Graph signals, structures and sketches

Nicolas Tremblay

Université Grenoble Alpes

HABILITATION À DIRIGER DES RECHERCHES

Pour obtenir le diplôme

DE L'UNIVERSITÉ GRENOBLE ALPES

Spécialité : **Traitement du Signal et Apprentissage Statistique**

Arrêté ministériel : 25 mai 2016

Présentée par

Nicolas Tremblay

préparée au sein du **Laboratoire Grenoble Images Parole Signal Automatique (GIPSA-lab)**

dans **l'École Doctorale Electronique, Electrotechnique, Automatique et Traitement du Signal (EEATS)**

Graph signals, structures and sketches

HDR soutenue publiquement le **13/06/2024**,
devant le jury composé de :

M. Marc Lelarge
Directeur de Recherches, INRIA, Rapporteur

Mme. Mylène Maïda
Professeure, Université de Lille, Rapporteuse

M. Antonio Ortega
Professeur, University of Southern California, Rapporteur

Mme Laura Grigori
Professeure, Ecole Polytechnique Fédérale de Lausanne, Examinatrice

M. Luc Pronzato
Directeur de Recherches, CNRS, Examinateur

M. Pierre Vandergheynst
Professeur, Ecole Polytechnique Fédérale de Lausanne, Examinateur

To all my relations

Acknowledgments

The timescale of research is much larger than a mere human life, and the contributions presented here are but a tiny stone of a longstanding adventure. I would like to thank those that came and went before my time, for their perseverance and work that have inspired me on my own path. You left so many jewels behind you!

Science is nothing when it is not disseminated and confronted. I would like to thank all my past and current students – be it my PhD or MsC students, or students I've had in classrooms, at university but also in primary schools – as well as people outside of academia (friends, family). Thank you for your questions, and especially the challenging ones, on scientific but also ethical and political topics.

I would like to thank all my fellow researchers out there, those that I know personally and all those I don't know. Research does not really make sense at the individual level: we are in this together! In particular, I would like to thank all my co-authors with whom I've had a great time discussing, filling blackboards, proving things, *etc.* None of this would have been possible without you.

I would also like to thank my team here in Grenoble. When I arrived as a young researcher back in 2016, my team (called CICS back then; now called GAIA) has protected me from the darkest sides of research, including: i) its disproportionate competition and race to excellence (whatever that means) stimulating self-importance, egos, over-publication where it is to me clear that team-work, slow-science and horizontality is the best path forward, ii) the impressive waste of time and money induced by the grant system, iii) the multi-scale layers of the administration, *etc.* I am very grateful for my team that told me as a welcome gift: "don't worry about money and grants, don't worry about the system and how it works, you'll learn soon enough. We will provide you with what you need, just do your best science".

I am grateful to the administrative services of my lab (the Gipsa-lab) for their help navigating the intricate and overly frustrating maze of the French administration. I would also like to thank the CNRS, my employer, for its continued belief that it makes sense to offer "lifetime" employment to young researchers at PhD + 2 or 3 years; with a salary and status that are not h-indexed. I continue to believe that employment security is a better and more human way than brute force competition to perform research: take time to read last century's literature to avoid re-inventing the wheel, create long-term fruitful collaborations, take time to publish what is necessary and not necessarily what is publishable.

I am very grateful to all the members of my jury (with a special thanks to the three reviewers) for accepting to be part of this: I appreciate your time and consideration. My work presented here is only worth what it's worth, but I am honored and pleased (and, I admit, intimidated) to present it to you six in particular.

Finally, and while I am at it, I am joyfully, humbly and profoundly grateful to Life, for the opportunities it gave me to explore this intriguing, fascinating and, let's face it, sometimes bizarre, scientific path.

Foreword

Dear reader,

for those of you unfamiliar with the French *Habilitation à Diriger les Recherches*, it is the "last" academic diploma one can obtain at University in France. Common in other (mainly European) countries, the aim of the exercise, as I understand it, is to take some perspective on the work done since the PhD, show the work of supervised PhD students, and draw a vision for future work. It comes with i) this manuscript, submitted for review to a jury, ii) an oral defense in front of the same jury.

There are several schools in France on how to write this manuscript, ranging from a 40-page copy/paste of a few papers to a whole 350-page book of new material with a pedagogical introduction of one's different research themes. I aimed somewhere in the middle of this range. I started by copy/pasting parts of a few selected (published and unpublished) works, and then concentrated on harmonizing, drawing links, removing details, to provide a larger picture and reveal the underlying thread. From time to time, in the flow of my writing, I obtained a few new results that I added as well.

I am painfully aware that this manuscript is at times not very pedagogical: it delves into very different concepts, from coresets to loop-erased graph random walks, from multi-scale graph representations to accelerated sampling algorithms for determinantal point processes. Properly introducing all these domains would have taken me much more time and space, and is out-of-scope of this "habilitation" document. I do hope that these chapters (that can be read independently) can still be useful to some readers, interested in the general topics of (graph) signals, (graph) structures, and (graph) sketching.

This document relates the work of a selection of ~ 15 research papers published since my PhD: I made my choice to allow for a sort of continuity and harmony in the manuscript. A part of my work is thus not represented, a chunk of which I find is truly missing (apart from two examples I briefly discuss in the introduction): my work on applications. I have always had an applicative project. In fact, one of the things I love about working on such versatile objects as graphs is that they can be found in many areas of research. And I have had the great pleasure to work with people of different fields in the past: neurosciences, physicists, art historians.

I now wish you a pleasant reading. And don't hesitate to drop me an e-mail if you have questions, comments, ideas, epiphanies that arise while reading!

Contents

List of Acronyms

Chapter 1

Introduction

Two words. If two words could epitomize my academic work since even before my PhD, it would be: *graphs* and *sketching*.

On the one hand are *graphs* and their ability to represent the complexity and interconnectedness of systems. They can be as large as needed, as connected as required by the system, with edges that can be weighted, directed, and/or multiple. With node or edge features[1] that augment the graph in its modeling capacity: from spiking activity in neuronal networks in neuroscience, to the dynamics of an epidemic on a human contact network, from chemical affinities on molecular networks to the spread of fake news in social networks.

On the other hand is *sketching* (via sampling, projecting, coarsening, ...) and its ability to simplify, to reduce, to detect invariants, to capture the essence of the truly useful information. Understood here in a very broad sense, sketching is the art of reducing the size of data in a way that keeps the information deemed important. Sketching is used for mainly two purposes. The first one is for description/visualization purposes: the idea here is to reduce the data in order to be able to represent it in a way that is understandable by a human. The second main purpose is for computational purposes: the idea here is to summarize data either for efficient storage/transmission or to run a costly processing algorithm on the reduced data (an algorithm that would have been prohibitive to run on the whole data). In this case, the summarized data –the *sketch*– does not necessarily need to be in a human-friendly form, it can take any form as long as it keeps what is important.

These two concepts, graphs and sketching, sit at both ends of a simple-to-complex spectrum and linking them together may appear to be doomed to failure. There is indeed an underlying tension –that I think is visible throughout my work– stemming from the unattainable goal to combine the best of two worlds: the fine-grain details and modeling power of very large graphs, and the coarse-grain vision and understanding enabled by sketches. This tension has nourished my creativity for the last 15 years, and is an underlying thread of my academic work, a part of which is presented in this manuscript.

A few graphs and graph-based questions. Let us start by showing a few examples of how graphs and graph-based questions arise in different scientific fields.

The neuronal network [Bassett & Sporns 2017, De Vico Fallani et al. 2014]. We humans have around 10^{10} neurons in our brain and around 10^{15} inter-neuronal

[1] additional information associated to nodes and/or edges

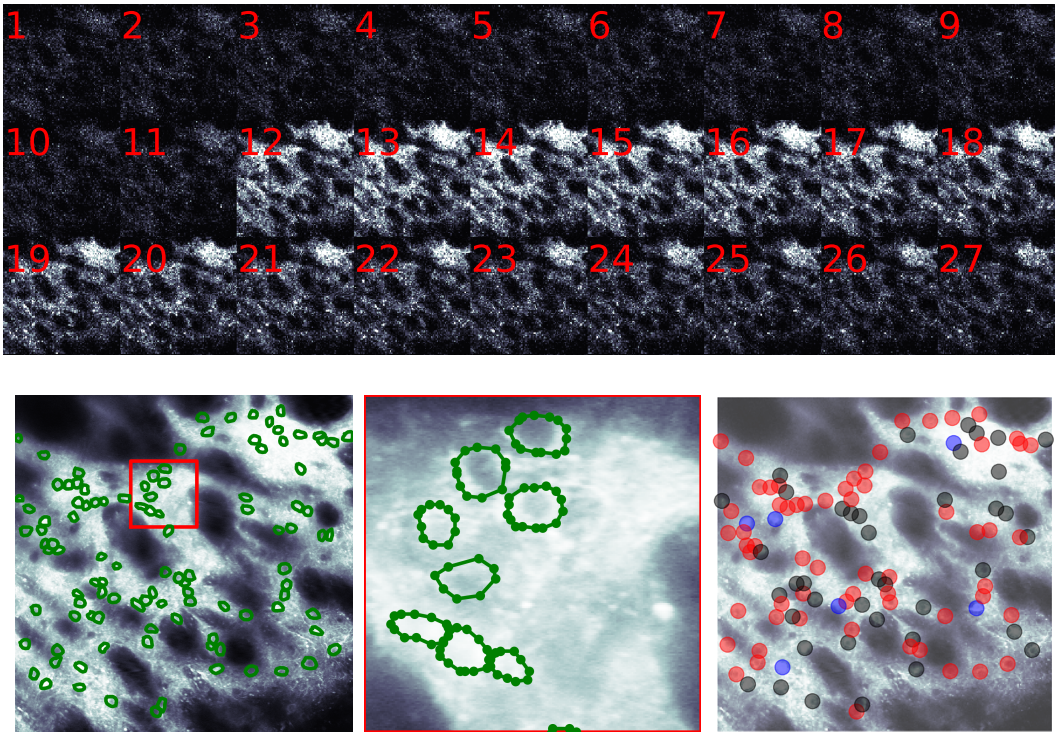

Figure 1.1: **Example from neurosciences.** Top: 27 images extracted from a movie of calcium activitiy in the striatum of a mouse under stimulation (obtained via two-photon fluorescence microscopy). The electric stimulation happens between images 11 and 12, and induces a calcium activity that is visible by fluorescence. Bottom left: one such image (400×400 μm) with neurons that have been detected (in green). Bottom center: zoom of the previous figure's red square. Bottom right: nodes of the graph that models the underlying neuronal network. Each node represents one neuron. Edges are not represented (but we worked here with nearest-neighbor graphs). Different colors are used to represent different types of neurons.

connections. This huge network raises many questions, such as the link between topology and function of brain regions. To illustrate such a type of neuronal data, see Fig. 1.1 extracted from my work in [Becq *et al.* 2019, Badreddine *et al.* 2022, Vertaure 2021]. This figure represents calcium imaging data of a slice of mouse striatum (a part of the brain responsible for procedural memory). We see here a (very) small portion of the mouse's brain at the individual neuron scale. About a hundred neurons are visible in the movie. Modeling this data as a graph: each neuron is a node of the graph, edges are drawn between nearest neighbor neurons, and the fluorescence of each neuron is seen as a time series associated to the corresponding node. The question that concentrated our attention is: when a mouse learns a motor task, is a reorganization of the graph measurable? If so, how does it reorganize itself?

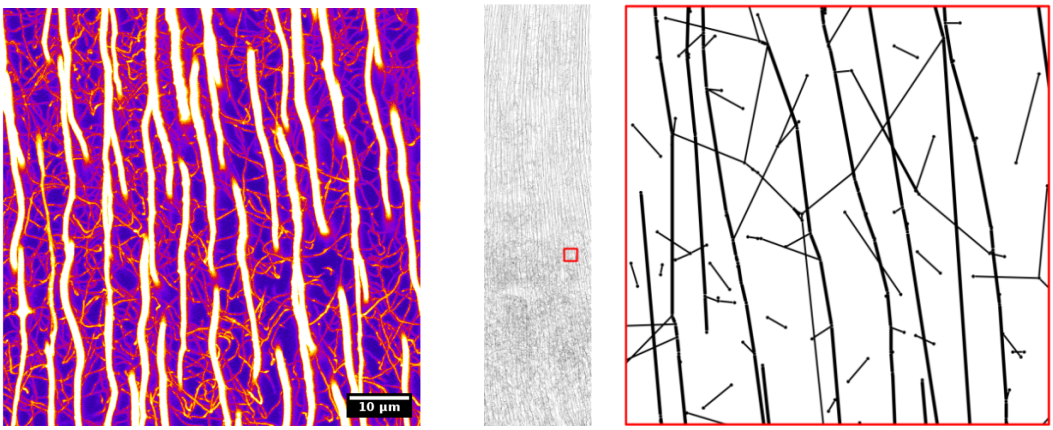

Figure 1.2: **Another example from biology.** Left: zoom on a typical 2D confocal fluorescence image of dentin (porous tissue inside teeth). So-called *tubules* appear as thick somewhat straight and elongated white structures, whereas so-called *branches* are less intense, thinner and show more curvature. Middle: an entire image in which tubules and branches are modeled as edges of a graph (and nodes are the intersection of these edges). Right: zoom of the red square on the previous image. For convenience, the data is shown here in "flattened-2D", but it is in fact 3 dimensional.

Another biological network. Another biological network stemming from direct imaging, and on which I currently work on, is the cellular porosity network in bone tissues and in the dentin [Sigal *et al.* 1984, Vennat *et al.* 2017]. See Fig. 1.2 for some illustrations extracted from my work in [Chatelain *et al.* 2023]. The overall topology and connectivity of these porous networks are believed to be key determinants of the mechanosensing capacity of teeth. The graph-based questions that arise here concentrate on connectivity questions such as: how is the graph connected? Is its local connectivity stationary along the different spatial dimensions of the image? What can we say about its robustness to random localized "attacks" (removal of edges and nodes in a small region)?

Other networks. There exist many different other types of networks in the field of biology, some of them from direct imaging techniques like the two presented above, some of them coming from modeling (sub)systems, such as gene regulatory networks. Now, networks also arise in many other fields of research other than biology, and we give here some classical examples:

- *The internet network [Pastor-Satorras & Vespignani 2004, Tu 2000]* formed by all computers and routers interconnected. On this type of network, security questions are often raised: how does it react to different types of attacks? How should it be wired to improve its resistance to random attacks? Where should data be stored / processed to avoid saturation?

- *The transportation network [Strano et al. 2012, Gallotti & Barthelemy 2015]*

of a city formed by the inter-twined bus, subway, bicycle route networks. Where are the traffic bottlenecks and how to modify the network to avoid them?

- *The network of face-to-face interactions [Isella et al. 2011, Colosi et al. 2022]* in a social event. How fast does a disease (or any type of information) spread? How can we most efficiently contain the spread?
- *Other well-known examples*: the electricity grid, online social networks, financial networks, ..., all with their own sets of questions and challenges.

General methodology. All these networks share the common property that they can be modeled with graphs, mathematical objects made of nodes representing, depending on the application, neuronal regions, routeurs, bus stops, ..., interconnected with edges of varying intensity. This part of the modeling is necessarily done by an expert of the application field. Then, once a graph has been extracted from the network at hand, comes the question of extracting relevant information from it, to answer the specific questions of the research field it came from.

Analyzing such networks via graphs requires specific methods and theoretical tools that have been developed over the last ~ 80 years in the fields of statistics, statistical physics, probability, computer science, combinatorics, signal processing, spectral graph theory, geometry, machine learning, ... These fields of research, when focused on graph analysis, are sometimes put under the umbrella term of *network science*. Under this umbrella, my work has been driven by two main aspects: how can we summarize graphs and/or signals defined over graphs? How can we do so efficiently (in terms of computation cost), especially in the growing number of cases where graphs have large number of nodes and edges? The next few paragraphs give preliminary comments and context on these two questions.

The art of sketching. Let us discuss here a few examples of data sketching, from different fields of research: sampling continuous signals, subsampling rows of matrices, subsampling elements of a point cloud, community detection in graphs, *etc.* Some of these examples are illustrated in Fig. 1.3.

Example 1: Sampling continuous signals for perfect reconstruction [Shannon 1949]. Consider a continuous time signal $f(t)$ defined on an interval $I \subset \mathbb{R}$, a cornerstone theorem of signal processing states the following:

Theorem 1 (Nyquist-Shannon sampling theorem)**.** *If a signal $f(t)$ does not contain any frequency larger than $B(>0)$ hertz, then it can be perfectly recovered from regular samples taken at intervals smaller than $1/2B$ seconds.*

In other words, a sufficient sample rate for perfect reconstruction of $f(t)$ is anything larger than $2B$ samples per second. From a sketching point-of-view, this is absolutely remarkable: the information of a *continuous* signal $f(t)$ (that *a priori* takes an infinite memory to store in a computer) can be perfectly (*i.e.*, without any

loss of information) encoded in a *finite* number of samples! In a sense, this is the greatest sketching scenario of all: an extreme reduction of data size that keeps all the information. Of course, all this works by supposing that the signal is B-bandlimited (*i.e.*, does not contain frequencies larger than B); which, in many instances, is either false or unknown.

Example 2: sampling continuous signals for integral estimation [Davis & Rabinowitz 2007]. Another classical topic regarding sampling of continuous signals is numerical integration. How to numerically integrate functions defined on continuous spaces (that by definition cannot be stored in a computer)? Sampling (may it be deterministic or stochastic) is in general the only way to go in this scenario. In numerical integration, perfect estimation of the integral is in general impossible and theorems usually find a way to bound the error made by the sampling/estimation algorithm.

Example 3: sampling rows of a matrix for a fast estimation of its Singular Value Decomposition (SVD) [Drineas & Mahoney 2016]. Computing the SVD of a tall $n \times m$ matrix costs $\mathcal{O}(nm^2)$ number of elementary operations. When n and/or m become tool large, this cost may be prohibitive with regard to the available computing budget. In this example, the idea is to i/ sample a subset of rows of the matrix, ii/ compute the SVD on this reduced matrix, and iii/ lift back the result in the original dimension. This procedure can also be done via element-wise sampling, and can also be preceded by a random-projection-type "preconditioning" operation, in order to "spread out" or uniformize the information in the original matrix (making it more amenable to uniform random sampling strategies). Given an admissible error on the result obtained, one of the main objective is to find sampling procedures that minimize the number of samples needed to reach that error.

Example 4: sampling points of a large point cloud for fast clustering. In the context of clustering large point clouds in several (hopefully) well-separated clusters, when the number of points and/or the dimension in which they are defined are too large, it may become prohibitive to run off-the-shelf clustering algorithms (such as k-means heuristics or hierarchical clustering algorithms). An elegant way of accelerating these algorithms is to i/ sample a small proportion of the point cloud, ii/ run the clustering algorithms on this small "sketch", iii/ lift back the result to all points of the original dataset. There are many ways to do so, by defining landmarks [Vladymyrov & Carreira-Perpiñán 2017], by sampling coresets [Jubran *et al.* 2021], *etc.*

Graph sketching. When it comes to data in the form of graphs, there are two main ways to approach sketching.

Graph sparsifiers. The first one consists in reducing the number of edges while keeping the same number of nodes; and leads to methods referred to as graph sparsifi-

cation. This line of work is devoted to showing the existence and designing sampling algorithms of sparse subgraphs of the original graph for the approximation of various tasks: cut sparsifiers [Benczúr & Karger 1996, Karger 1999] approximate every graph cut, spectral sparsifiers [Spielman & Teng 2011, Spielman & Srivastava 2011] approximate the spectrum, spanners [Peleg & Schäffer 1989, Ahmed *et al.* 2020] approximate all pairwise distances between nodes, etc. A famous example of graph-based numerical linear algebra task for which such sparsifiers have been found useful is to approximate the pseudo-inversion $\boldsymbol{x} = \mathsf{L}^{\dagger}\boldsymbol{b}$ with L the Laplacian matrix of a graph with n nodes. Whereas exact (pseudo-)inversion has a worst-case cost in $\mathcal{O}(n^3)$, spectral sparsifiers enable an approximate pseudo-inversion in nearly linear time in the number of edges [Vishnoi 2013, Spielman & Teng 2014].

Graph coarsening. A different point-of-view on graph sketching is the coarsening paradigm: rather than sampling some edges of a graph and discarding the others, the graph is locally aggregated (coarsened) to provide a higher scale vision with fewer nodes. To this end, nodes are either sampled or locally merged, and edges are in general sampled or re-wired. Early influential ideas in this direction, originally designed for solving large-scale systems of equations such as those formed by the discretization of partial differential equations, can be found in the algebraic multigrid literature [Ruge & Stüben 1987, Xu & Zikatanov 2017]. The founding concept of iterating cycles of multilevel coarsening and lifting has been since generalized to graphs in various forms and is nowadays a cornerstone of many modern graph analysis algorithms, may they be for clustering [Dhillon *et al.* 2007, Loukas 2019], graph visualization [Walshaw 2006], graph signal analysis [Gavish *et al.* 2010], or graph convolutional neural networks [Bronstein *et al.* 2017, Wu *et al.* 2021]. In some sense, *community detection* [Fortunato 2010, Fortunato & Hric 2016] (where a *community* is loosely defined as a connected subset of nodes that is more connected to itself than with rest of the graph) can be understood as a –rather drastic– graph coarsening operation.

Sketching algorithms. In this introduction, we have up until now only explicitly mentioned one algorithm: the regular, periodic sampling algorithm of the Nyquist-Shannon theorem. This is possibly the most straightforward sampling strategy one can think of. It has however its limitations and, in many cases, one has to resolve to other types of strategies, when periodic sampling is not appropriate for some reason or when it is not well-defined (what does "periodic sampling" even mean on a graph, or in a point cloud?). We now discuss several families of sampling strategies, as well as the main challenges that come up when designing novel sketching algorithms.

Deterministic vs stochastic. A first major choice one needs to make when designing sketching methods is whether to choose a deterministic or stochastic strategy. Depending on the context, there are pros and cons in both worlds. In the deter-

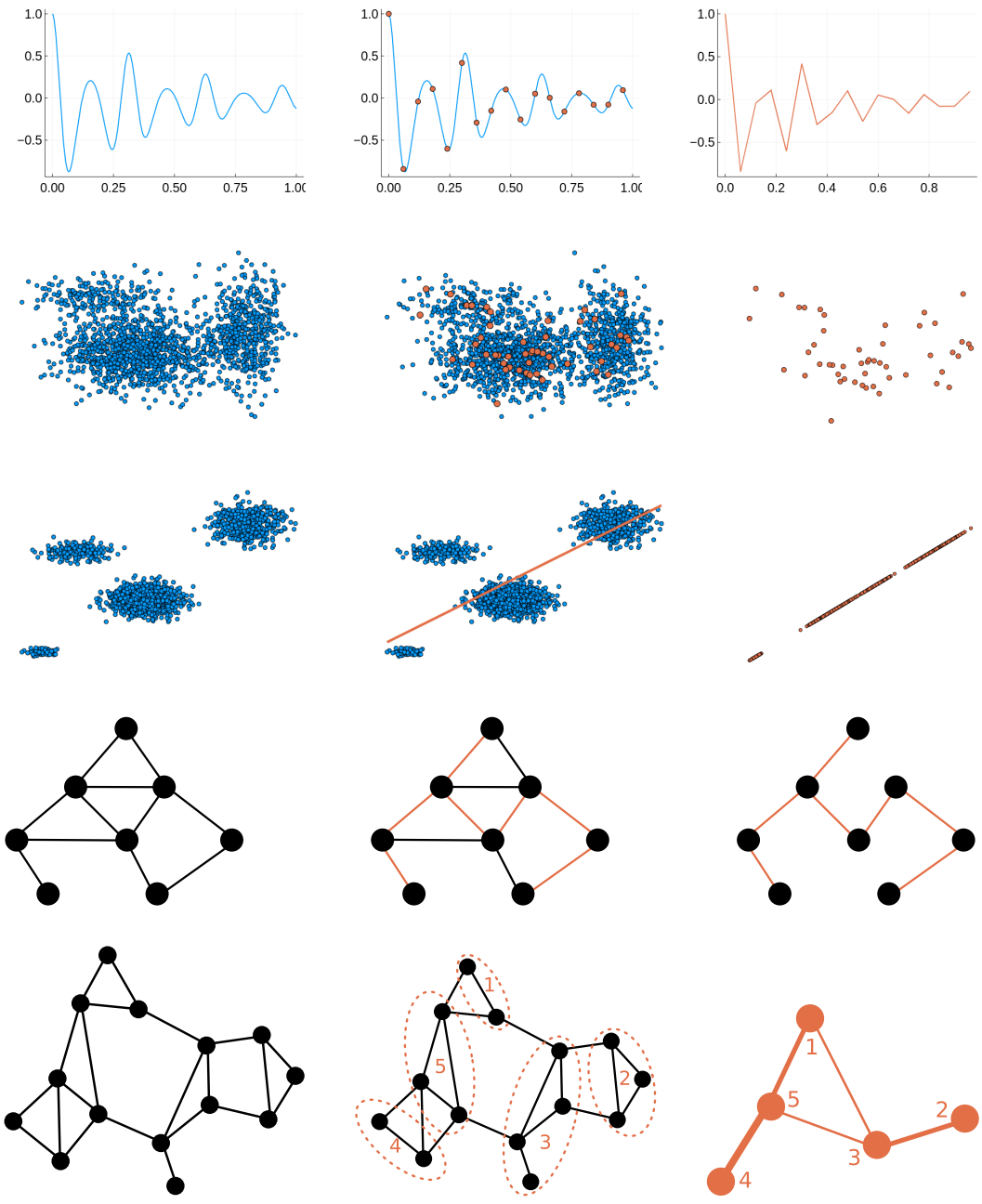

Figure 1.3: **Illustrating different forms of the art of sketching.** *Left column*: original data. *Middle column*: illustration of a sketching strategy. *Right column*: sketched data. *Top line*: periodic sampling of a continuous signal. *2nd line*: uniform sampling of a point cloud. *3rd line*: projection on a smaller dimensional space: here, an orthogonal projection of a point cloud onto a line. *4th line*: a uniform spanning tree (*i.e.*, a uniform sampling from all possible spanning trees) of a graph. *Bottom line*: coarsening of a graph. In all cases, the size of the sketched data is –sometimes drastically– reduced, while some information from the original data is preserved. The central question of the art of sketching asks: *how exactly should one reduce the data (and to what extent it is reducible) in order to preserve the information necessary to the compression, visualization or computation task at hand?*

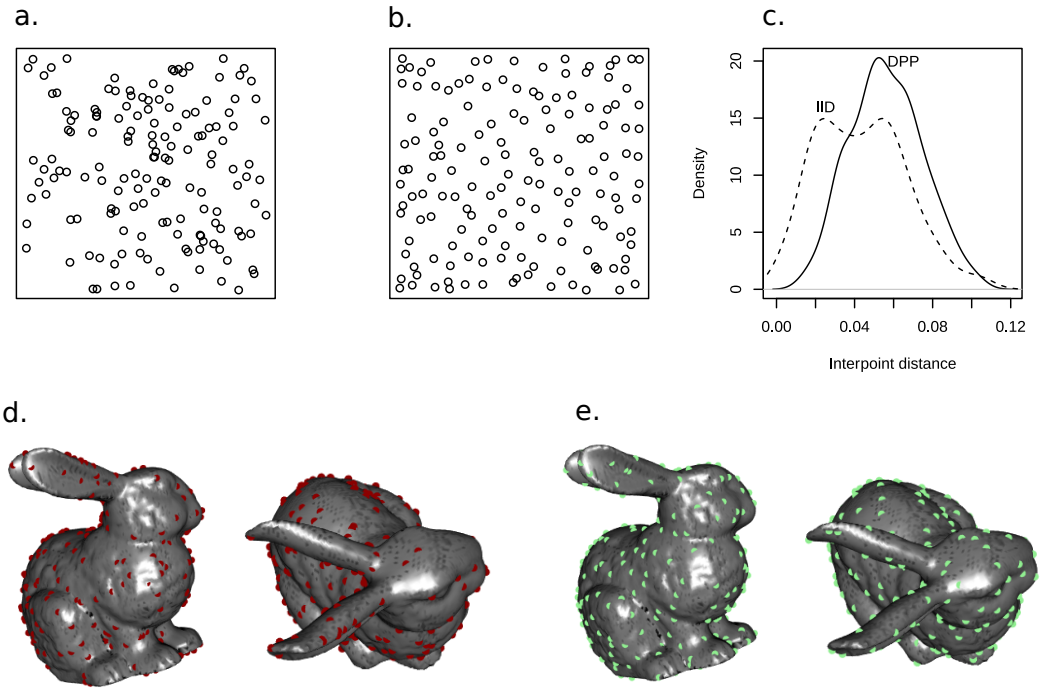

Figure 1.4: **Illustration of the difference between iid and DPP sampling.** Top (on the 2D square): (a) iid sample, (b) DPP sample, (c) density of interpoint distances. Bottom (on the graph associated with the Bunny mesh): (d) iid sample, (e) DPP sample. On both examples, clusters of similar samples are much less probable when using DPPs.

ministic paradigm, greedy algorithms [DeVore & Temlyakov 1996] are often used: these algorithms, at each step, choose the next sample by maximizing a chosen cost function. In the stochastic paradigm, the first go-to sampling strategy is iid (independent and identically distributed) sampling, as it can rely on powerful results from decades of research in concentration theory [Sridharan 2002, Tropp 2015]. Independent sampling has however some obvious drawbacks as it allows to sample elements of the original data that are very similar to each other (each new sample potentially adding only incremental information on the original data). Ideally, and to avoid such issues, one would need random sampling procedures that incorporate some sort of repulsion in order to obtain sampling sets capable of efficiently capturing the diversity of the original data. Determinantal Point Processes (DPPs) [Macchi 1975, Kulesza & Taskar 2012], that will be the main object of interest of Chapter 3, are such an example of repulsive sampling. An illustration comparing iid sampling and DPP sampling is shown in Figure 1.4.

With or without guarantees. Designing a sketching algorithm is usually easy: there are so many ways of doing so! Designing such algorithms in a way that guarantees its invariants (*i.e.*, the information that is kept in the sketch) is a whole

different story. Sketching theorems usually have three ingredients.

Firstly, they need to suppose *something* about the data. The Nyquist-Shannon sampling theorem, for instance, supposes that the signal is B-bandlimited. Community reconstruction theorems need to suppose that the graph has some sort of community structure to start with (a classical hypothesis is that the graph is drawn from a Stochastic Block Model for instance). In matrix row-sampling examples, it is usually assumed that the matrix is approximately low-rank. Secondly, a precise sampling algorithm has to be defined, may it be deterministic or stochastic. Thirdly, one or several properties of the data have to be identified and a suitable error on those properties has to be considered. Sketching theorems are then of the form: under such hypothesis, this algorithm reduces the size of the data while keeping this and that property with a controlled error.

Of course, many algorithms do not come with guarantees; and some of them don't necessarily *need* guarantees: for instance, even though the majority of community detection algorithms do not have any guarantees, they are still largely used to represent complex graph data in an easy-to-understand small sketch. Also, even though it is nice to have guarantees, the hypothesis they require on the data are often not verified in practice, which sometimes make the hard-earned guarantees completely collapse. So, guarantees are nice, but they are not the only focus one should have when designing sketching methods.

Complexity and computation costs. Let us give an example: say we want to compute the SVD of a large $n \times m$ matrix with $n \gg m \gg 1$. The exact computation takes $\mathcal{O}(nm^2)$ and this is too much for our computation budget. We thus resort to element-wise or row sampling strategies to approximate the result. An obvious –yet sometimes overlooked– constraint is the computation cost of the sampling itself! Often, unfortunately, the best sampling strategies –those that provide the tightest guarantees with a minimal number of samples– are those that require computations that are of the same order of magnitude (or even more costly) than the original problem itself! When summarizing to accelerate a given computational task, there is usually a very delicate line to follow between tight guarantees and prohibitive sampling cost. When summarizing for other purposes (to find a useful representation of the data for instance), then this problem does not exist. However, in this case, the best representation is often defined as the optimal solution of a cost function – which often turns out to be a NP problem. In this case, computational complexity is also an issue and one has to come up with sketching strategies that approximate the desired global optimum while being in practice computable.

Structure of the manuscript. This manuscript is structured in the following way. Chapter 2 talks about graph signal processing and spectral clustering algorithms, and concentrates on unveiling the intimate link between graph signals and graph structure. Chapter 3 does not talk about graphs at all and concentrates on

DPPs, with theoretical, algorithmical and applicative aspects. Both chapters are completely independent. Chapter 4 proposes a selection of my work that bridges the previous two chapters and focuses on particular DPPs defined over edges and nodes of graphs (called Kirchoff Forests), and how they can be advantageously used to estimate graph properties. All three chapters have the same structure: after a short introduction, they first have a background section, before three contribution sections. At the end of each contribution section, a "What now?" subsection discusses future directions stemming from the work discussed. Finally, Chapter 5 discusses other future directions of research and perspectives, with some ongoing projects and other more long term visions. An (unusual) Appendix (Appendix A) closes the manuscript: it details my Curriculum Vitae (as this manuscript is, after all, my "habilitation" manuscript).

Chapter 2

Processing graph signals and detecting graph structures

Contents

This chapter delves into my work in *Graph Signal Processing* (GSP). This relatively recent field of research (initiated in the late 2000's) aims at developing tools to process signals defined on the nodes (or edges) of graphs: loads on the Internet network, atoms on a molecular network, preferences on digital social networks, infections propagating on the human contact network, etc. The data of all these examples can be modeled as a signal (possibly multi-variate) that "lives" on a specific structure defined by a graph (that models the underlying network). Initially, the field focused on developing generalizations of classical signal processing tools,

such as the Fourier transform, wavelets transforms, multi-scale filtering, sampling methods, to arbitrary graph structures.

I would say that, after a decade of very active efforts in this direction, the main legacy of GSP is a solid theoretical framework, that can be seen as a "spin-off' of the vast theory of signal processing, that allows for more intuition, and more understanding, of many graph-based processing approaches from different fields, an active example of which is the field of *Graph Neural Networks (GNNs)*.

As the properties of signals on graphs is non-dissociable from the specific graph structure it is defined over, my work in GSP sits at the frontier between GSP and an important subfield of graph structure analysis: community detection. Community detection is the art of finding a partition of the nodes of a graph that separates the nodes in so-called *communities*: groups of nodes that are more connected to one another than with the rest of the graph. There exists many, *many*, community detection methods and in this chapter we will mainly discuss *spectral clustering methods*: methods that perform this partitioning based on the leading eigenvectors of graph-representative matrices, such as the Laplacian matrix, the adjacency matrix, or well-chosen variants of these matrices.

In this chapter, I will shed light on some of my work on this frontier between GSP and community detection. It is separated in four sections:

- Section 2.1, a background section, in which I briefly set a few notations and describe some notions of graph signal processing (graph Fourier transform, graph filtering) and graph structure detection (spectral clustering)
- Section 2.2 describes how community detection tools inspired my work on graph filterbanks and graph signal multiscale analysis.
- Section 2.3 describes how graph signal processing tools, initially created to analyze *signals on graphs*, can be used to explore the *structure of graphs*.
- The last section, Section 2.4, describes my work on how to choose the best representative matrix for spectral clustering, and how this can impact GSP.

2.1 Background

2.1.1 Notations and first definitions

Graphs Throughout this document, and unless otherwise specified, I will consider undirected, weighted graphs, and denote them $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathsf{W})$ where $\mathcal{V}$ is the set of $|\mathcal{V}| = n$ nodes, $\mathcal{E}$ is the set of $|\mathcal{E}|$ edges, and $\mathsf{W} \in \mathbb{R}^{n \times n}$ is the symmetric weighted adjacency matrix. If (ij) does not belong to the set of edges $\mathcal{E}$, $\mathsf{W}_{ij} = \mathsf{W}_{ji} = 0$. Otherwise, if $(ij) \in \mathcal{E}$, $\mathsf{W}_{ij} = \mathsf{W}_{ji} > 0$ is the weight of edge (ij). In cases where the graph is unweighted (all weights are equal to 1), I will denote the adjacency matrix by its more traditional notation: A.

The degree of node i is denoted by $d_i = \sum_j \mathsf{W}_{ij}$, the vector of degrees by $\boldsymbol{d} = (d_1, d_2, \dots, d_n)^\top \in \mathbb{R}^n$, and the diagonal degree matrix by $\mathsf{D} = \mathrm{diag}(\boldsymbol{d}) \in \mathbb{R}^{n \times n}$. Note that in the unweighted case, d_i is simply the number of neighbors of node i.

$\mathsf{L} = \mathsf{D} - \mathsf{W} \in \mathbb{R}^{n \times n}$ denotes the usual (combinatorial) Laplacian matrix of the graph.

Graph signals *Univariate graph signals* $\boldsymbol{x} \in \mathbb{R}^n$ are signals defined over the set of nodes $\mathcal{V}$ of a graph: $x_i = x(i) \in \mathbb{R}$ is the signal's value at node i. The vector $\mathbf{1} \in \mathbb{R}^n$ refers to the constant signal equal to 1, and $\boldsymbol{\delta}_i \in \mathbb{R}^n$ to the Kronecker delta verifying $\delta_i(j) = 1$ if $j = i$, and 0 otherwise.

Graph Signal Processing (GSP) is the art of *processing* (decompose, filter, compress, denoise, interpolate, infer, *etc.*) such graph signals. Some applications require to process graph signals that are defined on the edges of the graph ($x_{(ij)}$ representing, for instance, a flow on edge (ij)). One can come back to the previous case by considering edge-based graph signals on $\mathcal{G}$ as node-based graph signals on the so-called *line graph*[1] of $\mathcal{G}$. Other applications come with multivariate graph signals: to each node is associated a vector of real values and not only a scalar. In the following, we will, for simplicity, only consider univariate graph signals.

