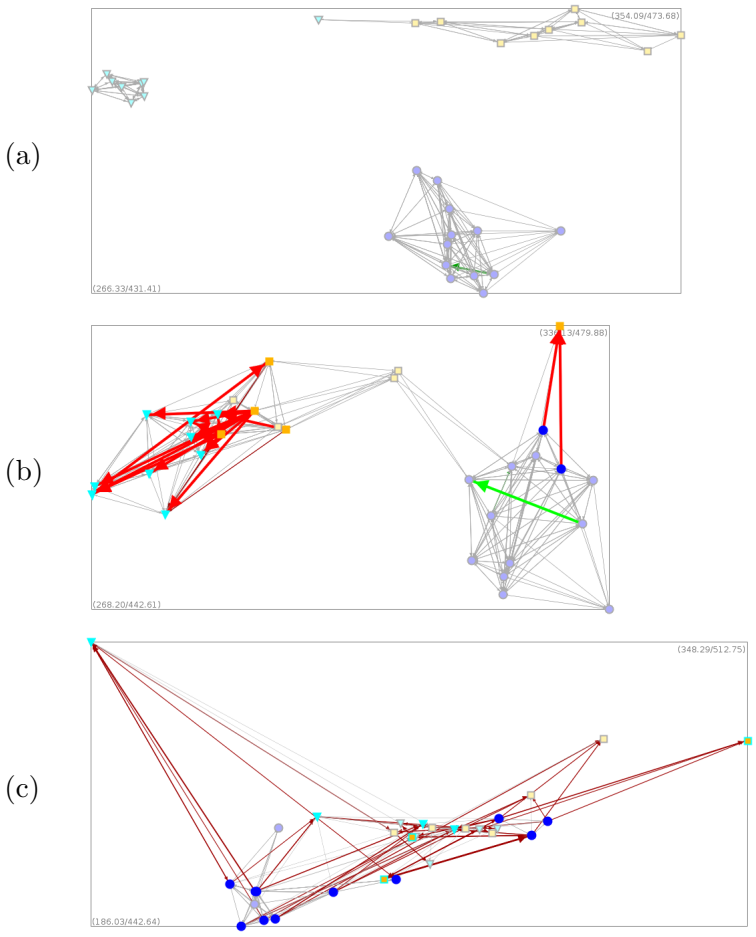


Figure 4.6.4: True teams versus effective teams. (a) before the combat starts, all players are assigned to the no-team (grayed out). The only available information is by spatial closeness between players. (b) to the beginning of the combat, the clustering process suggests that there are four teams: to the left a team of size 9 against another team of size 4, and to the right a team of size 2 against a single player. (c) the clustering process suggests that there is one team of size 11 in a combat against another team of size 7, although the ground truth suggests a three-on-three combat of 11 versus 4 versus 3 remaining players.

spatial similarities (such as the friendly close time). At the top of the map, some player i of the triangle team spies for some time on the square team. During its stay, player i looses the spatial similarity to its true team, but gains some spatial similarity to the opponent team. Every clustering algorithm on this graph would return the three connected graph components as three separate clusters, and a human observer (without seeing the true teams) would agree that this is indeed the best solution. However, with respect to the true team associations, this clustering is suboptimal because player i is known to be a member of the left cluster. No reasonable clustering algorithm would ever assign i to the left cluster, since there is absolutely no intrinsic reason (without additional knowledge) why i should belong to the left cluster. Thus, rather than rating this clustering as erroneous, we should consider our ground truth (based on true teams) as erroneous, and put our focus on effective teams instead. In terms of effective teams it is totally acceptable to assign player i to the cluster at the top. In Figure 4.6.4 (b) the combat has just begun with currently 16 participants. The clustering algorithm splits these 16 players into four teams rather than three. This is reasonable, since the two sub-combats (left: 9 versus 4, right: 2 versus 1) are only connected indirectly via some spatial similarities. In particular it is plausible to not use the spatial similarities for merging the 4 teams into a smaller number of teams: the "correct" (with respect to true teams) merging strategy would join the singleton cluster (the attacked player in the top right) with the team of square players. However, there is no reason provided by the graph for doing so. Any plausible merging strategy would rather merge the intensively connected circle team with the square team, while the single victim to the top right would either form its own team, or it would be merged with the triangle team. Therefore, rather than penalizing the clustering algorithm for this 4-clustering, we should treat it as the correct solution for this graph. In Figure 4.6.4 (c) to the end of the combat there are still 18 players fighting. The clustering algorithm suggests to cluster them into just two teams, one of size 11 and another of size 7. The true teams would instead further split the 7 players into 3 and 4, since they originate from different true teams. However, a closer look at the interactions reveals that these 7 players started to cooperate once they realized that they are even together in an inferior position to the square team. So again it is acceptable that the clustering algorithm splits the combat in a different way than given by the true teams.

These examples show that although we have access to some ground truth, a rigorous quantitative analysis of the data is hardly possible. In order to get a clear understanding of the overall clustering performance it is rather necessary to explore manually all kinds of special cases, while comparing several different parameter settings. The true team labels provide a helpful guidance for this exploratory analysis. I followed this approach up to a certain extent, and my impression is that the overall performance is less a matter of choosing the right algorithm (SCC with satisfaction-pruning works "very well" without quantifying this further), but the performance depends much more on the details how to define the graph weights and all other parameters of the algorithmic superstructure. This parameter tuning cannot be achieved just from the artificial test scenarios. So as a next step it is required

to plug the Anti-Zerg Clustering Software into the real-time productive system in order to evaluate the clustering performance on real combats and to continue tuning the parameters. But this would still not give the final solution. No experiment and no live inspection is able to simulate the feedback loop that is caused by the change in behavior of players that receive a penalty in the game because of computed debuff scores. So the final step will be to implement such penalty strategies into the game physics, and then continue the live inspection and parameter tuning. All these further steps are out of the scope of this thesis. My part finishes with this feasibility study and the insight that the idea of "clustering against Zergs" appears to be possible to become true.

During my visit to the game company's site I learned that a major part of game development is the patient process of tuning hundreds of core parameters plus thousands of deeper parameters. Obviously, game developers prefer parameters that provide a plain intuition rather than cryptic parameters with unforeseeable effects. The threshold τ of the satisfaction-pruned variant of the SCC algorithm is such an intuitive parameter to tune. This is demonstrated in Figure 4.6.5 and Figure 4.6.6 by showing the expected effect of satisfaction-pruning on the clustering results in case of cheating. Both figures show the course of the test scenario "*two-front attack of one team against another*" at five different times, one per row. The left half of Figure 4.6.5 visualizes the original stream data and the achieved clustering results. The right half of Figure 4.6.5 as well as Figure 4.6.6 show the clustering results for the same data modified according to two different strategies of cheating by adding friendly fire:

- **moderate, focused friendly fire**. Every 5th player fixes one other player of its own team and then repeatedly reduces the other's health points by 1% per second.
- **heavy, spread friendly fire**. All players choose randomly every second another player of their own team and reduce its health points by 1%.

Along each row, the combat graph is plotted together with a heat plot of the corresponding non-symmetric adjacency/correlation matrix $[c_{ij}]$. Rows and columns of the matrix are sorted according to the ground truth. This allows to easily observe from the plot that the graph contains two clusters. Next to the correlation plot, the confusion matrix of the SCC algorithm's output is shown. The entry m_{ij} of the confusion matrix shows the number of players that are clustered into team i, while originally belonging to the true team j. In the non-cheating case, the confusion matrix of the unpruned SCC algorithm coincides with the confusion matrix of the satisfaction-pruned SCC-algorithm (for $\tau = -2$). Both show perfect results with respect to the true teams. In the presence of moderate cheating, the unpruned SCC algorithm starts to over-segment the true teams. For example at second 50, six players are split off from the larger team, so the cheaters reached their goal of hiding their association. However, with satisfaction-pruning enabled, this error is corrected and the true teams are recovered. Satisfaction-pruning is able to correct all but one of the cases. This positive effect is even stronger in case of heavy friendly fire as in Figure 4.6.6. The unpruned algorithm gets strongly misguided by the several dissimilarities and tends to put every player in its own singleton team.