Section 2.1.2 will briefly recall a few notations and basic tools of GSP.

Graph structures Processing graph signals is obviously intimately linked to the structure of the underlying graph that encodes in a sense the geometry on which the signal lives. What I call *graph structures* include any marginal information of the structure of the graph: degree distribution, shortest path matrix, different centrality measures, and also its potential modular structure in communities. Where the notion of *community* is loosely defined as a subset of nodes that are more connected to themselves than to the rest of the graph.

Section 2.1.3 will briefly recall a few notations and basic tools of a cornerstone of graph structure analysis: the spectral clustering framework.

2.1.2 Graph filters

A graph filter operates on graph signals in the same way that classical filters operate on classical signals: by decomposing the signal in a Fourier space and multiplying its spectral components by a filter function [Shuman *et al.* 2013, Sandryhaila & Moura 2013]. Many possible definitions of graph Fourier space exist and I will not enter here into the details, pros and cons, of all the ways to properly construct them. For more information on this topic, one may refer to my book chapter [Tremblay *et al.* 2018] reviewing the variety of ways to define this fundamental building block of GSP. I will simply outline the overall strategy common to all these constructions: i) define

[1] Let $\mathcal{G}$ be a undirected, unweighted graph. Its line graph is another graph in which each node represents an edge of $\mathcal{G}$ and there exists a link between two nodes if the two corresponding edges in $\mathcal{G}$ are connected to the same node in $\mathcal{G}$.

a representative matrix (let's call it generically R) that encodes the graph's structure by imposing $\mathsf{R}(i,j) \neq 0$ for all $(i,j) \in \mathcal{E}$, and $\mathsf{R}(i,j) = 0$ otherwise. ii) Denote by $\mathsf{U} = (\boldsymbol{u}_1|\dots|\boldsymbol{u}_n) \in \mathbb{R}^{n\times n}$ the matrix storing the normalized eigenvectors of R, and define the graph Fourier transform of a graph signal $\boldsymbol{x}$ as its transformation from the canonical "node" basis to its representation in the eigenvector basis of R: $\mathbb{F}_{\mathcal{G}}\ \boldsymbol{x} = \mathsf{U}^{-1}\boldsymbol{x}$. iii) To complement this transform and give sense to a notion of filtering, find a clever way to associate to each eigenvector – *i.e.*, each graph Fourier mode – a notion of "frequency" (which, on a graph, is loosely defined as the level of variability along paths on the graph).

The basis of GSP is thus laid down as we have access to a graph frequency analysis: decompose the graph signal $\boldsymbol{x}$ on R's eigenvector basis and obtain the result versus the frequency. One can then filter the signal by a simple multiplication in the graph Fourier space – similarly to classical signal processing.

A classical choice of representative matrix. A classical choice for R is the Laplacian L. Note first that L is symmetric such that $\mathsf{U}^{-1} = \mathsf{U}^\top$. Also, L is *Positive Semi-Definite* (PSD), as shown by Eq. (2.1): its eigenvalues can be ordered as $0 = \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$, where λ_i is the eigenvalue associated to eigenvector $\boldsymbol{u}_i$. Denoting by $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ the diagonal matrix of eigenvalues, L reads:

$$\mathsf{L} = \mathsf{U}\Lambda\mathsf{U}^\top.$$

An attractive property of this choice for R is that it comes with a natural definition of graph frequency: the frequency of mode $\boldsymbol{u}_\ell$ is simply defined as its associated eigenvalue λ_ℓ, as λ_ℓ is directly linked to the variability of $\boldsymbol{u}_\ell$ via the well-known formula:

$$\lambda_\ell = \boldsymbol{u}_\ell^\top \mathsf{L}\boldsymbol{u}_\ell = \frac{1}{2}\sum_{(i,j)\in\mathcal{E}} \mathsf{W}_{ij}(u_\ell(j) - u_\ell(i))^2. \tag{2.1}$$

In other words, the more the graph Fourier mode $\boldsymbol{u}_\ell$ oscillates, the larger its associated eigenvalue λ_ℓ. For instance, plot a) of Fig. 2.1 represents a low-frequency graph signal (here: $\boldsymbol{u}_2$, the eigenvector associated with the second lowest frequency); and plot b) represents a high-frequency graph signal.

Example of graph filtering: Tikhonov denoising. Arguably the most straightforward example of graph signal filtering is low-pass filtering via Tikhonov regularization. Given a noisy signal $\boldsymbol{x}$ measured on the nodes of the graph, the idea is to find the solution to the regularized least square problem, where the Dirichlet form $\boldsymbol{z}^\top\mathsf{L}\boldsymbol{z}$ is used as a regularization promoting smoothness on the graph.

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{z}\in\mathbb{R}^n}{\arg\min}\, q\,\|\boldsymbol{x} - \boldsymbol{z}\|^2 + \boldsymbol{z}^\top\mathsf{L}\boldsymbol{z}$$

for $q > 0$ the regularization parameter. The larger q, the more we want to find a solution that is close to $\boldsymbol{x}$, and the smaller q, the more we want to put emphasis on

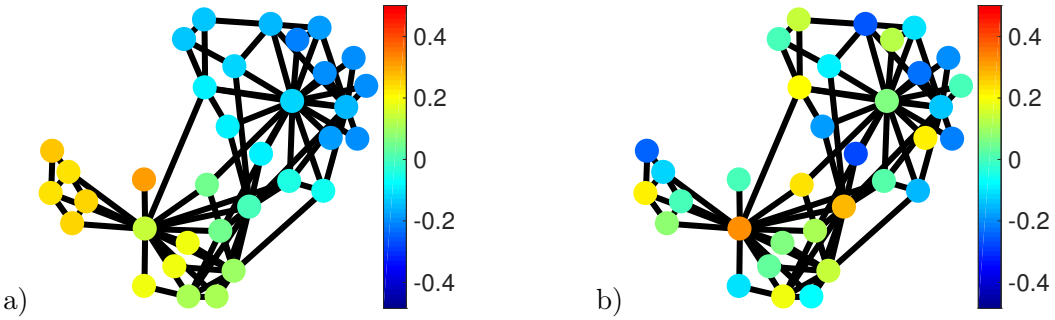

Figure 2.1: Plots a) and b) represent respectively, a low-frequency and a high-frequency graph signals on the binary Karate club graph [Zachary 1977].

the smoothness of the solution. The exact solution to this problem is

$$\hat{\boldsymbol{x}} = q(q\mathsf{I} + \mathsf{L})^{-1}\boldsymbol{x},$$

which can be "spectrally" re-written:

$$\hat{\boldsymbol{x}} = \mathsf{U}h(\Lambda)\mathsf{U}^\top\boldsymbol{x}$$

with $h(\Lambda) = \mathrm{diag}(h(\lambda_1), \ldots, h(\lambda_n))$ and h the function $h(\lambda) = \frac{q}{q+\lambda}$. The operation $\mathsf{U}h(\Lambda)\mathsf{U}^\top\boldsymbol{x}$ thus denotes a low-pass graph filtering operation: it is the succession of the graph Fourier transform of $\boldsymbol{x}$, $\mathsf{U}^\top\boldsymbol{x}$, a multiplication in the graph Fourier space by a low-pass function $h(\lambda)$ (attenuating in this case the components of $\boldsymbol{x}$ with high frequencies), and finally an inverse Fourier transform via the left multiplication by U. An illustration of this low-pass filtering is given in Fig. 2.2, for the choice $q = 1$.

Graph multiscale representations. To process and filter signals on graphs at different scales, it is attractive to develop an equivalent of multiscale transforms such as wavelets, *i.e.*, ways to decompose a graph signal on components at different scales and frequency ranges. The road to multiscale transforms on graphs has originally been tackled in the vertex domain [Crovella & Kolaczyk 2003] to analyze data on networks. Thereafter, a general design of multiscale transforms was based on the diffusion of signals on the graph structure, leading to the powerful framework of Diffusion Wavelets [Coifman & Maggioni 2006, Gavish *et al.* 2010]. These early works were already based on a diffusion operators, usually a Laplacian or a random walk operator, whose powers are decomposed in order to obtain a multiscale orthonormal basis. The objective was to build a kind of equivalent of discrete wavelet transforms for graph signals. Then, came developments of continuous wavelet transforms on graphs such as the spectral graph wavelets [Hammond *et al.* 2011]. Finally, there are approaches that combine filters on graphs with graph decompositions through decimation (pioneered in [Narang & Ortega 2012] with decimation of bipartite graphs) or aggregation of nodes; in a nutshell, these methods are very close to the filter banks implementation of discrete wavelets [Strang & Nguyen 1996]. I will, in Section 2.2, detail one of my contribution on such graph filterbanks.

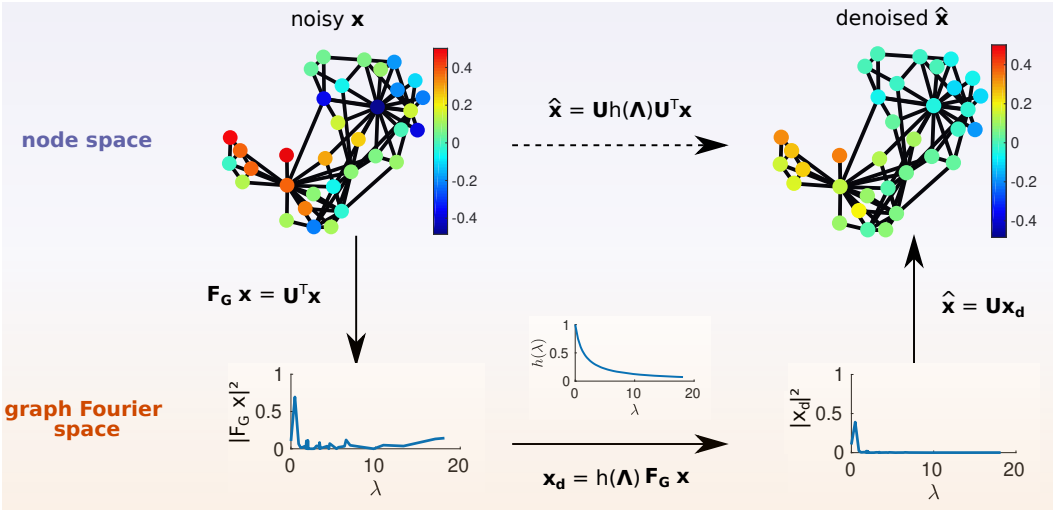

Figure 2.2: **Illustration of a graph filter: a denoising toy experiment**. The input signal $\boldsymbol{x}$ is a noisy version (additive Gaussian noise) of the low-frequency graph signal displayed in Fig. 2.1. We detail here the Tikhonov denoising process, for $q = 1$: a graph Fourier transform, followed by a multiplication in the Fourier space by the filter's coefficients, and ended by an inverse Fourier transform.

2.1.3 Spectral clustering

Spectral clustering refers to a widespread family of unsupervised learning algorithms that compute a spectral embedding of the nodes of a graph to cluster them in communities (nodes closest to each other in the embedding space will end up in the same cluster). The typical spectral clustering algorithm follows Alg. 1.

Much has been written about this elegant, simple algorithm in the last quarter of century, the majority working on improving this algorithm on two main challenges: making it computationally efficient and obtaining theoretical guarantees on the cluster reconstruction.

Algorithm 1 A typical spectral clustering algorithm on a graph $\mathcal{G}$ with k classes

1: **Input** : $\mathcal{G}$, k
2: Choose the matrix $\mathsf{R} \in \mathbb{R}^{n \times n}$, a representative matrix of the graph
3: Find the k largest (or smallest, depending on the choice of R) eigenvalues of R and stack the associated eigenvectors $\{\boldsymbol{u}_\ell\}_{\ell=1,\dots,k}$ in the columns of a matrix $\mathsf{X} \in \mathbb{R}^{n \times k}$
4: (*Optional*) Normalize the rows of X
5: Estimate community labels as the output of some small dimensional clustering method (such as *k-means* or *expectation maximization*) performed on the points in $\mathbb{R}^k$ defined by the rows of X
6: **return** Estimated label community vector $\boldsymbol{\ell}$.

2.1.3.1 Accelerating spectral clustering.

Given a graph $\mathcal{G}$ consisting of n nodes and $|\mathcal{E}|$ edges, the first step is to compute the spectral embedding X (line 2 of Alg. 1). Among several options, the Lanczos method is usually favored: it approximates the first k eigenvectors in roughly $\mathcal{O}(|\mathcal{E}|k + nk^2)$ time. This procedure is often numerically unstable resulting to a loss of orthogonality of the computed Krylov subspace basis. The most common way to circumvent this problem is by implicit restart [Calvetti *et al.* 1994], whose computational complexity is not easily derived. We refer the interested reader to [Bai *et al.* 2000] for an in-depth perspective. The second step (line 5 of Alg. 1) entails solving the k-means problem, typically by using the Lloyd-Max algorithm [Lloyd 1982]. Since there is no guarantee that this heuristic will find a good local minimum, it is usually rerun multiple times, starting in each case from randomly selected centroids. Overall, the computational complexity of this step is $\mathcal{O}(trnk^2)$, where t is a bound on the number of iterations required until convergence and r is the number of retries (typically around 10).

Overall, spectral clustering thus runs in $\mathcal{O}(|\mathcal{E}|k + nk^2)$, with multiplicative constants that strongly depend on the particular spectrum of R (and especially eigengaps[2]), and whether the embedding actually has well formed clusters.

Many works have been devoted to accelerate this computation time and one may refer to my book chapter reviewing many different sampling methods to do so [Tremblay & Loukas 2020], between very efficient methods that do not have strong approximation guarantees and very accurate approximate methods that are not much more efficient than the original spectral clustering algorithm. Later on in this manuscript, in Section 2.3, I will detail one of my contributions to provably accelerate spectral clustering – based on GSP tools.

2.1.3.2 A few theoretical guarantees on the performance of spectral clustering.

A possible way of formalizing spectral clustering consists in defining the class labels as the solution of an optimization problem such as `MinCut`, `RatioCut`, and `NormalizedCut`. In fact, it has been shown that the spectral clustering algorithm (using R equal to different types of Laplacian matrices) can be understood as a relaxation of these traditional NP-hard `Cut` problems [von Luxburg 2007]. Another classical theoretical result on spectral clustering is its consistency [Lei & Rinaldo 2015, Von Luxburg *et al.* 2008]. In the following, we will discuss more in detail yet another classical result of spectral clustering: when applied to graphs generated from the Stochastic Block Model (a random graph model with planted community structure), spectral clustering can recover the planted communities all the way down to the theoretical detectability threshold.

[2] for instance: the smaller is the eigengap $\lambda_{k+1} - \lambda_k$, the longer will Lanczos take to compute X

The Stochastic Block Model. A famous approach to community detection proposes to consider it as a statistical inference problem. The graph $\mathcal{G}$ is seen as a realization of a random graph model in which the class-label assignment is encoded by some hidden parameters of the generative model that have to be inferred. One such very common model is the *Stochastic Block Model* (SBM) [Karrer & Newman 2011]: all edges are drawn independently according to a Bernoulli random variable verifying

$$\mathbb{P}(\mathsf{A}_{ij} = 1 | \ell_i, \ell_j) = \frac{\mathsf{C}_{\ell_i,\ell_j}}{n} \quad 1 \leq i < j \leq n \tag{2.2}$$

in which $\boldsymbol{\ell} = \{1, \cdots, k\}^n$ denotes the node-wise labelling vector ($\ell_i = p$ if node i is in class p). We denote with $\mathsf{C} \in \mathbb{R}^{k \times k}$ the symmetric class-affinity matrix, with entries independent of n. The term $1/n$ in Eq. (2.2) bounds the average degree $\bar{d} = \frac{1}{n}\mathbf{1}_n^T \mathsf{A} \mathbf{1}_n = \mathcal{O}(1)$ to an n-independent value, setting the problem in the sparse regime. I in fact directly set the context in the sparse regime as this is the regime where it is difficult to show that spectral clustering works. In denser regimes, that is, in cases where the average degree is larger than $\mathcal{O}(\log n)$, random matrix theory tools show that spectral clustering works on the SBM under mild conditions, and using many classical representative matrices R, such as A and L (see for instance [Lei & Rinaldo 2015]). Note that, in this context, stating that "spectral clustering works" means that the correlation between the true hidden labels and the output of the algorithm is in expectation bounded away from zero, as n tends to infinity.

Existence of a detectability threshold in the sparse regime A major advantage of a well defined generative model is that it can come with statistically tractable properties, like the existence in asymptotically large graphs ($n \to \infty$) of a limiting *detectability threshold* in the sparse regime. Specifically, one can in general identify a parameter α, function of the entries of C and the number of classes k and their size, and a threshold α_c such that

- beyond the threshold (that is, when $\alpha > \alpha_c$), partial label reconstruction can be theoretically achieved
- whereas below the threshold ($\alpha < \alpha_c$) no algorithm can perform better than random guess.

In the very specific setting where $k = 2$ and both classes are of equal size ($n/2$), the detectability threshold assumes a simple expression. Letting $\mathsf{C}_{\ell_i,\ell_j} = c_{\text{in}}$ if $\ell_i = \ell_j$ and c_{out} otherwise, in the SBM, it was initially conjectured [Decelle *et al.* 2011] and later proved [Mossel *et al.* 2017, Massoulié 2014] that the condition for non-trivial clustering is given by

$$\alpha \hat{=} \frac{c_{\text{in}} - c_{\text{out}}}{\sqrt{c}} \quad > \quad 2 \hat{=} \alpha_c, \tag{2.3}$$

where $c = (c_{\text{in}} + c_{\text{out}})/2$.

For $k > 2$ classes, the situation is more involved: it is conjectured [Decelle *et al.* 2011] that there exists an *easy* detection zone in which a non-trivial clustering can be

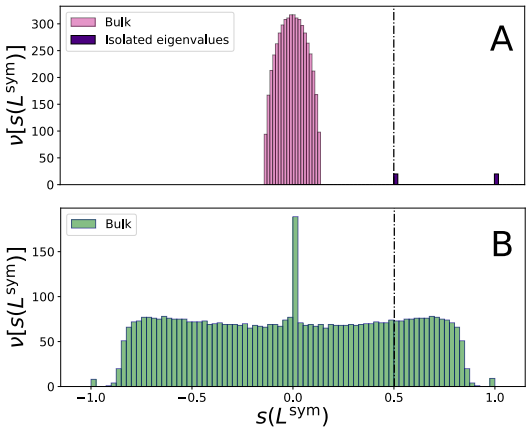

Figure 2.3: For both plots: $n = 5000$, $k = 2$. Spectrum of the matrix $\mathsf{R} = \mathsf{D}^{-\frac{1}{2}}\mathsf{A}\mathsf{D}^{-\frac{1}{2}}$ in (A) the dense regime ($c_{\text{in}}/n = 0.06$, $c_{\text{out}}/n = 0.02$); B) the sparse regime ($c_{\text{in}} = 6$, $c_{\text{out}} = 2$).

found in polynomial time, a *hard* detection zone in which non-trivial clustering can be found but only in exponential time and an *impossible* detection zone in which no algorithm can find a non-trivial partition.

Reaching the threshold with a well-chosen R matrix. Sparsity makes it difficult to theoretically predict the behavior and performance of Algorithm 1 and to choose a proper matrix R to work with. From a linear algebra viewpoint, sparsity has been shown to spread the eigenvalues of the Laplacian matrix, with the deleterious effect to "swallow" the isolated (smallest or largest) informative eigenvalues within the so-called "bulk" of uninformative eigenvalues: as a result, the corresponding informative eigenvectors are no longer found to be associated with dominant (largest or smallest) eigenvalues, as shown in Figure 2.3B, and as opposed to Figure 2.3A. Besides, by losing the isolation of informative eigenvalues, the associated eigenvectors tend to merge with the eigenvectors associated to close-by (non-informative) eigenvalues.

One prominent line of research was devoted to defining spectral algorithms capable of retrieving communities as soon as theoretically possible ($\alpha > \alpha_c$ in Equation (2.3)) on sparse graphs generated from the SBM. The authors of [Krzakala *et al.* 2013] proposed an algorithm based on the (non-symmetric) non-backtracking matrix

$$\mathsf{B}_{(ij)(lm)} = \delta_{jl}(1 - \delta_{im}),$$

with $\mathsf{B} \in \{0,1\}^{2|\mathcal{E}|\times 2|\mathcal{E}|}$. In [Massoulié 2014] it was indeed shown that the eigenvectors associated to the largest (in modulus) eigenvalues of B have a non-trivial correlation with the actual underlying communities, as soon as $\alpha > \alpha_c$. A closely related algorithm is the one proposed in [Saade *et al.* 2014] which instead uses the eigenvectors attached to the smallest eigenvalues of the Bethe-Hessian matrix

$$\mathsf{H}_r = (r^2 - 1)\mathsf{I}_n + \mathsf{D} - r\mathsf{A} \in \mathbb{R}^{n\times n}, \tag{2.4}$$

for $r = \sqrt{\rho(B)}$ ($\rho(\cdot)$ refers to the spectral radius). Very similar propositions have been suggested in other works [Amini *et al.* 2013, Joseph & Yu 2016, Le *et al.* 2018, Qin & Rohe 2013] where they argue that one should use regularized or so-called "deformed" Laplacians (*i.e.*, using $\mathsf{D}_\tau = \mathsf{D} + \tau \mathsf{I}_n$ and $\mathsf{A}_\tau = \mathsf{A} + \tau \mathbf{1}_n \mathbf{1}_n^T$ in place of D and A) as a solution to maintain the location of the informative eigenvalues in their dominant (smallest or largest) positions.

I will, in Section 2.4, detail a contribution of mine that extends these works to a larger class of random graphs, the so-called Degree Corrected Stochatsic Block Models (DCSBMs): models that are able to generate random community-structured graphs (similarly to the SBM) but with arbitrary degree distributions. We will see that the Bethe-Hessian and other regularized Laplacians perform well up to the threshold provided that they are well-parametrized (parameter r for the Bethe-Hessian, parameter τ for the regularized Laplacians).

2.1.4 The link between graph filters and spectral clustering

At first sight, graph filters and spectral clustering may seem to be very different subjects and one may wonder why they end up in the same chapter of this document. The reason is that properties of graph signals and graph structures are intimately linked. In Fig. 2.4, one sees for instance how the graph Fourier transform of the same signal can change as the underlying structure changes. This intimate link between signal and structure has been driving my scientific curiosity since my PhD and is at the heart of this chapter. In fact, the contributions I present here are precisely investigating this link:

- in Section 2.2, I show how one can use detection of graph structures to define multiscale graph transforms
- in Section 2.3, I show how one can use graph signal sampling and filtering to detect graph structures
- finally, in Section 2.4, I show how one should ideally choose the representative matrix R (via a sort of "graph structure regularization") to have meaningful low frequency modes (thus lending themselves naturally for spectral clustering) when the graph becomes too sparse and its degree distribution too heterogeneous for the usual options to work well

2.2 Graph structure detection for graph signal processing: a contribution to graph filterbanks

Section 2.2.1 provides a bit of context on graph filterbanks. For readers who have never heard of classical filterbanks, the reading of the first few chapters of the book by Strang and Nguyen [Strang & Nguyen 1996] is probably a pre-requisite. The next section then details my contribution in this domain.

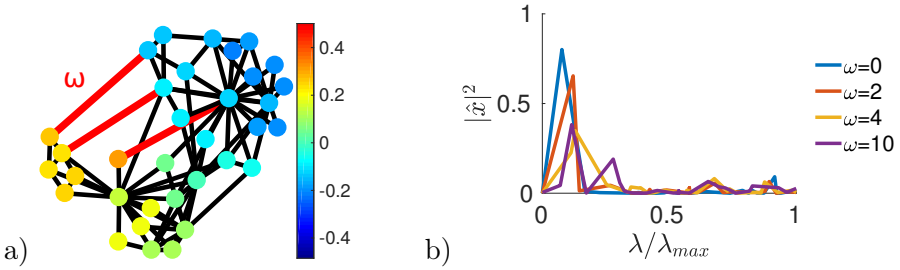

Figure 2.4: Plot a) represents the same low-frequency graph signal as in Fig. 2.1-a), but the underlying graph structure is altered by adding three edges with same weight ω (in red). Plot b) represents the variation of its GFT with respect to the edges' weight ω.

2.2.1 Preliminaries on graph filterbanks

A generic way of constructing multiscale transforms is via a succession, as depicted in Fig. 2.5, of filtering and decimation operations (traditionally, for a time series for instance, decimation consists in keeping only one every two values of the signal). This scheme is usually cascaded as in Fig. 2.6, and each level of the cascade represents a scale of description of the input signal. Again, for an introduction to this particular approach to multiscale transforms in classical signal processing, we refer to [Strang & Nguyen 1996]. In the following, we directly consider the graph-based context. Let us first settle notations:

- The decimation operator may be generally defined by partitioning the set of nodes $\mathcal{V}$ into two sets $\mathcal{V}_0$ and $\mathcal{V}_1$. As this subdivision is a partition, we have $\mathcal{V}_0 \cup \mathcal{V}_1 = \mathcal{V}$ and $\mathcal{V}_0 \cap \mathcal{V}_1 = \emptyset$. Moreover, let us define $\downarrow_{\mathcal{V}_i}$ the downsampling operator associated with $\mathcal{V}_i$: given any graph signal $\mathbf{x}$, $\mathbf{y}_i = \downarrow_{\mathcal{V}_i} \mathbf{x}$ is the reduction of $\mathbf{x}$ to $\mathcal{V}_i$. We also define the upsampling operator $\uparrow_{\mathcal{V}_i} = (\downarrow_{\mathcal{V}_i})^\top$: given $\mathbf{y}_i$ a signal defined on $\mathcal{V}_i$, $\uparrow_{\mathcal{V}_i} \mathbf{y}_i$ is the zero-padded version of $\mathbf{y}_i$ on the whole graph. The combination of both operators reads: $\uparrow_{\mathcal{V}_i}\downarrow_{\mathcal{V}_i} = \mathrm{diag}(\mathcal{I}_{\mathcal{V}_i})$, where $\mathcal{I}_{\mathcal{V}_i}$ is the indicator function of $\mathcal{V}_i$. Moreover, we define

$$\mathbf{J} = 2 \uparrow_{\mathcal{V}_0}\downarrow_{\mathcal{V}_0} - \mathbf{I} = \mathrm{diag}(\mathcal{I}_{\mathcal{V}_0}) - \mathrm{diag}(\mathcal{I}_{\mathcal{V}_1}). \tag{2.5}$$

- We define two analysis filters: a low-pass graph filter $\mathbf{H}_0$ and a high-pass graph filter $\mathbf{H}_1$, as well as two synthesis graph filters $\mathbf{G}_0$ and $\mathbf{G}_1$. All filters are associated with their frequency responses $h_0(\lambda)$, $h_1(\lambda)$, $g_0(\lambda)$ and $g_1(\lambda)$.

The signal $\mathbf{y}_0 = \downarrow_{\mathcal{V}_0} \mathbf{H}_0\mathbf{x}$ is called the approximation of $\mathbf{x}$, whereas $\mathbf{y}_1 = \downarrow_{\mathcal{V}_1} \mathbf{H}_1\mathbf{x}$ is generally understood as the necessary high-frequency details to recover $\mathbf{x}$ from its approximation.

Given the scheme of Fig. 2.5, one writes the processed signal $\tilde{\mathbf{x}}$ as:

$$\tilde{\mathbf{x}} = \left(\mathbf{G}_0 \uparrow_{\mathcal{V}_0}\downarrow_{\mathcal{V}_0} \mathbf{H}_0 + \mathbf{G}_1 \uparrow_{\mathcal{V}_1}\downarrow_{\mathcal{V}_1} \mathbf{H}_1\right)\mathbf{x} \tag{2.6}$$

$$= \frac{1}{2}\left(\mathbf{G}_0\mathbf{H}_0 + \mathbf{G}_1\mathbf{H}_1\right)\mathbf{x} + \frac{1}{2}\left(\mathbf{G}_0\mathbf{J}\mathbf{H}_0 - \mathbf{G}_1\mathbf{J}\mathbf{H}_1\right)\mathbf{x}. \tag{2.7}$$

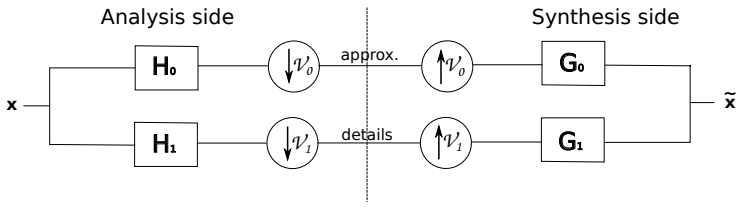

Figure 2.5: A filterbank seen as a succession of filtering and decimating operators.

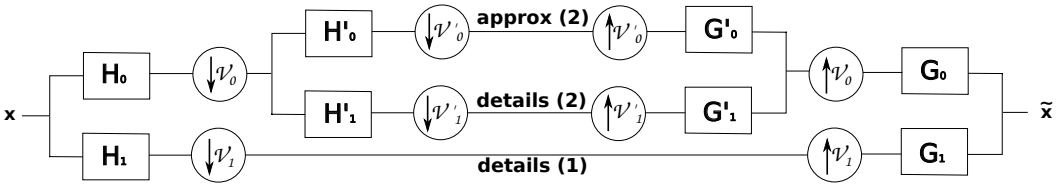

Figure 2.6: A cascaded filterbank (here two levels).

When designing such filterbanks, and in order to enable perfect reconstruction ($\forall \mathbf{x} \in \mathbb{R}^N \quad \tilde{\mathbf{x}} = \mathbf{x}$), one deals with two main equations linking all four filters and matrix $\mathbf{J}$:

$$\mathbf{G}_0\mathbf{H}_0 + \mathbf{G}_1\mathbf{H}_1 = 2\mathbf{I} \tag{2.8}$$

$$\mathbf{G}_0\mathbf{J}\mathbf{H}_0 - \mathbf{G}_1\mathbf{J}\mathbf{H}_1 = \mathbf{0} \tag{2.9}$$

Left-multiplying by $\mathbf{U}^\top$ and right-multiplying by $\mathbf{U}$, one obtains equivalently:

$$g_0(\mathbf{\Lambda})h_0(\mathbf{\Lambda}) + g_1(\mathbf{\Lambda})h_1(\mathbf{\Lambda}) = 2\mathbf{I} \tag{2.10}$$

$$g_0(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{J}\mathbf{U}h_0(\mathbf{\Lambda}) - g_1(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{J}\mathbf{U}h_1(\mathbf{\Lambda}) = \mathbf{0} \tag{2.11}$$

Eq.(2.10) is purely spectral, and may be seen as a set of N equations:

$$\forall \lambda_i \quad g_0(\lambda_i)h_0(\lambda_i) + g_1(\lambda_i)h_1(\lambda_i) = 2. \tag{2.12}$$

On the other hand, Eq. (2.11) is not so simple due to the decimation operation and needs to be investigated in detail.

Filterbanks on the circle or the tore graph. In 1D (resp 2D) classical signal processing, which is equivalent to the undirected circle graph (resp. tore graph), the classical choice for the decimation operator is to sample one every two nodes. Moreover, given $\mathbf{x}' = \uparrow_2\downarrow_2 \mathbf{x}$, one classically has the following aliasing phenomenon (Theorem 3.3 of [Strang & Nguyen 1996]), where $\mathbb{F}$ denotes the Fourier transform:

$$\mathbb{F}[\mathbf{x}'](\omega) = \frac{1}{2}\left(\mathbb{F}[\mathbf{x}](\omega) + \mathbb{F}[\mathbf{x}](\omega + \pi)\right). \tag{2.13}$$

This means that the decimation operations may be explicitly described in the Fourier space, which greatly simplifies calculations by enabling to write Eq. (2.11) as a purely spectral equation as well. The hardest part is done and now one only needs to decide

the forms of the filters g and h depending on the task at hand.

Filterbanks on bipartite graphs. Bipartite graphs are graphs where the nodes are partitioned in two sets of nodes $\mathcal{A}$ and $\mathcal{B}$ such that all links of the graph connect a node in $\mathcal{A}$ with a node in $\mathcal{B}$. On bipartite graphs, the "one-every-two-node" paradigm has a natural extension: decimation ensembles are set to $\mathcal{V}_0 = \mathcal{A}$ and $\mathcal{V}_1 = \mathcal{B}$. Leveraging the fact that bipartite graphs' spectra are symmetrical[3] around the value 1, Narang and Ortega [Narang & Ortega 2012] show the bipartite graph spectral folding phenomenon:

$$\forall \lambda \quad \mathbf{Pr}_\lambda \mathbf{J} = \mathbf{J}\mathbf{Pr}_{2-\lambda}, \tag{2.14}$$

with $\mathbf{Pr}_\lambda$ the eigenspace associated to λ. This means that for any filter $\mathbf{G} = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^\top$ one has:

$$\mathbf{GJ} = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^\top\mathbf{J} = \mathbf{J}\mathbf{U}g(2\mathbf{I} - \mathbf{\Lambda})\mathbf{U}^\top. \tag{2.15}$$

Eq. (2.11) therefore boils down to a second set of n purely spectral equations:

$$\forall \lambda_i \quad g_0(2-\lambda_i)h_0(\lambda_i) - g_1(2-\lambda_i)h_1(\lambda_i) = 0. \tag{2.16}$$

Again, the hardest part is done: together, Eqs. (2.12) and (2.16) give us $2n$ equations linking the $4n$ parameters of the four filters to ensure perfect reconstruction. The other $2n$ degrees of liberty are free to be used to adapt the filterbanks to the specific task at hand.

Filterbanks on other regular graphs. Extending these ideas, several authors have proposed similar approaches to define filterbanks on other regular structures such as M-block cyclic graphs [Teke & Vaidyanathan 2017] or circulant graphs [Ekambaram *et al.* 2013, Kotzagiannidis & Dragotti 2016]. In all these cases, the key is to be able to write decimation operations exactly as graph filters, which requires regular graph structures inducing at least some regularity in the spectrum one may take advantage of. Now, the central question that remains is how to mimic these filtering/decimation approaches on arbitrary graphs that have no trivial regularities?

Filterbanks on arbitrary graphs. In order to extend the filterbanks approach to arbitrary graphs, one needs to either generalize the decimation operator, or to bypass decimation via aggregation operators. For that, the two key questions are:

- how to generalize the decimation operator on arbitrary graphs? Generalized decimation operators of the literature either try to mimic the classical decimation and attempt to sample "one every two nodes", or aggregate nodes to form supernodes according in general to some graph cut objective function.

[3] with the normalized Laplacian $\mathcal{L} = \mathsf{D}^{-\frac{1}{2}}\mathsf{L}\mathsf{D}^{-\frac{1}{2}}$: if λ is an eigenvalue of $\mathcal{L}$, then so is $2-\lambda$

- how to build the new coarser-scale graph from the decimated nodes (or aggregated supernodes)? In fact, after each decimation, if one wants to cascade the filterbank, a new coarse-grain graph has to be built in order to define the next level's graph filters. The nodes (resp. supernodes) are set thanks to decimation (resp. aggregation): but how do we link them together?

For a detailed account on how the literature deals with these questions, I refer to my book chapter [Tremblay *et al.* 2018].

2.2.2 A Haar-like graph filterbank

The main idea I explore in this contribution lets go of the "one every two nodes" decimation paradigm, and concentrates on graph coarsening: given a partition in connected subgraphs of the initial graph, the approximation and detail(s) will be obtained on each "supernode" that represent each connected subgraph.

Hence we will not attempt to define separately analogies of downsampling ($\downarrow$) and filtering $\boldsymbol{H}_0$ and $\boldsymbol{H}_1$. Instead, we mimick Haar's filterbank that directly define decimated sliding average (and difference) operators.

The central building block is thus a partition $\mathcal{P}$ of the nodes $\mathcal{V}$ in connected subgraphs. It turns out that to obtain a filterbank that can stay sufficiently sensitive to high frequency details, one needs a partition in small subgraphs. The average size of these subgraphs is in fact a parameter of the method, that I will denote by η. Typically, we use the first (or second) round of Louvain's algorithm [Blondel *et al.* 2008] to obtain such a partition[4]. Louvain's algorithm is a greedy heuristic to optimize modularity [Newman & Girvan 2004] and has a multi-scale implementation. By looking at the result after its first round, we do not necessarily have a partition that has a very good modularity score, but we have – in a very efficient linear computation time – a partition in substructures that can be interpreted as small communities. Note that in this case using spectral clustering would be completely counterproductive as it is only efficient for $k \ll n$ and not for a number of communities of the order of n as is our case here!

Then, given this partition, the construction is straightforward –even though cumbersome to write formally, which I will not do here–:

- Instead of 2 channels as depicted in Fig. 2.5, the analysis has a number of channels equal to the maximum size of the detected communities

- The first channel –the approximation channel– outputs a graph signal defined on a coarsened graph with: i) each "supernode" of this coarsened graph represents one community of $\mathcal{P}$, ii) the weight of the link between two supernodes is the sum of the weights of the links connecting the associated communities, iii) the value of the approximation signal on a supernode of this coarsened graph is simply the average of the original signal over the community it represents.