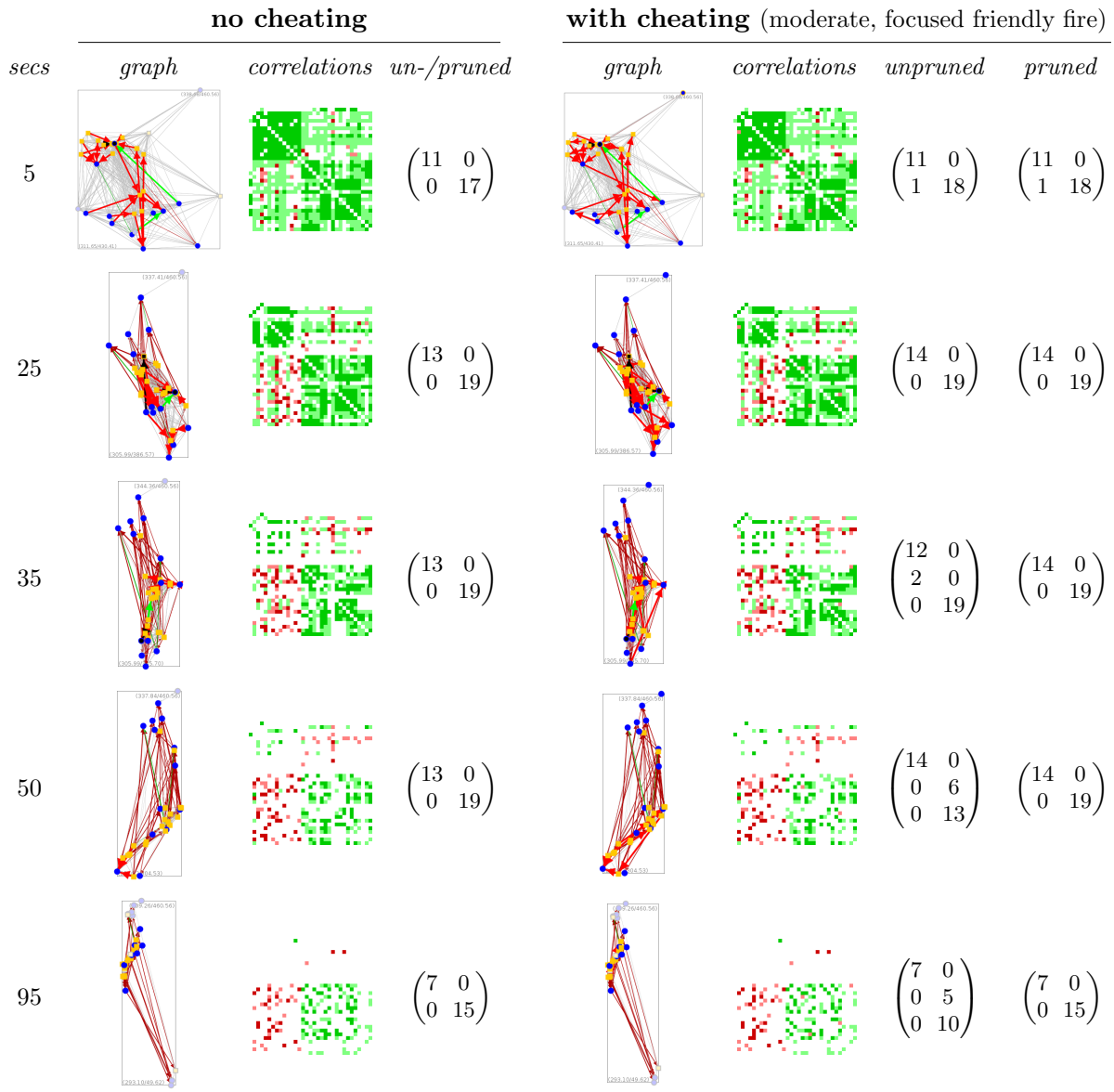


Figure 4.6.5: Satisfaction-pruning in case of moderate friendly fire. Course of a combat at five different times (rows). (left half) original test scenario data. The three individual columns show the combat graph, a heat plot of the correlation weights (light/dark green indicates weak/strong similarities, light/dark red indicates weak/strong dissimilarities), and the confusion matrix of the clustering results (row = cluster, column = true team). (right half) the same test scenario, but with some friendly fire added. While the unpruned SCC algorithm starts to over-segment, the satisfaction-pruned SCC algorithm is still able to recover the true teams almost perfectly.

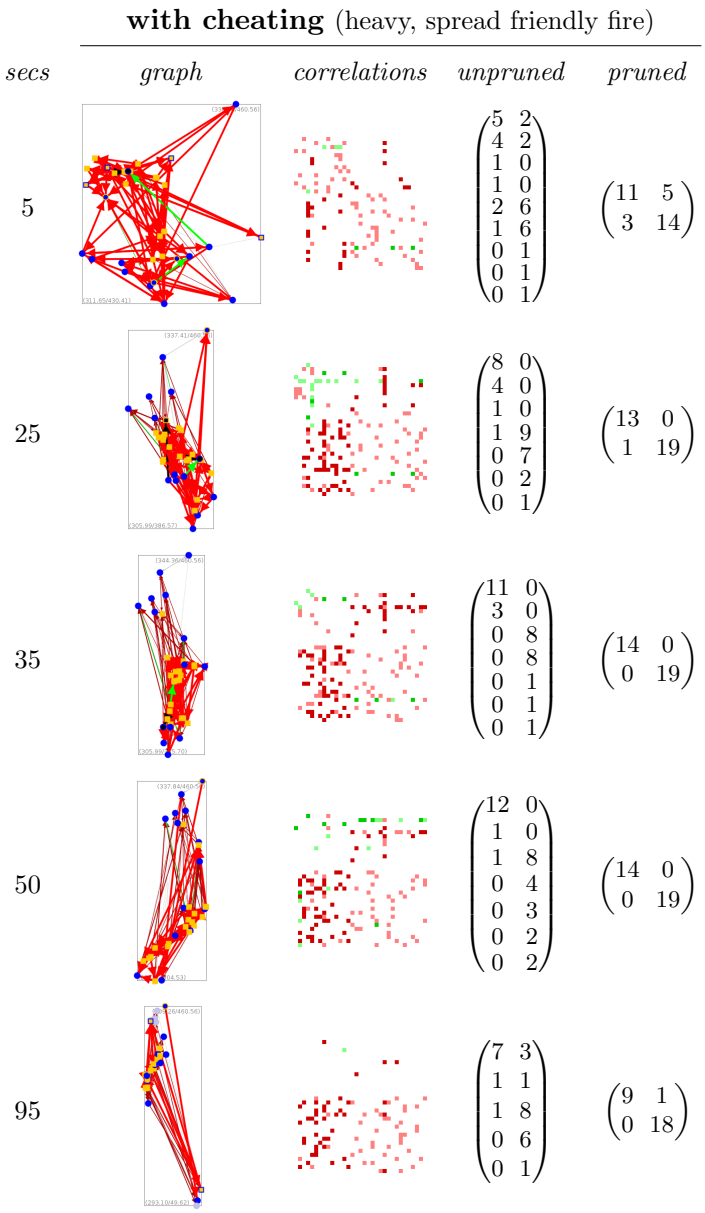


Figure 4.6.6: Satisfaction-pruning in case of heavy friendly fire. Similar to the right half of Figure 4.6.5, but here for more frequent and distributed friendly fire. The SCC algorithm with satisfaction-pruning is still able to recover the true teams to a great extent.

With satisfaction-pruning enabled, the SCC algorithm is again able to recover almost all cases. Note that it may even be true that in the "incorrect" cases the clustering result provides a more reasonable solution in terms of effective teams than the "correct" clustering. From the definition of satisfaction-pruning we get that a player is only assigned to any team if all team members (including itself) tolerate this assignment. This gives us the calming insight that a player can only ever be misclassified because of a debatable gray-zone decision rather than because of an epic fail.

4.7 Chapter summary

This chapter studies another variant of graph Laplacian matrices. Like in both chapters before, a tiny modification to the standard definitions causes strong differences in the algebraic properties and their interpretation. Just by defining the entries in the degree matrix in terms of *absolute* edge weights rather than by the original real-valued edge weights, the interpretation of the Laplacian's smallest eigenvector changes fundamentally: while the smallest eigenvector of the standard Laplacian is meaningless in similarity-based clustering, the smallest eigenvector of the signed Laplacian becomes the most informative one for correlation-clustering. I prove this by a novel spectral relaxation of the NP-hard problem of determining the optional real-weighted MinCut, denoted as OptMinCut. I particularly argue that one should think about the smallest eigenvector of the signed Laplacian as an approximation of OptMinCut, rather than as an approximation of SignedRatioCut as currently stated in the literature.

These theoretical insights result in an algorithm: the SCC algorithm. Its main idea is that of recursive spectral clustering, but it shows two additional interesting properties. First, it refers to unbalanced cuts, which I claim to be desired in correlation clustering rather than balanced cuts. It is a fundamental paradigm shift to no longer focus on balanced cuts, but to study unbalanced cuts for spectral clustering. Second, it automatically prunes the cluster-tree as soon as no cut of negative weight exists. I compare this algorithm to several alternative approaches to correlation clustering. It turns out that the SCC algorithm is able to outperform existing spectral and non-spectral methods, so it is at least on par with state-of-the-art. It shows a particular strength on sparse graphs with noisy edge weights.

This chapter is originally motivated by a practical application from online gaming, where the goal is to robustly detect a certain type of cheaters. My suggested solution is a pruned variant of the SCC algorithm that provides an intuitive parameter of tolerance to misclassification. In order to transfer the theoretical contributions to practice, several additional algorithmic challenges were solved as presented in this chapter.

Conclusions

Studying graphs by linear algebra is still a fascinating and effective approach, even after decades of highly active research across different communities. There are still countless new results published every year that reveal surprising and useful properties of graphs in terms of algebraic properties of their graph matrices. This thesis contributes a couple of novel results to this field, with a focus on machine learning and its applications. Since every result in research is accompanied by even more questions that point into possible directions of future work, I am going to conclude this thesis by describing some of these challenges.

Chapter 2 provides a quite complete picture of the technique of **f**-adjusting, but it still leaves many questions open. The geometric interpretation refers to the convergence of volumes and cut weights to well-understood limit quantities, given that the number of sample points goes to infinity. As demonstrated by applications, these asymptotic results provide already meaningful explanations for samples of realistic sizes. Future work on **f**-adjusting could focus on elaborating explicit approximation guarantees for finite sample sizes.