[4]Instead of Louvain, one should now use the more modern and strictly better version called Leiden's algorithm [Traag 2015]

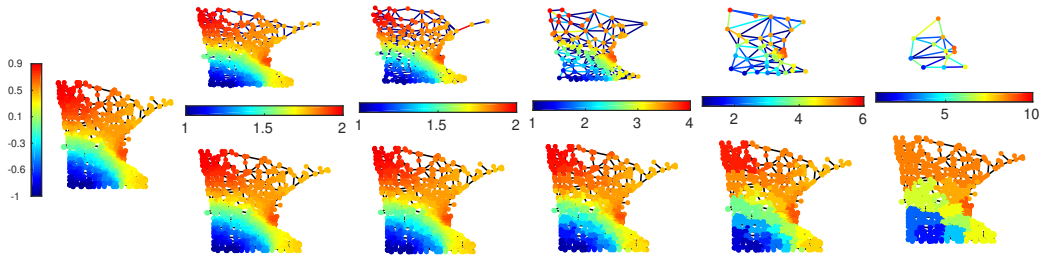

Figure 2.7: An example of multiresolution analyis of a graph signal. Left: original smooth graph signal (sum of the five lowest Fourier modes normalized by its maximum absolute value) defined on the Minnesota traffic graph. The vertical colorbar of this figure is valid for all graph signals represented on this figure. Top row: successive approximations of the graph signal. The horizontal colorbar on the bottom of each figure corresponds to the weights of the links of the corresponding coarsened graph. Figures which do not have a bottom horizontal colorbar represent binary graphs. Lower row: for each of the successive approximations, we represent the upsampled reconstructed graph signal obtained from the corresponding approximations.

- All the other channels encode the various details needed for reconstruction via the "local" graph Fourier transforms associated to each subgraph induced by each community. All these detail signals also live on coarsened structures.

Illustrations. I show in Fig. 2.7 an example of multiresolution analysis of a graph signal, via our proposed filterbank. In Fig. 2.8, we also show on a small toy graph the graph wavelets indirectly encoded by this multiscale transform.

2.2.3 What now?

The two major assets of this work are that

- it defines a graph multiscale transform that is very cheap to compute. It indeed costs only $\mathcal{O}(\eta^2 n \log(n))$ to compute it at all scales and $\mathcal{O}(n\eta)$ to apply to a graph signal, where we recall that η is the hyper-parameter giving the average size of subgraphs in partitions. This cost is to compare to the $\mathcal{O}(n^3)$ diagonalizing cost to obtain the graph Fourier modes and the $\mathcal{O}(n^2)$ cost to apply it to a graph signal. The proposed filterbanks can thus be deployed on very large graphs. This low cost is truly enabled by the efficiency of Louvain-like greedy algorithms to find relevant partitions in connected subgraphs.

- it is well-adapted to the multi-scale community structure of many networks. In fact, improvement over other existing methods becomes truly apparent for irregular graphs for which a community structure exists. Existence of communities is a very common, if not universal, property of real-world graphs; and our filterbanks rely on this particular organization of complex networks. For

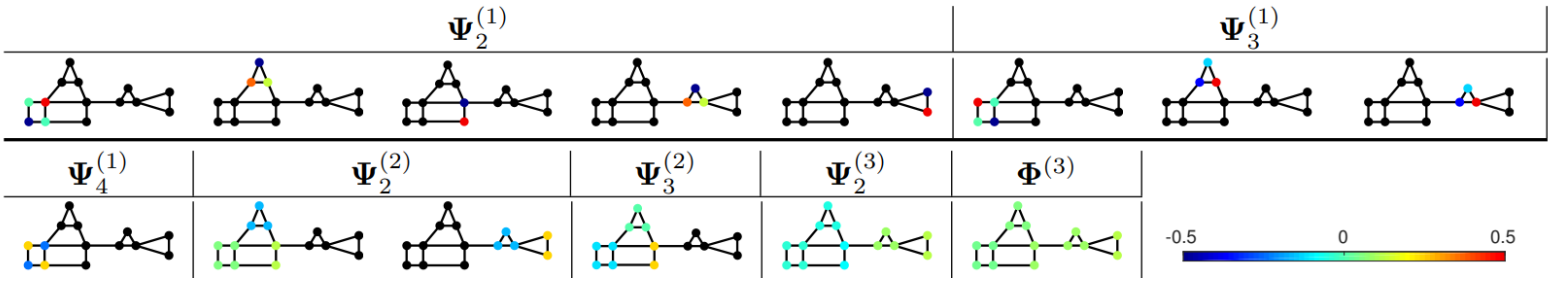

Figure 2.8: The 14 analysis atoms (or wavelets) of the proposed filterbank on a toy graph. From left to right, and top to bottom, are listed the 13 detail atoms, from small to large scales. The bottom right atom is the approximation atom (computes the average over the graph). A node's color represents the atom's value on that node, as indicated by the colorbar. A black node corresponds to a strictly null value: indeed, the obtained wavelets have compact-support.

such graphs, our proposition outperforms existing ones on non-linear approximation experiments and equals state-of-the art on denoising experiments.

I think that this framework has been under-used in the GSP community. A future direction I would like to pursue on this topic consists in investigating how this transform fares when integrated in GNN architectures. It provides yet a different framework than the two main frameworks of today's literature: the message passing framework and the spectral framework (based on the Laplacian's spectrum). Also, instead of separately defining message-passing layers and pooling (downsampling/coarsening) layers, we would define only one layer that would integrate both operations. The coarsening scale, η, as well as the parameters of a sort of generalized filter (yet to be defined properly) would then be the learnable parameters.

2.3 Graph signal processing for graph structure detection: a contribution to efficient spectral clustering

We have seen in the previous section how detecting graph structures can be useful to design meaningful and efficient graph signal transforms. We will see in this section how the converse is also true: how graph signal processing tools such as graph sampling and graph filtering can be useful to efficiently detect graph structures.

2.3.1 Preliminaries: graph sampling of bandlimited signals

In classical signal processing, the Nyquist-Shannon sampling theorem (see Theorem 1) famously states that a ω-bandlimited signal can be perfectly recovered from periodic samples, provided that the sampling frequency is at least 2ω. In the GSP literature, there has been a line of works investigating how this can be generalized to graph signals. Writing $\mathsf{U}_k = (\boldsymbol{u}_1|\dots|\boldsymbol{u}_k) \in \mathbb{R}^{n\times k}$ the matrix concatenating

the eigenvectors of L associated with its k smallest eigenvalues, we have the formal definition:

Definition 1 (k-bandlimited graph signals)**.** *A signal $\boldsymbol{x} \in \mathbb{R}^n$ defined on the nodes of a graph is k-bandlimited if $\boldsymbol{x} \in span(\mathsf{U}_k)$, i.e., $\exists\ \boldsymbol{\alpha} \in \mathbb{R}^k$ such that $\boldsymbol{x} = \mathsf{U}_k\boldsymbol{\alpha}$.*

In words, a k-bandlimited signal is thus defined as a linear combination of the k lowest frequency graph Fourier modes. As "periodic sampling" is ill-defined for arbitrary graphs, the question of generalizing Shannon's theorem is not straightforward and many different forms were proposed: see [di Lorenzo *et al.* 2018] for a review of existing schemes. They all try to answer the following question: how should one sample bandlimited graph signals in order to guarantee perfect reconstruction?

Let us formalize the problem and link it to the theory of optimal experimental design. Suppose $\boldsymbol{x}$ is k-bandlimited, *i.e.*, there exists $\boldsymbol{\alpha}$ such that $\boldsymbol{x} = \mathsf{U}_k\boldsymbol{\alpha}$. Now, to any sampling set $\mathcal{X}$ of size m we associate a sampling matrix $\mathsf{M} \in \mathbb{R}^{m\times n}$: it is designed to verify $\forall \boldsymbol{z} \in \mathbb{R}^n, \mathsf{M}\boldsymbol{z} = \boldsymbol{z}_{\mathcal{X}} \in \mathbb{R}^m$, the reduction of $\boldsymbol{z}$ to its entries indexed by $\mathcal{X}$. It also verifies $\forall \mathsf{A} \in \mathbb{R}^{n\times n}, \mathsf{MAM}^\top = \mathsf{A}_{\mathcal{X}} \in \mathbb{R}^{m\times m}$, the reduction of matrix A to rows and columns indexed by $\mathcal{X}$. The measurement of $\boldsymbol{x}$ on $\mathcal{X}$ reads:

$$\boldsymbol{y} = \mathsf{M}\boldsymbol{x} + \boldsymbol{n} = \mathsf{MU}_k\boldsymbol{\alpha} + \boldsymbol{n} \in \mathbb{R}^m, \tag{2.17}$$

where $\boldsymbol{n}$ models any kind of measurement noise.

The problem thus consists in choosing $\mathcal{X}$ of size m such that the measured signal $\boldsymbol{y}$ enables to perfectly (up to the noise level) recover $\boldsymbol{\alpha}$ for all $\boldsymbol{\alpha}$. In fact, recovering $\boldsymbol{\alpha}$ is equivalent to recovering $\boldsymbol{x}$. Of course, if $m < k$ then there is no hope as there are less measurements than degrees of freedom of $\boldsymbol{x}$. So we suppose $m \geq k$. One of the most straight-forward (and agnostic) strategy to recover $\boldsymbol{x}$ from $\boldsymbol{y}$ is to solve the least-square problem:

$$\begin{aligned}\boldsymbol{x}_{\text{rec}} &= \underset{\boldsymbol{z}\in\text{span}(\mathsf{U}_k)}{\arg\min} \|\mathsf{M}\boldsymbol{z} - \boldsymbol{y}\|^2 \\ &= \mathsf{U}_k\left(\mathsf{MU}_k\right)^\dagger \boldsymbol{y} \\ &= \mathsf{U}_k\left(\mathsf{MU}_k\right)^\dagger \mathsf{MU}_k\boldsymbol{\alpha} + \mathsf{U}_k\left(\mathsf{MU}_k\right)^\dagger \boldsymbol{n}.\end{aligned}$$

Now, all sets $\mathcal{X}$ such that the smallest singular value of MU_k is strictly positive enable perfect reconstruction up to the noise level. In fact, in this case, $\mathsf{U}_k^\top\mathsf{M}^\top\mathsf{MU}_k$ is invertible and the Moore-Penrose pseudo-inverse $(\mathsf{MU}_k)^\dagger$ has a closed form $(\mathsf{MU}_k)^\dagger = (\mathsf{U}_k^\top\mathsf{M}^\top\mathsf{MU}_k)^{-1}\mathsf{U}_k^\top\mathsf{M}^\top$ such that the least-square solution to recover $\boldsymbol{x}$ from its measurements reads:

$$\boldsymbol{x}_{\text{rec}} = \boldsymbol{x} + \mathsf{U}_k(\mathsf{U}_k^\top\mathsf{M}^\top\mathsf{MU}_k)^{-1}\mathsf{U}_k^\top\mathsf{M}^\top\boldsymbol{n}. \tag{2.18}$$

A first line of research investigates the tight-budget scenario where m is set to k. In this case, the constraint on the smallest singular value of MU_k is just an invertibility constraint on MU_k and Eq. (2.18) simplifies to

$$\boldsymbol{x}_{\text{rec}} = \boldsymbol{x} + \mathsf{U}_k\left(\mathsf{MU}_k\right)^{-1}\boldsymbol{n}.$$

The invertiblity constraint on $\mathsf{M}\mathsf{U}_k$ leaves in general much room to choose the best sample set $\mathcal{X}$ and several strategies are possible to find the best $\mathcal{X}$ depending on what criterion is preferred to deal with the distorted noise term $\mathsf{U}_k \left(\mathsf{M}\mathsf{U}_k\right)^{-1} \boldsymbol{n}$.

A classical option, called A-optimal design, is to find the subset that minimizes the mean square error (assuming $\mathbb{E}(\boldsymbol{n}\boldsymbol{n}^\top)$ is the identity) :

$$\mathcal{X}_A = \underset{\mathcal{X}\,\text{s.t.}\,|\mathcal{X}|=k}{\arg\min}\ \mathrm{Tr}\left[\left(\mathsf{M}\mathsf{U}_k\mathsf{U}_k^\top\mathsf{M}^\top\right)^{-1}\right] \tag{2.19}$$

where Tr is the trace operator. Another option, called D-optimal design, is to find the following set –which stems from an analysis where one supposes the noise Gaussian and looks for the set minimizing a differential entropy:

$$\mathcal{X}_D = \underset{\mathcal{X}\,\text{s.t.}\,|\mathcal{X}|=k}{\arg\max}\ \det\left(\mathsf{M}\mathsf{U}_k\mathsf{U}_k^\top\mathsf{M}^\top\right). \tag{2.20}$$

The list continues with other types of designs (E-optimal design, T-optimal design, *etc.*) depending on what one wishes to define as "optimal".

Now, computationally, all these optimization problems are NP-complete (given their combinatorial nature). A standard way of tackling them is via greedy approaches (see for instance the generic Alg. 1 in [Tremblay *et al.* 2017]). They typically run in time $\mathcal{O}(nk^3)$ and provide a local extremum. Those are the strategies developed in, *e.g.*, [Chen *et al.* 2015, Chamon & Ribeiro 2018, Tsitsvero *et al.* 2016].

2.3.2 GSP sampling tools to accelerate spectral clustering

Random iid sampling of bandlimited graph signals. A radically different strategy to sample subsets $\mathcal{X}$ to recover bandlimited graph signals is via random strategies. In the iid random sampling paradigm, m samples are drawn independently with replacement to obtain a random subset X. At each draw, the probability to sample node i is denoted by p_i. We have $\sum_i p_i = 1$ and write $\mathsf{P} = \mathrm{diag}(p)$. Under this sampling scheme, we showed that the following Restricted Isometry Property holds for the associated measurement matrix M [Puy *et al.* 2016].

Theorem 2 (Restricted Isometry Property)**.** *For any $\delta, \varepsilon \in (0,1)$, with probability at least $1-\delta$:*

$$(1-\varepsilon)\|\boldsymbol{x}\|_2^2 \leq \frac{1}{m}\|\mathsf{M}\mathsf{P}^{-1/2}x\|_2^2 \leq (1+\varepsilon)\|\boldsymbol{x}\|_2^2 \tag{2.21}$$

for all $\boldsymbol{x} \in$ span(U_k) provided that

$$m \geq \frac{3}{\varepsilon^2}(\nu_p^k)^2 \log\frac{2k}{\delta} \tag{2.22}$$

where ν_p^k is the so-called graph weighted coherence:

$$\nu_p^k = \max_i \left\{p_i^{-1/2}\|\mathsf{U}_k^\top\boldsymbol{\delta}_i\|_2\right\}. \tag{2.23}$$

This property is important as it says, in a nutshell, that any two bandlimited signals will be identifiable post-sampling provided the number of samples is large enough. The concept of large enough depends on $(\nu_p^k)^2$: a measure of the interplay between the probability distribution and the norms of the lines of U_k. In the uniform iid case where $\forall i \quad p_i = 1/n$, one has $(\nu_p^k)^2 = n \max_i \|\mathsf{U}_k^\top \boldsymbol{\delta}_i\|_2^2$ which stays under control only for very regular graph. The good news is that there exists an optimal sampling distribution that adapts to the graph at hand:

$$\forall i \qquad p_i^* = \frac{\|\mathsf{U}_k^\top \boldsymbol{\delta}_i\|_2^2}{k}.$$

In fact, in this case, $(\nu_{p^*}^k)^2$ matches its lower bound k and the necessary number of samples to embed all bandlimited signals drops to the order of $k \log k$. Note that p_i^* is called the leverage score of node i in the random numerical linear algebra literature [Drineas & Mahoney 2016].

Reconstruction of the sampled signal. Now, for reconstruction, we may use the unbiased decoder:

$$\boldsymbol{x}_{\text{rec}} = \underset{\boldsymbol{z} \in \text{span}(\mathsf{U}_k)}{\arg\min} \|\mathsf{P}_X^{-1/2}(\mathsf{M}\boldsymbol{z} - \boldsymbol{y})\|^2 \tag{2.24}$$

where $\boldsymbol{y} = \mathsf{M}\boldsymbol{x} + \boldsymbol{n}$ is the noisy measurement of the signal $\boldsymbol{x}$ on X and $\mathsf{P}_X^{-1/2} = \mathsf{M}\mathsf{P}^{-1/2}\mathsf{M}^\top$. We have the following reconstruction result [Puy *et al.* 2016]:

Theorem 3. *Let X be the i.i.d. nodes sampled with distribution p and M be its associated sampling matrix. Let $\varepsilon, \delta \in (0,1)$ and suppose that m satisfies Eq. (2.22). With probability at least $1-\delta$, the following holds for all signal $\boldsymbol{x} \in span(\mathsf{U}_k)$ and all noise $\mathsf{n} \in \mathbb{R}^m$. The solution $\boldsymbol{x}_{rec}$ of Eq. (2.24) verifies:*

$$\|\boldsymbol{x}_{rec} - \boldsymbol{x}\|_2 \leq \frac{2}{\sqrt{m(1-\varepsilon)}} \|\mathsf{P}_X^{-1/2}\boldsymbol{n}\|_2.$$

Remark 4. *The theory is easier to write for iid sampling with replacement. In practice, we will perform iid sampling without resampling (there is no point in measuring twice the same node!). In terms of sampling time, uniform sampling is obviously the most efficient by far as it is independent of n. However, if one wishes to take advantage of the superior performance of leverage score sampling, the cost of iid sampling with p^* is dominated by the computation of p^* itself: $\mathcal{O}(|\mathcal{E}|k + nk^2)$ to compute U_k (via Lanczos).*

Circumventing the exact computation of U_k. Exact knowledge of U_k is needed to compute the leverage score distribution p^*, and for the signal's reconstruction (Eq. (2.24)). In some cases however, the cost of computing U_k in the first place may be too expensive and different authors have been designing methods bypassing the exact computation of U_k (see, *e.g.*, [Anis *et al.* 2016]). In such cases, and in our context of iid random sampling, i) we have developed a method, based on random

projections, to estimate the leverage scores in time $\mathcal{O}(|\mathcal{E}| \log n)$. ii) Instead of the decoder of Eq. (2.24), we suggest to resort to graph signal interpolation reminiscent to graph Tikhonov denoising:

$$\boldsymbol{x}_{\text{rec}} = \underset{\boldsymbol{z} \in \mathbb{R}^n}{\arg\min} \quad q \, \|\mathsf{P}_X^{-1/2}(\mathsf{M}\boldsymbol{z} - \boldsymbol{y})\|^2 + \boldsymbol{z}^\top \mathsf{L} \boldsymbol{z} \tag{2.25}$$

with $q > 0$ the regularization parameter. In order to penalize even more high-frequencies while keeping the decoding step computationally fast, one can replace the regularization term $\boldsymbol{z}^\top \mathsf{L} \boldsymbol{z}$ by $\boldsymbol{z}^\top \mathsf{L}^2 \boldsymbol{z}$ or any low-order power of L.

Application to spectral clustering. So what's the link between bandlimited graph sampling and spectral clustering? Consider a graph with k disconnected components, and $\boldsymbol{i}_l \in \mathbb{R}^n$ the indicator vector of component l: $\boldsymbol{i}_l(j) = 1$ if j is in component l and 0 otherwise. It is a classical property of the Laplacian that $\{\boldsymbol{i}_1, \ldots, \boldsymbol{i}_k\}$ form a set of orthogonal eigenvectors of L associated to eigenvalue 0 [Chung 1997]: the set of indicator vectors $\boldsymbol{i}_l$ form a basis of span(U_k). Understanding arbitrary graphs with block structure as a perturbation of the ideal disconnected component case, we assume that the indicator vectors $\boldsymbol{i}_l$ of the blocks live close to span(U_k), *i.e.*: the difference between any $\boldsymbol{i}_l$ and its orthogonal projection onto span(U_k) is small. In other words: the information we are actually looking for in spectral clustering, the $\boldsymbol{i}_l$'s, are approximately bandlimited.

As a consequence, one can apply the previous results and only needs to: i) sample $X \subset \mathcal{V}$: $m = \mathcal{O}(k \log k))$ iid sampled nodes according to the approximate leverage score distribution, ii) cluster these m nodes in k classes, iii) lift the result back on the whole graph via the graph interpolation decoder of Eq. (2.25).

The last question that needs answering is how to perform step ii). There are many ways of doing this, but to stay in the spectral clustering spirit, we decided in this work to embed these m nodes in a r-dimensional space (let us denote by $\boldsymbol{f}_s \in \mathbb{R}^r$ the embedding of node $s \in X$ in that space) in which the distances between any pair of nodes s and s' is a good approximation of the distance between these nodes in the original spectral clustering framework, that is:

$$\|\boldsymbol{f}_s - \boldsymbol{f}_{s'}\|^2 \approx \|\mathsf{U}_k^\top \boldsymbol{\delta}_s - \mathsf{U}_k^\top \boldsymbol{\delta}_{s'}\|^2.$$

In fact, the k-means step of spectral clustering works on a distance matrix: it does not require U_k *per se*, but only the distance matrix of the lines in U_k. The trick is here to approximate directly this distance matrix, thus circumventing the computation of U_k. Now, how do we do we compute these embeddings $\boldsymbol{f}$ without computing U_k? The answer lies in two key ideas: polynomial approximation of graph filters and random projections. The full details of this approximation algorithm, along with the theoretical guarantees obtained via the Johnson-Lindenstrauss lemma are fully documented in our paper [Tremblay *et al.* 2016], and are out of the scope of this document.

In the end, combining random graph sampling, approximate graph filtering of random signals, and graph interpolation techniques, we obtain an algorithm –that

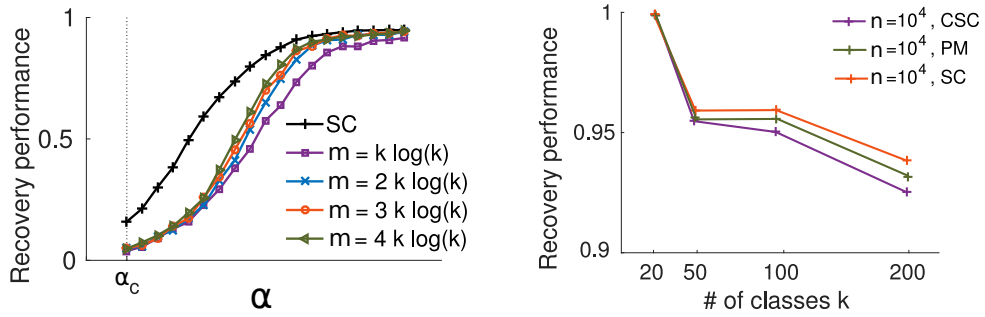

Figure 2.9: Left: recovery performance of CSC on a SBM with $n = 10^3$, $k = 20$ for different choices of number of samples m (results are averaged over 20 graph realisations); versus the difficulty parameter α discussed in Section 2.1.3.2. Right: recovery performance on a SBM with $n = 10^4$ nodes and diffferent values of k; for CSC (our approximation algorithm), PM (the Power Method approximation) and SC (the exact spectral clustering algorithm).

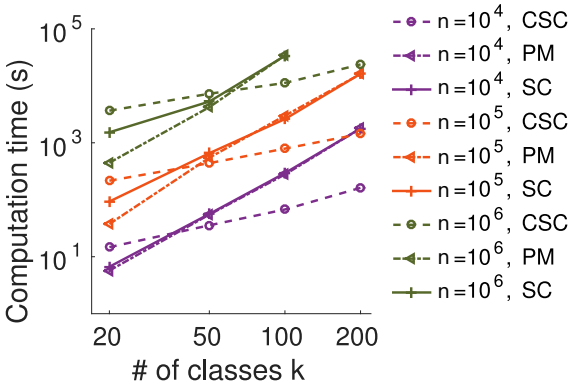

Figure 2.10: Computation time of CSC (our algorithm), PM (Power Method) and SC (spectral clustering) on a SBM with diffferent values of n and k. Log-log scale.

we call Compressive Spectral Clustering– that is a good approximation of spectral clustering, and that runs in time

$$\mathcal{O}\left(|\mathcal{E}|(k + \log n) + k^2 \log^2 k\right)$$

which makes a significant difference with the cost of spectral clustering –which we recall is $\mathcal{O}(|\mathcal{E}|k + nk^2)$– especially as k increases. Theoretically, we have some guarantees for parts of the algorithm, but we are far from an end-to-end guarantee that could certify how far we are in terms of approximation.

Empirically, the obtained performance is satisfying. In a nutshell: compared to the state-of-the-art approximation technique based on the power method (PM) [Boutsidis *et al.* 2015], our algorithm is in general a bit less accurate but much more efficient (by orders of magnitude for large k). We show in Fig. 2.9 (left) how the number of samples m affects the approximation on a SBM. On the right of the same figure, we show

performance results compared with SC (spectral clustering) and the PM method. Fig. 2.10 illustrates how our algorithm (CSC) is orders of magnitude faster than SC or PM, as the number of classes increase.

2.3.3 What now?

An annoying thorn in the side of this method is that we spend a very long time estimating the value of the k-th eigenvalue of the Laplacian matrix, λ_k (it is in fact the only spectral information we need). We do this by dichotomy on the whole spectrum and via approximate eigencounting techniques. It is unreasonable to spend so much time on this step. We however need it to properly parametrize low-pass filtering approximations and thus by-pass the full computation of U_k. A connection with Boutsidis' power method approximation [Boutsidis *et al.* 2015] should be envisioned; or a specific work direction should focus on how to estimate as fast and accurately as possible the k-th eigenvalue of a PSD matrix.

2.4 At the heart of graph signal processing and structure detection: choosing the best graph representation matrix

*The work presented in this section is the work of Lorenzo Dall'Amico, my first PhD student, that I co-advised (*50%*) with Romain Couillet (the other* 50%*). Lorenzo defended his thesis in October 2021.*

2.4.1 Motivation

My original motivation in this project was actually a GSP motivation: how should one choose the representative matrix R to have the most meaningful graph Fourier transform? This is a very broad and vague question and can be answered in various ways.

One possible way of answering this question is via very simple questions on statistical models. Let us consider again a SBM with two equal-size classes, and consider its first two Fourier modes $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$. First of all, it is evident that a good matrix R should yield a constant (or at least a constant sign) signal for $\boldsymbol{u}_1$ in order to have a proper sense of the average of the signal $\boldsymbol{u}_1^\top \boldsymbol{x}$. Now, what should $\boldsymbol{u}_2$ look like? In order to be coherent with a Fourier interpretation, $\boldsymbol{u}_2$ should show a very slow oscillation along the graph. In fact, I argue that the sign of the entries of $\boldsymbol{u}_2$ should be constant within each community of the graph, and the two communities should have opposite signs (as $\boldsymbol{u}_1^\top \boldsymbol{u}_2$ should be equal to zero). Although this fact has been observed for the choice $\mathsf{R} = \mathsf{L}$ as early as 1973 in [Fiedler 1973], and is in fact one of the most classical heuristic argument justifying spectral clustering, it has however been shown (as recalled in Section 2.1.3.2) that in sparse SBM graphs the choice $\mathsf{R} = \mathsf{L}$ fails to consistently provide a $\boldsymbol{u}_2$ that is correlated with the 2-class structure.

In other words, GSP based on $\mathsf{R} = \mathsf{L}$, at least on the SBM in the sparse regime and in the context of weakly structured communities, fails to provide graph Fourier modes that verify a basic "oscillatory" intuition.

The problem of course is that a vast majority of "natural graphs" (graphs modeling networks such as transportation networks, brain neuron networks, social networks, *etc.*) on which GSP tools are deployed, are in fact sparse – and their are very few safeguards out there to be certain that choosing classical options such as $\mathsf{R} = \mathsf{L}$ or $\mathsf{R} = \mathsf{A}$ is in fact a good idea.

And this is unfortunately not the only problem. Heterogeneity in node degrees is also another fundamental element blurring the first few eigenvectors of R. To study the effect of degree heterogeneity on the first graph Fourier modes, one needs to go beyond the SBM. In fact, SBM graphs have a very homogeneous degree distribution: it is a mixture (with k modes) of multinomial distributions. There is no chance with such graphs to model broad distribution such as power law distributions or other heavy tailed distributions *very* often observed in real graphs. To enrich the SBM framework, statisticians have come up with the following *Degree Corrected Stochatsic Block Model* (DCSBM) [Karrer & Newman 2011]. Similarly to the SBM, it is a model that draws each edge independently from a Bernoulli. It follows:

$$\mathbb{P}(\mathsf{A}_{ij} = 1 | \ell_i, \ell_j, \theta_i, \theta_j) = \theta_i \theta_j \frac{\mathsf{C}_{\ell_i, \ell_j}}{n} \quad 1 \leq i < j \leq n \tag{2.26}$$

in which $\boldsymbol{\theta}$ is the vector of intrinsic "sociability" (the larger θ_i, the larger its expected degree) which are used to produce an arbitrary degree distribution and are independent of the label vector $\boldsymbol{\ell}$. The entries of $\boldsymbol{\theta}$ are positive independent random variables $\theta_i \in [\theta_{\min}, \theta_{\max}]$, with $\theta_{\min} > 0$ and $\theta_{\max}$ suitably upper-bounded in order for $\theta_i \theta_j \mathsf{C}_{\ell_i, \ell_j}/n$ to never exceed 1. The entries of $\boldsymbol{\theta}$ have first moment $\mathbb{E}[\theta_i] = 1$ and second moment $\mathbb{E}[\theta_i^2] = \Phi$. Setting $\boldsymbol{\theta} = \mathbf{1}$, one recovers the SBM.

When one chooses $\mathsf{R} = \mathsf{L}$, degree heterogeneity induces a severe negative effect on the low frequency eigenvectors: it modifies the amplitude of the i-th entry (for all $1 \leq i \leq n$) of the low-frequency eigenvectors by a non-trivial function of the degree of node i. In the case of $k = 2$ classes of equal size, we illustrate this effect in Figure 2.11.

Objective Upon this observation that, in sparse heterogeneous graphs, classical choices of $\mathsf{R} = \mathsf{L}$ or $\mathsf{R} = \mathsf{A}$ do not consistently yield low-frequency Fourier modes that are correlated with the underlying community structure, the question is: how should we choose R? Translating the question in terms of spectral clustering, how should we choose R such that the correlation between the class structure and R's first k eigenvectors is the largest?

Results achieved, in a nutshell. We abandon the original spectral clustering setup that computes the k most extremal eigenvalues of a *unique* matrix R. We rather build the spectral embedding as a set of eigenvectors computed from k different matrices (one eigenvector per matrix is computed). These matrices have all the

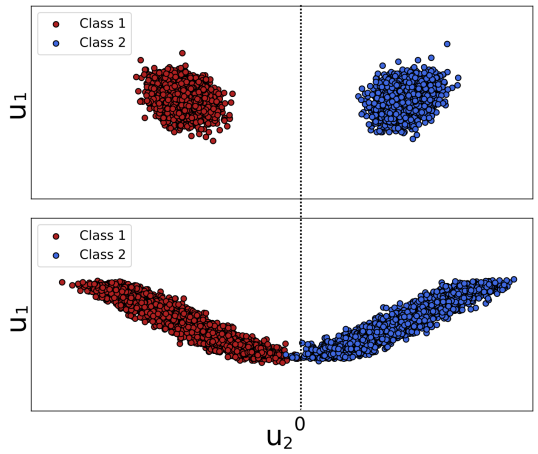

Figure 2.11: For both plots: DCSBM graphs with $n = 5000$ and $k = 2$ classes of equal size. First eigenvector $\boldsymbol{u}_1$ vs second eigenvector $\boldsymbol{u}_2$ of $\mathsf{R} = \mathsf{D}^{-\frac{1}{2}}\mathsf{A}\mathsf{D}^{-\frac{1}{2}}$ in the dense regime for: (top) the homogeneous case $\boldsymbol{\theta} = \mathbf{1}_n$; (bottom) the heterogeneous case $\theta_i \sim [\mathcal{U}(3, 10)]^3$ where $\mathcal{U}(3, 10)$ is the uniform distribution between 3 and 10. Colors represent ground truth classes.

same structure: they are Bethe-Hessian matrices H_r (see Eq. (2.4)), for k different choices of parameter r. We show how should these parameters be chosen optimally in the DCSBM context and end up with an algorithm that can be applied to arbitrary graphs and that outperforms the state-of-the-art on numerous examples.

2.4.2 State-of-the-art

As we have already discussed in the background section 2.1.3.2, sparse SBMs have a detectability threshold α_c after which detection is possible, and before which it is impossible. Also it has been shown that the first k eigenvectors of the Bethe-Hessian matrix $\mathsf{H}_r = (r^2 - 1)\mathsf{I}_n + \mathsf{D} - r\mathsf{A}$ for $r = \sqrt{\rho(\mathsf{B})}$ where $\rho(\mathsf{B})$ is the spectral radius of the non-backtracking matrix, achieve a non-trivial correlation with the block structure down to the threshold [Saade 2016].

Since these results appeared more or less a decade ago, they have been generalized to DCSBMs. The detectability condition of Equation (2.3) was generalized in [Gulikers *et al.* 2018] to DCSBMs, with an updated threshold $\alpha_c = 2/\sqrt{\Phi}$ (we recall that $\Phi = \mathbb{E}[\theta_i^2]$). Later, in [Gulikers *et al.* 2016], the authors show that spectral clustering on B also works down to this threshold.

These results are powerful as they propose algorithms capable of reaching the information-theoretic threshold, but they also have inherent weaknesses as they only guarantee a positive correlation of their output classification with the underlying true structure: they do not try to *maximize* this correlation, which, in a nutshell, is the goal of our contribution.

2.4.3 An optimal parametrization of the Bethe-Hessian for sparse heterogeneous graphs

A first look at our algorithm. Taking a pragmatic approach, let us start by looking at the spectral clustering algorithm we end up with (Algorithm 2). Several comments are in order:

- k, the number of classes, is no longer required as an input: we in fact come up with an automatic way of performing this model selection. It simply consists in counting the number of negative eigenvalues of $\mathsf{H}_{\sqrt{\rho(\mathsf{B})}}$. This is not novel *per se* as $\mathsf{H}_{\sqrt{\rho(\mathsf{B})}}$ is in fact the matrix that authors in [Saade *et al.* 2014] suggest to use as a matrix R for spectral clustering.
- What is in fact very novel is our way to build the spectral embedding not from one unique matrix, but from a collection of Bethe-Hessian matrices, parametrized by the $\{\zeta_p\}_{p=2,\dots,k}$.

Now, there are many implementation subtleties in this algorithm that I will not delve into here, even though I find them very interesting. I refer the readers to Section 4 of [Dall'Amico *et al.* 2021] for details on –for instance– how to actually compute the ζ_p's (up to the machine's precision error) in reasonable time.

Algorithm 2 Community Detection with optimal Bethe-Hessians

1: **Input** : adjacency matrix of undirected graph $\mathcal{G}$
2: Estimate the number of classes $\hat{k}$ as the number of negative eigenvalues of $\mathsf{H}_{\sqrt{\rho(\mathsf{B})}}$
3: Compute the $\hat{k}$ parameters $(\zeta_1, \dots, \zeta_{\hat{k}})$, where ζ_p is the smallest value of the parameter $r \geq 1$ verifying $\lambda_p(\mathsf{H}_r) = 0$.
4: **for** $1 \leq p \leq \hat{k}$ **do**
5: $\quad \mathsf{X}_{:,p} \leftarrow \boldsymbol{u}_p(\mathsf{H}_{\zeta_p})$ (*i.e.*, set $\boldsymbol{u}_p(\mathsf{H}_{\zeta_p})$ as the p-th column of X)
6: **end for**
7: Estimate community labels $\hat{\boldsymbol{\ell}}$ from the node embedding $\mathsf{X} = [\mathsf{X}_{:,1}, \dots, \mathsf{X}_{:,\hat{k}}]$.
8: **return** Estimated number $\hat{k}$ of communities and label vector $\hat{\boldsymbol{\ell}}$.

Why this parametrization? ζ_p has a –seemingly bizarre– definition: it is defined as the smallest value of the parameter $r \geq 1$ such that the p-th smallest eigenvalue of H_r, $\lambda_p(\mathsf{H}_r)$, is null.

Empirically, on as many connected graphs as we could think of, we have observed (and partly proven) that the function $\lambda_p(H_r)$ for $r \geq 1$ either never crosses zero (in which case ζ_p does not exist), crosses zero exactly twice and is convex between these two crossings (in which case ζ_p is the lowest of the two values), or, in very symmetric cases, touches zero exactly once without crossing it (in which case ζ_p is that value). Also, we can prove, on a connected graph, that if $\exists r$ s.t. H_r has k negative eigenvalues, then ζ_p exists for $p = 1$ to k with $\zeta_1 < \zeta_2 \leq \ldots \leq \zeta_k$.

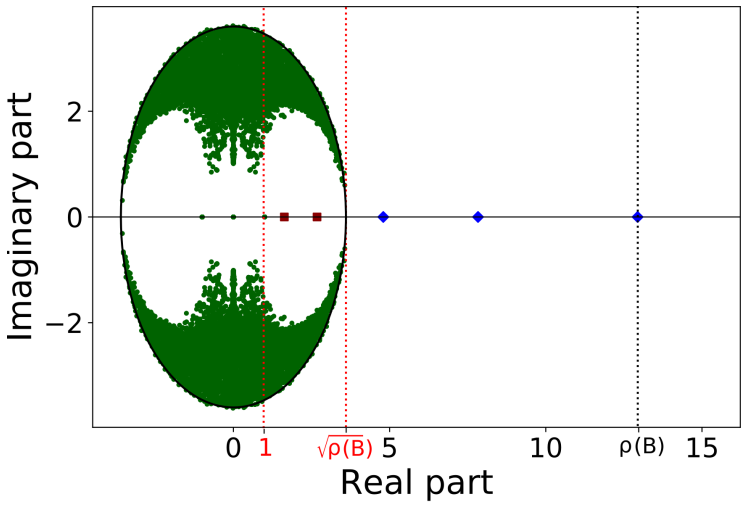

Figure 2.12: Spectrum of the matrix B on the complex plane, for a DCSBM graph with $n = 5000$ and $k = 3$. The green dots are the uninformative eigenvalues. The blue diamonds are the real eigenvalues outside the bulk and the red squares are the real eigenvalues inside the bulk (more precisely on the segment $]1, \sqrt{\rho(\mathsf{B})}]$).

To briefly explain the reasons behind this parametrization, let us first recall the links between the non-backtracking matrix and the Bethe-Hessian matrix:

- A consequence of the Ihara-Bass formula [Terras 2010] is the following: $\forall \gamma \in \mathbb{R}$, γ is an eigenvalue of B if and only if H_γ has a zero eigenvalue.
- Moreover if $\boldsymbol{v} \in \mathbb{R}^{2|\mathcal{E}|}$ is an eigenvector of B associated to γ, and defining $\boldsymbol{u} \in \mathbb{R}^n$ verifying $u_i = \sum_j v_{(ij)}$, one has that $\boldsymbol{u}$ is eigenvector of H_γ associated to its null eigenvalue [Krzakala *et al.* 2013].
- In other words, studying the eigenvectors in X that we build in our algorithm boils down –in terms of information– to studying the eigenvectors associated to the real eigenvalues of B that are on the (real) segment $[1, \sqrt{\rho(\mathsf{B})}]$. See Figure 2.12 for an illustration.

The reason why we choose to work with the Bethe Hessian rather than directly with B is twofold: first of all it is more convenient and computationally lighter to work with matrices of size n rather than $2|\mathcal{E}|$ but, most importantly, it is much simpler to compute extremal eigenvalues than eigenvalues that are "hidden" in the bulk's semi-circle, as shown in Fig. 2.12. Note in passing that the choice $r = \sqrt{\rho(\mathsf{B})}$ for the estimation of the number of classes stems from this well-known observation made by many authors that the spectrum of B, on many different graphs, concentrate around a circle of radius $\sqrt{\rho(\mathsf{B})}$, again as we see in Fig. 2.12.

The fact that there are isolated real eigenvalues outside of the bulk (the blue diamonds) as soon as we are above the detection threshold α_c is a very well-known

fact in the literature since more-or-less a decade for the SBM [Krzakala *et al.* 2013] and since 2017 for the DCSBM [Gulikers *et al.* 2017]. In fact, the proofs show that the eigenvectors associated to these isolated eigenvalues contain non-trivial community information down to the threshold –thus proving the effectiveness of spectral clustering on B. However, the fact that for each eigenvalue outside the bulk (apart from $\rho(\mathsf{B})$ whose associated eigenvector is not instructive anyways for community reconstruction) there is a *matched* real eigenvalue *inside* the bulk, has never been picked up by the literature. It is however the first central point of our contribution. The second central point is that the eigenvectors associated to these inner eigenvalues also contain non-trivial community information down to the threshold. The third central point is that these eigenvectors are also much less polluted by the degree heterogeneity of the underlying graph – thus enabling a better community recovery when the graph is heterogeneous.

These central contributions were intuited via a body of several arguments, in which I will not delve into here, and that can be found in [Dall'Amico *et al.* 2019] for $k = 2$ and in [Dall'Amico *et al.* 2021] for arbitrary k. Some come from statistical physics and others from linear algebra considerations. To my knowledge, only the existence of "inside eigenvalues" has been formally proven to-date: in [Coste & Zhu 2021], and only in the almost sparse regime where the average degree is in $\mathcal{O}(\log(n))$ instead of $\mathcal{O}(1)$.

Performances. We compare the performance of our method versus the state-of-the-art in terms of community recovery as a function of the difficulty α of the task, for DCSBM graphs in Figure 2.13. We also have a table in Figure 2.14 that shows performance comparisons on real-world graphs. As there is no ground truth in this case, we use modularity as a performance measure. We are of course very satisfied with these results.

On top of shear performance, the philosophy behind our algorithm is quite elegant in the way it adapts automatically to the difficulty of the problem. In fact, in easy regimes where the real eigenvalues inside the bulk of B tend to 1, the values of ζ_p also tend to 1, and the eigenvectors we extract are in fact tending towards the first k eigenvectors of the classical combinatorial Laplacian matrix! (recall that $\mathsf{H}_{r=1} = \mathsf{L}$) Now, in difficult regimes where we are close to the detectability threshold, and the inner real eigenvalues of B tend to the bulk spread as a semi-circle of radius $\sqrt{\rho(\mathsf{B})}$, the ζs will all tend to $\sqrt{\rho(\mathsf{B})}$, and the eigenvectors we extract will tend to the first k eigenvectors of $\mathsf{H}_{\sqrt{\rho(\mathsf{B})}}$: which is the "pessimistic" scenario of [Saade *et al.* 2014]. One can see that the ζs are a form of data-driven, structure regularization that enables the eigenvectors to both stay i) in extremal positions of the spectrum, ii) not be too polluted by degree heterogeneity.

Another way of seeing this structure regularization is by observing the following facts:

- Let $\mathsf{D}_\tau = \mathsf{D} + \tau \mathsf{I}_n$ for $\tau \in \mathbb{R}^+$ be a regularized degree matrix.

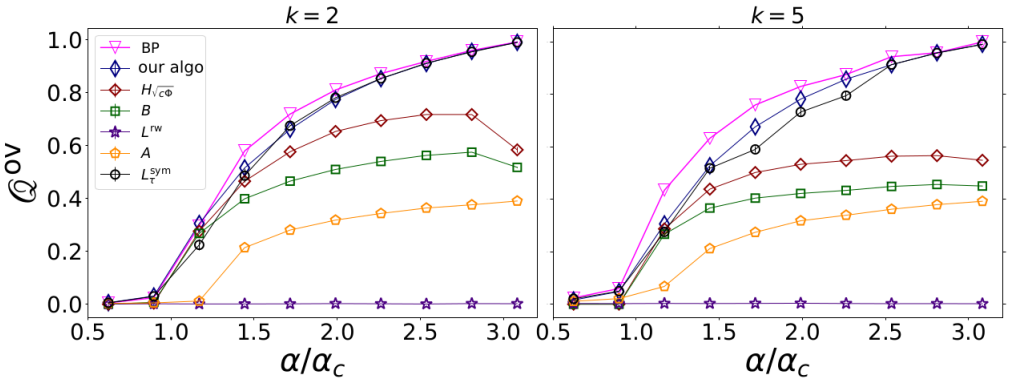

Figure 2.13: Performance vs competing methods on DCSBM synthetic graphs. $n = 5 \cdot 10^4$, $c = 5$. BP stands for belief propagation (optimal method but much slower than spectral clustering). All other methods shown are spectral clustering results on various choices of representative matrix R: $\mathsf{H}_{\sqrt{c\Phi}}$ is the Bethe-Hessian with parameter $\sqrt{c\Phi}$ as proposed by [Saade *et al.* 2014], B is the non-backtracking matrix, L^{rw} is the random walk Laplacian, A the adjacency matrix and $\mathsf{L}_\tau^{\text{sym}}$ is a normalized and regularized Laplacian with τ set to be the average degree, as suggested by [Qin & Rohe 2013]. Note that L^{rw} drastically fails in this case as the eigenvectors show strong localization effects due to the degree heterogeneity.

- Define $\mathsf{L}_\tau^{\text{rw}} = \mathsf{I} - \mathsf{D}_\tau^{-1}\mathsf{A}$ and $\mathsf{L}_\tau^{\text{norm}} = \mathsf{I} - \mathsf{D}_\tau^{-1/2}\mathsf{A}\mathsf{D}_\tau^{-1/2}$, respectively a regularized random walk Laplacian and a regularized normalized Laplacian.
- One can show the following equivalence:

$$\forall (r, \boldsymbol{x}), \quad \mathsf{H}_r \boldsymbol{x} = \left[(r^2-1)\mathsf{I}_n + \mathsf{D} - r\mathsf{A}\right]\boldsymbol{x} = 0 \quad \Longleftrightarrow \quad \mathsf{L}_{r^2-1}^{\text{rw}} \boldsymbol{x} = \left(1 - \frac{1}{r}\right)\boldsymbol{x}$$

As a consequence: looking into the spectrum of the Bethe-Hessian is equivalent to looking into the spectrum of a regularized Laplacian, that is, the Laplacian of a regularized structure. Remarking that $\mathsf{L}_\tau^{\text{rw}}$ and $\mathsf{L}_\tau^{\text{norm}}$ share the same spectrum, this also shows a strong connection between the Bethe-Hessian and other community detection works based on regularized Laplacians, such as [Qin & Rohe 2013]. As we give strong arguments to optimally choose the parameters of the Bethe-Hessian, our work also extends to the optimal parametrization of these deformed Laplacians – a question that was not previously properly addressed in the literature (with very heuristic arguments to choose τ).

Computation time. Our implementation runs in time $\mathcal{O}(|\mathcal{E}|k^2 + nk^3)$ which is k times longer than classical spectral clustering. This is intrinsically due to the fact that we have to deal with the spectrum of k different matrices, not just one.

In practice, to give an order of magnitude of computation times[5], running our

[5] on a laptop with 7.7Gb of RAM and Intel Core i7-6600U CPU @ 2.6GHz x 4

Dataset	n	c	k	H_{ζ_p}	A	$H_{\sqrt{c\Phi}}$	B	L^{rw}	L_τ^{sym}
Karate	34	4.6	2	**0.37**	**0.37**	**0.37**	**0.37**	0.36	**0.37**
Dolphins	62	5	2	**0.38**	0.21	0.34	0.22	**0.38**	**0.38**
Polbooks	105	8.4	3	**0.50**	0.47	**0.50**	0.45	**0.50**	**0.50**
Football	115	10.7	12	**0.60**	**0.60**	**0.60**	**0.60**	**0.60**	**0.60**
Mail	1133	9.6	21	**0.52**	0.32	0.40	0.37	0.48	**0.52**
Polblogs	1222	27,4	2	**0.43**	0.25	0.27	0.23	0.00	**0.43**
Tv	3892	8.9	41	**0.85**	0.60	0.56	0.55	0.55	0.80
Facebook	4039	43.7	55	**0.76**	0.42	0.49	0.48	0.70	0.58
GrQc	4158	6.5	29	**0.80**	0.52	0.51	0.51	0.34	**0.80**
Power grid	4941	2.7	25	**0.92**	0.18	0.33	0.31	**0.92**	0.85
Politicians	5908	14.1	62	**0.82**	0.48	0.54	0.51	0.74	0.74
GNutella P2P	6299	6.6	4	**0.40**	0.20	0.14	0.14	0.00	0.35
Wikipedia	7066	28.3	22	0.27	0.14	0.18	0.16	**0.34**	0.27
HepPh	11204	21.0	60	**0.57**	0.46	0.42	0.42	0.27	0.52
Vip	11565	11.6	53	**0.62**	0.28	0.32	0.32	0.16	0.54

Figure 2.14: Performance (as measured by modularity, as there is no ground truth) comparison on real networks. When k is known, it appears as an underlined number. If not known, it is estimated by our method.

Julia implementation CoDeBetHe.jl of our algorithm on a SBM with $k = 4$ classes of equal size, $n = 10^5$ (resp. 10^6), $c_{\mathrm{in}} = 26$ on the diagonal elements of C and $c_{\mathrm{out}} = 5$ for the off diagonal elements, takes approximately 15 (resp., 450) seconds on a laptop. If k is known in advance, this times drop to approximately 6 (resp., 100) seconds. Given the added precision of the clustering, this added computation price seems reasonable.

2.4.4 What now?

A first theoretical direction is to try to prove parts of the conjectures, perhaps by starting in the easier regime where the average degree is in $\mathcal{O}(\log n)$. The existence of inner real eigenvalues of B has already been proven in this case in [Coste & Zhu 2021]. However, the facts that the associated eigenvectors i) contain relevant community information down to the threshold, ii) are less polluted by degree heterogeneity than their counterparts outside the bulk, remain to be proven formally.

A second, more methodological direction, would be design a new GSP framework where one does not define the whole graph spectral theory on one unique matrix R like what has been done before (classically choosing $\mathsf{R} = \mathsf{A}$ or L or normalized versions of these two matrices), but by defining Fourier modes in an iterative and data-driven manner, in a similar fashion than how Algorithm 2 builds the spectral embedding for our improved spectral clustering method. In fact, eigenvectors that perform better for spectral clustering should ideally be used as low-frequency graph Fourier modes! There are however complications that will need to be addressed, one of them being the added computation time needed to compute these new Fourier

modes, another that Fourier modes thus defined are not orthogonal anymore (in fact, the eigenvectors concatenated in X in Alg. 2 are not necessarily orthogonal to each other as they come from different Bethe-Hessian matrices).

Chapter 3

Determinantal point processes: fundamentals and applications

Contents

This chapter relates a selection of my work on *Determinantal Point Processes (DPPs)*, random processes that have the rare property of jointly being tractable and repulsive. For the story, I came across these processes back in 2016, frustrated by the iid framework of the random graph sampling strategy I had contributed to (and recalled in Section 2.3.2): the *independent* property of our sampling was admittedly very convenient for theoretical purposes as we could draw from powerful concentration inequalities, but it seemed inefficient to allow sampling nodes that

were close to each other in the graph. We indeed intuitively know that close-by nodes in a graph hold similar information on bandlimited, slow-varying, graph signals; and that some kind of repulsion in the sampling process would be a great asset. In the extreme case, iid sampling with replacement allowed to sample the *same* node several times – which is clearly a waste of our sampling budget. I needed random processes that could produce some kind of repulsion in the samples to hope to sample nodes holding information on the signal that was not too redundant. Unfortunately, repulsion naturally induces dependence in the sampling process and the theory of concentration largely collapses in dependent sampling contexts – making theoretical analysis of repulsive processes a major scientific challenge in today's field of statistics.

About at the same period, I came across DPPs by reading Avena and Gaudillière's paper [Avena & Gaudillière 2017] on random spanning forests, in which they show that random forests' roots were a DPP defined on the nodes of the graph: in other words, the root process was in effect a sampling process over the nodes of the graph that was both tractable and repulsive. The marginal sampling distribution was however not exactly what I was needing for my purpose of bandlimited graph sampling, but it sparked a long-lasting interest in DPPs in general, and DPPs over graphs in particular.

Since then, I have been concentrating my efforts on these two fronts: DPPs in general and DPPs over graphs. This chapter describes my work on DPPs in general, as a fascinating sampling process in its own right –and we put aside for now all notions of graph. In fact, this chapter can be read totally independently from Chapter 2. My work on DPPs specifically defined over graphs, that brings together my joint interest in graphs and in DPPs, will be described in the next chapter.

Organization. This chapter is organized similarly to Chapter 2: it first contains a background section (Section 3.1) where a few notations and necessary definitions and intuitions are (hopefully sufficiently smoothly) introduced. The next three sections are a selection of three of my works on DPPs. The first contribution, detailed in Section 3.2, is algorithmical: it provides a new sampling algorithm for DPPs. The second contribution, in Section 3.3, is theoretical in nature: it fills a gap in the theory of DPPs. Finally, Section 3.4 details an applicative contribution: it studies how DPPs can be used for sketching under the computer science framework known as coresets.

3.1 Background

Determinantal point processes are by now perhaps one of the most famous example of repulsive point processes. They first appear as a model for the position of fermionic particles in an energy potential [Macchi 1975], but also occur in random matrix theory and graph theory, for instance as the distribution of edges in uniform spanning trees [Burton & Pemantle 1993]. DPPs can also be used as tools designed for particular computational tasks: for instance, they have

been advocated in machine learning as a way of providing samples with guaranteed diversity [Kulesza & Taskar 2012], as a way of performing Monte Carlo integration [Bardenet & Hardy 2020, Coeurjolly *et al.* 2021], or accelerating classical linear algebra tasks such as regression or low rank approximation of matrices [Belhadji *et al.* 2020, Derezinski & Mahoney 2021].

We place ourselves in the discrete sampling setting. In that framework, one has a set of n items Ω, and one desires to produce a subset $\mathcal{X} \subset \Omega$ of size m (usually verifying $m \ll n$) such that no two items in $\mathcal{X}$ are excessively similar. A key aspect of DPPs is that "diversity" is defined relative to a notion of similarity represented by a positive-definite kernel. For instance, if the items are vectors in $\mathbb{R}^d$, similarity may be defined via the squared-exponential (Gaussian) kernel:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\|\boldsymbol{x} - \boldsymbol{y}\|^2\right) \tag{3.1}$$

Here $\boldsymbol{x}$ and $\boldsymbol{y}$ are two items of Ω, and similarity is a decreasing function of distance.

The goal of DPPs is to sample subsets $\mathcal{X}$ that are *diverse*, that is, well spread-out according to the chosen metric κ. In this sense they are an instance of repulsive processes. In fact, DPPs sit at the edge between very repulsive processes for which we hardly know anything analytically (such as the hard sphere process) and iid processes that do not offer any repulsion but which have been theoretically thoroughly developed for more than a century. To be more specific, DPPs are only mildly repulsive (as we will discuss from an original viewpoint in Section 3.2.3), which is both a blessing and a malediction. First, a blessing because they are "close enough" to iid processes that they are *tractable*: the marginal distributions at any order are known and even computable in polynomial time, concentration inequalities can be obtained [Pemantle & Peres 2014], *etc.* Second, a malediction because they can only offer a weak repulsion, and there are many cases where we would intuitively need more negative correlation between the samples. But this is the price to pay to work with correlated samples: one cannot, at this stage, have the best of both worlds.

In the following, we briefly recall some definitions on DPPs along with fixed-size DPPs, a useful variant (as well as L-ensembles and fixed-size L-ensembles). For details we refer the reader to [Kulesza & Taskar 2011], [Tremblay *et al.* 2023] or [Bardenet 2022]. All of the results below are classical.

Notations. Let A be a $n \times n$ matrix, and $\mathcal{Y}$, $\mathcal{Z}$ be two subsets of indices. Then $\mathsf{A}_{\mathcal{Y},\mathcal{Z}}$ is the submatrix of A formed by retaining the rows in $\mathcal{Y}$ and the columns in $\mathcal{Z}$. Furthermore, $\mathsf{A}_{:,\mathcal{Y}}$ (resp. $\mathsf{A}_{\mathcal{Y},:}$) is the matrix made of the full columns (resp. rows) indexed by $\mathcal{Y}$. Finally, we let $\mathsf{A}_{\mathcal{Y}} = \mathsf{A}_{\mathcal{Y},\mathcal{Y}}$. Also, for a matrix V, by $\mathrm{span}(\mathsf{V})$ we denote its column span, and by $\mathrm{orth}(\mathsf{V})$ the orthogonal complement of $\mathrm{span}(\mathsf{V})$. Finally, for a *Positive Semi-Definite* (PSD) matrix A, I mean by $a\mathsf{I} \preceq \mathsf{A} \preceq b\mathsf{I}$ that A's spectrum is included in the interval $[a, b]$.

3.1.1 DPPs

Let $\Omega = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\} \subset \mathbb{R}^d$ be a collection of vectors called the *ground set*. A finite point process is a random subset $X \subseteq \Omega$. Abusing notation, we sometimes use X to designate the indices of the items, rather than the items themselves. Which one we mean should be clear from context.

Definition 2 (Determinantal Point Process)**.** *Let $\mathsf{K} \in \mathbb{R}^{n \times n}$ be a symmetric Positive Semi-Definite (PSD) matrix verifying $\mathsf{0} \preceq \mathsf{K} \preceq \mathsf{I}$. X is a DPP with marginal kernel K, denoted by $X \sim \mathtt{DPP}(\mathsf{K})$, if*

$$\forall \mathcal{A} \subseteq \Omega \qquad \mathbb{P}(\mathcal{A} \subseteq X) = \det \mathsf{K}_{\mathcal{A}}, \tag{3.2}$$

where by convention, $\det \mathsf{K}_{\varnothing} = 1$.

This definition determines what we will refer to as the *class of DPPs*.

Remark 5. *In all generality, K only needs to be PSD, not necessarily symmetric. Even if there is some recent evidence that non-symmetric PSD kernels may be of interest [Han et al. 2021] for some machine learning tasks, we decide in our work to restrict ourselves to symmetric kernels as it is much simpler to deal with, and much more intuitive. In the following, even if I don't repeat it everywhere, the matrices K and L are always supposed symmetric.*

Now, manipulating inclusion probabilities as in Eq. (3.2) rather than the joint probability distribution itself is often cumbersome (nor is it very intuitive at first glance). This usually leads authors to consider a slightly less general class of DPPs: L-ensembles [Borodin & Rains 2005].

Definition 3 (L-ensemble)**.** *Let $\mathsf{L} \in \mathbb{R}^{n \times n}$ designate a Positive Semi-Definite (PSD) matrix. An L-ensemble based on L, , denoted by $X \sim \mathtt{Lens}(\mathsf{L})$, is a point process X defined as*

$$\forall \mathcal{X} \in \Omega \qquad \mathbb{P}(X = \mathcal{X}) = \frac{\det \mathsf{L}_{\mathcal{X}}}{Z}, \tag{3.3}$$

where by convention, $\det \mathsf{L}_{\varnothing} = 1$ (implying for instance that $\mathbb{P}(X = \varnothing) = 1/Z$).

The normalisation constant $Z = \sum_{\mathcal{X} \subseteq \Omega} \det \mathsf{L}_{\mathcal{X}}$ can be shown [Kulesza & Taskar 2012] to equal $\det(\mathsf{I} + \mathsf{L})$.

Why are such processes repulsive? For instance, consider the L-ensemble based on the $n \times n$ PSD matrix $\mathsf{L} = [\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)]_{i,j}$, with κ as in Eq. (3.1). If two or more points in a set $\mathcal{X}$ are very similar (in the sense of the kernel function), then the matrix $\mathsf{L}_{\mathcal{X}}$ has rows that are nearly collinear and its determinant is small. This in turns makes it unlikely that such a set $\mathcal{X}$ will be selected by the L-ensemble.

Another way to gain intuition is to look at it from the marginal point-of-view and consider the marginal probability of order 2:

$$\mathbb{P}(\{i, j\} \in X) = \det \mathsf{K}_{\{i,j\},\{i,j\}} = \mathsf{K}_{ii}\mathsf{K}_{jj} - \mathsf{K}_{ij}^2$$

which is necessarily smaller than $\mathsf{K}_{ii}\mathsf{K}_{jj}$, that is, what one would have obtained by independent sampling. Also, the more similar are items indexed by i and j, the larger K_{ij}^2, the less chance they have of ending up jointly in the DPP sample.

Note that all L-ensembles are DPPs:

Lemma 6. *An L-ensemble is a DPP. Let $X \sim \mathtt{Lens}(\mathsf{L})$ be an L-ensemble based on the PSD matrix L. Then, X is a DPP with marginal kernel*

$$\mathsf{K} = \mathsf{L}(\mathsf{I} + \mathsf{L})^{-1}. \tag{3.4}$$

The converse is not true: all DPPs are *not* L-ensembles.

Lemma 7. *A DPP with marginal kernel K is an L-ensemble if and only if K verifies $\mathsf{0} \preceq \mathsf{K} \prec \mathsf{I}$ (note the $\prec$ sign, implying that no eigenvalue of K is allowed to be equal to one). Let $X \sim \mathtt{DPP}(\mathsf{K})$ be a DPP with such a marginal kernel. Then X is a L-ensemble with a PSD kernel L verifying:*

$$\mathsf{L} = \mathsf{K}(\mathsf{I} - \mathsf{K})^{-1}.$$

Remark 8. *The class of DPPs can thus be separated in two: the L-ensembles (all DPPs with marginal kernel verifying $\mathsf{0} \preceq \mathsf{K} \prec \mathsf{I}$), and the rest (all DPPs with marginal kernel whose spectrum contains at least one eigenvalue equal to one).*

In DPPs, the size (cardinal) of X, denoted by $|X|$, is a random variable. Its distribution is as follows [Hough *et al.* 2006]:

Lemma 9. *Let $\mathsf{0} \preceq \mathsf{K} \preceq \mathsf{I}$ be a marginal kernel with eigenvalues $\mu_1, \ldots, \mu_n$. Let $X \sim \mathtt{DPP}(\mathsf{K})$. Then, $|X|$ has the same distribution as $\sum_{i=1}^{n} B_i$, where B_i is a Bernoulli random variable with expectation $\mathbb{E}(B_i) = \mu_i$, and the B_i's are distributed independently. In particular, the expected size of the DPP, $\mathbb{E}(|X|)$, can be directly deduced from the above to be*

$$\mathbb{E}(|X|) = \sum \mu_i = \mathrm{Tr}(\mathsf{K}) \tag{3.5}$$

3.1.2 Fixed-size DPPs

The cardinal of a DPP is thus in general random. Such varying-sized samples are not practical in many applications (one often desires a subset of size 50, not a subset of size 50 on average but which may be of size 35 or 56); which led the authors of [Kulesza & Taskar 2011] to define fixed-size DPPs[1].

Definition 4 (Fixed-size Determinantal Point Process)**.** *A fixed size DPP of size m is a DPP conditioned on $|X| = m$. It is denoted $X \sim |\mathtt{DPP}|_m(\mathsf{K})$.*

[1] They are often called k-DPPs in the literature, but we prefer "fixed-size DPPs" in order not to overload the symbol k too much.

It is important to understand that, in general, fixed-size DPPs are *not* DPPs. The only exception are projection DPPs, that we will define later as DPPs associated to projection kernels. There are several ways to see why fixed-size DPPs are not DPPs. For instance, whereas all DPPs have a marginal kernel, fixed-size DPPs (again with the exception of projection DPPs) do not have marginal kernels: there does not exist a matrix whose principal minors are the marginal probabilities. For more information, the question of inclusion probabilities in fixed-size DPPs is treated at length in our paper [Barthelmé *et al.* 2019].

A subclass of fixed-size DPPs is the class of fixed-size L-ensembles:

Definition 5 (Fixed-size L-ensemble)**.** *Let* L *be a PSD matrix. A fixed-size L-ensemble, denoted by* $X \sim |\mathtt{Lens}|_m(\mathsf{L})$*, is a L-ensemble conditioned on* $|X| = m$*. It thus verifies:*

$$\forall \mathcal{X} \in \Omega \qquad \mathbb{P}(X = \mathcal{X}) = \frac{\det \mathsf{L}_\mathcal{X}}{Z_m} \mathbb{I}(|\mathcal{X}| = m). \tag{3.6}$$

where Z_m *is the normalisation constant, and* $\mathbb{I}(\cdot)$ *is the indicator function.*

Similarly to the varying-size case, fixed-size L-ensembles are a strict subclass of fixed-size DPPs:

Lemma 10. *A fixed-size L-ensemble is a fixed-size DPP. The converse is not true: there exist fixed-size DPPs that are not fixed-size L-ensembles.*

The normalization constant $Z_m = \sum_{\mathcal{X},|\mathcal{X}|=m} \det \mathsf{L}_\mathcal{X}$ can be shown to be equal to the m-th *elementary symmetric polynomial* of L, a quantity that depends only on the spectrum of L. We will however not delve into this in this document.

3.1.3 Two useful special cases.

There are two special cases of (possibly fixed-size) DPPs that are useful to study on their own, both from a practical and theoretical viewpoint. These are the DPPs with *diagonal* kernels and those with *projection* kernels. As we will see in section 3.1.4, these two examples are the key components for sampling any DPP using its mixture representation.

3.1.3.1 Diagonal DPPs

Diagonal DPPs are in a way the most basic kind of DPPs (although the fixed-size case is surprisingly intricate).

Lemma 11. *Let* $\mathsf{D} = \mathrm{diag}(d_1, \ldots, d_n)$ *be a diagonal matrix with entries* $\{d_i\}$ *between* 0 *and* 1*.* $X \sim \mathtt{DPP}(\mathsf{D})$ *is a Bernoulli process: each event* $i \in X$ *is independent and occurs with probability* $\pi_i = d_i$*.*

Remark 12. *For fixed-size DPPs this is no longer true:* $X \sim |\mathtt{DPP}|_m(\mathsf{D})$ *is not a Bernoulli process, as the events are no longer independent but indeed negatively associated. To see why, note that since the total size is fixed, conditional on* $i \in X$ *other points are less likely to be included.*

Using Lemma 6, one directly obtains the equivalent result for L-ensembles:

Lemma 13. *Let* $\mathsf{\Lambda} = \mathrm{diag}(\lambda_1, \dots, \lambda_n)$ *be a diagonal matrix with* $\forall i, \lambda_i \geq 0$. $X \sim \mathtt{Lens}(\mathsf{\Lambda})$ *is a Bernoulli process: each event* $i \in X$ *is independent and occurs with probability* $\pi_i = \frac{\lambda_i}{1+\lambda_i}$.

3.1.3.2 Projection DPPs

Projection DPPs have many unique features that make them a very central object of the theory of DPPs. The definition of a projection DPP is as follows:

Definition 6 (Projection DPP)**.** *Let* Q *be an* $n \times m$ *matrix with* $\mathsf{Q}^\top\mathsf{Q} = \mathsf{I}_m$. *A projection DPP is a DPP with marginal kernel* $\mathsf{K} = \mathsf{Q}\mathsf{Q}^\top$.

The name "projection DPP" comes from the fact that $\mathsf{Q}\mathsf{Q}^\top$ is a projection matrix (its eigenvalues are 1, with multiplicity m, and 0 with multiplicity $n-m$). This has several consequences:

- As by Lemma 9, the size of a projection DPP $X \sim \mathtt{DPP}(\mathsf{Q}\mathsf{Q}^\top)$ is constant and equals to m. Thus, a projection DPP is trivially a fixed-size DPP. In fact, as already briefly mentioned, the class of projection DPPs is the exact intersection of the class of fixed-size DPPs and the class of DPPs.
- As by Lemma 7, and as K's spectrum contains at least one eigenvalue equal to 1: a projection DPP is *not* an L-ensemble.

However, one can show that projection DPPs are *fixed-size* L-ensembles:

Lemma 14 (See e.g., [Barthelmé *et al.* 2019])**.** *Let* $\mathsf{Q} \in \mathbb{R}^{n\times m}$ *matrix verifying* $\mathsf{Q}^\top\mathsf{Q} = \mathsf{I}_m$. *A projection DPP with marginal kernel* $\mathsf{Q}\mathsf{Q}^\top$ *is a fixed-size L-ensemble* $X \sim |\mathtt{Lens}|_m(\mathsf{Q}\mathsf{Q}^\top)$.

Remark 15. *Moreover, the normalisation constant* Z_m *as defined in Definition 5 is particularly simple in this case and equals* 1. *This means that the joint probability distribution of a projection DPP simply verifies*

$$\forall \mathcal{X} \in \Omega \qquad \mathbb{P}(X = \mathcal{X}) = \det \mathsf{L}_\mathcal{X}\, \mathbb{I}(|\mathcal{X}| = m).$$

Remark 16. *Projection DPPs are thus DPPs, fixed-size DPPs and fixed-size L-ensembles: given* $\mathsf{Q} \in \mathbb{R}^{n\times m}$ *verifying* $\mathsf{Q}^\top\mathsf{Q} = \mathsf{I}_m$, *notations* $X \sim \mathtt{DPP}(\mathsf{Q}\mathsf{Q}^\top)$, $X \sim |\mathtt{DPP}|_m(\mathsf{Q}\mathsf{Q}^\top)$ *and* $X \sim |\mathtt{Lens}|_m(\mathsf{Q}\mathsf{Q}^\top)$ *denote the exact same distribution.*

3.1.4 Mixture representation

Determinantal point processes have a well-known representation as a mixture of projection DPPs (also sometimes called "elementary DPPs" in the literature). This mixture representation (due to [Hough *et al.* 2006]) is fundamental, both for theoretical and computational purposes, since it serves as the basis for exact sampling of DPPs. There are two variants, one for DPPs and one for fixed-size DPPs. Both can be proven using the Cauchy-Binet formula.

Lemma 17 (Mixture representation of DPPs)**.** *Let* K *be a marginal kernel (thus verifying* $\mathsf{0} \preceq \mathsf{K} \preceq \mathsf{I}$*) and* $\mathsf{K} = \mathsf{Q}\mathsf{D}\mathsf{Q}^\top$ *its eigendecomposition. Then,* $X \sim \texttt{DPP}(\mathsf{K})$ *may be obtained from the following mixture process:*

1. *Sample indices* $Y \sim \texttt{DPP}(\mathsf{D})$
2. *Form the projection matrix* $\mathsf{M} = \mathsf{Q}_{:,Y}(\mathsf{Q}_{:,Y})^\top$
3. *Sample from the projection DPP* $X|Y \sim |\texttt{DPP}|_{|Y|}(\mathsf{M})$

The mixture representation can be understood as (a) first sample which eigenvectors to use and (b) sample a projection DPP with the selected eigenvectors.

The counterpart for fixed-size DPPs looks highly similar.

Lemma 18 (Mixture representation of fixed-size DPPs)**.** *Let* K *be a marginal kernel and* $\mathsf{K} = \mathsf{Q}\mathsf{D}\mathsf{Q}^\top$ *its eigendecomposition. Then,* $X \sim |\texttt{DPP}|_m(\mathsf{K})$ *may be obtained from the following mixture process:*

1. *Sample* m *indices* $Y \sim |\texttt{DPP}|_m(\mathsf{D})$
2. *Form the projection matrix* $\mathsf{M} = \mathsf{Q}_{:,Y}(\mathsf{Q}_{:,Y})^\top$
3. *Sample from the projection DPP* $X|Y \sim |\texttt{DPP}|_m(\mathsf{M})$

The only step that varies is the first one, where we sample from $|\texttt{DPP}|_m(\mathsf{D})$ instead of $\texttt{DPP}(\mathsf{D})$. Note that as L-ensembles are a subclass of DPPs, these mixture representations naturally apply to L-ensembles as well.

3.1.5 Standard DPP sampling algorithm

Steps 1 above necessitate to sample diagonal DPPs. Note that although sampling from a diagonal DPP is straightforward and costs $\mathcal{O}(n)$ as explained in Lemma 11, sampling from a fixed-size diagonal DPP can be surprisingly tricky in practice. It necessitates to either compute elementary polynomials and follow Algorithm 8 of [Kulesza & Taskar 2012] (but it is prone to numerical instabilities when n and/or m are too large), or to use approximate techniques such as the ones we developed in our paper [Barthelmé *et al.* 2019]. In both cases, this first step can be performed in time $\mathcal{O}(nm)$.

Step 3 of the above mixture representation requires to sample from a projection DPP. The standard algorithm to do so is described in Algorithm 3.

Computation cost of DPP sampling. For arbitrary kernels, the cost of DPP sampling is dominated by the cost of the eigendecomposition and thus scales as $\mathcal{O}(n^3)$. A popular workaround to this prohibitive cost is to sample DPPs defined by low-rank kernels. In fact, one often considers L-ensembles associated to kernels of the form $\mathbf{\Psi}\mathbf{\Psi}^\top$ with $\mathbf{\Psi} \in \mathbb{R}^{n \times p}$. Sampling from this kernel costs only $\mathcal{O}(np^2 + nm^2)$. The first term, $\mathcal{O}(np^2)$, comes from a SVD-like step of $\mathbf{\Psi}$ and the second term, $\mathcal{O}(nm^2)$ is the cost of Algorithm 3 (as one needs to at least compute m times the matrix-vector product of line 7).

Algorithm 3 Standard sampling algorithm for a projection DPP $X \sim \texttt{DPP}(\mathsf{Q}\mathsf{Q}^\top)$

Inputs: $\mathsf{Q} \in \mathbb{R}^{n\times m}$ verifying $\mathsf{Q}^\top\mathsf{Q} = \mathsf{I}_m$.
Initialise $\pi(x) \leftarrow \sum_{j=1}^m Q_{x,j}^2$, $X = \emptyset$, Gram-Schmidt basis $\mathsf{S} = []$
for $t = 1$ to m **do**
    Sample x from $\frac{\pi(x)}{m-(t-1)}$, add to X
    Compute residual $\boldsymbol{z}_t = (\mathsf{I} - \mathsf{S}\mathsf{S}^\top)(\mathsf{Q}_{x,:})^\top \in \mathbb{R}^m$
    Add column $\boldsymbol{s}_t = \frac{\boldsymbol{z}_t}{\|\boldsymbol{z}_t\|}$ to S
    Compute $\boldsymbol{v} = \mathsf{Q}\boldsymbol{s}_t \in \mathbb{R}^n$
    Update probabilities $\pi(x) \leftarrow \pi(x) - (v_x)^2$
end for
return X

3.2 A faster sampler for DPPs

We are now over with the background section. The contribution we discuss in this section, published in [Barthelme *et al.* 2023a], is an algorithmical contribution: it shows that the cost of sampling projection DPPs, $\mathcal{O}(nm^2)$, can always be reduced to $\mathcal{O}(nm + m^3 \log m)$ without loss on the sampling's exactness. As projection DPPs are a building block of both DPPs and fixed-size DPPs in their mixture representations, our contribution also accelerates the sampling time of any arbitrary (possibly fixed-size) DPP. The extent of this acceleration depends on the rank of the DPP's kernel. Indeed, if the DPP's kernel is of full rank, one still needs to perform the full eigendecomposition, which is in any case overwhelmingly time-dominant compared to the projection DPP step: in these cases, our acceleration is negligible. However, for the common cases where the DPP is defined as an L-ensemble associated to a low-rank kernel, that is $\mathsf{L} = \Psi\Psi^\top$ with $\Psi \in \mathbb{R}^{n\times p}$, then, the closer is m to p the more substantial becomes our acceleration. In this document, we will not delve further on this link between rank and the extent of the acceleration enabled by our algorithm.

We will rather concentrate on the projection DPP building block and suppose that our objective is simply to sample $X \sim \texttt{DPP}(\mathsf{Q}\mathsf{Q}^\top)$ with $\mathsf{Q} \in \mathbb{R}^{n\times m}$ given and verifying $\mathsf{Q}^\top\mathsf{Q} = \mathsf{I}_m$. Our contribution is to show a sampling algorithm for projection DPPs that costs $\mathcal{O}(nm + m^3 \log m)$ rather than the $\mathcal{O}(nm^2)$ cost of the standard variant. In practice, we observe orders-of-magnitude speedups compared to the classical algorithm as soon as $n > 1000$. Our algorithm described here is a close variant of the standard algorithm for sampling continuous DPPs, and uses rejection sampling. We also show that the term in $\mathcal{O}(nm)$ of our algorithm's cost comes in fact from a pre-processing step: a consequence is that any *additional* sample from the same DPP can be drawn in time only $\mathcal{O}(m^3 \log m)$, instead of $\mathcal{O}(nm^2)$ for the standard algorithm.