Another line of future work could focus on weakening the assumptions that are put on the underlying density as well as on the cut surface to be a hyperplane. The current assumptions are made because of technical requirements of certain proof steps, and in order to keep the overall complexity of the proof manageable. Probably one can find an elegant argument to generalize the proof from hyperplanes to more general smooth cut surfaces, or one can even base the proof on less restrictive techniques such as Γ-convergence (Trillos and Slepčev, 2016).

Finally, it could be interesting to connect **f**-adjusting to manifold learning, for example to Locally-linear embedding (Roweis and Saul, 2000), since these techniques often work poorly in case of non-uniform sampling densities. Probably it is possible to assist these techniques by removing the non-uniform distortion in a similar way as achieved by `UnbiasedSpectralIntensityClustering` (Algorithm 2.6.3).

In Chapter 3 it is shown that the SIPF sequence equals the m-sequence for $m = m_G$ chosen as the geometric mean function. In contrast to PSIPF, the m-sequence converges to a different limit for a different choice of m. It is a challenging question how to generalize the convergence of m-sequences to other mean functions than m_G, and how to interpret their limit matrix. As shown in Figure 3.3.1, for m the minimum or the maximum function no convergence to a feasible solution can be guaranteed. However, there is some evidence that the m-sequence

converges to a feasible solution for any other Hölder p-mean function, that is for any finite p. Probably the limit matrix can then be expressed as being optimal with respect to a more general family of f-divergences, yielding relative-entropy just as a special case for $m = m_G$. Since the proof of convergence of SIPF exploits the factorizing property of the geometric mean, proving the above conjecture will require other techniques.

Another interesting direction is the comparison of the single-step $\mathbf{f}$-scaled matrix $W^{(1)}$ of degree vector $\tilde{\mathbf{d}}$ to the $\mathbf{f}$-fitted matrix $W^{(\infty)}$ of degree vector $\mathbf{f}$. A larger relative deviation between $\tilde{d}_i$ and f_i can indicate critical vertices that are "stressed" more than others. Similarly, the deviation between $w_{ij}^{(1)}$ and $w_{ij}^{(\infty)}$ could indicate certain edges that are crucially important for a successful balancing. Both scores may provide interesting semantics in studying networks.

Finally, it would further be interesting to study the rate of convergence of SIPF and of general m-sequences, in particular as a function of the sparsity of W or of the ratios f_i/d_i.

For the SCC algorithm introduced in Chapter 4, an interesting challenge is to study in deep the various improvements that are stated in Section 4.4.2. This can be done in a heuristic, experimental way, or formally. One step into the latter direction could be the formal analysis of the "ideal" cluster-tree that is generated by providing the SCC algorithm with a perfect solver for OptMinCut. The ideal cluster-tree is a hierarchical partition into subsets of non-negative minimum cut. Probably one can give theoretical guarantees on the cut weight or the cardinality of such an ideal partition.

Another interesting line of future work is to connect the SCC algorithm deeper to the literature on the well-studied MaxCut problem, see also the brief summary in Section 4.3.3. Since many results on MaxCut focus on proving constant-factor approximation guarantees, these results could help to find such for the SCC algorithm.

Finally, one could study the spectral relaxation of OptMinCut and SignedRatioCut in more detail. The approach in Section 4.5.2 suggests a large class of alternative non-relaxed optimization problems, all of which can be relaxed to the same eigenvalue problem. Among all these alternative interpretations, it is likely that OptMinCut is the best approximation in some precise sense. A rigorous proof for this kind of optimality would demonstrate again that the smallest eigenvector of the signed Laplacian should be interpreted as an approximate solution to the OptMinCut optimization problem rather than to any other.

In whatever direction the hypothetic future work would go, it should continue to equip its results with an intuition that is as simple as possible. The bridge between Theoretical Machine Learning and Applied Machine Learning, and even more general between Data Science and Data Engineering, requires intuitive abstractions of the technical details on both sides. In order to creatively apply theoretical results in practice, an intuitive access to the application-relevant aspects is required. The other way round, physical, economic and other domain-dependent constraints must be abstracted away by an adequate model in order to be accessible for theoretical research. The collective continuous advancement of this bridge is one of the biggest challenges in this day and age.

Bibliography

E. Abbe and C. Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *Foundations of Computer Science (FOCS)*, pages 670–688, 2015.

E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486, 2009.

M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. OPTICS: Ordering points to identify the clustering structure. In *International Conference on Management of Data (SIGMOD)*, volume 28, pages 49–60, 1999.

B. Arsić, D. Cvetković, S. Simić, and M. Škarić. Graph spectral techniques in computer sciences. *Applicable Analysis and Discrete Mathematics*, pages 1–30, 2012.