Algorithm 4 Sampling of a projection DPP $X \sim \texttt{DPP}(\mathsf{Q}\mathsf{Q}^\top)$ via rejection sampling

1: Initialise $\pi(x) \leftarrow \sum_{j=1}^m Q_{x,j}^2$, $X \leftarrow \emptyset$, Gram-Schmidt basis $\mathsf{S} \leftarrow []$
2: Initialise alias table for Walker's alias algorithm to sample from π.
3: **for** $t \in 1 \dots m$ **do**
4: accept $\leftarrow$ false
5: **while** not accept **do**
6: Draw x from π using the alias method
7: Compute acceptance ratio $r = 1 - \frac{1}{\pi(x)} \sum_{j=1}^{t-1} (\mathsf{Q}_{x,:}\boldsymbol{s}_t)^2$
8: **if** rand() $< r$ **then**
9: accept $\leftarrow$ true
10: **end if**
11: **end while**
12: Add x to X
13: Compute residual $\boldsymbol{z}_t = (\mathsf{I} - \mathsf{S}\mathsf{S}^\top)(\mathsf{Q}_{x,:})^\top$
14: Add column $\boldsymbol{s}_t = \frac{\boldsymbol{z}_t}{\|\boldsymbol{z}_t\|}$ to S
15: **end for**
16: **return** X

3.2.1 A rejection sampling algorithm to accelerate projection DPP sampling

The main idea behind our rejection sampling algorithm is that, instead of sampling exactly from a probability distribution $\pi(x)$ that is costly to update at every step of the *for* loop of Alg. 3, we decide to sample everytime from $\pi(x) = \sum_{j=1}^m Q_{x,j}^2$ and then use an adaptative rejection probability to correct the bias. This is a standard rejection sampling framework (that is in fact used by necessity to sample from continuous DPPs but has never really been precisely studied in the discrete case). In the end, the algorithm we end up with is given as Alg. 4. To recapitulate the different computational costs:

- Preprocessing cost: computing π for all entries comes at cost $\mathcal{O}(nm)$ (line 1) and setting up Walker's alias method at cost $\mathcal{O}(n)$ (line 2)
- The Gram-Schmidt process (computing $\boldsymbol{z}_t$ then $\boldsymbol{s}_t$) costs $\mathcal{O}(tm)$ at step t. Summing this figure for $t = 1$ to m gives a cost of $\mathcal{O}(m^3)$
- We now need to compute the average cost of the *while* loop. At step t, it can be shown that the rejection sampler has probability $\rho_t = \frac{m-t+1}{m}$ of succeeding. The number of proposals R_t that are required until acceptance is thus a random variable that follows a geometric distribution with success probability ρ_t. Thus:
$$\mathbb{E}(R_t) = \frac{1}{\rho_t} = \frac{m}{m-t+1}. \tag{3.7}$$
Since computing the acceptance ratio $r = 1 - \frac{1}{\pi(x)} \sum_{j=1}^{t-1} (\mathsf{Q}_{x,:}\boldsymbol{s}_t)^2$ costs $\mathcal{O}(mt)$

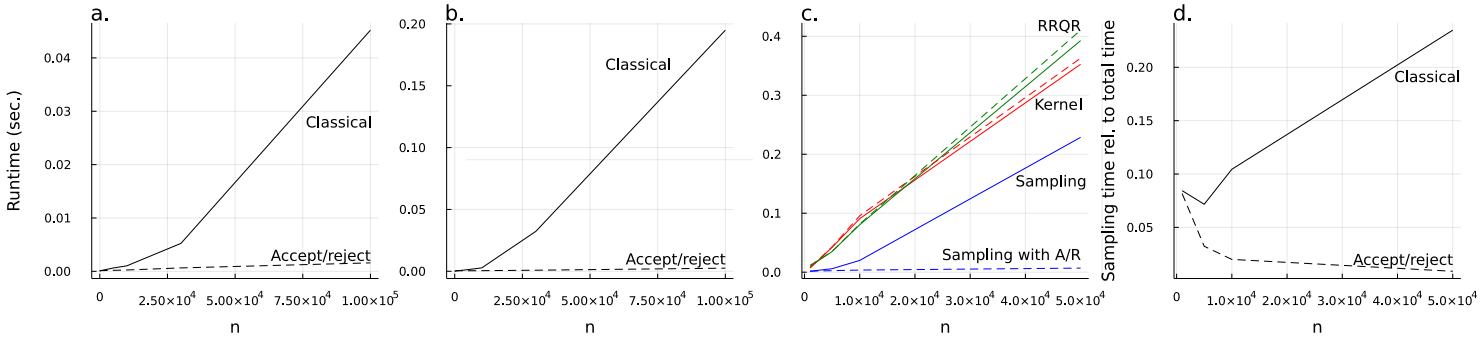

Figure 3.1: *Left panels*: median time needed to sample a projection DPP, using the standard approach (Alg. 3) vs. A/R (Alg. 4). For two values of m: a) $m = 30$, b) $m = 60$. Note that here the time taken to compute an orthogonal basis is not taken into account (see text). *Right panels*: simulation of a full workload that includes feature generation and orthogonalisation (see text). c) time taken by each step in the computation as a function of n. Solid lines are runtimes when sampling using the classical method, dashed, when using A/R. The first two steps (labeled "kernel" and "RRQR", in red and green respectively) are identical (up to time measurement noise). A/R sampling becomes beneficial at around $n = 1,000$ and is orders of magnitude faster at $n = 100,000$. d) Sampling time related to total computation time, as a function of n.

for each trial, the expected cost of the *while* loop at step t scales as $\mathcal{O}\left(\frac{m^2 t}{m-t+1}\right)$. Summing this over t:

$$\sum_{t=1}^{m} \frac{m^2 t}{m-t+1} \leq m^3 \sum_{t=1}^{m} \frac{1}{m-t+1}$$

yields a total expected cost scaling as[2] $\mathcal{O}(m^3 \log m)$.

We obtain the following theorem.

Theorem 19. *Alg. 4 samples a projection DPP, with an expected runtime scaling as* $\mathcal{O}(nm + m^3 \log m)$. *Also, any additional sample from the same DPP can be obtained in an extra* $\mathcal{O}(m^3 \log m)$ *expected runtime.*
Moreover, these expected runtimes are representative. Indeed, $\forall \delta \in \left(0, \frac{1}{2}\right)$, *the total number of proposals* $R = \sum_{t=1}^{m} R_t$ *satisfies, with probability greater than* $1 - \delta$*:*

$$R \leq 2m \log m + 3m \log \frac{1}{\delta}$$

3.2.2 Empirical validation

We compare the Accept/Reject algorithm (Alg. 4) to its classical counterpart (Alg. 3) for different values of n and m. Both algorithms are implemented in the Julia lan-

[2] $\sum_{t=1}^{m} \frac{1}{m-t+1} = \sum_{t=1}^{m} \frac{1}{t}$ scales as $\mathcal{O}(\log m)$: see, *e.g.*, Chapter 6 of [Abramowitz & Stegun 1964]

guage and are publicly available[3]. For each value of n, we sample a random projection matrix of size $n \times m$ (via QR decomposition of a matrix with Gaussian entries). We then pre-compute π, and run each algorithm 100 times. Fig. 3.1 a) and b) show the measured median runtimes.

For very small values of n, the classical algorithm is faster, which can be explained by the efficiency of BLAS calls. At each step the whole conditional distribution is computed, the main cost being a matrix multiplication (*i.e.*, a BLAS call), which benefits from efficient multithreaded code. However, the different asymptotic scalings ($\mathcal{O}(nm^2)$ vs. $\mathcal{O}(m^3 \log m)$, as we have pre-computed π in this experiment) soon makes the classical algorithm uncompetitive.

An experiment in a realistic data processing pipeline. Comparing computation times of projection DPPs for which Q is already computed, is *not* a realistic scenario: one very often needs to compute Q in the first place! As we have seen in the background section, sampling is only one of the steps in a processing pipeline that involves feature computation, orthogonalisation, *etc.* So, we illustrate here to what extent our algorithm accelerates sampling in a full data processing pipeline. For instance, a natural idea to produce a diverse set X from $\Omega = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n)$ n points in $\mathbb{R}^d$, is to draw $X \sim \texttt{Lens}(\mathsf{L})$ with L the Gaussian kernel of Eq. (3.1): $\mathsf{L}_{ij} = \gamma \exp\left(-\frac{1}{\sigma^2} \|\boldsymbol{x} - \boldsymbol{y}\|^2\right)$, where γ is a tuning parameter that determines the expected size. Producing an exact sample from this DPP requires the eigendecomposition of L which is impractical (even computing L is prohibitive for large n!).

However, Gaussian kernel matrices have rapidly decaying spectra (see, *e.g.*, [Wathen & Zhu 2015]) and this can be exploited. It implies in particular that L-ensembles associated to Gaussian kernels are well approximated by projection DPPs with kernels P_m where P_m projects onto the dominant eigenspace of L of order m. Thus, all we need is a good basis for the dominant eigenspace. Methods from randomised linear algebra offer good practical tools ("range finders") to obtain a basis for such a space [Martinsson & Tropp 2020]. For these simulations, we used the following approximation:

1. Select (and compute) $5m$ columns uniformly from L. Call this matrix A. We call this the "kernel step".

2. Use Rank-Revealing QR (RRQR, [Chan 1987]) and random projections, as implemented in the Julia package LowRankApprox.jl, to produce Q, an orthonormal matrix of size $n \times m$ that approximates the image of A. We call this the "RRQR step".

3. Sample a DPP with projection kernel $\mathsf{Q}\mathsf{Q}^\top$ using either the classical or the A/R algorithm.

[3] we've added a folder in our Determinantal.jl repository containing the code necessary to reproduce the figures, available here: `https://github.com/dahtah/Determinantal.jl/tree/main/misc/sampling_paper`. In addition, the A/R sampler is available as part of the Determinantal.jl software package, `https://github.com/dahtah/Determinantal.jl`.

We set $m = 100$ and time each step. This results in a total runtime of around 1.2 sec. at $n = 10^5$ with the A/R sampler, which challenges the notion that DPPs are very slow to sample from. With this procedure, the time spent sampling the actual DPP goes up to 20% of total time for the classical algorithm at $n = 10^5$, but using the A/R sampler sampling time becomes negligible. See Fig. 3.1 c) and d) to see how these times vary with n.

3.2.3 What now?

On top of the improvement on the sampling time of DPPs, our results imply the following intriguing by-product. Let X be a projection DPP of size m. A set of $\mathcal{O}(m \log m)$ points sampled iid from the inclusion probability distribution $\mathbb{P}(x \in X) = \pi(x)/m$ (also known as *leverage scores*) contains with high probability a realisation from the DPP. The consequences of this fact are worth discussing. First, let us state the result a bit more formally.

Definition 7. *Let X be a DPP on Ω and $\mathcal{Y} \subseteq \Omega$. We call $\Phi(\mathcal{Y})$ a* thinning algorithm *if it returns a subset of $\mathcal{Y}$. Moreover, we say $\Phi(\mathcal{Y})$ is* successful *when it returns a realisation from X.*

Corollary 20. *Let X be a projection DPP of size m, and Y be a set of iid points sampled with replacement with probability proportional to the leverage scores: $\mathbb{P}(x \in X) = \pi(x)/m$. Let $\delta \in (0, 1/2)$. A simple modification of Alg. 4 gives a thinning algorithm Φ that verifies: $\Phi(Y)$ is successful with probability greater than $1 - \delta$ provided that $|Y| \geq 2m \log m + 3m \log\left(\frac{1}{\delta}\right)$.*

Note that this is a substantial improvement over the work of [Derezinski *et al.* 2019], which gives this result only for $|Y| \geq \mathcal{O}(m^2)$ iid points.

A natural question is to ask if this result is optimal: can we find a thinning algorithm that succeeds with high probability for even smaller i.i.d sets? The answer is no in general (proofs of Corrollary 20 and of the following proposition are in [Barthelme *et al.* 2023a])

Proposition 21. *Corollary 20 is optimal in the following sense. Let X and Y be as previously. There does not exist a generic thinning algorithm able to succeed with fixed non-null probability if $|Y| = o(m \log m)$.*

This transition occuring at $\mathcal{O}(m \log m)$ calls for discussion, and paves the way to future interesting lines of research. First of all, Corollary 20 shows, from an original angle, that the repulsiveness of DPPs is weak. Indeed, other repulsive processes such as hard-core processes cannot verify such property in all generality. For instance, in the high density limit of a hard-sphere model, the probability that the position of m non-overlapping spheres can be found within a set of only $\mathcal{O}(m \log m)$ iid points drawn uniformly, tends to 0. In addition, these results ask the following question: in what cases should one pay the extra cost of sampling m elements from a DPP, rather than simply sampling $\mathcal{O}(m \log m)$ elements iid from the leverage score distribution?

Of course, when the objective is to sample a diverse set for visualization purposes, such as in search engines, it is always worthwhile to sample the DPP. However, in the case of integration [Bardenet & Hardy 2020, Coeurjolly *et al.* 2021]; or in the case of coresets [Tremblay *et al.* 2019], when the objective is to reduce the size for computational purposes, the answer is not so clear and requires further investigation.

3.3 Filling a gap in the theory: extended L-ensembles

We now move on to the second DPP contribution I would like to discuss in this chapter: this contribution is theoretical in nature. From the background section 3.1, let us recall the following, as this will help us precisely describe where this contribution stands.

Concerning DPPs, we observe that i/ DPPs whose marginal kernel do not contain any eigenvalue equal to 1 are L-ensembles and thus have an explicit joint probability distribution (using Lemma 6 and Eq.(3.3)), ii/ projection DPPs, that is, DPPs whose marginal kernel have eigenvalues equal to exactly 0 or 1, are fixed-size L-ensembles, and thus also have an explicit joint probability distribution (using Lemma 14 and Eq. (3.6)). The current state-of-the-art however does not provide[4] easy-to-use explicit joint distributions for DPPs whose marginal kernels lie in-between those two cases: DPPs whose marginal kernel have some eigenvalues equal to 1 and others lying strictly between 0 and 1. In other words: the class of DPPs that are *not* L-ensembles. According to Lemma 9, those are the cases where the size of the DPP is the sum of a deterministic part (the number of eigenvalues equal to 1) and a random part. As these DPPs did not even have a name prior to our work, we named them *partial-projection DPPs* for reasons that will become clear by the end of this section.

Concerning fixed-size DPPs, we observe that all fixed-size L-ensembles are fixed-size DPPs but the contrary is false. In other words, the current state-of-the-art does not provide explicit joint distributions for those fixed-size DPPs that are not fixed-size L-ensembles. These are the partial-projection DPPs conditioned on size, that we naturally call fixed-size partial-projection DPPs.

Our main contribution is to provide a unifying framework to write handy, explicit joint distributions for *all* DPPs and fixed-size DPPs and thus to fill in the holes of the current theory. To this end, we introduce *extended L-ensembles*, a novel way of representing the class of DPPs. Before we give their formal definition (Definition 10) we need a few preliminary results.

3.3.1 Conditionally positive (semi-)definite matrices

L-ensembles are naturally formed from positive semi-definite matrices, because L being positive semi-definite is a sufficient condition for $\det \mathsf{L}_X$ being non-negative. Ex-

[4] A formula due to [Macchi 1975] exists in this case but it is unwieldy. See also the discussion around Corollary 1.D.3 in [Gautier 2020a].

tended L-ensembles can accomodate a broader set of matrices called *Conditionnally Positive Semi-Definite* (CPSD) matrices.

Definition 8. *A matrix* $\mathsf{L} \in \mathbb{R}^{n\times n}$ *is called conditionally positive (semi) definite with respect to a rank* $p \geq 0$ *matrix* $\mathsf{V} \in \mathbb{R}^{n\times p}$ *if* $\boldsymbol{x}^\top \mathsf{L}\boldsymbol{x} > 0$ *(resp.,* $\boldsymbol{x}^\top \mathsf{L}\boldsymbol{x} \geq 0$*) for all* $\boldsymbol{x}$ *verifying* $\mathsf{V}^\top \boldsymbol{x} = 0$*.*

Remark 22. *Note that we authorize* $p = 0$ *in the definition: in this case, the definition simply boils down to that of positive semi-definite matrices.*

The set of vectors such that $\mathsf{V}^\top \boldsymbol{x} = 0$ is the space orthogonal to the span of V, which we denote be $\mathrm{orth}(\mathsf{V})$. The CPSD requirement may be read as a requirement for L to be PSD within $\mathrm{orth}(\mathsf{V})$. Positive-definite matrices are therefore also conditionally positive-definite, but matrices with negative eigenvalues may also be conditionally positive-definite.

Proposition 23. *Let* L *be a CPSD matrix with respect to* $\mathsf{V} \in \mathbb{R}^{n\times p}$*, where* V *has full column rank. Let* $\mathsf{Q} \in \mathbb{R}^{n\times p}$ *designate an orthonormal basis for* $\mathrm{span}\,\mathsf{V}$*, so that* $\mathsf{I} - \mathsf{Q}\mathsf{Q}^\top$ *is the orthogonal projection on* $\mathrm{orth}\,\mathsf{V}$*. Let* $\widetilde{\mathsf{L}} = (\mathsf{I} - \mathsf{Q}\mathsf{Q}^\top)\mathsf{L}(\mathsf{I} - \mathsf{Q}\mathsf{Q}^\top)$*.* $\widetilde{\mathsf{L}}$ *is symmetric, thus diagonalisable in* $\mathbb{R}$*. Moreover, all its eigenvalues are non-negative.*

The following example of a CPSD matrix is classical (but surprising), and is a special case of a class of conditionally positive definite kernels studied in [Micchelli 1986].

Example 24 ([Micchelli 1986])**.** *Let*

$$\mathsf{D} = [\|\boldsymbol{x}_i - \boldsymbol{x}_j\|]_{i,j}$$

the distance matrix between n *points in* $\mathbb{R}^d$*. Then* $-\mathsf{D}$ *is conditionally positive definite with respect to* $\mathsf{V} = \mathbf{1}_n$ *the all-ones vector.*

Some extensions of this example can be found in our paper [Tremblay *et al.* 2023].

3.3.2 Nonnegative Pairs

The central object when defining extended L-ensembles is what we call a Non Negative Pair (NNP).

Definition 9. *A Nonnegative Pair, denoted by* $(\mathsf{L};\mathsf{V})$*, is a pair* $\mathsf{L} \in \mathbb{R}^{n\times n}$*,* $\mathsf{V} \in \mathbb{R}^{n\times p}$*,* $0 \leq p \leq n$*, such that* L *is symmetric and CPSD with respect to* V*, and* V *has full column rank. Wherever a NNP* $(\mathsf{L};\mathsf{V})$ *appears below, we consistently use the following notation:*

- $\mathsf{Q} \in \mathbb{R}^{n\times p}$ *is an orthonormal basis of* $\mathrm{span}\,\mathsf{V}$*, such that* $\mathsf{I} - \mathsf{Q}\mathsf{Q}^\top$ *is a projector on* $\mathrm{orth}\,\mathsf{V}$

- $\widetilde{\mathsf{L}} = (\mathsf{I} - \mathsf{Q}\mathsf{Q}^\top)\mathsf{L}(\mathsf{I} - \mathsf{Q}\mathsf{Q}^\top) \in \mathbb{R}^{n\times n}$. *From Proposition 23, we know that all its eigenvalues are non-negative. We will denote by* q *the rank of* $\widetilde{\mathsf{L}}$. *Note that* $q \leq n - p$ *as the* p *columns of* Q *are trivially eigenvectors of* $\widetilde{\mathsf{L}}$ *associated to* 0. *We write*

$$\widetilde{\mathsf{L}} = \widetilde{\mathsf{U}}\widetilde{\mathbf{\Lambda}}\widetilde{\mathsf{U}}^\top$$

its truncated spectral decomposition; where $\widetilde{\mathbf{\Lambda}} = diag(\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_q) \in \mathbb{R}^{q\times q}$ *is the diagonal matrix of nonzero eigenvalues and* $\widetilde{\mathsf{U}} \in \mathbb{R}^{n\times q}$ *the corresponding eigenvector matrix of* $\widetilde{\mathsf{L}}$.

Remark 25. *Again, note that we authorize* $p = 0$ ($\mathsf{V} = \emptyset$) *in the definition: in this case,* $\widetilde{\mathsf{L}}$ *boils down to* L.

3.3.3 DPPs via extended L-ensembles

Definition 10 (Extended L-ensemble)**.** *Let* $\left(\mathsf{L}; \mathsf{V}\right)$ *be a NNP. An extended L-ensemble* X *based on* $\left(\mathsf{L}; \mathsf{V}\right)$, *denoted by* $X \sim \texttt{ExtLens}\left(\mathsf{L}; \mathsf{V}\right)$, *is a point process verifying:*

$$\forall \mathcal{X} \subseteq \Omega, \qquad \mathbb{P}(X = \mathcal{X}) \propto (-1)^p \det \begin{pmatrix} \mathsf{L}_\mathcal{X} & \mathsf{V}_{\mathcal{X},:} \\ (\mathsf{V}_{\mathcal{X},:})^\top & \mathbf{0}_p \end{pmatrix}. \tag{3.8}$$

This indeed defines a valid distribution as the right-hand side of Eq. (3.8) can be proven to always be non-negative.

Remark 26. *We stress that an extended L-ensemble reduces to an L-ensemble only in the case* $p = 0$. *If* $p \geq 1$, *it can be shown that an extended L-ensemble is not an L-ensemble.*

Two of our main results are the following theorems showing the equivalence between the class of DPPs and the class of extended L-ensembles.

Theorem 27. *Let* $\left(\mathsf{L}; \mathsf{V}\right)$ *be a NNP, and* $X \sim \texttt{ExtLens}\left(\mathsf{L}; \mathsf{V}\right)$. *Then,* X *is a DPP with marginal kernel*

$$\mathsf{K} = \mathsf{Q}\mathsf{Q}^\top + \widetilde{\mathsf{L}}(\mathsf{I} + \widetilde{\mathsf{L}})^{-1}. \tag{3.9}$$

The converse is also true: any DPP is an extended L-ensemble:

Theorem 28. *Let* $\mathsf{0} \preceq \mathsf{K} \preceq \mathsf{I}$ *be a marginal kernel and* $X \sim \texttt{DPP}(\mathsf{K})$. *Denote by* $\mathsf{V} \in \mathbb{R}^{n\times p}$ *the matrix concatenating the* $p \geq 0$ *orthonormal eigenvectors of* K *associated to eigenvalue* 1 *and* $\mathsf{L} = \mathsf{K}\left(\mathsf{I} - \mathsf{K}\right)^\dagger$ *with* $\dagger$ *representing the Moore-Penrose pseudo-inverse. Then,* X *is an extended L-ensemble based on the NNP* $\left(\mathsf{L}; \mathsf{V}\right)$.

The main argument to prove these theorems is a generalization of the Cauchy-Binet formula (see Theorem 3.1 in [Tremblay *et al.* 2023] for more details).

As a consequence, we readily obtain the explicit joint distributions of all DPPs:

Corollary 29. *Let* $\mathsf{0} \preceq \mathsf{K} \preceq \mathsf{I}$ *be a marginal kernel and* $X \sim \texttt{DPP}(\mathsf{K})$*. Let* $\left(\mathsf{L}; \mathsf{V}\right)$ *be the NNP as defined in theorem 28. Then*

$$\forall \mathcal{X} \subseteq \Omega, \qquad \mathbb{P}(X = \mathcal{X}) \propto (-1)^p \det \begin{pmatrix} \mathsf{L}_{\mathcal{X}} & \mathsf{V}_{\mathcal{X},:} \\ (\mathsf{V}_{\mathcal{X},:})^\top & \mathbf{0}_p \end{pmatrix}. \tag{3.10}$$

Recalling that, as per definition 4, a fixed-size DPP is simply a DPP conditioned on size, we also obtain the following explicit expression of the probability mass function of any fixed-size DPP:

Corollary 30. *Let* $\mathsf{0} \preceq \mathsf{K} \preceq \mathsf{I}$ *be a marginal kernel and* $X \sim |\texttt{DPP}|_m(\mathsf{K})$*. Let* $\left(\mathsf{L}; \mathsf{V}\right)$ *be the NNP as defined in theorem 28. Then*

$$\forall \mathcal{X} \subseteq \Omega, \qquad \mathbb{P}(X = \mathcal{X}) \propto (-1)^p \det \begin{pmatrix} \mathsf{L}_{\mathcal{X}} & \mathsf{V}_{\mathcal{X},:} \\ (\mathsf{V}_{\mathcal{X},:})^\top & \mathbf{0}_p \end{pmatrix} \mathbb{I}(|\mathcal{X}| = m). \tag{3.11}$$

Remark 31. *The normalization constants in Eqs. (3.10) and (3.11) are tractable. See our paper [Tremblay et al. 2023] for more details.*

Extended L-ensembles and complements of DPPs. A known (see, *e.g.*, [Kulesza & Taskar 2012], section 2.3) result about DPPs is that the complement of a DPP in Ω is also a DPP:

Theorem 32. *Let* $X \sim \texttt{DPP}(\mathsf{K})$*. Then the complement of* X*, denoted by* X^c*, is also a DPP, and its marginal kernel is* $\mathsf{I} - \mathsf{K}$*.*

Thanks to our framework, we can easily write the equivalent for L-ensembles:

Corollary 33. *Let* $X \sim \texttt{Lens}(\mathsf{L})$*, with* L *a rank* r *matrix and* $r \leq n$*. Then* $X^c \sim \texttt{ExtLens}\left(\mathsf{L}^\dagger; \mathsf{V}\right)$ *with* V *a basis for* $\text{orth}(\mathsf{L})$*, where* $\dagger$ *stands for the Moore-Penrose pseudo-inverse. In particular, if* $r = n$ *(if* L *is full rank), we have* $X^c \sim \texttt{Lens}(\mathsf{L}^{-1})$*.*

This is not a novel result but is is a remarkably concise way of putting it. We also have formulas to generalize this to fixed-size DPPs –that are novel up to our knowledge– but they are bit more involved and left out of this document.

An example of partial projection DPP: roots of random spanning forests. Recall that we call partial projection DPPs those DPPs that are not L-ensembles (all extended L-ensembles verifying $p \geq 1$). We give here an example of partial projection DPP, that is very much linked to the objects of interest of the next chapter (chapter 4): random spanning forests (that I call Kichhoff forests in the next chapter). It is known (*e.g.*, [Avena *et al.* 2017]) that the roots of the trees in a Kirchhoff forest over a graph with n nodes and Laplacian[5] $\mathcal{L}$ are distributed according to a DPP with marginal kernel $\mathsf{K} = q(q\mathsf{I} + \mathcal{L})^{-1}$ for some real parameter $q > 0$. Figure 3.2 illustrates what a spanning forest over a graph is. Let us denote

[5]The notation L is already taken in this chapter for the kernel matrix of L-ensembles, so I exceptionally use the notation $\mathcal{L}$ to refer to the standard Laplacian matrix of a graph.

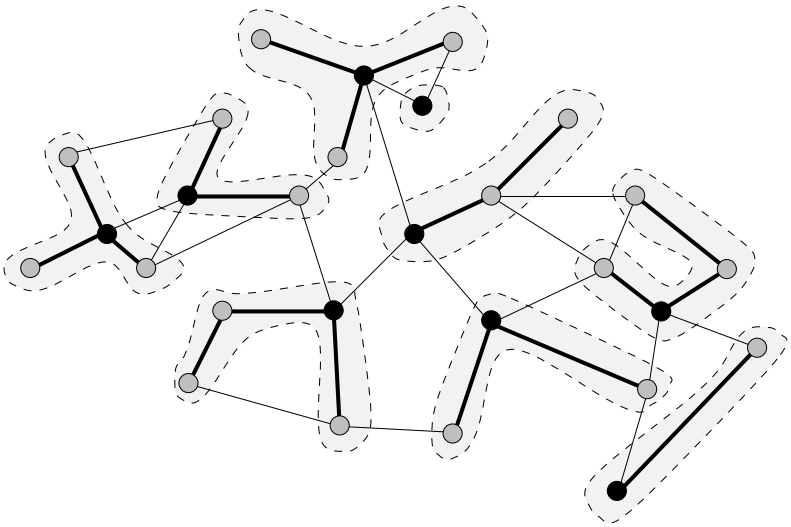

Figure 3.2: Roots of uniform random forests over a graph are distributed according to a partial projection DPP. Vertices or nodes are depicted in gray; edges as thin lines. A random forest is depicted : its trees are surrounded by light gray zones; edges of the trees are thicks black lines; roots are the black nodes. The forest is *spanning* the graph as each node of the graph is spanned by the forest.

as $\lambda_1 \leq \ldots \leq \lambda_n$ the eigenvalues of $\mathcal{L}$, and $\{\boldsymbol{u}_i\}_i$ its associated set of orthonormal eigenvectors. It is well known that $\lambda_1 = 0$ for any graph [Chung 1997]: K has thus at least one eigenvalue equal to 1 and, as such, the associated DPP is not an L-ensemble: it is a partial-projection DPP. It can thus be described by an extended L-ensemble:

Proposition 34. *The set of roots in a uniform random spanning forest over a connected graph with Laplacian $\mathcal{L}$ is distributed according to a partial projection DPP with NNP $(q\mathcal{L}^\dagger; \mathbf{1})$.*

Remark 35. *This example also provides a nice illustration for the properties of complements we have seen. Since $\mathcal{L}$ is a positive-definite matrix, we may define $Y \sim \texttt{Lens}(\frac{1}{q}\mathcal{L})$. According to Corollary 33, the complement of Y verifies $Y^c \sim \texttt{ExtLens}\left(q\mathcal{L}^\dagger; \mathbf{1}\right)$, which from the result above corresponds to the root process. Y therefore samples every node except the roots of a random forest on the graph.*

3.3.4 An application: perturbative limits of L-ensembles

Partial-projection DPPs arise as limits of certain L-ensembles, and this is in fact one of the main reason we studied them in the first place. We exhibit here one such limit: the L-ensemble based on the linear perturbation of a (low-rank) positive semi-definite matrix; *i.e.*, we consider L-ensembles based on matrices of the form:

$$\mathsf{L}_\varepsilon \triangleq \varepsilon \mathsf{A} + \mathsf{V}\mathsf{V}^\top \tag{3.12}$$

where $\mathsf{A} \in \mathbb{R}^{n\times n}$ has full rank n, $\mathsf{V} \in \mathbb{R}^{n\times p}$ has full column rank $p < n$, and ε is the parameter that will tend to 0.

Thus L_ε defined in (3.12) is a regular matrix pencil. One should think about this scenario as constructing a kernel as a sum of (a) a few important features contained in $\mathsf{V}\mathsf{V}^\top$ and (b) a generic kernel in A.

Limit of fixed-size L-ensembles based on $\varepsilon\mathsf{A}+\mathsf{V}\mathsf{V}^\top$. We begin with the more straightforward fixed-size case. We seek the limiting process $X_\star$ of $X_\varepsilon \sim |\texttt{Lens}|_m(\mathsf{L}_\varepsilon)$ as $\varepsilon \to 0$. The following theorem establishes the limiting distribution:

Theorem 36. *Let $X_\varepsilon \sim |\texttt{Lens}|_m(\mathsf{L}_\varepsilon)$ with L_ε as Eq. (3.12). The limiting process is:*

$$X_\varepsilon \to X_\star \sim \begin{cases} |\texttt{Lens}|_m(\mathsf{V}\mathsf{V}^\top), & \text{if } \ m \leq p \\ |\texttt{ExtLens}|_m\left(\mathsf{A};\mathsf{V}\right), & \text{if } \ m > p. \end{cases}$$

Limits of L-ensembles based on $\varepsilon\mathsf{A} + \mathsf{V}\mathsf{V}^\top$. The variable-size version of the results requires a bit more care. In fixed-size L-ensembles, the law of X is invariant to a rescaling of the positive semi-definite matrix it is based on: $X \sim |\texttt{Lens}|_m(\mathsf{L})$ is equivalent to $X \sim |\texttt{Lens}|_m(\alpha\mathsf{L})$ for any $\alpha > 0$. For regular (variable-size) DPPs this is not true. That feature both enriches and complicates a little the asymptotic analysis.

Let us start with a straightforward limit, namely $X_\varepsilon \sim \texttt{Lens}(\mathsf{L}_\varepsilon)$:

Proposition 37. *Let $X_\varepsilon \sim \texttt{Lens}(\mathsf{L}_\varepsilon)$. Then the limiting process $X_\star$ is $X_\star \sim \texttt{Lens}(\mathsf{V}\mathsf{V}^\top)$.*

The result is not very surprising. It has a noteworthy consequence, which is that as $\varepsilon \to 0$, the expected sample size will be bounded by p from above.

If we wish to sample a larger number of points on average, then it appears that we are out of luck. We may instead look at a more interesting limit: instead of taking L_ε, we will now take

$$\mathsf{L}'_\varepsilon = \varepsilon^{-1}\mathsf{L}_\varepsilon = \mathsf{A} + \varepsilon^{-1}\mathsf{V}\mathsf{V}^\top,$$

which carries the same intuition of giving more importance to $\mathsf{V}\mathsf{V}^\top$ than A.

Proposition 38. *Let $X_\varepsilon \sim \texttt{Lens}(\mathsf{A} + \varepsilon^{-1}\mathsf{V}\mathsf{V}^\top)$. Then the limiting process is $X_\star \sim \texttt{ExtLens}\left(\mathsf{A};\mathsf{V}\right)$.*

Importantly, the expected sample size of this rescaled L-ensemble allows for sample sizes larger than p.

Example 39. *We return to the distribution of roots in a random forest, and show that it arises naturally as a limit. The "resistance distance" in a graph (see for instance [Klein & Randić 1993]) between two nodes i and j is defined as:*

$$\mathsf{D}_{ij} = \mathcal{L}^\dagger_{ii} + \mathcal{L}^\dagger_{ii} - 2\mathcal{L}^\dagger_{ij}$$

where $\mathcal{L}^\dagger$ is the pseudo-inverse of the Laplacian. The formula is analoguous to $\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 = \boldsymbol{x}^\top\boldsymbol{x} + \boldsymbol{y}^\top\boldsymbol{y} - 2\boldsymbol{x}^\top\boldsymbol{y}$, and in that sense the resistance distance is actually a squared distance. A graph kernel can be constructed from the resistance distance as $\mathsf{L}_{ij} = \exp(-\mathsf{D}_{ij})$, analoguously to the Gaussian kernel. If we parametrise the kernel as $\mathsf{L}_\varepsilon(i,j) = \varepsilon^{-1}\exp(-\varepsilon\mathsf{D}_{ij})$, then by taking a Taylor series in ε we have

$$\mathsf{L}_\varepsilon = \varepsilon^{-1}\mathbf{1}\mathbf{1}^\top - \mathsf{D} + \mathcal{O}(\varepsilon)$$

Let $X_\varepsilon \sim \texttt{Lens}(\mathsf{L}_\varepsilon)$, a DPP on the nodes on the graph. Then using Proposition 38 we find that the limit of X_ε is the roots process $X_\star \sim DPP\left(\boldsymbol{\mathcal{L}}^\dagger; \mathbf{1}\right)$, as discussed in and around Proposition 34.

The interested reader on the topic of such flat limits of DPPs is referred to [Barthelmé *et al.* 2023b], in which we study flat limits of general DPPs (and not only flat limits of a matrix pencil), making extensive use of the extended L-ensemble formalism.

3.3.5 What now?

This work does not necessarily call for future work *per se*, as the gap it was meant to fill is now filled. However, the jungle of definitions existing around DPPs (DPPs, L-ensembles, now partial-projection DPPs, as well as their fixed-size versions) has become unfortunately difficult to navigate. This is due to historical reasons and the step-by-step progression of this field. However, the result is that it has become a non-inviting family of random processes for the beginner. I honestly think that, on our way to provide a unified picture of DPPs and L-ensembles, our paper [Tremblay *et al.* 2023] failed to give a clearer picture of these processes. I have in mind to, one day, attempt to write a paper on DPPs that could be both pedagogical and complete, with concrete and practical implementations. These processes deserve it.

3.4 An application: DPPs for coresets

The last contribution I would like to present in this chapter is a practical application of DPPs to sketching, in the computer science framework of *coresets*.

Given a learning task, if an algorithm is too slow on large datasets, one can either speed up the algorithm or reduce the amount of data. The theory of "coresets" gives theoretical guarantees on the latter option. A coreset is a weighted sub-sample of the original data, with the guarantee that for any learning parameter, the task's cost function estimated on the coreset is equal to the cost computed on the entire dataset up to a controlled relative error.

An elegant consequence of such a property is that one may run learning algorithms solely on the coreset, allowing for a significant decrease in the computational cost while guaranteeing almost-equal performance. Of course, the difficulty lies in sampling these coresets in the first place. There are many algorithms that produce coresets, with some tailored for a specific task (such as k-means, k-medians, logistic regression, etc.), and others more generic. Also, there exists coreset sampling strategies both for the streaming setting and the offline setting: we choose here to focus on the offline setting.

We refer the reader to the review [Munteanu & Schwiegelshohn 2017] for a good introduction to the various types of coreset construction techniques that have been developed in the past. Of the four families of construction techniques, our work is part of the family of *random sampling* strategies, along with, for instance: [Chen 2009, Langberg & Schulman 2010, Braverman *et al.* 2016, Bachem *et al.* 2017]. The state of the art for many different tasks such as k-means or k-median is currently via iid non-uniform sampling, recalled in Section 3.4.1.3. In this framework, and for optimal performance, the probability to sample an element should be set proportional to a quantity known as its *sensitivity* (introduced by [Langberg & Schulman 2010], see Definition 12 for its formal definition).