M. Bacharach. Estimating nonnegative matrices from marginal data. *International Economic Review*, 6(3):294–310, 1965.

A. Bagga and B. Baldwin. Entity-based cross-document coreferencing using the vector space model. In *International Conference on Computational Linguistics (COLING-ACL)*, volume 1, pages 79–85, 1998.

S. Bagon and M. Galun. Large scale correlation clustering optimization. *CoRR*, arXiv:1112.2903, 2011.

N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

R. B. Bapat, S. J. Kirkland, and S. Pati. The perturbed Laplacian matrix of a graph. *Linear and Multilinear Algebra*, 49(3):219–242, 2001.

H. H. Bauschke and A. S. Lewis. Dykstra's algorithm with Bregman projections: a convergence proof. *Optimization*, 48:409–427, 2000.

R. E. Behrend. Fractional perfect b-matching polytopes. I: General theory. *Linear Algebra and its Applications*, 439(12):3822–3858, 2013.

J. Berg and M. Jarvisalo. Optimal correlation clustering via MaxSAT. In *International Conference on Data Mining Workshops (ICDMW)*, pages 750–757, 2013.

J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 259–302, 1986.

R. Bhatia. *Matrix Analysis*. Springer, New York, 1997.

R. Bhatia. Linear algebra to quantum cohomology: The story of Alfred Horn's inequalities. *The American Mathematical Monthly*, 108(4):289–318, 2001.

C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

D. Boley, G. Ranjan, and Z.-L. Zhang. Commute times for a directed graph using an asymmetric Laplacian. *Linear Algebra and its Applications*, 435(2):224–242, 2011.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

J. P. Boyle and R. L. Dykstra. A method for finding projections onto the intersection of convex sets in Hilbert spaces. *Lecture Notes in Statistics*, 37:28–47, 1985.

L. M. Bregman. The method of successive projection for finding a common point of convex sets. *Soviet Mathematics*, 6:688–692, 1965.

L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7(3):200–217, 1967a.

L. M. Bregman. Proof of the convergence of Sheleikhovskii's method for a problem with transportation constraints. *USSR Computational Mathematics and Mathematical Physics*, 7(1):191–204, 1967b.

M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: identifying density-based local outliers. In *International Conference on Management of Data (SIGMOD)*, volume 29, pages 93–104, 2000.

R. A. Brualdi. Convex sets of nonnegative matrices. *Canadian Journal of Mathematics*, 20: 144–157, 1968.

T. Bühler and M. Hein. Spectral clustering based on the graph p-Laplacian. In *International Conference on Machine Learning (ICML)*, pages 81–88, 2009.

T.-H. Chan, M. Dinitz, and A. Gupta. Spanners with slack. In *European Symposium on Algorithms (ESA)*, pages 196–207, 2006.

M. Charikar and A. Wirth. Maximizing quadratic programs: extending Grothendieck's inequality. In *Foundations of Computer Science (FOCS)*, pages 54–60, 2004.

M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. In *Foundations of Computer Science (FOCS)*, pages 524–533, 2003.

K. Chaudhuri, S. Dasgupta, S. Kpotufe, and U. von Luxburg. Consistent procedures for cluster tree estimation and pruning. *IEEE Transactions on Information Theory*, 60(12): 7900–7912, 2014.

Y. Chen, S. Sanghavi, and H. Xu. Clustering sparse graphs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2204–2212, 2012.

Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 17(8):790–799, 1995.

F. Chung. *Spectral Graph Theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. American Mathematical Society, 1997.

F. Chung. Laplacians and the Cheeger inequality for directed graphs. *Annals of Combinatorics*, 9(1):1–19, 2005.

C. Cooper, M. Dyer, and A. Handley. The flip Markov chain and a randomising P2P protocol. In *Principles of Distributed Computing (PODC)*, pages 141–150, 2009.

A. Cornejo and R. Nagpal. Distributed range-based relative localization of robot swarms. In *Algorithmic Foundations of Robotics XI*, pages 91–107. Springer, 2015.

I. Csiszar. I-divergence geometry of probability distributions and minimization problems. *Annals of Probability*, 3(1):146–158, 1975.

M. Cucuringu, I. Koutis, S. Chawla, G. Miller, and R. Peng. Simple and scalable constrained clustering: A generalized spectral method. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 445–454, 2016.

M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2292–2300, 2013.

M. Cuturi and A. Doucet. Fast computation of Wasserstein barycenters. In *International Conference on Machine Learning (ICML)*, pages 685–693, 2014.

D. Cvetković, M. Doob, and H. Sachs. *Spectra of graphs: theory and application*, volume 87. Academic Press, 1980.

D. Cvetković, P. Rowlinson, and S. Simić. *An introduction to the theory of graph spectra*, volume 75. Cambridge University Press Cambridge, 2010.