Independent processes being ignorant of the past, and thus liable to sample similar points repeatedly, an avenue for improvement is to produce samples that are less redundant than what iid sampling produces. A natural idea is to consider negatively correlated point processes, *i.e.*, point processes for which sampling jointly two similar datapoints is less probable than sampling two very different datapoints. Unsurprisingly given the context of this chapter, we propose to use DPPs to create coresets and to study their performance on generic tasks. To the best of our knowledge, we provide the first general coreset guarantee using non-iid random sampling.

In this document, I will discuss only the two following results we obtained in our work [Tremblay *et al.* 2019]:

- Theorem 41. If a fixed-size DPP is defined in such a way that its first order inclusion probability is proportional to the sensitivity, then its coreset performance is at least as good as in the iid case – regardless of the higher-order marginals of the fixed-size DPP.
- Theorem 43. A projection DPP sample necessarily has a lower variance than its iid sampling counterpart with same inclusion probabilities.

The other results we obtained are briefly summarized in Section 3.4.3.

3.4.1 Background

Let $\Omega = \{\boldsymbol{x}_1, \dots, \boldsymbol{x}_n\}$ be a set of n datapoints in $\mathbb{R}^d$. Let Θ be a parameter space endowed with a metric d_Θ. Let $\theta \in \Theta$. We consider cost functions of the form:

$$L(\Omega, \theta) = \sum_{\boldsymbol{x} \in \Omega} f(\boldsymbol{x}, \theta), \tag{3.13}$$

where f is a non-negative γ-Lipschitz function ($\gamma > 0$) with respect to θ, *i.e.*:

$$\begin{aligned} &\forall \boldsymbol{x} \in \Omega \qquad \forall \theta \in \Theta \qquad\quad f(\boldsymbol{x}, \theta) \geq 0, \\ &\forall \boldsymbol{x} \in \Omega \qquad \forall (\theta, \theta') \in \Theta^2 \qquad |f(\boldsymbol{x}, \theta) - f(\boldsymbol{x}, \theta')| \leq \gamma \; d_\Theta(\theta, \theta'). \end{aligned}$$

Many classical machine learning cost functions fall under this model: k-means, k-median, logistic or linear regression, support-vector machines, low-rank approximations of matrices, *etc.*

A standard learning task is to minimize the cost L over all $\theta \in \Theta$. We write:

$$\theta^{\mathrm{opt}} = \arg\min_{\theta \in \Theta} L(\Omega, \theta), \qquad L^{\mathrm{opt}} = L(\Omega, \theta^{\mathrm{opt}}) \qquad \text{and} \qquad \langle f \rangle_{\mathrm{opt}} = \frac{L^{\mathrm{opt}}}{n}. \quad (3.14)$$

In some instances of this problem, *e.g.*, if n is very large and/or if f is expensive to evaluate and should be computed as rarely as possible, one may rely on sampling strategies to accelerate this optimization task. This is where coresets come in.

3.4.1.1 Coresets

Let $\mathcal{X}$ be a subset of Ω (possibly with repetitions) of size m. To each element $\boldsymbol{x} \in \mathcal{X}$, associate a weight $\omega(\boldsymbol{x}) \in \mathbb{R}^+$. Define the following estimated cost associated to the weighted subset $\mathcal{X}$:

$$\hat{L}(\mathcal{X}, \theta) = \sum_{\boldsymbol{x} \in \mathcal{X}} \omega(\boldsymbol{x}) f(\boldsymbol{x}, \theta). \quad (3.15)$$

Definition 11 (Coreset)**.** *Let $\varepsilon \in (0, 1)$. The weighted subset $\mathcal{X}$ is a ε-coreset for L if,* for any parameter θ*, the estimated cost is equal to the exact cost up to a relative error, i.e.:*

$$\forall \theta \in \Theta \qquad \left| \frac{\hat{L}}{L} - 1 \right| \leq \varepsilon. \quad (3.16)$$

This is the so-called "strong" coreset definition, as the ε-approximation is required for all $\theta \in \Theta$. A weaker version of this definition exists in the literature where the ε-approximation is only required in a neighborhood of θ^{opt}. In the following, we focus on theorems guaranteeing the strong coreset property.

Let us write $\hat{\theta}^{\mathrm{opt}}$ the optimal solution computed on the weighted subset $\mathcal{X}$: $\hat{\theta}^{\mathrm{opt}} = \arg\min_{\theta \in \Theta} \hat{L}(\mathcal{X}, \theta)$. An important consequence of the coreset property is:

$$(1-\varepsilon) L(\Omega, \theta^{\mathrm{opt}}) \underset{(1)}{\leq} (1-\varepsilon) L(\Omega, \hat{\theta}^{\mathrm{opt}}) \underset{(2)}{\leq} \hat{L}(\mathcal{X}, \hat{\theta}^{\mathrm{opt}}) \underset{(3)}{\leq} \hat{L}(\mathcal{X}, \theta^{\mathrm{opt}}) \underset{(4)}{\leq} (1+\varepsilon) L(\Omega, \theta^{\mathrm{opt}}),$$

where inequality (1) is because θ^{opt} is the argmin of $L(\Omega, \theta)$, inequality (2) comes from the coreset property, inequality (3) is because $\hat{\theta}^{\mathrm{opt}}$ is the argmin of $\hat{L}(\Omega, \theta)$ and inequality (4) also comes from the coreset property. As a consequence one thus has:

$$(1-\varepsilon) L(\Omega, \theta^{\mathrm{opt}}) \leq \hat{L}(\mathcal{X}, \hat{\theta}^{\mathrm{opt}}) \leq (1+\varepsilon) L(\Omega, \theta^{\mathrm{opt}}).$$

In words, this means that running an optimization algorithm on the weighted sample $\mathcal{X}$ will result in a minimal learning cost that is a controlled ε-approximation of the learning cost one would have obtained by running the same algorithm on the entire data set Ω. Note that the guarantee is over costs only: the estimated optimal parameters $\hat{\theta}^{\text{opt}}$ and θ^{opt} may be different. Nevertheless, if the cost function is well suited to the problem: either there is one clear global minimum and the estimated parameters will almost coincide; or there are multiple solutions for which the learning cost is similar and selecting one over the other is not an issue.

In terms of computation cost, if the sampling scheme is efficient, n is very large and/or f is expensive to compute for each datapoint, coresets thus hold the promise of significant gains in computing time as the expensive optimization problem is now on m ($m \ll n$) elements rather than n originally.

3.4.1.2 Sensitivity

To define appropriate sampling schemes for coresets, authors in [Langberg & Schulman 2010] introduce the notion of sensitivity:

Definition 12 (Sensitivity)**.** *The sensitivity of a datapoint $\boldsymbol{x}_i \in \Omega$ with respect to a fuction $f : \Omega, \Theta \to \mathbb{R}^+$ is:*

$$\sigma_i = \max_{\theta \in \Theta} \frac{f(\boldsymbol{x}_i, \theta)}{L(\Omega, \theta)} \qquad \in [0, 1]. \tag{3.17}$$

Also, the total sensitivity is defined as $\mathfrak{S} = \sum_{i=1}^{n} \sigma_i$.

The sensitivity is related to the concept of statistical leverage score [Drineas & Mahoney 2018, Drineas *et al.* 2012], which plays a crucial role in iid random sampling theorems in the randomized numerical linear algebra literature [Mahoney 2011]. Both notions are similar, but not equivalent. For instance, we show in Lemma 25 of our paper [Tremblay *et al.* 2019] that sensitivities for the linear regression task are different from the usual definition of leverage score in this context. Thus, in general, leverage scores used in the randomized linear algebra literature are not sensitivities, *i.e.*, they do not necessarily verify Eq. (3.17).

In words, the sensitivity σ_i is the worse case contribution of datapoint x_i in the total cost. Informally, the larger it is, the larger its "outlierness" [Lucic *et al.* 2016].

3.4.1.3 iid importance sampling and state of the art results

Say the sample set X consists in m samples drawn iid with replacement from a (discrete) probability distribution $\boldsymbol{p} \in \mathbb{R}^n$ (with p_i the probability of sampling $\boldsymbol{x}_i$ at each draw, and $\sum_i p_i = 1$). Denote by ε_i the random variable counting the number of occurences of $\boldsymbol{x}_i$ in X. One may define $\hat{L}_{\text{iid}}$, the so-called importance sampling estimator of L, as :

$$\hat{L}_{\text{iid}}(X, \theta) = \sum_i \frac{f(\boldsymbol{x}_i, \theta)\varepsilon_i}{m p_i}. \tag{3.18}$$

One can show that $\mathbb{E}(\varepsilon_i) = mp_i$, such that $\hat{L}_{\text{iid}}$ is an unbiased estimator of L:

$$\mathbb{E}(\hat{L}_{\text{iid}}(X, \theta)) = L(\Omega, \theta).$$

The concentration of $\hat{L}_{\text{iid}}$ around its expected value is controlled by the following state of the art theorem:

Theorem 40 (Coresets with iid random sampling)**.** *Let $\boldsymbol{p} \in [0,1]^n$ be a probability distribution over all datapoints Ω with p_i the probability of sampling $\boldsymbol{x}_i$ and $\sum_i p_i = 1$. Draw m iid samples with replacement according to $\boldsymbol{p}$. Associate to each sample $\boldsymbol{x}_i$ a weight $\omega(\boldsymbol{x}_i) = 1/mp_i$. The weighted subset obtained is a ε-coreset with probability $1-\delta$ provided that:*

$$m \geq m^*$$

with

$$m^* = \mathcal{O}\left(\frac{1}{\varepsilon^2}\left(\max_i \frac{\sigma_i}{p_i}\right)^2 (d' + \log(1/\delta))\right),$$

where d' is the pseudo-dimension of Θ (a generalization of the Vapnik-Chervonenkis dimension). The optimal probability distribution minimizing m^* is $p_i = \sigma_i/\mathfrak{S}$. In this case, the weighted subset is a ε-coreset with probability $1-\delta$ provided that:

$$m \geq \mathcal{O}\left(\frac{\mathfrak{S}^2}{\varepsilon^2}(d' + \log(1/\delta))\right).$$

For instance, in the k-means setting[6], $d' = dk\log k$ and $\mathfrak{S} = \mathcal{O}(k)$ such that the coreset property is guaranteed with probability $1-\delta$ provided that:

$$m \geq \mathcal{O}\left(\frac{k^2}{\varepsilon^2}(dk\log k + \log(1/\delta))\right).$$

This theorem is taken from [Bachem *et al.* 2017]. Its original form goes back to [Langberg & Schulman 2010]. Note that sensitivities cannot be computed rapidly. It is much worse than that in fact: prior to our work[7], no analytical form was even known for the sensitivity, for any choice of f. This theorem is thus unpractical. Thankfully, bi-criteria approximation schemes (such as Alg. 2 of [Bachem *et al.* 2017], or other propositions such as in [Feldman & Langberg 2011, Makarychev *et al.* 2016]) may be used to efficiently work around this problem via (computable) upper bounds of the sensitivity.

[6] In the literature [Feldman & Langberg 2011, Balcan *et al.* 2013], d' is often taken to be equal to dk in the k-means setting. We nevertheless agree with [Bachem *et al.* 2017] and their discussion in Section 2.6 regarding k-means' pseudo-dimension and thus write $d' = dk\log k$

[7] We slighlty changed that state-of-the-art by obtaining an analytical expression of the sensitivity in the 1-means and linear regression contexts.

3.4.1.4 Correlated importance sampling

Eq. (3.18) is not suited to correlated sampling and, in the following, we will use a slightly different importance sampling estimator, more adapted to this case. Consider a point process defined on Ω that outputs a random sample $X \subset \Omega$. For each data point $\boldsymbol{x}_i$, denote by π_i its inclusion (or marginal) probability:

$$\pi_i = \mathbb{P}\left(\boldsymbol{x}_i \in X\right). \tag{3.19}$$

Moreover, denote by ε_i the random Boolean variable such that $\varepsilon_i = 1$ if $\boldsymbol{x}_i \in X$, and 0 otherwise. In this paper, we focus on the following definition[8] of the importance sampling cost estimator $\hat{L}$:

$$\hat{L}(X, \theta) = \sum_i \frac{f(\boldsymbol{x}_i, \theta)\varepsilon_i}{\pi_i}. \tag{3.20}$$

By construction, $\mathbb{E}(\varepsilon_i) = \pi_i$, such that $\hat{L}$ is an unbiased estimator of L:

$$\mathbb{E}(\hat{L}(X, \theta)) = L(\Omega, \theta).$$

Studying the coreset property in this setting boils down to studying the concentration properties of $\hat{L}$ around its expected value.

3.4.2 DPP-based coreset theorems

We now detail parts of our main theoretical contributions. In Section 3.4.2.1, we present a coreset theorem for fixed-size DPPs providing sufficient conditions on the marginal probabilities $\{\pi_i\}_i$ to guarantee the coreset property. We will see that, similar to the iid case (Theorem 40), the optimal marginal probability should be set proportional to the sensitivity. This theorem is valid for any choice of higher order inclusion probabilities (the conditions are only on the first-order inclusion probabilities $\{\pi_i\}$). We further discuss in Section 3.4.2.2 how one may take advantage of these additional degrees of freedom encoding the negative correlations of DPPs to improve the coreset performance over iid sampling.

3.4.2.1 Determinantal Point Processes for coresets

We write here the simplified version of our theorem, that requires to suppose that the sensitivity is lower-bounded by $1/n$. One can show that this is in fact verified in the k-means case for instance. We could work with unconstrained $\sigma_{\min}$ if needed, but it requires much more cumbersome notations and with little effects on the main messages we would like to convey.

[8] Note that in fact $\hat{L}_{\text{iid}}$ and $\hat{L}$ are the same objects if one defines ε_i to be the number of times i is sampled (which will be in practice Boolean in the DPP case as the same sample can never be sampled twice in this context) and write $\hat{L}(X, \theta) = \sum_i \frac{f(\boldsymbol{x}_i, \theta)\varepsilon_i}{\mathbb{E}(\varepsilon_i)}$. We prefer to introduce both notations to avoid confusions.

Theorem 41 (fixed-size DPP for coresets)**.** *Suppose* $n\sigma_{min} \geq 1$*. Let* $X \sim \texttt{Lens}(\mathsf{L})$*,* $\varepsilon \in (0,1)$*, and* η *the minimal number of balls of radius* $\frac{\varepsilon\langle f\rangle_{opt}}{6\gamma}$ *necessary to cover* Θ*, with* γ *the Lipschitz parameter of* f*.* X *is a* ε*-coreset with probability larger than* $1-\delta$ *provided that:*

$$m \geq m^* = \frac{32}{\varepsilon^2}\left(\max_i \frac{\sigma_i}{\bar{\pi}_i}\right)^2 \log\frac{4\eta}{\delta} \tag{3.21}$$

with $\forall i, \bar{\pi}_i = \pi_i/m$*.*

Note that m^* is not independent of m as $\bar{\pi}_i = \pi_i/m$. While this formulation may be surprising at first, this is due to the fact that in non-iid settings, separating m from π_i is not as straightforward as in the iid case (in Theorem 40, m and p_i are independent). Also, we give this particular formulation of the theorem to mimic classical concentration results obtained with iid sampling.

Now, one would like to have the coreset guarantee for a minimal number of samples, that is: to find the marginal probabilities π_i minimizing m^*. A quick glance at Eq. (3.21) tells us to set $\pi_i = m\sigma_i/\mathfrak{S}$ in order to minimize the bound m^* while satisfying the constraint $\sum_i \pi_i = m$. One obtains:

Corollary 42. *If one sets the* π_i*'s such that* $\forall i, \pi_i = m\sigma_i/\mathfrak{S}$*, then* X *is a* ε*-coreset with probability at least* $1-\delta$ *provided that*

$$m \geq \frac{32}{\varepsilon^2}\mathfrak{S}^2 \log\frac{4\eta}{\delta}.$$

In practice, however, computing the sensitivities is often untractable and we derived in [Tremblay *et al.* 2019] results that consider only a bounded knowledge of the sensitivity. We will not delve into this here.

3.4.2.2 Links with the iid case and a variance argument

Let us first compare these results with Theorem 40 obtained in the iid setting. A few remarks are in order:

1. we need to compare the minimal number of samples obtained in the fixed-size DPP setting, $\frac{32\mathfrak{S}^2}{\varepsilon^2}(\log\eta + \log\frac{4}{\delta})$, with $\mathcal{O}(\frac{\mathfrak{S}^2}{\varepsilon^2}(d' + \log(1/\delta)))$ of Theorem 40, where d' is the pseudo-dimension of Θ. η being the number of balls of radius $\frac{\varepsilon\langle f\rangle_{\text{opt}}}{6\gamma}$ necessary to cover Θ, it will typically be $\frac{\varepsilon\langle f\rangle_{\text{opt}}}{6\gamma}$ to the power of the ambient dimension of Θ (analogous to d'). For instance, in the k-means case, $d' = dk\log k$ (see footnote 6), whereas, as we show in our paper [Tremblay *et al.* 2019], $\log\eta = dk\log\left(\frac{12\rho\gamma}{\varepsilon\langle f\rangle_{\text{opt}}} + 1\right)$ where ρ is the diameter of the minimum enclosing ball of the data Ω. Up to the log term, d' and $\log\eta$ are the same. The difference observed in the log term is due to the fact that coreset theorems in the iid case [Bachem *et al.* 2017] take advantage of powerful results from the Vapnik-Chervonenkis (VC) theory, as detailed in [Li *et al.* 2001]. Unfortunately, these fundamental results are valid in the

iid case only, and are not easily generalized to the correlated case. Possible improvements to reduce this small gap could take advantage of chaining arguments *à la* Talagrand such as in [Baraud 2010], in order to improve over the repeated loose union bounds we have used in the proof.

2. The DPP coreset theorem that we obtain is in a sense disappointing: it does not show that the concentration is tighter in the fixed-size DPP case than in the iid case. Our proof is in fact limited by the current state-of-the-art in concentration of strongly Rayleigh measures [Pemantle & Peres 2014]. On the bright side, our results take *only* into account first-order inclusion probabilities: the $\{\pi_i\}$'s; meaning that these results are valid for any choice of higher-order inclusion probabilities (encoding the correlation between samples). We will now see how these extra degrees of freedom enable to provably decrease the variance of the cost estimator, compared to the iid case.

A variance argument: improvement over the iid estimator. We now compare the variance of the iid estimator with replacement $\hat{L}_{\text{iid}}$ of Eq. (3.18) and the variance of the DPP estimator $\hat{L}$ of Eq. (3.20). To simplify, we consider a projection DPP with a rank m marginal kernel K verifying $\forall i \quad \pi_i = \mathsf{K}_{ii}$ the marginal probability of sampling element i. We compare the variance of $\hat{L}$ with such a DPP and the variance of $\hat{L}_{\text{iid}}$ with m independent draws with replacement with $p_i = \pi_i/m$ (in order to have a fair comparison).

Before we state the result: K being positive-semi definite and of rank m, there exists $\mathsf{V} = (\boldsymbol{v}_1|\boldsymbol{v}_2|\dots|\boldsymbol{v}_n) \in \mathbb{R}^{m\times n}$ a set of n vectors in dimension m such that $\mathsf{K} = \mathsf{V}^\top\mathsf{V}$. By construction, $\forall i \quad \|\boldsymbol{v}_i\|^2 = \mathsf{K}_{ii} = \pi_i$. For each vector $\boldsymbol{v}$, consider its diagram vector [Copenhaver *et al.* 2014, Definition 2.3], denoted by $\tilde{\boldsymbol{v}}$, defined as:

$$\tilde{\boldsymbol{v}} = \frac{1}{\sqrt{r-1}} \begin{bmatrix} v(1)^2 - v(2)^2 \\ \vdots \\ v(r-1)^2 - v(r)^2 \\ \sqrt{2r}\, v(1)v(2) \\ \vdots \\ \sqrt{2r}\, v(r-1)v(r) \end{bmatrix} \in \mathbb{R}^{m(m-1)}, \tag{3.22}$$

where the difference of squares $v(i)^2 - v(j)^2$ and the product $v(i)v(j)$ occur exactly once for $i < j, i = 1, 2, \cdots, r-1$.

Theorem 43. *One has:*

$$\forall \theta \in \Theta \qquad \mathrm{Var}(\hat{L}) = \mathrm{Var}(\hat{L}_{\text{iid}}) - \frac{m-1}{m} \left\| \sum_i \frac{f(\boldsymbol{x}_i, \theta)}{\pi_i} \tilde{\boldsymbol{v}}_i \right\|^2. \tag{3.23}$$

The variance is thus necessarily improved when using a projective DPP compared to its iid counterpart. This result is remarkable: the variance reduction is independent of the sign of f (supposed positive in the coreset context). This opens

interesting generalizing perspectives to a more general class of cost functions L.

To recap, we showed that a projective DPP sampling scheme has necessarily a lower variance than its iid counterpart *regardless of the choice of off-diagonal elements of* K, provided that K stays projective.

3.4.3 Other results not discussed here

I did not discuss the other theoretical results we obtained:

- We show that the previous results also hold in the DPP case, and not only the fixed-size DPP case.
- We show that DPP samples from particular polynomial kernels based on the Vandermonde matrix of the data asymptotically have a rebalancing property. For instance in the k-means setting, this rebalancing property ensures that, asymptotically, such a DPP produces samples in each cluster, even if some are much smaller than others.
- Of independent interest, we provide for the first time analytical formulas for the sensitivity, in two specific settings: the 1-means and the linear regression cases.

I did not discuss neither our empirical contributions: I will only briefly summarize them here. In the iid setting, for optimal performance, the probability of sampling an element should be set proportional to its sensitivity. In general, the sensitivity is not computable in polynomial time, thus out of reach in practice. For the specific 1-means and linear regression tasks, now that we have provided analytical formulas, these quantities become computable in polynomial time but turn out to be heavier to compute than solving the task on the whole data set –thus useless in practice. However, this setting serves as a strong performance basis we can compare our DPP algorithms to: it turns out that we outperform this performance basis in many settings, meaning that we outperform all the existing iid methods that attempt to circumvent the exact computation of the sensitivity (usually by computing bounds).

We apply our results to both the k-means and the linear regression problems where the initial data consists in n points in $\mathbb{R}^d$. Our theoretical analyses show that the ideal choice of kernel L for DPP sampling is untractable in practice, and we thus provide two efficient heuristics: one based on random Fourier features of the Gaussian kernel, the other on polynomial features. These propositions are of course computationally heavier than iid sampling, especially as m increases. We show nonetheless extensive empirical evidence demonstrating that this additional cost stays reasonable, given the enhanced performance it clearly provides –especially in the small m regime. Finally, a Julia toolbox called DPP4Coresets is available online[9].

[9]The DPP4Coresets toolbox is also available at https://gricad-gitlab.univ-grenoble-alpes.fr/tremblan/dpp4coresets.jl .

3.4.4 What now?

The weakness of our results compared to their iid counterparts are remarkably frustrating, given the improvement we observe empirically. The road towards improvement is either via tighter concentration results of DPPs or via a closer look at what the ideal DPP kernel looks like.

A link with tight frames In order to design the ideal marginal kernel K, and according to the previous discussion of the end of section 3.4.2.2, one wants K to verify:

- Theorem 43 suggests that K should be a projective DPP, that is: $\mathsf{K} = \mathsf{V}^\top\mathsf{V}$ with $\mathsf{V}\mathsf{V}^\top = \mathsf{I}_m$.
- Theorem 41 suggests to set $\pi_i = \mathsf{K}_{ii} = \frac{m\sigma_i}{\mathfrak{S}}$.

Finding such a marginal kernel boils down to finding $\mathsf{V} = (\boldsymbol{v}_1|\dots|\boldsymbol{v}_n)$ a set of n vectors in dimension m with specified norms $\|\boldsymbol{v}_i\|^2 = \pi_i$, such that $\sum_i \pi_i = m$ and $\mathsf{V}\mathsf{V}^\top = \mathsf{I}_m$. This is exactly the problem of finding a tight frame of n vectors in dimension m, with specified norms [Casazza & Kutyniok 2012]. Using the Schur-Horn theorem, one can show that:

Lemma 44. *Such a tight frame exists.*

Also, a tight frame not only exists, but several solutions exist in general, and efficient algorithms have been designed to build one [Tropp *et al.* 2004]. Out of all these possibilities, the ideal would be to find the tight frame that minimizes the variance of Eq.(3.23).

Up to our knowledge, this is an open and difficult question, rooted in frame theory: a fascinating avenue for future research directions on this topic!

Chapter 4

Kirchhoff forests

Contents

This chapter details a selection of my work at the intersection between the topics discussed in the two previous chapters, that is, at the intersection between graphs and DPPs. I indeed relate here my work around DPPs defined over nodes and/or edges of graphs.

The foundation on which this chapter is built upon is the *Uniform Spanning Tree* (UST). A UST of an unweighted and undirected graph[1] $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a random tree uniformly drawn from the set of all spanning trees of $\mathcal{G}$. The following properties of USTs are widely known.

A UST, seen as a sampling process of edges, is a DPP defined over $\mathcal{E}$, the edges of the graph. See for instance [Pemantle 2004] for details. As we have seen in the previous section, this implies that there exists a UST sampling algorithm running in

[1] there are variants for weighted and/or directed cases

$\mathcal{O}(|\mathcal{E}|^3)$: the time to perform the eigendecomposition of the $|\mathcal{E}| \times |\mathcal{E}|$ kernel matrix of the DPP. Now, the number of spanning trees in a graph growing very fast with its number of edges (the worst-case scenario, the complete graph of size n, has n^{n-2} spanning trees), a cubic sampling time in $|\mathcal{E}|$ seems already a very reasonable time to uniformly sample from such a large set of trees! This is notwithstanding the 90's during which a few researchers designed very elegant –and *very* efficient– sampling algorithms for USTs. In a beautiful paper that I invite anyone to read, Wilson [Wilson 1996] proposed an algorithm, known today as *Wilson's algorithm*, to do so based on loop-erased random walks, that runs in time (roughly) linear in the number of edges $|\mathcal{E}|$.

In this chapter, I detail results we obtained on a generalization of USTs that we call *Kirchhoff Forests (KFs)*. KFs are a type of random spanning forests, where we recall that a spanning forest of $\mathcal{G}$ is a union of disjoint trees that together span all the nodes of $\mathcal{G}$. Note that *Kirchhoff forests* is my own vocabulary. In fact, even though these forests refer to a specific distribution over the set of spanning forests, they are only very vaguely referred to under the generic term of *random spanning forests* in the literature. For clarity's sake, and to avoid confusions, my co-authors and I thus decided to give these specific random forests a name.

The background Section 4.1 will provide the necessary definitions and known facts about KFs. I will then discuss four of my recent lines of work on this subject. First of all, in Section 4.2, we show how Kirchhoff forests can be used for graph signal sampling, in the same k-bandlimited framework as my work previously described in Section 2.3.2. Then, in Section 4.3, I show how KFs can be efficiently leveraged to estimate the trace of the regularized inverse of diagonally dominant matrices (a subclass of which are graph Laplacians). In Section 4.4, I will show how KFs naturally enable very original and efficient (in computation time and in memory footprint) computation of i) the graph Tikhonov regularization of a signal (as introduced in Section 2.1.2), ii) and the related problem of graph signal interpolation.

4.1 Background

4.1.1 Uniform spanning trees and Wilson's algorithm

Spanning trees. Consider a connected, undirected, possibly weighted, graph[2] $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. A tree is a subset of $\mathcal{E}$ that does not create any cycle in $\mathcal{G}$. A spanning tree, generically denoted by $\mathcal{T}$, is a tree that spans all nodes in $\mathcal{V}$. We denote by $\mathfrak{T}(\mathcal{G})$ the set of all spanning trees of $\mathcal{G}$. An elementary property of a spanning tree is for instance that it contains exactly $n-1$ edges (more would necessarily create cycles, less would fail to span all nodes).

[2]USTs –and Kirchhoff forests as well– also exist for directed graphs but we prefer here to stick with undirected graphs, for simplicity. We do suppose that $\mathcal{G}$ is connected. If it is not, then a UST on $\mathcal{G}$ is the union of USTs drawn on each of its connected components.

Uniform Spanning Trees. A UST T is a randomly generated spanning tree from the following distribution:

$$\forall \mathcal{T} \in \mathfrak{T}(\mathcal{G}) \qquad \mathbb{P}(T = \mathcal{T}) \propto \prod_{(i,j)\in\mathcal{T}} w_{i,j}. \tag{4.1}$$

It is called *uniform* because its distribution becomes uniform over the set of all spanning trees whenever the graph is unweighted. That is, when $\forall (i,j) \in \mathcal{E}, w_{i,j} = 1$, the UST distribution boils down to:

$$\forall \mathcal{T} \in \mathfrak{T}(\mathcal{G}) \qquad \mathbb{P}(T = \mathcal{T}) = \frac{1}{|\mathfrak{T}(\mathcal{G})|}. \tag{4.2}$$

$|\mathfrak{T}(\mathcal{G})|$, the number of spanning trees in $\mathcal{G}$ is highly dependent on the structure of $\mathcal{G}$. If $\mathcal{G}$ is itself a tree, then $|\mathfrak{T}(\mathcal{G})| = 1$ of course. However, this number grows very fast with the number of edges in the graph. In the extreme case, which is the complete graph, Cayley's formula counts n^{n-2} spanning trees. One can in fact precisely count the number of trees in a graph via Kirchhoff's famous matrix-tree theorem:

Theorem 45 (Matrix-tree theorem)**.** *The normalization constant of the distribution of Eq.* (4.1) *is:*

$$Z = \sum_{\mathcal{T}\in\mathfrak{T}(\mathcal{G})} \prod_{(i,j)\in\mathcal{T}} w(i,j) = \frac{1}{n}\,\mathrm{Det}(\mathsf{L}) \tag{4.3}$$

where L *is the graph's Laplacian and the notation* $\mathrm{Det}(\mathsf{L})$ *stands for the pseudo determinant of* L *that consists in the product of all non-null eigenvalues of* L*. For unweighted graphs,* Z *simply counts the number of spanning trees in* $\mathcal{G}$*, i.e.:* $Z = |\mathfrak{T}(\mathcal{G})| = \frac{1}{n}\,\mathrm{Det}(\mathsf{L})$.

Remark 46. *This is one of my favorite theorem in the field!*

Sampling USTs. Now, how to sample USTs? There has been a lot of research efforts put into efficient sampling of USTs. In his celebrated work [Wilson 1996], Wilson proposes a sampling strategy based on loop-erased random walks. The algorithms proposed in his paper reduce to Alg. 5 for undirected graphs (which is our case in this document). At termination, the size-n array $Next$ contains all the necessary information to build the UST. To be precise, $Next$ encodes a directed spanning tree where all edges are directed towards an arbitrary node r (chosen at line 3 of the algorithm) and $Next[u]$ is the unique outgoing neighbor of u in this directed spanning tree. Forgetting the edges' orientation, one obtains the UST.

In line 8 of Alg. 5, `RandomSuccessor`$(u, \mathcal{G})$ is a function that outputs a random neighbor ℓ of node u according to a probability proportional to $w(u,\ell)$: $\mathbb{P}(\ell) = w(u,\ell)/d_u$ where d_u is the degree of node u. In simpler words, it is one step of the "natural" random walk on $\mathcal{G}$.

The proof that this Algorithm samples uniformly a spanning tree rooted in r draws from properties of loop-erased random walks. One of the fundamental step of

Algorithm 5 Wilson's algorithm to sample a UST from an undirected graph.

1: **Inputs:**
   A (possibly weighted) graph $\mathcal{G}$.
2: **Initialize:**
   # Initially, the tree is empty
   $\forall i \in \mathcal{V}$, InTree[$i$] $\leftarrow$ `false`
   $\forall i \in \mathcal{V}$, Next[i] $\leftarrow -1$
3: Choose a node r arbitrarily and set InTree[r] $\leftarrow$ `true`
4: **for** $i \leftarrow 1$ to $|\mathcal{V}|$ **do**
5: $u \leftarrow i$
6: *# Start a random walk to create a tree branch*
7: **while not** InTree[u] **do** *# Stop if u is in the tree*
8: Next[u] $\leftarrow$ `RandomSuccessor`$(u, \mathcal{G})$
9: $u \leftarrow$ Next[u]
10: **end while**
11: $u \leftarrow i$ *# Go back to the initial node*
12: *# Add the newly created branch to the tree*
13: **while not** InTree[u] **do**
14: InTree[u] $\leftarrow$ `true`
15: $u \leftarrow$ Next[u]
16: **end while**
17: **end for**
18: `return` Next

the proof amounts to showing that the result is independent of the choice of node r at line 3, and of the order with which we go through the set of nodes in the for loop starting at line 4: the indexing of the nodes can be arbitrary.

In line 3, node r may be chosen arbitrarily: this means that any choice of r will output a tree according to the UST distribution. However, the choice of r *does* have an impact on the expected computation time of the algorithm. Intuitively, if r is only very weakly connected with the rest of the graph, the first *while* loop might take ages before stopping. However, if r is a well connected hub, this won't be the case and sampling will be faster. More formally, Marchal [Marchal & others 2000] showed (Proposition 1) that the expected number of calls to `RandomSuccessor` verifies:

$$\mathrm{Tr}\left[\left(\mathsf{I} - \left(\mathsf{D}^{-1}\mathsf{W}\right)_{\mathcal{V}\setminus\{r\}}\right)^{-1}\right] \tag{4.4}$$

One should in principle choose r accordingly (that is, minimizing this trace). However, choosing the best r amounts to solving an optimization problem that will, in a vast majority of cases, cost much more than just sampling a UST with any arbitrary node. Such that one usually prefers to come up with heuristics such as the one suggested by Wilson in his paper: sample an edge with a probability proportional to its weight and pick one of its adjacent node for r.

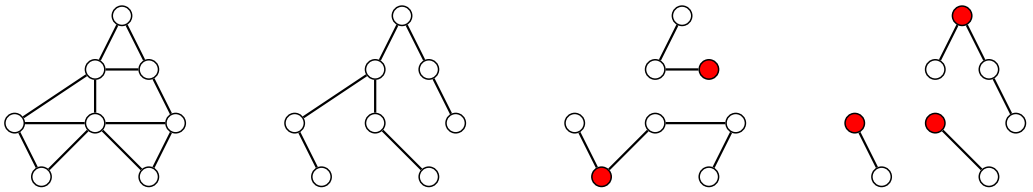

Figure 4.1: From left to right, a graph $\mathcal{G}$, a spanning tree on $\mathcal{G}$, a rooted spanning forest on $\mathcal{G}$ and another rooted spanning forest on $\mathcal{G}$ (roots are colored in red).

4.1.2 Kirchhoff forests

Rooted spanning forests. A forest is simply a set of disjoint trees, and a spanning forest is a forest spanning all nodes in $\mathcal{V}$. Now, we will consider *rooted* spanning forests: spanning forests that are made of rooted trees, which are trees from which one node is singled out and called root. We will generically denote rooted spanning forests by ϕ, and the set of all rooted spanning forests by $\mathfrak{F}(\mathcal{G})$. Let ρ be the function that maps any rooted spanning forests to its set of roots. The number of roots $|\rho(\phi)|$ is between 1 and n. See Fig. 4.1 for illustrations.

Kirchhoff Forests. Among the many possible distributions over $\mathfrak{F}(\mathcal{G})$, we focus on the following parametric distribution. For a fixed parameter $q > 0$, a Kirchhoff forest is a random rooted spanning forest, denoted by Φ_q, and distributed according to:

$$\forall \phi \in \mathfrak{F}(\mathcal{G}), \quad \mathbb{P}(\Phi_q = \phi) \propto q^{|\rho(\phi)|} \prod_{(i,j)\in\phi} w_{i,j}. \tag{4.5}$$

Link with USTs. KFs are a generalization of USTs, in the following sense. Consider an extended graph $\mathcal{G}_{\text{ext}} = (\mathcal{V}_{\text{ext}}, \mathcal{E}_{\text{ext}})$, that we will often consider in this chapter. It is built from $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ by simply adding an extra node, denoted by Γ, connected to all other nodes: $\mathcal{V}_{\text{ext}} = \mathcal{V} \cup \Gamma$ and $\forall i \in \mathcal{V}, (i, \Gamma) \in \mathcal{E}_{\text{ext}}$ with a weight denoted by $q > 0$.

Definition 13. *Let $\mathcal{G}$ be a graph and $\mathcal{G}_{\text{ext}}$ its extended graph. We define the function:*

$$\begin{aligned} f : \mathfrak{F}(\mathcal{G}) &\to \mathfrak{T}(\mathcal{G}_{\text{ext}}) \\ \phi &\mapsto f(\phi) \end{aligned}$$

where $f(\phi)$ is the spanning tree over $\mathcal{G}_{\text{ext}}$ obtained by connecting all ϕ's roots to Γ and erasing all root knowledge.

Lemma 47. *f is trivially bijective. Moreover, given a spanning tree $\mathcal{T}_{\text{ext}}$ of $\mathcal{G}_{\text{ext}}$, $f^{-1}(\mathcal{T}_{\text{ext}})$ is the rooted spanning forest of $\mathcal{G}$ obtained by cutting all $\mathcal{T}_{\text{ext}}$'s edges connected to Γ and setting all ex-neighbors of Γ to roots.*

Lemma 48. *The following two assertions are verified:*

i) Let Φ_q be a KF of $\mathcal{G}$. Then, $f(\Phi_q)$ is a UST of $\mathcal{G}_{\text{ext}}$.

ii) Let T_{ext} be a UST of $\mathcal{G}_{\text{ext}}$. Then, $f^{-1}(T_{\text{ext}})$ is a KF of $\mathcal{G}$.

Sampling KFs. Given these equivalence results, an algorithm to sample KFs is

1. sample a UST T_{ext} on $\mathcal{G}_{\text{ext}}$ with Alg. 5
2. return $\Phi = f^{-1}(T_{\text{ext}})$

Now, in the following, as a choice of r in line 3 of Alg 5 for step 1 above, we will choose $r = \Gamma$. As discussed earlier, even though this is a somewhat natural choice for r in our context, it is not necessarily the best choice in terms of computation time. In fact, we can readily know that this is *not* a smart choice for very small values of q: as hinted earlier, the length of the first while loop tends to ∞ as q tends to 0! However, it makes for simplified notations and algorithms so we will make that choice nonetheless.

Algorithm 6 Kirchhoff Forest sampling algorithm

Inputs:
  $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$
  $q > 0$
Initialize:
  # Initially, the forest is empty
  $\forall i \in \mathcal{V}$, InForest[$i$] $\leftarrow$ `false`
  $\forall i \in \mathcal{V}$, Next[i] $\leftarrow -1$
  $\forall i \in \mathcal{V}$, d[$i$] $\leftarrow \sum_{j \in \mathcal{N}(i)} w(i,j)$ *# Degrees*
for $i \leftarrow 1$ to $|\mathcal{V}|$ **do**
  $u \leftarrow i$
  # Start a random walk to create a forest branch
  while not InForest[u] **do** *# Stop if u is in the forest*
    if `rand` $\leq \frac{q}{q+\text{d}[u]}$ **then** *#If true, u becomes a root*
      InForest[u] $\leftarrow$ `true` *# Add u to the forest*
      Next[u] $\leftarrow -1$ *# Set next of u to null*
    else *# If false, continue the random walk*
      Next[u] $\leftarrow$ `RandomSuccessor`$(u, \mathcal{G})$
      $u \leftarrow$ Next[u]
    end if
  end while
  $u \leftarrow i$ *# Go back to the initial node*
  # Add the newly created branch to the forest
  while not InForest[u] **do**
    InForest[u] $\leftarrow$ `true`
    $u \leftarrow$ Next[u]
  end while
end for
`return` Next

An implementation of this Alg. 5 with the choice $r = \Gamma$ is detailed in Alg. 6. Applying Eq. (4.4) with this choice of r, and replacing D and W in the formula by the degree matrix and the adjacency matrix of the extended graph $\mathcal{G}_{\text{ext}}$, one can show that the expected number of calls to `RandomSuccessor` in Alg. 6 is

$$\mathrm{Tr}\left[(\mathsf{L}+q\mathsf{I})^{-1}(\mathsf{D}+q\mathsf{I})\right],$$

a loose upper-bound[3] of which is $n + \mathrm{Vol}(\mathcal{G})/q$, where the volume of the graph is defined as $\mathrm{Vol}(\mathcal{G}) = \sum_i d_i$. In unweighted graphs, this is linear in the number of edges ($\mathrm{Vol}(\mathcal{G}) = 2|\mathcal{E}|$ in this case). Also, as expected, the time explodes as $q \to 0$.

Varying q over nodes. The original graph can also be extended by setting $\forall i \in \mathcal{V}, w(i,\Gamma) \leftarrow q_i > 0$, that is, by connecting the added node Γ with links of unequal weights. In fact, the q_i's do not all have to be strictly positive: some can be chosen to be null so long as there is at least one non-null q_i. In this case, the distribution of the associated KFs becomes:

$$\forall \phi \in \mathfrak{F}(\mathcal{G}) \qquad \mathbb{P}(\Phi_Q = \phi) \propto \prod_{i \in \rho(\phi)} q_i \prod_{(i,j)\in\phi} w(i,j) \tag{4.6}$$

where $Q = \{q_1, q_2, \ldots, q_n\}$ is the collection of parameters. Algorithm 6 can easily be adapted by replacing the scalar input q by $Q = \{q_1, q_2, \ldots, q_n\}$, and $\frac{q}{q+d[u]}$ by $\frac{q_u}{q_u+d[u]}$ at line 7. In addition, the average run time in this case becomes

$$\mathrm{Tr}\left((\mathsf{L}+\mathsf{Q})^{-1}(\mathsf{D}+\mathsf{Q})\right),$$

with $\mathsf{Q} = \mathrm{diag}(q_1, \ldots, q_n)$. This varying q setting is richer than the fixed q setting for only a small added complexity to the notations: from now on, unless otherwise specified, we thus set ourselves in the varying q setting. To go back to the fixed q setting, one needs only to set Q to $q\mathsf{I}$. Also, from now on, we will –unless we really need them– drop the notation q or Q and simply generically denote KFs by Φ. The associated values of Q should be clear from context.

Random partitions. A partition of $\mathcal{V}$, denoted by $\mathcal{P}$, is a set of disjoint subsets whose union equals $\mathcal{V}$. To any KF Φ can be associated a random partition of $\mathcal{V}$ by splitting it into $|\rho(\Phi)|$ disjoint subsets. Let π be the function that outputs the partition for a given spanning forest. Enumerating Φ's trees from 1 to $|\rho(\Phi)|$ and denoting the vertex set of the k-th tree as $\mathcal{V}_k \subset \mathcal{V}$, one has: $\pi(\Phi) = (\mathcal{V}_1, \ldots, \mathcal{V}_{|\rho(\Phi)|})$. Note that this function is a many-to-one mapping because different spanning forests may correspond to the same partition (see Figure 4.2 for an illustration).

[3] To show this, we need to use Thm 49 stating that the root process of KFs, $\rho(\Phi)$, is a DPP of kernel $\mathsf{K} = (\mathsf{L}+q\mathsf{I})^{-1}(q\mathsf{I})$. One has: $\mathrm{Tr}\left[(\mathsf{L}+q\mathsf{I})^{-1}(\mathsf{D}+q\mathsf{I})\right] = \mathrm{Tr}\left[\mathsf{K}\left(\mathsf{D}/q+\mathsf{I}\right)\right] = \sum_i \mathsf{K}_{i,i}\frac{d_i+q}{q} = \mathbb{E}_\Phi\left(\sum_{i\in\rho(\Phi)} 1 + \frac{d_i}{q}\right)$ where the last equality comes from $\mathsf{K}_{i,i} = \mathbb{P}(i \in \rho(\Phi))$. From there, we obtain the upper-bound by writing $\forall \phi \in \mathfrak{F}(\mathcal{G})$, $\sum_{i\in\rho(\Phi)} 1 + \frac{d_i}{q} \leq \sum_{i\in\mathcal{V}} 1 + \frac{d_i}{q} = n + \mathrm{Vol}(\mathcal{G})/q$

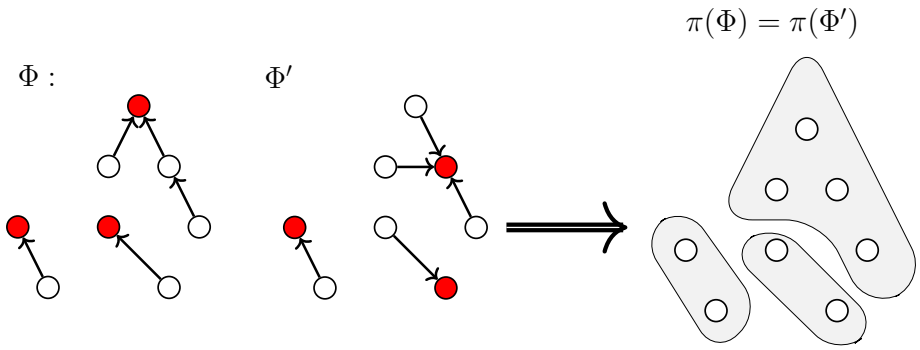

Figure 4.2: Two different rooted spanning forests (on the left) with the same corresponding partition (on the right)

4.1.3 Useful properties of Kirchhoff forests

Recent studies in [Avena & Gaudillière 2017, Avena & Gaudillière 2013] have established some theoretical properties of Φ that we reproduce here.

Theorem 49 (The root process, see Proposition 2.2 in [Avena & Gaudillière 2017])**.** *The root process of KFs, $\rho(\Phi)$, is a determinantal point process: $\rho(\Phi) \sim \mathtt{DPP}(\mathsf{K})$ with marginal kernel:*

$$\mathsf{K} = (\mathsf{Q} + \mathsf{L})^{-1}\mathsf{Q}. \tag{4.7}$$

Remark 50 (Cardinality of $\rho(\Phi)$.)**.** *As a consequence, in the fixed q setting where $\mathsf{Q} = q\mathsf{I}$, the first two moments of $|\rho(\Phi)|$ verify, given the fact that it is a Bernoulli process (see Lemma 9):*

$$\begin{aligned} \mathbb{E}[|\rho(\Phi)|] = \mathrm{Tr}(\mathsf{K}) &= \sum_i \frac{q}{q+\lambda_i}, \\ \mathrm{Var}(|\rho(\Phi)|) = \mathrm{Tr}(\mathsf{K} - \mathsf{K}^2) &= \sum_i \frac{\lambda_i q}{(q+\lambda_i)^2} \end{aligned} \tag{4.8}$$

where the λ_i's are the eigenvalues of the graph's Laplacian matrix L. In the varying q setting, K is not co-diagonalizable with L anymore: the expected number of roots $\mathbb{E}[|\rho(\Phi)|]$ is not writable in terms of the λ_i's. It is however still equal to $\mathrm{Tr}(\mathsf{K})$.

Definition 14. *Let ϕ be a rooted spanning forest. The* root function *$r_\phi : \mathcal{V} \to \rho(\phi)$, takes as input any node i and outputs the root of the tree which i belongs to.*

Theorem 51 (see [Avena & Gaudillière 2013, Avena & Gaudillière 2017])**.** *Let Φ be a KF. For any pair of nodes (i, j), the probability that i is rooted in j reads:*

$$\forall i, j \in \mathcal{V} \quad \mathbb{P}(r_\Phi(i) = j) = \mathsf{K}_{ij}. \tag{4.9}$$

with K as in Eq. (4.7).

Definition 15. *Let $\phi \in \mathfrak{F}(\mathcal{G})$. We define the function $t : \mathcal{V} \to \{1, 2, \dots, |\rho(\Phi)|\}$ as a mapping between any node and its tree number in Φ. Example: $t(i) = k$ if $i \in \mathcal{V}_k \in \pi(\Phi)$.*

Theorem 52 (Conditioning on a partition, Proposition 2.3 in [Avena & Gaudillière 2017])**.** *In the fixed q setting, conditioning the root probability over a fixed partition $\mathcal{P}$, one obtains:*

$$\forall i, j \in \mathcal{V} \quad \mathbb{P}(r_\Phi(i) = j | \pi(\Phi) = \mathcal{P}) = \frac{\mathbb{I}(j \in \mathcal{V}_{t(i)})}{|\mathcal{V}_{t(i)}|} \tag{4.10}$$

where $\mathbb{I}$ is the indicator function. In other words, given a fixed partition $\mathcal{P}$, the root probability within each subset $\mathcal{V}_k$ is uniform over the nodes in $\mathcal{V}_k$. In the varying q setting, this conditional probability in Eq. (4.10) becomes:

$$\forall i, j \in \mathcal{V} \quad \mathbb{P}(r_\Phi(i) = j | \pi(\Phi) = \mathcal{P}) = \frac{q_j \mathbb{I}(j \in \mathcal{V}_{t(i)})}{\sum_{k \in \mathcal{V}_{t(i)}} q_k}. \tag{4.11}$$

4.2 KFs to sample bandlimited graph signals

In this first contributive section, I discuss the existent links between the problem of sampling k-bandlimited graph signals detailed in Section 2.3.1 and Kirchhoff forests.

4.2.1 Recall the graph sampling context

Let us put ourselves back in the setting of Section 2.3.1: the main elements where the following. Let $\mathsf{U}_k \in \mathbb{R}^{n \times k}$ be the concatenation of the first k eigenvectors of a graph's Laplacian matrix. Let $\boldsymbol{x} \in \mathbb{R}^n$ be a k-bandlimited graph signal, *i.e.*, there exists $\boldsymbol{\alpha} \in \mathbb{R}^k$ such that $\boldsymbol{x} = \mathsf{U}_k \boldsymbol{\alpha}$. Given a sampling set $\mathcal{X} \subseteq \mathcal{V}$, we measure the signal $\boldsymbol{x}$ on $\mathcal{X}$ to obtain the measurements

$$\boldsymbol{y} = \mathsf{M}\boldsymbol{x} + \boldsymbol{n} \in \mathbb{R}^m, \tag{4.12}$$

where $\mathsf{M} \in \mathbb{R}^{m \times n}$ is the sampling matrix associated to $\mathcal{X}$ and $\boldsymbol{n}$ models noise.

We recall that, in the tight budget setting (that is, $m = k$), the condition for perfect reconstruction (up to the noise level) of $\boldsymbol{x}$ from $\boldsymbol{y}$ is that $\mathcal{X}$ should be chosen such that MU_k is invertible. Indeed, in this case, the solution to the least-square recovery problem gives:

$$\begin{aligned} \boldsymbol{x}_{\text{rec}} &= \operatorname*{arg\,min}_{\boldsymbol{z} \in \operatorname{span}(\mathsf{U}_k)} \|\mathsf{M}\boldsymbol{z} - \boldsymbol{y}\|^2 \\ &= \boldsymbol{x} + \mathsf{U}_k \left(\mathsf{MU}_k\right)^{-1} \boldsymbol{n}. \end{aligned}$$

Given that there are often many sets $\mathcal{X}$ for which MU_k is invertible, the "best" sample set $\mathcal{X}$ is a question of how we want to deal with the distorted noise. Many possible best choice for $\mathcal{X}$ are possible: the A-optimal set, the E-optimal set, the D-optimal set, *etc.*

Finding any of these optimal sets is a NP-complete problem, given the combinatorial nature of the problem (they are optimization problems defined on all sets of size k). So one usually resorts to two general families of strategies: greedy strategies and random strategies.

In the literature, many greedy strategies were suggested for any of the above choices of optimality. Their naive cost is generally in $\mathcal{O}(nk^4)$ with possible improvements to $\mathcal{O}(nk^3)$, if one supposes that U_k is pre-computed of course.

In section 2.3.2, I detailed our contribution using iid sampling with respect to a well tailored distribution p^*. We show reconstruction guarantees (with high probability) but only in the almost-tight budget where the number of samples is allowed to be $\mathcal{O}(k \log k)$. The good news however is that the cost of this sampling is in $\mathcal{O}(nk)$: it is the cost of computing the norm of each line of U_k to obtain the optimal sampling distribution p^*.

4.2.2 A projection DPP for graph sampling

We show in [Tremblay *et al.* 2017] that we can do strictly better than this iid framework by using DPPs. How should we build this DPP? From our investigations in DPPs for coresets (detailed in Section 3.4), we know that the inclusion probability of order 1, $\mathbb{P}(i \in X)$, should be set according to the leverage scores p^*. We also know that projection DPPs have in general a lower variance –or are at least easier to control theoretically.

We obtain the following result:

Proposition 53. *Let $\mathsf{K}_k = \mathsf{U}_k\mathsf{U}_k^\top$ be a projection kernel. Let $X \sim \mathtt{DPP}(\mathsf{K}_k)$. The following two assertions are verified:*

i) $\mathsf{M}\mathsf{U}_k$ is invertible such that perfect recovery of $\boldsymbol{x}$ (up to noise) is verified.

ii) The most probable set of the DPP is $\mathcal{X}_D$, the D-optimal set, as defined in Eq. (2.20).

Proof. i) Let $X \sim \mathtt{DPP}(\mathsf{K}_k)$ with $\mathsf{K}_k = \mathsf{U}_k\mathsf{U}_k^\top$. By definition, $\det(\mathsf{U}_k\mathsf{U}_k^\top)_X > 0$ (or else X cannot be sampled). This is equivalent to $\det(\mathsf{M}\mathsf{U}_k\mathsf{U}_k^\top\mathsf{M}^\top) > 0$, itself equivalent to $\det(\mathsf{M}\mathsf{U}_k) \neq 0$, *i.e.*, that $\mathsf{M}\mathsf{U}_k$ is invertible.
ii) By definition, the most probable set of this DPP is the set $\mathcal{X}$ that maximizes $\det(\mathsf{M}\mathsf{U}_k\mathsf{U}_k^\top\mathsf{M}^\top)$: it is thus $\mathcal{X}_D$, the D-optimal set. □

Remark 54. *Note that the inclusion probability of this DPP is indeed proportional to the optimal iid sampling distribution p^* we earlier described in Section 2.3.2:*

$$\mathbb{P}(i \in X) = \mathsf{K}_k(i,i) = \sum_j \mathsf{U}_k(i,j)^2$$

In this sense, this DPP is a sort of "matched DPP" to our previous iid framework. The DPP enables simply to have a tighter sampling: $m = k$ instead of $m =$

$\mathcal{O}(k \log k)$ *–while keeping the reconstruction guarantee. Also, given our fast sampling algorithm of Section 3.2, and given* U_k*, sampling the DPP is only marginally longer (*$\mathcal{O}(nk + k^3 \log k)$*) than the iid sampling framework (*$\mathcal{O}(nk)$*), especially in the usual setting where* $k \ll n$*.*

Remark 55. *We are satisfied to obtain, via this projection DPP, a tight-budget framework that enables perfect reconstruction up to the noise level, while being orders of magnitude faster to sample than its greedy counterparts:* $\mathcal{O}(nk + k^3 \log k)$ *versus* $\mathcal{O}(nk^3)$ *for greedy algorithms.*

How does it fare in the noisy case? Comparing sampling frameworks is not only about perfect reconstruction in the noiseless case or about a strict number of samples. It is also about the behavior of the distorted noise term $\boldsymbol{x}_{\text{rec}} - \boldsymbol{x} = \mathsf{U}_k \left(\mathsf{M}\mathsf{U}_k\right)^{-1} \boldsymbol{n}$ in Eq. (4.13).

Well, again, this depends on how we want to optimally deal with this noise. If we look at the worst-case error, we can bound $\|\boldsymbol{x}_{\text{rec}} - \boldsymbol{x}\|_2^2$ with:

$$\|\boldsymbol{x}_{\text{rec}} - \boldsymbol{x}\|_2^2 \leq \lambda_{\max} \left(\left(\mathsf{M}\mathsf{U}_k \mathsf{U}_k^\top \mathsf{M} \right)^{-1} \right) \|\boldsymbol{n}\|_2^2$$

of which I do not know the expected value over DPP samples $X \sim \texttt{DPP}(\mathsf{U}_k \mathsf{U}_k^\top)$.

What we *can* upper-bound is $\mathbb{E}_X \left(\mathbb{E}_{\boldsymbol{n}} \left(\|\boldsymbol{x}_{\text{rec}} - \boldsymbol{x}\|_2^2 \right) \right)$ in the case of a noise $\boldsymbol{n}$ verifying $\mathbb{E}_{\boldsymbol{n}} \left(\boldsymbol{n}\boldsymbol{n}^\top \right) = \sigma^2 \mathsf{I}$. One has, using the circularity of the trace:

$$\mathbb{E}_X \left(\mathbb{E}_{\boldsymbol{n}} \left(\|\boldsymbol{x}_{\text{rec}} - \boldsymbol{x}\|_2^2 \right) \right) = \sigma^2 \, \mathbb{E}_X \left(\text{Tr} \left(\left(\mathsf{M}\mathsf{U}_k \mathsf{U}_k^\top \mathsf{M} \right)^{-1} \right) \right) \tag{4.13}$$

$$\leq \sigma^2 \, k(n - k + 1) \tag{4.14}$$

where the equality is reached if all sets $\mathcal{X}$ have a non-null probability of occuring as a result of the DPP sampling. To obtain the upper-bound in Eq. (4.14), we use here a result from [Derezinski & Warmuth 2017].

Empirically, we observe (not shown here) that the robustness to noise of DPP samples $X \sim \texttt{DPP}(\mathsf{U}_k \mathsf{U}_k^\top)$ is comparable to its associated greedy strategy (in which each new sample is chosen in order to maximize the determinant associated to the current set). We however did not make extensive simulations and this should be studied more in depth.

4.2.3 What to do without knowledge of U_k?

A common goal in the graph sampling literature is to find sampling strategies that side-step the full knowledge of U_k. For instance, one can refer to [Anis *et al.* 2016, Tsitsvero *et al.* 2016] for some examples. In our own iid framework, we came up with an algorithm that estimates the ideal distribution p^* and we then work with that distribution.

One can think of many different heuristics to adapt this to our DPP context. There is one original way that I would like to particularly discuss in the context of this chapter: the use of Kirchhoff forests.

In fact, Thm 49 states that the root process of KFs are a DPP with marginal kernel (in the fixed-q case):

$$\mathsf{K} = q(\mathsf{L} + q\mathsf{I})^{-1}$$

which can be written as a graph filter as:

$$\mathsf{K} = \mathsf{U}h(\Lambda)\mathsf{U}^\top \tag{4.15}$$

with $h(\lambda) = \frac{q}{q+\lambda}$ a low-pass filter, and where we recall that U and $\mathsf{\Lambda}$ stem from the eigendecomposition of $\mathsf{L} = \mathsf{U}\mathsf{\Lambda}\mathsf{U}^\top$. Let us recall that $\mathsf{U}_k\mathsf{U}_k^\top$ can be seen as the ideal low-pass graph filter:

$$\mathsf{U}_k\mathsf{U}_k^\top = \mathsf{U}h_k^*(\Lambda)\mathsf{U}^\top \tag{4.16}$$

with $h_k(\lambda) = 0$ if $\lambda > \lambda_k$ and 1 otherwise.

In other words, while Proposition 53 requires us to sample from a DPP with marginal kernel $\mathsf{K}_k = \mathsf{U}_k\mathsf{U}_k^\top$ but which is impractical in cases where matrix U_k is unavailable, the roots of KFs provide us with a sample from a DPP with marginal kernel $\mathsf{K} = \mathsf{U}h(\Lambda)\mathsf{U}^\top$ that can be seen as an approximation of K_k; and they do so without any diagonalization (even partial). Roots of KFs appear thus as a possible proxy to sample nodes in our context of bandlimited graph signal reconstruction.

What can we say about their performance? Well, first let us recall the mixture representation of $X \sim \mathtt{DPP}(\mathsf{K} = \mathsf{U}h(\Lambda)\mathsf{U}^\top)$ (see Section 3.1.4)

1. Sample $Y \sim \mathtt{DPP}(h(\mathsf{\Lambda}))$. It is a Bernoulli process: $\forall i$, add i to Y with probability $\frac{q}{q+\lambda_i}$. For instance, as $\lambda_1 = 0$, $i = 1$ is always sampled.
2. Sample a fixed-size DPP of size $|Y|$: $X \sim |\mathtt{DPP}|_{|Y|}\left(\mathsf{U}_{:,Y}\left(\mathsf{U}_{:,Y}\right)^\top\right)$

Intuitively, in order to recover k-bandlimited signals, the first eigenvector selection step of this mixture representation, needs to at least sample the k first eigenvectors. We obtain the following result (unpublished):

Proposition 56. *Let $X \sim \mathtt{DPP}(\mathsf{K})$ be the root process of a KF, for a given parameter q. Measuring the signal on X enables the recovery of all k-bandlimited signals with probability larger than:*

$$\prod_{1\leq i\leq k} \frac{q}{q+\lambda_i}$$

Proof. With probability larger than $\prod_{1\leq i\leq k} \frac{q}{q+\lambda_i}$, the sampled set of eigenvector Y in step 1 of the above mixture representation, contains (at least) all elements between 1 and k.

Suppose the event $\{1,\dots,k\} \in Y$ occurs. This implies that $\mathrm{span}(\mathsf{U}_k) \in \mathrm{span}(\mathsf{U}_{:,Y})$. As X is a projection DPP based on the marginal kernel $\mathsf{U}_{:,Y}\left(\mathsf{U}_{:,Y}\right)^\top$, it can recover any signal in $\mathrm{span}(\mathsf{U}_{:,Y})$ (see proof of Proposition 53). As $\mathrm{span}(\mathsf{U}_k) \in \mathrm{span}(\mathsf{U}_{:,Y})$, it can thus recover any signal in $\mathrm{span}(\mathsf{U}_k)$, that is: any k-bandlimited signal. □

One obtains the following corollary:

Corollary 57. *Let $X \sim \mathtt{DPP}(\mathsf{K})$ be the root process of KFs, for a given parameter q. Let $\delta \in (0,1)$. With probability larger than $1-\delta$, the following is verified: X can recover any k-bandlimited signal provided that*

$$q \geq \frac{k\lambda_k}{\delta} \tag{4.17}$$

Proof. We need to set set q sufficiently large to verify

$$\prod_{1\leq i\leq k} \frac{q}{q+\lambda_i} \leq 1-\delta$$

Lower-bounding the lhs by $(1+\lambda_k/q)^{-k}$, one obtains the following necessary condition:

$$1+\lambda_k/q \leq (1-\delta)^{-\frac{1}{k}}$$

Lower-bounding the rhs by $1+\delta/k$, one obtains that if $q \geq \frac{k\lambda_k}{\delta}$ then $\prod_{1\leq i\leq k} \frac{q}{q+\lambda_i} \leq 1-\delta$. Applying proposition 56 finishes the proof. □

4.2.4 What now?

Corollary 57 is not a very strong result as in many cases the value of q suggested by Eq. (4.17) is so large that the expected number of root it implies,

$$\mathrm{Tr}(\mathsf{K}) = \sum_{1\leq i\leq n} \frac{q}{q+\lambda_i} = \sum_{1\leq i\leq n} \frac{1}{1+\frac{\delta\lambda_i}{k\lambda_k}}$$

is exceedingly large (compared to k). We are admittedly able to recover all k-bandlimited signals at very low sampling cost[4], but in a regime very far from tight sampling!

Still, Corollary 57 enables to intuitively see the conditions necessary for the roots of KFs to be good proxies: they are good proxies if, for the proposed $q = \frac{k\lambda_k}{\delta}$, the expected number of roots $\mathrm{Tr}(\mathsf{K})$ is not too large compared to k. This is a condition formulated on the spectrum of the Laplacian. One sees for instance that the larger the spectral gap between λ_k and λ_{k+1}, the closer we are to the tight sampling scenario. This means that this KF-based technique is much more adapted to graphs with k well-separated communities (for which such a spectral gap is expected) than on graphs that do not have a spectral graph, such as a 2d-grid or an ε-graph of uniformly distributed points in the square.

One possible direction of research is to attempt to take advantage of the degrees of freedom we in fact have to define Q. In the previous paragraphs, we have studied the constant q case: $\mathsf{Q} = q\mathsf{I}$. We can however "play" with these extra degrees of freedom to incorporate some extra information we might have computed by other

[4] although we still need to estimate λ_k in order to set q as in Eq. (4.17)

means from the graph. We can even set q_i to 0 or ∞ if we want to remove i from the possibly sampled nodes or, in the contrary, make sure it is sampled. This comes however with additional difficulty that we would need to address, a foreseeable one is that with a heterogeneous Q we loose the alignment between the eigenvectors of the DPP kernel K and the eigenvectors of L.

4.3 KFs for the regularized inverse trace of diagonally dominant matrices

The work presented in the following two sections (Sections 4.3 and 4.4) is the work of Yigit Pilavci, my second PhD student, that I co-advised (40%) with Simon Barthelmé (40%) and Pierre-Olivier Amblard (the 20% left). Yigit defended his thesis in October 2022.

Randomized methods are often used to approximate the trace of matrices that are not known explicitly, of the form

$$\mathrm{Tr}\left(f(\mathsf{A})\right)$$

where $\mathsf{A} \in \mathbb{R}^{n\times n}$ is typically a large matrix (*e.g.* $n \geq 10^6$), $f(\mathsf{A})$ is a matrix transformation that we cannot compute explicitly due to memory or computation time constraints. $f(\mathsf{A})$ is typically a matrix that we can only access in reasonable time via matrix-vector multiplications of the form $f(\mathsf{A})\boldsymbol{x}$. These methods come into play in various problems (see, for instance, [Fan *et al.* 2020]).

In this section, we describe how KFs can be leveraged for this approximation problem narrowed down to the following setting:

- $f(\cdot)$ is the regularized inverse: $f(\mathsf{A}) = (\mathsf{A}+\mathsf{Q})^{-1}\mathsf{Q}$ where $\mathsf{0} \prec \mathsf{Q}$ is a diagonal matrix of non-negative elements.
- matrices A are supposed to be *Symmetric Diagonally Dominant* (SDD) matrices: they verify $\forall i, |\mathsf{A}_{i,i}| \geq \sum_{j\neq i} |\mathsf{A}_{i,j}|$.

Note that graph Laplacian matrices are a subclass of SDD matrices. For the sake of the following presentation, we will suppose that A is a graph Laplacian and we will thus denote it by L. I will show at the end of this section how our solutions can be easily generalized to the larger class of SDD matrices.

Now, a natural use case in which one needs to compute $\mathrm{Tr}(q(\mathsf{L}+q\mathsf{I})^{-1})$ arises in the graph Tikhonov regularization problem, as described in Section 2.1.2 that we briefly recall here. Suppose we measure a noisy signal over n vertices of a graph: $\boldsymbol{y} = [y_1, y_2, \ldots, y_n]^\top$. Our aim is to recover the true signal $\boldsymbol{x}$ by solving the following denoising problem:

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{z}\in\mathbb{R}^n}{\arg\min}\, q||\boldsymbol{y}-\boldsymbol{z}||_2^2 + \boldsymbol{z}^\top\mathsf{L}\boldsymbol{z}, \quad q>0 \tag{4.18}$$

where the hyper-parameter $q > 0$ controls the regularization. The explicit solution $\hat{\boldsymbol{x}}$ reads $\mathsf{K}\boldsymbol{y}$ where $\mathsf{K} = q(\mathsf{L} + q\mathsf{I})^{-1}$. Notice that the recovery error, $||\boldsymbol{x} - \hat{\boldsymbol{x}}||_2^2$, highly depends on q and there are several methods to automatically choose the value of q such that the solution $\hat{\boldsymbol{x}}$ is a good approximation of $\boldsymbol{x}$. Many of them, such as generalized cross-validation, Akaike or Bayesian information criteria, use $\mathrm{Tr}(\mathsf{K})$ as a measure of the degrees of freedom of the linear smoother K [Hastie *et al.* 2005].

State-of-the-art. The standard estimator for $\mathrm{Tr}(\mathsf{K})$ is due to Hutchinson[5] [Hutchinson 1989]. Given N random Bernoulli vectors $\{\boldsymbol{a}^{(\ell)}\}_{\ell=1,\dots,N}$ built via $\forall \ell, \forall i$, $\mathbb{P}\left(a_i^{(\ell)} = \pm 1\right) = 1/2$, Hutchinson's estimator is defined as $h \coloneqq \frac{1}{N}\sum_{\ell=1}^{N} \boldsymbol{a}^{(\ell)^\top} \mathsf{K}\boldsymbol{a}^{(\ell)}$. This estimator is indeed unbiased due to the circularity of the trace:

$$\mathbb{E}(h) = \mathbb{E}_{\boldsymbol{a}}\left(\boldsymbol{a}^\top \mathsf{K}\boldsymbol{a}\right) = \mathbb{E}_{\boldsymbol{a}}\left(\mathrm{Tr}\left(\mathsf{K}\boldsymbol{a}\boldsymbol{a}^\top\right)\right) = \mathrm{Tr}\left(\mathsf{K}\mathbb{E}_{\boldsymbol{a}}\left(\boldsymbol{a}\boldsymbol{a}^\top\right)\right) = \mathrm{Tr}(\mathsf{K})$$

as $\mathbb{E}_{\boldsymbol{a}}\left(\boldsymbol{a}\boldsymbol{a}^\top\right) = \mathsf{I}$ by construction. Note in passing that this algorithm works with random vectors $\boldsymbol{a}$ with elements distributed according to any distribution, so long as its covariance matrix is the identity. The trick is to find the distribution for which the estimator h will converge as fast as possible towards its expected value. The Bernoulli vectors are often a good choice in practice.

Now, this is not the full story yet. In fact, computing $\mathsf{K}\boldsymbol{a}^{(\ell)}$ is still expensive due the matrix inverse. Even by leveraging the possible sparsity of L by using a sparse Cholesky decomposition, it has a time complexity of $\mathcal{O}(n^3)$ in the worst case. For large n, this cost is prohibitive. The state-of-the-art that avoids this cubic cost consists of (possibly preconditioned) conjugate gradient [Shewchuk *et al.* 1994], algebraic multigrid [Ruge & Stüben 1987] and polynomial approximations. These methods compute $\mathsf{K}\boldsymbol{a}$ with very small error (often much less than the Monte Carlo error induced by Hutchinson's estimator) and their cost scales linearly with the number of edges $|\mathcal{E}|$.

Our contribution. In a series of works [Barthelmé *et al.* 2019, Pilavcı *et al.* 2022a, Pilavci *et al.* 2022b], we propose a Kirchhoff forest-based method to estimate $\mathrm{Tr}(\mathsf{K})$. Empirical evidence on various graphs shows that the proposed methods perform at least as well as the state-of-the-art, while outperforming it in many settings.

In Section 4.3.1, I describe our KF-based estimator and illustrate with a few experiments how it behaves with respect to the state-of-the-art. Then, in Section 4.3.2, I will show how to generalize these results to SDD matrices.

4.3.1 A KF-based estimator

Given that we have already laid down the background on KFs, our estimator is straightforward

Theorem 58. *Let Φ be a KF. Then, $|\rho(\Phi)|$, that is, the number of roots of Φ, is an unbiased estimator of* $\mathrm{Tr}(\mathsf{K})$.

[5] a close variant of which was proposed by Girard [Girard 1987]

Proof. We know from Thm 49 that $\rho(\Phi) \sim \mathtt{DPP}(\mathsf{K})$. As such (see Remark 50), the expected value[6] of $|\rho(\Phi)|$ is $\mathrm{Tr}(\mathsf{K})$. □

Estimating $\mathrm{Tr}(\mathsf{K})$ is thus a very simple endeavour:

1. sample N KFs
2. count their number of roots
3. compute the average

The performance of this very simple algorithm is actually very good already. We showed in [Pilavci *et al.* 2022b, Pilavcı *et al.* 2022a] that its performance can still increase by smartly adapting two well-known variance reduction techniques: control variates and stratified sampling. We will not enter here into the details of these implementations. We will just show the results we obtain on a few experiments.

Experiments. We empirically compare our proposed estimators to Hutchinson's estimator over various graphs. To do so in the fairest possible way, first notice that all estimators here are Monte Carlo. Therefore, the asymptotic relation $\sigma_N \approx \frac{\sigma_1}{N^{1/2}}$ between one-sample standard deviations σ_1 and N-sample ones σ_N, is valid for all estimators in our comparison. We leverage this fact to compare the effective runtimes of all methods, *i.e.*, the time needed to reach a fixed relative error $\varepsilon > 0$. First, we run all methods with $N = 100$. This gives us the average runtime for the computation per sample, τ and the sample variance $\hat{\sigma}_N^2$. Then, we approximate $\hat{\sigma}_1 = \sqrt{N}\hat{\sigma}_N$ for each method. Finally, we solve for k the equation

$$\varepsilon = \frac{\hat{\sigma}_1}{\mathrm{Tr}(K)\sqrt{k}}$$

which gives us the number of iterations k needed for reaching an ε error. The effective runtime to attain an error ε is thus k times τ. In the following, we show results for the choice $\varepsilon = 2\%$.

In Hutchinson's estimator, we compute $\mathsf{K}\boldsymbol{a}$ using either

- `cg`: Conjugate Gradient[7]. Note that CG methods benefit from block implementations [O'Leary 1980].
- `amg`: Algebraic Multigrid[8],
- `pcg`: CG with AMG preconditioning
- `direct`: sparse Cholesky decomposition using CHOLMOD [Chen *et al.* 2008]

We compare these with our proposed estimators:

[6] In passing, we also know that the variance of this estimator verifies $\mathrm{Tr}(\mathsf{K} - \mathsf{K}^2)$.

[7] `https://docs.juliahub.com/KrylovMethods`

[8] `https://github.com/JuliaLinearAlgebra/AlgebraicMultigrid.jl`

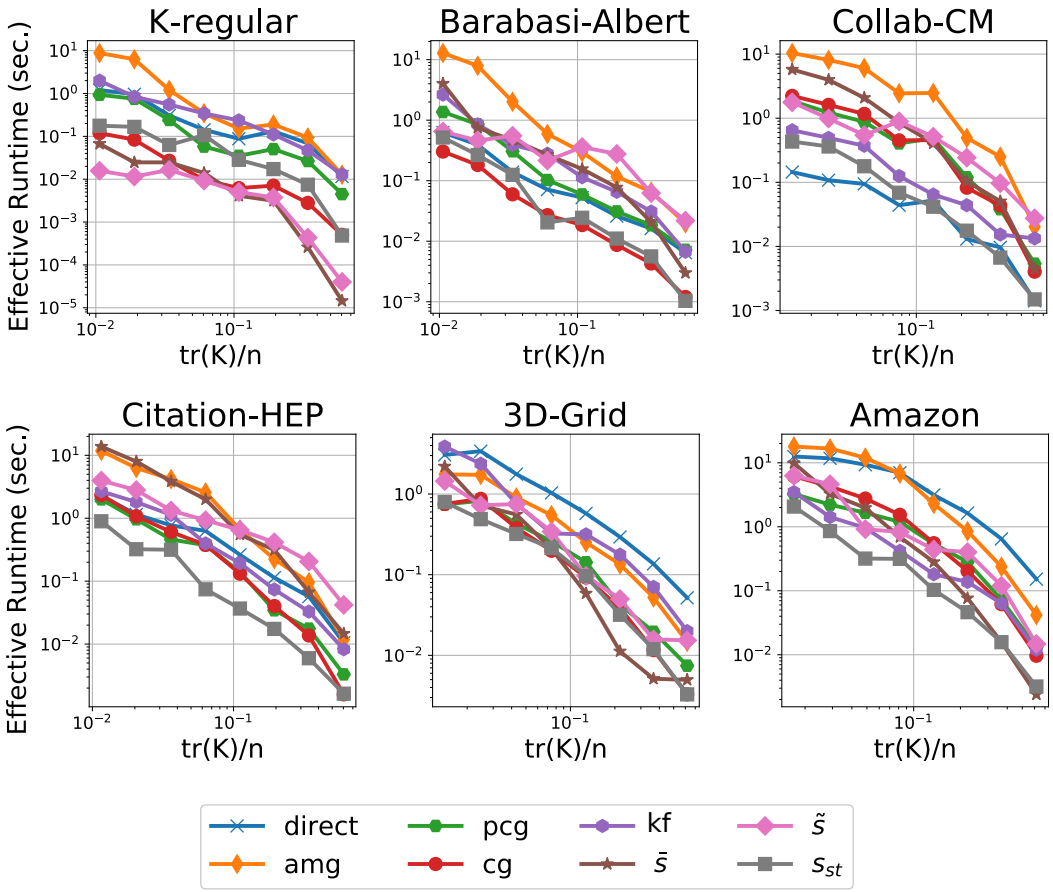

Figure 4.3: Effective Runtime vs $\mathrm{Tr}(\mathsf{K})/n$, for different graphs. The four methods 'kf', '$\tilde{s}$', '$\bar{s}$' and 's_{st}' are different variants of our KF-based methods. The other four methods are state-of-the-art (see text).