E. D. Demaine and N. Immorlica. Correlation clustering with partial information. In *Approximation, Randomization and Combinatorial Optimization: Algorithms and Techniques*, pages 1–13. 2003.

E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2):172–187, 2006.

L. Devroye and G. Lugosi. *Combinatorial Methods in Density Estimation*. Springer, New York, 2001.

W. Donath and A. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973.

R. L. Dykstra. An algorithm for restricted least squares regression. *Journal of the American Statistical Association*, 78(384):837–842, 1983.

D. Emanuel and A. Fiat. Correlation clustering – minimizing disagreements on arbitrary weighted graphs. In *European Symposium on Algorithms (ESA)*, pages 208–220. 2003.

M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining (KDD)*, volume 96, pages 226–231, 1996.

M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2): 298–305, 1973.

F. Fouss, A. Pirotte, J.-M. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

D. Gale. A theorem on flows in networks. *Pacific Journal of Mathematics*, 7(2):1073–1082, 1957.

K. Germina, S. Hameed, and T. Zaslavsky. On products and line graphs of signed graphs, their eigenvalues and energy. *Linear Algebra and its Applications*, 435(10):2432–2450, 2011.

Z. Ghahramani. Graphical models: parameter learning. *Handbook of Brain Theory and Neural Networks*, 2:486–490, 2002.

M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

O. Goldschmidt and D. S. Hochbaum. A polynomial algorithm for the k-cut problem for fixed k. *Mathematics of Operations Research*, 19(1):24–37, 1994.

L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems*, 11(9): 1074–1085, 1992.

K. Hall. An r-dimensional quadratic placement algorithm. *Management Science*, 17(3): 219–229, 1970.

M. Hein and S. Setzer. Beyond spectral clustering – tight relaxations of balanced graph cuts. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2366–2374, 2011.

A. J. Hoffman. Some recent applications of the theory of linear inequalities to external combinatorial analysis. *Proceedings of Symposia in Applied Mathematics*, 10:113–128, 1960.

P. Holland, K. Laskey, and S. Leinhardt. Stochastic blockmodels: first steps. *Social networks*, 5(2):109–137, 1983.

Y. Hou, J. Li, and Y. Pan. On the Laplacian eigenvalues of signed graphs. *Linear and Multilinear Algebra*, 51(1):21–30, 2003.

M. Imakaev, G. Fudenberg, R. P. McCord, N. Naumova, A. Goloborodko, B. R. Lajoie, J. Dekker, and L. A. Mirny. Iterative correction of Hi-C data reveals hallmarks of chromosome organization. *Nature Methods*, 9(10):999–1003, 2012.

C. T. Ireland and S. Kullback. Contingency tables with given marginals. *Biometrika*, 55(1): 179–188, 1968.

Y. Jia and M. Harman. An analysis and survey of the development of mutation testing. *IEEE Transactions on Software Engineering*, 37(5):649–678, 2011.

G. Kirchhoff. Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird. *Annalen der Physik*, 148(12): 497–508, 1847.

P. Knight, D. Ruiz, and B. Uçar. A symmetry preserving algorithm for matrix scaling. *SIAM Journal on Matrix Analysis and Applications*, 35(3):931–955, 2014.

J. Kruithof. Telefoonverkeersrekening. *De Ingenieur*, 52(8):E15–E25, 1937.

J. Kunegis, S. Schmidt, S. Albayrak, C. Bauckhage, and M. Mehlitz. Modeling collaborative similarity with the signed resistance distance kernel. In *European Conference on Artificial Intelligence (ECAI)*, pages 261–265, 2008.

J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. W. De Luca, and S. Albayrak. Spectral analysis of signed graphs for clustering, prediction and visualization. In *SIAM International Conference on Data Mining (SDM)*, volume 10, pages 559–559, 2010.

S. Kurras. Symmetric iterative proportional fitting. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 526–534, 2015.

S. Kurras, U. von Luxburg, and G. Blanchard. The f-adjusted graph Laplacian: a diagonal modification with a geometric interpretation. In *International Conference on Machine Learning (ICML)*, pages 1530–1538, 2014.

B. Lamond and N. F. Stuart. Bregman's balancing method. *Transportation Research B*, 15 (4):239–248, 1981.

J. R. Lee, S. O. Gharan, and L. Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):37, 2014.

T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999.

D. Levin, Y. Peres, and E. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2009.

D. O. Loftsgaarden and C. P. Quesenberry. A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3):1049–1051, 1965.

L. Lovász. Random walks on graphs: a survey. In *Combinatorics, Paul Erdös is eighty*, pages 353 – 397. János Bolyai Matematikai Társulat, Budapest, 1993.

G. Lugosi and A. Nobel. Consistency of data-driven histogram methods for density estimation and classification. *The Annals of Statistics*, 24(2):687–706, 1996.

M. Maier, U. von Luxburg, and M. Hein. Influence of graph construction on graph-based clustering measures. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1025–1032, 2009.

M. V. Menon. Matrix links, an extremization problem, and the reduction of a non-negative matrix to one with predescribed row and column sums. *Canadian Journal of Mathematics*, 20:225–232, 1968.

R. Merris. Laplacian matrices of graphs: a survey. *Linear Algebra and its Applications*, 197: 143–176, 1994.

B. Nadler, N. Srebro, and X. Zhou. Statistical analysis of semi-supervised learning: The limit of infinite unlabelled data. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1330–1338, 2009.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems (NIPS)*, pages 849–856. MIT Press, 2002.

M. Penrose. *Random geometric graphs*, volume 5. Oxford University Press, 2003.

O. Pretzel. Convergence of the iterative scaling procedure for non-negative matrices. *Journal of the London Mathematical Society*, 21(2):379–384, 1980.

F. Pukelsheim. Biproportional scaling of matrices and the iterative proportional fitting procedure. *Annals of Operations Research*, 215(1):269–283, 2014.

C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 22(8):888–905, 2000.

R. Sibson. SLINK: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

R. Sinkhorn. Diagonal equivalence to matrices with prescribed row and column sums. *The American Mathematical Monthly*, 74(4):402–405, 1967.

R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

S. Stańczak, M. Wiczanowski, and H. Boche. Chapter 1: On the Perron root of irreducible matrices. In *Resource Allocation in Wireless Networks*, volume 4000 of *Lecture Notes in Computer Science*, pages 3–49. 2006.

W. Stute. The oscillation behavior of empirical processes: the multivariate case. *Annals of Probability*, pages 361–379, 1984.

L. Trevisan, G. B. Sorkin, M. Sudan, and D. P. Williamson. Gadgets, approximation, and linear programming. *SIAM Journal on Computing*, 29(6):2074–2097, 2000.

N. G. Trillos and D. Slepčev. Continuum limit of total variation on point clouds. *Archive for Rational Mechanics and Analysis*, 220(1):193–241, 2016.

D. Verma and M. Meila. A comparison of spectral clustering algorithms. *UW CSE Technical Report 03-05-01*, 1:1–18, 2003.

U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395 – 416, 2007.

U. von Luxburg and M. Alamgir. Density estimation from unweighted k-nearest neighbor graphs: a roadmap. In *Advances in Neural Information Processing Systems (NIPS)*, pages 225–233, 2013.

U. von Luxburg, A. Radl, and M. Hein. Getting lost in space: Large sample analysis of the commute distance. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2622–2630, 2010.

J. von Neumann. *Functional operators, volume 2: the geometry of orthogonal spaces*, volume 2. Princeton University Press, 1950.

X. Wang, B. Qian, and I. Davidson. On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, 28(1):1–30, 2014.

H. Wilf. The eigenvalues of a graph and its chromatic number. *Journal of the London Mathematical Society*, 42(1967):330, 1967.

X. Wu, Z. Li, A. So, J. Wright, and S. Chang. Learning with partially absorbing random walks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3077–3085, 2012.

J. Xie and L. Qi. Spectral directed hypergraph theory via tensors. *Linear and Multilinear Algebra*, 64(4):780–794, 2016.

Z. Yang and E. Oja. Quadratic nonnegative matrix factorization. *Pattern Recognition*, 45(4): 1500–1510, 2012.

W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168, 2008.

D. C. Youla. Mathematical theory of image restoration by the method of convex projections. *Image Recovery: Theory and Application*, pages 29–77, 1987.

O. Younis and S. Fahmy. HEED: a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks. *IEEE Transactions on Mobile Computing*, 3(4):366–379, 2004.

S. X. Yu and J. Shi. Understanding popout through repulsion. In *Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–752–II–757, 2001.

T. Zaslavsky. Signed graphs. *Discrete Applied Mathematics*, 4(1):47–74, 1982.

R. Zass and A. Shashua. A unifying approach to hard and probabilistic clustering. In *International Conference on Computer Vision (ICCV)*, volume 1, pages 294–301, 2005.

R. Zass and A. Shashua. Doubly stochastic normalization for spectral clustering. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1569–1576, 2006.

X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002.

V. Zlatić, A. Gabrielli, and G. Caldarelli. Topologically biased random walk and community finding in networks. *Physical Review E*, 82(6):066109, 2010.

Index

F

G

H

I

K

L

T

U

V

W

Z

Eidesstattliche Versicherung

Hiermit erkläre ich an Eides statt, dass ich die vorliegende Dissertationsschrift selbst verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Hamburg, den 31. Oktober 2016,

_____