- `kf`: the simple KF-based estimator of Thm 58
- `s̃` and `s̄`: two different version of controle variates used to reduce the variance of the previous `kf` method
- `s_st`: stratified sampling to reduce the variance of the previous `kf` method

The graphs that we use in these experiments are:

- **Barabasi-Albert**: A random graph generated by Barabasi-Albert model ($k = 10$) with $n = 10^4$ and $m = 99900$,
- **K-random regular**: A random regular graph with $n = 10^4$ and $m = 10^5$ ($k = 20$),
- **Collab-CM**: A collaboration network of $n = 21363$ authors in Arxiv on condense matter physics with $m = 91342$ links,

- **Citation-HEP**: A citation network of $n = 34401$ in Arxiv on high energy physics with $m = 420828$ links.
- **3D Grid**: 3-dimensional grid with $n = 50^3 = 125000$ nodes and $m = 375000$ edges.
- **Amazon**: A real-life network over $n = 262111$ products in Amazon with $m = 899792$. A link between two products indicates that the same client purchases these two products.[9]

We choose 8 logarithmically spaced values of q such that the ratio $\mathrm{Tr}(\mathsf{K})/n$ takes values up to 65%. All experiments are implemented in Julia and run in a single thread of a laptop.

Fig. 4.3 summarizes the results. For relatively small and sparse graphs, such as Collab-CM, the direct method gives the best performance, closely followed by the KF methods. However, the approximate ones beat the direct method when the graphs become larger or denser. In these cases, the proposed methods give either the best or a comparable performance with the other state-of-the-art methods.

We are very satisfied with these results, especially given the simplicity of our algorithms compared to the heavy machinery of (preconditioned) conjugate gradient.

4.3.2 A trick to generalize to diagonally dominant matrices

This trick works at least when $\mathsf{Q} = q\mathsf{I}$. For varying q's, one would need to check. We borrow a trick from the rich literature on Laplacian solvers (see for instance [Gremban 1996, Kelner *et al.* 2013, Hunter *et al.* 2014]). Let G be a SDD matrix, that we decompose as $\mathsf{G} = \mathsf{D}_1 + \mathsf{D}_2 + \mathsf{A}_p + \mathsf{A}_n$ where:

- A_p contains the positive off-diagonal elements, A_n contains the negative ones
- D_1 is diagonal, with $\mathsf{D}_1(i,i) = \sum_{j \neq i} |G_{ij}|$ (sum of off-diagonal elements)
- D_2 is also diagonal, with entries $\mathsf{D}_2(i,i) = \mathsf{G}_{ii} - \mathsf{D}_1(i,i)$. Diagonal dominance of G implies that $\forall i, \mathsf{D}_2(i,i) \geq 0$.

Note that as G is SDD, A_p and A_n are symmetric. We now form the following two graph Laplacians, both representing undirected weighted graphs, and of respective size n and $2n$:

$$\mathsf{L}_1 = \mathsf{D}_1 + \mathsf{A}_n - \mathsf{A}_p \tag{4.19}$$

$$\mathsf{L}_2 = \begin{pmatrix} \mathsf{D}_1 + \mathsf{D}_2/2 + \mathsf{A}_n & -\mathsf{D}_2/2 - \mathsf{A}_p \\ -\mathsf{D}_2/2 - \mathsf{A}_p & \mathsf{D}_1 + \mathsf{D}_2/2 + \mathsf{A}_n \end{pmatrix}. \tag{4.20}$$

It can be easily verified that an eigenvector basis for L_2 can be constructed as follows: n eigenvectors of the form $\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{u} \end{pmatrix}$, where $\boldsymbol{u}$ is an eigenvector of L_1; and n

[9] The real-life data sets can be found in https://snap.stanford.edu/data/

other eigenvectors of the form $\begin{pmatrix} \boldsymbol{v} \\ -\boldsymbol{v} \end{pmatrix}$, where $\boldsymbol{v}$ is an eigenvector of G. This implies that $\lambda(\mathsf{L}_2) = \lambda(\mathsf{L}_1) \cup \lambda(\mathsf{G})$ and consequently that:

$$\begin{aligned} \operatorname{Tr}\left((\mathsf{G}+q\mathsf{I})^{-1}q\right) &= \sum_{\lambda\in\lambda(\mathsf{G})} \frac{q}{q+\lambda} = \sum_{\lambda\in\lambda(\mathsf{L}_2)} \frac{q}{q+\lambda} - \sum_{\lambda\in\lambda(\mathsf{L}_1)} \frac{q}{q+\lambda} \\ &= \operatorname{Tr}\left((\mathsf{L}_2+q\mathsf{I})^{-1}q\right) - \operatorname{Tr}\left((\mathsf{L}_1+q\mathsf{I})^{-1}q\right) \end{aligned}$$

The extension to SDD matrices is thus straightforward: form the two Laplacians L_1 and L_2, run the algorithm on each graph, and subtract.

4.3.3 What now?

I would like to pursue further the theoretical side of this approach, by showing concentration rates of this "number of roots" estimator, to compare with concentration rates of Hutchinson's estimator. The tightest theorem to-date on the concentration of Hutchinson's estimator is the following (denoting by $\hat{s}_k^{\mathtt{H}}$ Hutchinson's estimator of $\operatorname{Tr}(\mathsf{K})$ with k random vectors)

Theorem 59 ([Skorski 2021], improved version of a theorem in [Avron & Toledo 2011])**.** *Let $\varepsilon \in (0, 3/8)$ and $\delta \in (0,1)$. If*

$$k \geq \frac{2(1-\frac{8}{3}\varepsilon)}{\varepsilon^2} \log\frac{1}{\delta}, \tag{4.21}$$

then, with probability larger than $1-\delta$, we have a relative error control:

$$\left|\frac{\hat{s}_k^{\mathtt{H}} - \operatorname{Tr}(K)}{\operatorname{Tr}(K)}\right| \leq \varepsilon. \tag{4.22}$$

Using Bernstein inequalities, we can show that our naive estimator (that we denote $\hat{s}_k^{KF}$ here, obtained after sampling k forests) that simply counts the number of roots of KFs concentrates faster:

Theorem 60. *Let $\varepsilon \in (0,1)$ and $\delta \in (0,1)$. If*

$$k \geq \frac{4}{\varepsilon^2 \operatorname{Tr}(K)} \log\frac{2}{\delta}, \tag{4.23}$$

then, with probability larger than $1-\delta$, we have a relative error control:

$$\left|\frac{\hat{s}_k^{KF} - \operatorname{Tr}(K)}{\operatorname{Tr}(K)}\right| \leq \varepsilon. \tag{4.24}$$

Thus, to reach a given multiplicative error ε, one needs less KFs than Hutchinson's estimator needs random vectors. This is a step towards showing theoretically the superiority of KFs on Hutchinson's estimator in this very specific scenario of estimating $\operatorname{Tr}(\mathsf{K})$, that I would like to pursue further in future work.

4.4 KFs for graph Tikhonov regularization and graph signal interpolation

We have just seen how we can use KFs to efficiently approximate the trace of a regularized inverse Laplacian. We will now see how these same Kirchhoff forests can also be used to approximate computations of the form:

$$\mathsf{K}\boldsymbol{x} = (\mathsf{L} + \mathsf{Q})^{-1}\,\mathsf{Q}\boldsymbol{x}$$

two examples of which are graph Tikhonov regularization and graph signal interpolation.

We recall the context of these two problems in Section 4.4.1. We then propose, in Section 4.4.2, several KF-based algorithms to approximate the above matrix-vector multiplication.

4.4.1 Tikhonov regularization and graph signal interpolation

Graph Tikhonov regularization. Graph Tikhonov regularization has already been discussed several times in this document, for instance in Section 2.1.2. We just briefly recall the context here. Given a graph $\mathcal{G}$ and measurements $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)^\top$ on its n vertices, the Tikhonov regularization of $\boldsymbol{y}$ reads:

$$\hat{\boldsymbol{x}} = \operatorname*{arg\,min}_{\boldsymbol{z}\in\mathbb{R}^n} q||\boldsymbol{y} - \boldsymbol{z}||^2 + \boldsymbol{z}^\top \mathsf{L}\boldsymbol{z} \tag{4.25}$$

the exact solution of which is:

$$\hat{\boldsymbol{x}} = \mathsf{K}\boldsymbol{y} \text{ with } \mathsf{K} = (\mathsf{L} + q\mathsf{I})^{-1} q\mathsf{I}. \tag{4.26}$$

In some cases, instead of $(\mathsf{L} + q\mathsf{I})^{-1} q\mathsf{I}\,\boldsymbol{y}$, the generalized solution $(\mathsf{L} + \mathsf{Q})^{-1}\mathsf{Q}\,\boldsymbol{y}$ is required where Q is an entry-wise non-negative diagonal matrix. For example, if we write the Tikhonov regularization of Eq. (4.25) with a regularization term based on another graph Laplacian such as the normalized Laplacian $\mathsf{L}_n = \mathsf{D}^{-1/2}\mathsf{L}\mathsf{D}^{-1/2}$, then the solution reads

$$\begin{aligned}\hat{\boldsymbol{x}} &= (\mathsf{L}_n + q\mathsf{I})^{-1} q\mathsf{I}\,\boldsymbol{y} && (4.27)\\ &= \mathsf{D}^{1/2}\,(\mathsf{L} + q\mathsf{D})^{-1}\,q\mathsf{D}\,\left(\mathsf{D}^{-1/2}\boldsymbol{y}\right) && (4.28)\end{aligned}$$

which requires a computation of the form $(\mathsf{L} + \mathsf{Q})^{-1}\mathsf{Q}\,\boldsymbol{z}$ with $\mathsf{Q} = q\mathsf{D}$ in this case. Another example occurs when the noise variance is known to be non-constant over vertices, *i.e.* heteroscedastic noise. In this case, as the confidence we have *a priori* on the measurements is node-dependent, one should set a different value of q to each node (the larger q_i, the more confidence we have on the measurement of that node).

Graph interpolation. Given a connected graph $\mathcal{G}$, a parameter $\mu \geq 0$, and $\ell \subset \mathcal{V}$ a set of nodes where a signal $\boldsymbol{x}$ is known, one way of defining the interpolated signal

is [Pesenson 2009]:

$$\begin{aligned} \hat{\boldsymbol{x}} &= \arg\min_{z\in\mathbb{R}^n} \boldsymbol{z}^\top (\mathsf{L}+\mu\mathsf{I})\boldsymbol{z} \\ &\text{subject to } \forall i \in \ell,\ z_i = x_i. \end{aligned} \tag{4.29}$$

Define $u = \mathcal{V}\backslash\ell$ the set of nodes for which $\boldsymbol{x}$ is not known and write L in block form:

$$\mathsf{L} = \begin{bmatrix} \mathsf{L}_{\ell|\ell} & \mathsf{L}_{\ell|u} \\ \mathsf{L}_{u|\ell} & \mathsf{L}_{u|u} \end{bmatrix}$$

where $\mathsf{L}_{\ell|u}$ is the Laplacian reduced to its rows and columns indexed by ℓ and u, respectively. The solution of Eq. (4.29) reads:

$$\hat{x}_i = \begin{cases} x_i & \text{if } i \in \ell \\ \left(-(\mathsf{L}_{u|u}+\mu\mathsf{I})^{-1}\,\mathsf{L}_{u|l}\,\boldsymbol{x}_\ell\right)_i & \text{otherwise} \end{cases} \tag{4.30}$$

where $\boldsymbol{x}_\ell \in \mathbb{R}^{|\ell|}$ is the signal $\boldsymbol{x}$ reduced to its entries in ℓ. This solution can almost always[10] be rewritten as:

$$\hat{\boldsymbol{x}}_u = \mathsf{K}\boldsymbol{y} \quad \text{with} \quad \begin{cases} \mathsf{K} = (\mathsf{L}_{\mathcal{G}\backslash\ell} + \mathsf{Q})^{-1}\mathsf{Q} \\ \boldsymbol{y} = -\mathsf{Q}^{-1}\mathsf{L}_{u|\ell}\boldsymbol{x}_\ell \end{cases} \tag{4.31}$$

where $\mathsf{L}_{\mathcal{G}\backslash\ell}$ is the Laplacian of the reduced graph obtained by removing the vertices (and the incident edges) in ℓ, $\mathsf{Q} \in \mathbb{R}^{|u|\times|u|}$ is a diagonal matrix with $\mathsf{Q}_{i,i} = \mu + \sum_{j\in\ell} w(i,j)$.

4.4.2 KF-based estimator

We propose two novel Monte Carlo estimators to approximate $\hat{\boldsymbol{x}} = \mathsf{K}\boldsymbol{y}$ with $\mathsf{K} = (\mathsf{L}+\mathsf{Q})^{-1}\mathsf{Q}$. These estimators leverage the probability distribution of the root process of KFs presented in Thm. 51.

The first estimator, denoted by $\tilde{\boldsymbol{x}}$, is simply defined as follows:

$$\forall i \in \mathcal{V} \quad \tilde{x}(i) = y(r_{\Phi_Q}(i)). \tag{4.32}$$

The strategy to obtain the approximation $\tilde{\boldsymbol{x}}$ is thus:

1. Sample a KF Φ
2. In each tree t of Φ, set $\forall i \in t$, $\tilde{x}(i)$ to the value of $\boldsymbol{y}$ measured at the root of t

One obtains a "piece-wise" constant signal $\tilde{\boldsymbol{x}}$ (constant over each tree). See the top-right illustration in Fig. 4.4.

[10] Q, as defined in the paragraph following Eq. (4.31), needs to be invertible for $\boldsymbol{y}$ to be well-defined. This is always the case if $\mu > 0$. When $\mu = 0$, it may not be the case, but the problem can be side-stepped (not discussed further here).

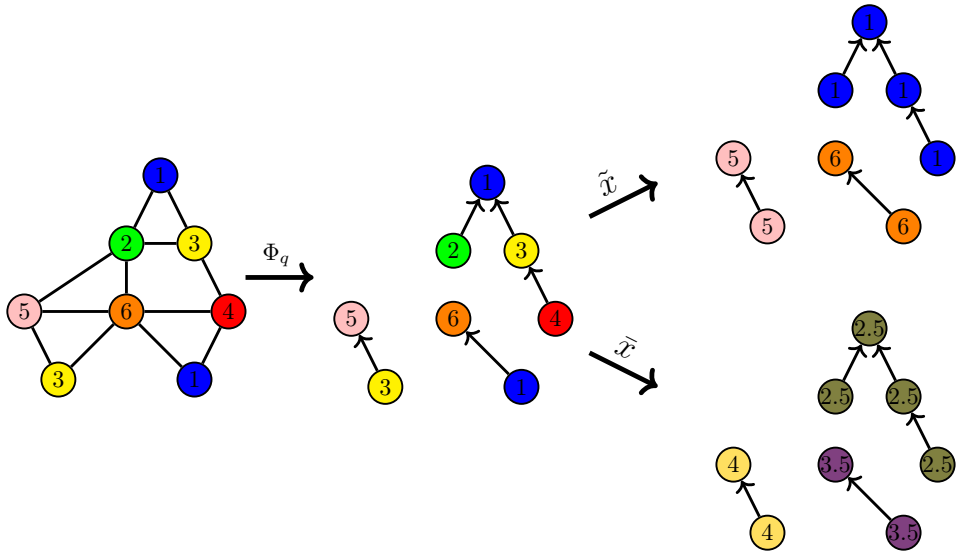

Figure 4.4: An illustration for the estimators where q is constant over all nodes. On the left, the graph signal is represented both via colors and by numbers. In the middle, a realization of Φ, a forest, is illustrated. On this forest, the estimators $\tilde{\boldsymbol{x}}$ and $\bar{\boldsymbol{x}}$ are illustrated in top-right and bottom right, respectively.

Proposition 61. *$\tilde{\boldsymbol{x}}$ is an unbiased estimator of $\hat{\boldsymbol{x}}$:*

$$\mathbb{E}\left[\tilde{\boldsymbol{x}}\right] = \hat{\boldsymbol{x}}.$$

Moreover, the weighted expected error of $\tilde{\boldsymbol{x}}$ is:

$$\mathbb{E}\big(||\hat{\boldsymbol{x}} - \tilde{\boldsymbol{x}}||_{\mathsf{Q}}^2\big) = \sum_{i \in \mathcal{V}} q_i \mathrm{Var}(\tilde{x}(i)) = \boldsymbol{y}^\top (\mathsf{Q} - \mathsf{K}^\top \mathsf{Q}\mathsf{K}) \boldsymbol{y}$$

where $||\boldsymbol{x}||_{\mathsf{Q}}^2 = \boldsymbol{x}^\top \mathsf{Q} \boldsymbol{x}$.

The second estimator, denoted by $\bar{\boldsymbol{x}}$, is the expectation of $\tilde{\boldsymbol{x}}$ conditioned on the partition induced by Φ:

$$\bar{x}(i) = \mathbb{E}[\tilde{x}(i) | \pi(\Phi) = \mathcal{P}] = \frac{\sum\limits_{j \in \mathcal{V}_{t(i)}} y(j) q_j}{\sum\limits_{j \in \mathcal{V}_{t(i)}} q_j}. \tag{4.33}$$

Due to the law of iterated expectations, this estimator is also unbiased, moreover it has a reduced variance compared to $\tilde{x}(i)$ due to the law of total variance:

$$\mathrm{Var}(\tilde{x}(i)) = \mathbb{E}[\mathrm{Var}(\tilde{x}(i) | \pi(\Phi) = \mathcal{P})] + \mathrm{Var}(\bar{x}(i))$$

which implies $\mathrm{Var}(\tilde{x}(i)) \geq \mathrm{Var}(\bar{x}(i))$. This idea of improving an estimator is often called Rao-Blackwellization [Blackwell 1947, Rao 1992].

The strategy to obtain the approximation $\bar{\boldsymbol{x}}$ is thus:

1. Sample a KF Φ
2. In each tree t of Φ, set $\forall i \in t$, $\tilde{x}(i)$ to the average (weighted by the q's) of $\boldsymbol{y}$ on t

See the bottom-right illustration in Fig. 4.4. Note that one still obtains a "piece-wise" constant approximation $\bar{\boldsymbol{x}}$ (constant over each tree). However, this estimator is always strictly better than the previous one.

Proposition 62. *$\bar{\boldsymbol{x}}$ is an unbiased estimator of $\hat{\boldsymbol{x}}$:*

$$\mathbb{E}[\bar{\boldsymbol{x}}] = \hat{\boldsymbol{x}}$$

Moreover, the weighted expected error reads:

$$\mathbb{E}\left(||\hat{\boldsymbol{x}} - \bar{\boldsymbol{x}}||_{\mathsf{Q}}^2\right) = \sum_{i \in \mathcal{V}} q_i \mathrm{Var}(\bar{x}(i)) = \boldsymbol{y}^\top (\mathsf{QK} - \mathsf{K}^\top \mathsf{QK}) \boldsymbol{y}.$$

We applied these estimators to different types of problems including graph semi-supervised learning problems in which a few known labels need to be propagated on the graph. We refer the reader to our paper [Pilavci *et al.* 2021] for the full details of our experiments. In a nutshell, our propositions work reasonably well but cannot compete with the state-of-the-art (which is conjugate gradient). The reason is that we have Monte-Carlo estimators, that have an inherently low variance decrease with the number of samples, whereas conjugate gradient is a very fast iterative method.

In the previous section where our goal was to estimate $\mathrm{Tr}(\mathsf{K})$, we were able to outperform the state-of-the-art because the state-of-the-art itself was limited by the Monte-Carlo nature of Hutchinson's estimator.

We still believe that these two estimators can find some use in some specific cases. It is important to note that our algorithms are *very different* in nature than CG or polynomial-based algorithms:

- they can be fully implemented in about 20 lines of code and they are *very* lightweight in memory footprint. Also, there is no pre-processing needed.
- our algorithms can be used in scenarios where the operator *does not have the full knowledge of the graph*. In fact, if we suppose that each node of a graph can communicate with its neighbors, the operator, at a distance, can contact a node, tell the node to start a KF sampling algorithm (Alg.6). All the following steps can be done locally, until a node "dies" and contacts the operator, *etc.* The operator does not need to know anything about the graph structure, and it will still end up with a sampled KF (and thus will still be able to use our estimators).

4.4.3 What now?

The questions that quite naturally arise from this work are how to generalize this to other types of graph filters. We have designed estimators for the graph filtering

operation of the form:

$$\mathsf{U} g(\Lambda) \mathsf{U}^\top \boldsymbol{y}$$

with $g(\lambda)$ the filter verifying

$$g(\lambda) = \frac{q}{q+\lambda}.$$

A possible direction of research is to to investigate whether we can we use KFs to estimate other graph filters. A first remark is that by using k independent, cascaded KFs, we can trivially estimate graph filtering operations of the form $\left(\frac{q}{q+\lambda_i}\right)^k$. Much more difficult, do we have a chance of finding KF-based algorithms that estimate filters of the form $\simeq \frac{1}{q+\lambda^2}$? Even more difficult, do we have a chance of using KFs or similar distributions of forests to estimate the ideal low pass filtering operation? This would be very valuable in many applications.

Preliminary investigations have explored the possibility to use graph duplication as a powerful tool to use KFs for generalized graph filtering (see section 4.3.2 of [Pilavci 2022]): the idea in a nutshell is to duplicate the graph, connect both copies with parametrized edges and sample KFs on this duplicated graph. Forests are of course more expensive to compute as the graph is larger, but it enables us to compute other forms of graph filters. The family of filters we currently attain is still very restricted and more work in this direction is needed.

Chapter 5

Future directions of research and perspectives

Contents

This last chapter briefly describes short term (Section 5.1), as well as longer term (Section 5.2), research directions of mine. Together with the ideas for future work depicted in the "What now?" sections of the previous chapters, they form a broad picture of my research vision for the next few years.

5.1 Short term

5.1.1 Generalized KFs for magnetic Laplacians: application to graph synchronization

This section is the PhD subject of my current student Hugo Jaquard, that I co-advise with Simon Barthelmé and Pierre-Olivier Amblard (both at Gipsa-lab, in Grenoble), and is already well under way.

Introduction to graph synchronization. Consider a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We will discuss here only the unweighted case for simplicity, even though everything is naturally generalizable to the weighted case. Suppose that on each node i is defined an unknown angle $\omega_i \in [0, 2\pi)$. *Graph synchronization* is the

task of recovering these angles (up to a global phase shift) from noisy pairwise measurements on each edge of the graph. That is, recover all $\{\omega_i\}_{i\in\mathcal{V}}$ from the edge measurements:

$$\forall (i,j) \in \mathcal{E} \qquad \theta_{i,j} = \omega_i - \omega_j + \varepsilon_{i,j}$$

where $\varepsilon_{i,j}$ represents the noise. Such problems arise in various applications, such as ranking problems, or cryo-EM problems where one has access to noisy images of the same molecule taken from unknown angles.

If the graph $\mathcal{G}$ is a tree, our best option is quite trivial: set an arbitrary node to $\hat{\omega}_i = 0$ (recall that we only look for the solution up to a global phase shift) and propagate the solution to all other nodes via the branches of the trees.

Difficulties start to arise when the graph $\mathcal{G}$ is not a tree anymore. In this case, consider the same strategy: set an arbitrary node to $\omega_i = 0$ and propagate that information along paths of the graph. Now, as the graph is not a tree, there are possibly several paths between any pair of node: which path should one choose to propagate the information and recover the signal? In the noiseless case, any path is equivalent. However, in the noisy case, choosing a path rather than another will yield different solutions.

A common optimization strategy in this case (first introduced in [Stella 2009]) is to solve the following global problem:

$$\hat{\boldsymbol{\omega}} = \underset{\boldsymbol{\omega}\in[0,2\pi)^n}{\arg\min} \sum_{(i,j)\in\mathcal{E}} 1 - \cos(\omega_i - \omega_j - \theta_{i,j}). \tag{5.1}$$

Unfortunately, this problem is non-convex and can be NP-hard to solve in general (see for instance [Boumal 2016]). On our way to a relaxation of this optimization problem, one can first show that Equation 5.1 is equivalent to

$$\hat{\boldsymbol{\omega}} = \underset{\boldsymbol{\omega}\in[0,2\pi]^n}{\arg\min} \sum_{(i,j)\in\mathcal{E}} \left| e^{i\omega_i} - e^{i\theta_{i,j}} e^{i\omega_j} \right|^2 \tag{5.2}$$

In fact:

$$\begin{aligned}
\left| e^{i\omega_i} - e^{i\theta_{ij}} e^{i\omega_j} \right|^2 &= \left(e^{i\omega_i} - e^{i\theta_{ij}} e^{i\omega_j} \right) \left(e^{-i\omega_i} - e^{-i\theta_{ij}} e^{-i\omega_j} \right) \\
&= 2 - e^{i(\omega_i - \theta_{ij} - \omega_j)} - e^{i(\theta_{ij} + \omega_j - \omega_i)} \\
&= 2 - \left(e^{i(\omega_i - \omega_j - \theta_{ij})} + e^{-i(\omega_i - \omega_j - \theta_{ij})} \right) \\
&= 2\left(1 - \cos\left(\omega_i - \omega_j - \theta_{ij}\right)\right)
\end{aligned}$$

Now, Equation (5.2) can be re-written in a quadratic form involving the so-called *magnetic Laplacian* that we will now define.

Definition 16. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an unweighted undirected graph. The* magnetic adjacency matrix, *denoted by A_θ is a Hermitian matrix that has the same sparsity pattern (same zeros) than the usual adjacency matrix A. The only difference is that, where $\mathsf{A}(i,j) = \mathsf{A}(j,i) = 1$ in the classical case, one has $\mathsf{A}_\theta(i,j) = e^{i\theta_{i,j}}$ and $\mathsf{A}_\theta(j,i) = e^{-i\theta_{i,j}}$.*

Definition 17. *Let* D *be the usual diagonal, real, degree matrix. The* magnetic Laplacian matrix *of a graph, denoted by* L_θ*, is a Hermitian matrix defined as:*

$$\mathsf{L}_\theta = \mathsf{D} - \mathsf{A}_\theta$$

Proposition 63. *The magnetic Laplacian is positive-semi-definite (via Gershgorin's theorem).*

Proposition 64. *The smallest eigenvalue of* L_θ*,* λ_1*, is equal to zero if and only if the graph, endowed with these pairwise measurements* θ_{ij} *is* coherent*, that is, if the sum of all* θ*'s along any cycle of the graph is null. Otherwise,* $\lambda_1 > 0$*.*

Remark 65. *One recovers the classical Laplacian by setting all* θ*'s to* 0*. In this case, the graph is trivially coherent, and we recover the fact that the smallest eigenvalue of the usual Laplacian is always zero.*

Now that we have defined the magnetic Laplacian, one can easily show that Equation (5.2) can be re-written in the following quadratic form:

$$\forall i \quad \hat{\omega}(i) = \arg\left(\hat{f}(i)\right) \qquad \text{with} \qquad \hat{\boldsymbol{f}} = \underset{\boldsymbol{f}\in U(1)^n}{\arg\min}\ \boldsymbol{f}^*\mathsf{L}_\theta\boldsymbol{f}, \tag{5.3}$$

where $U(1)$ is the unit complex circle, z^* is the complex conjugate of z, and $arg(z)$ the argument of the complex number z.

For now, we have only re-written the synchronization problem of Equation (5.1) and the problem is of course still NP-hard. However, this form calls for the following spectral relaxation:

$$\hat{\boldsymbol{f}} = \underset{\|\boldsymbol{f}\|^2=1}{\arg\min}\ \boldsymbol{f}^*\mathsf{L}_\theta\boldsymbol{f}, \tag{5.4}$$

whose solution is classically given by the eigenvector of L_θ associated to its smallest eigenvalue λ_1.

There are many ways to numerically obtain the solution to this problem, one of which is the (inverse) *power method* [Saad 2011], which performs iterative smoothing of an original guess $\boldsymbol{f}_0 \in \mathbb{C}^n$ by computing the iterates

$$\boldsymbol{f}_{k+1} = (\mathsf{L}_\theta + q\mathsf{I})^{-1}\, q \cdot \boldsymbol{f}_k \tag{5.5}$$

for some positive parameter $q \in R_+^*$, and where we of course recognize the regularized inverse form we have been working a lot with in Chapter 4. The difference here is that $\boldsymbol{f}$ is a complex vector and L_θ also has complex entries.

Preliminary results: Mixed Type Spanning Forests. Similarly to our work in Chapter 4, iterations of the form 5.5 can be either estimated by state-of-the-art methods such as (preconditioned) conjugate gradient, or can be estimated by graph-based DPPs, if only we are capable of i/ understanding what is the underlying DPP, ii /sampling from it efficiently.

Without entering too much into details, we have shown in Hugo's PhD (some results are published, others are submitted) that:

- instead of KFs, the underlying random objects are so-called Mixed Type Spanning Forests (MTSFs): they are a spanning collection of disjoint rooted trees (same as in Chapter 4) and unicycles. Unicycles are subgraphs that contain only one cycle. These unicycles are not rooted.

- in a somewhat similar fashion than our estimators of Chapter 4 (see Section 4.4), one can propagate the information from the roots to all trees of the MTSFs to obtain an unbiased estimator $(\mathsf{L}_\theta + q\mathsf{I})^{-1} q \cdot \boldsymbol{f}$. Each sampled MTSF gives one unbiased estimate.

- variance reduction techniques are also available in this case.

- these MTSFs can be efficiently sampled via a variant of Wilson's algorithm, but only when the graph is not too incoherent. When the incoherence is too strong, importance sampling tricks need to be applied to be able to take advantage of very fast Wilson-like algorithms.

Moving further. We are currently investigating the following directions:

- Find a Wilson-like algorithm, or an algorithm as efficient, able to sample MTSFs even in the very incoherent cases, without the need of importance sampling workarounds.

- Connect with the computer graphics literature, in which the so-called Vector Heat Method [Sharp *et al.* 2020] is a central method for parallel transport on manifolds, a foundational brick of which is precisely the computation of regularized inverses of magnetic Laplacians.

- Investigate the best choice of the parameter q in this context, perhaps with a varying q as the iterations of the power method unfold.

5.1.2 Convergence of Graph Neural Networks on large random graphs

This section is the PhD subject of my current student Matthieu Cordonnier, that I co-advise with Nicolas Keriven (Rennes) and Samuel Vaiter (Nice), and is already well under way.

One avenue of research regarding theoretical guarantees on GNNs is to investigate their properties on random graph models, in the limit when n, the number of nodes, tends to infinity. The proper definition, and the properties, of the limit operator that arises, can give us insights on the theoretical behavior of GNNs on large graphs.

Previous work on this topic [Keriven *et al.* 2020] introduces the concept of cGNN ("continuous GNN") and proves the convergence, on some random graph models, of GNNs towards their cGNN counterpart. However, they only consider so-called

spectral Graph Neural Networks [Defferrard *et al.* 2016] (SGNNs), based on the interpretation of the convolutional product as a product of frequencies in the Fourier domain. In most of the recent literature, the more versatile Message Passing Neural Networks (MPGNNs) [Gilmer *et al.* 2017, Kipf & Welling 2017] is usually preferred. The message passing paradigm consists of iteratively updating each node via the aggregation of messages from each of its neighbors. Its flexibility comes from the fact that messages and aggregation functions are unconstrained as long as they stay invariant to node reordering, *i.e.* as long as they match on isomorphic graphs. Recently, authors in [Maskey *et al.* 2022] proved a convergence result for MPGNNs but they only consider the case where the aggregation scheme is the local mean, which is a very restrictive case.

Our goals, in this thesis, is i) to establish convergence theorems of generic MPGNNs on random graph models, ii) study the resulting continuous operators that arise.

Random graphs. Let $\mathcal{X}$ be a compact subset of $\mathbb{R}^d$ and P a probability measure on $\mathcal{X}$. A *random graph model* is a couple (W, P) where $W : \mathcal{X} \times \mathcal{X} \mapsto [0, 1]$ is a kernel, *i.e.*, a symmetric measurable function.

Random graphs are generated from a random graph model (W, P) as follows. Given a positive integer n, first draw n iid random variables from P, represented by $X_1, \dots, X_n$: they form the vertex set of the graph. The random graph obtained is fully connected and its edge weights verify:

$$X_1, \dots, X_n \overset{iid}{\sim} P, \quad w_{i,j} = w_{j,i} = W(X_i, X_j).$$

We denote by $\mathcal{G}_n(W, P)$ the distribution of these random graphs.

Message Passing Graph Neural Networks (MPGNNs). A multilayer MPGNN iteratively propagates a signal over a graph. At each step, the current representation of every node's neighbors are gathered, transformed, and combined to update the node's representation. Broadly speaking, a MPGNN can be defined as a collection of L applications $(F^{(l)})_{1 \leq l \leq L}$ that act as follows. Let $G \in \mathcal{G}(V)$ be a graph with n nodes, and $Z = Z^{(0)} \in \mathbb{R}^{n \times d_0}$ be a signal on it. At each layer, denoting $Z^{(l)}$ as the current state of the signal, $Z^{(l+1)}$ is computed node-wise by:

$$z_i^{(l+1)} = F^{(l+1)}\left(z_i^{(l)}, \left\{\!\!\left\{ \left(z_j^{(l)}, w_{i,j}\right) \right\}\!\!\right\}_{v_j \in \mathcal{N}(v_i)}\right) \in \mathbb{R}^{d_{l+1}}. \tag{5.6}$$

So $Z^{(l+1)}$ is a $n \times d_{l+1}$ tensor. In Equation (5.6), the $F^{(l)}$ take as arguments a vector, which is the current node's representation, and a multiset of pairs. Each pair is composed of a node from the neighborhood of the aforementioned running node, along with the corresponding weight. In the literature, the $F^{(l)}$ are often referred to as *aggregations* [Jegelka 2022]. Their major property is to ignore the order in which the neighborhood information is collected, which is handled by the use of a multiset.

The final output of the MPGNN is a signal over the graph denoted as:

$$\Theta_G(Z) = Z^{(L)} \in \mathbb{R}^{n \times d_L} \tag{5.7}$$

A fundamental requirement for GNNs is to be consistent with graph isomorphism. More precisely, relabeling the nodes of the input graph signal must be the same as relabeling the nodes of the output in the *equivariant* case, and must leave the output unchanged in the *invariant* case.

Let us show four examples of MPGNNs, the first three of which are very popular in the literature:

Example 66 (Convolutional Message Passing [Kipf & Welling 2017, Defferrard *et al.* 2016, Wu *et al.* 2021])**.** . *Each neighbor representation is multiplied by its corresponding weight and we combine them with an arithmetic mean. Notice that this is equivalent to a SGNN with polynomial filters of degree one.*

$$z_i^{(l+1)} = \frac{1}{|\mathcal{N}(v_i)|} \sum_{v_j \in \mathcal{N}(v_i)} w_{i,j} \psi^{(l+1)}\left(z_j^{(l)}\right).$$

Example 67 (Degree normalized convolution)**.** *A weighted mean is performed [Maskey et al. 2022].*

$$z_i^{(l+1)} = \sum_{j \in \mathcal{N}(v_i)} \frac{w_{i,j}}{\sum_{k \in \mathcal{N}(v_i)} w_{i,k}} \psi^{(l+1)}\left(z_j^{(l)}\right).$$

Example 68 (Attention based Message Passing)**.** *The mean is weighted by the attentional coefficients, that may depend on all the possible parameters and are learnable [Veličković et al. 2017].*

$$z_i^{(l+1)} = \sum_{j \in \mathcal{N}(v_i)} \frac{c^{(l+1)}\left(z_i^{(l)}, z_j^{(l)}, w_{i,j}\right)}{\sum_{k \in \mathcal{N}(v_i)} c^{(l+1)}\left(z_i^{(l)}, z_k^{(l)}, w_{i,k}\right)} \psi^{(l+1)}\left(z_j^{(l)}\right).$$

Example 69 (Max Convolutional Message Passing)**.** *The aggregation maximum is often mentioned as a possibility in the literature but is rarely treated [Hamilton et al. 2017]. An element-wise maximum weighted by edge weights is used to combine everything:*

$$z_i^{(l+1)} = \max_{v_j \in \mathcal{N}(v_i)} w_{i,j} \psi^{(l+1)}\left(z_j^{(l)}\right).$$

Continuous MPGNNs (cMPGNNs) on random graph models. We define the continuous counterpart of MPGNNs, that we call continuous MPGNNs (cMPGNNs). Analogously to the discrete case, a cMPGNN is defined to be L operators $(\mathcal{F}^{(l)})_{1 \leq l \leq L}$ that propagate a function on $\mathcal{X}$ relatively to a random graph model. Let (W, P) be a random graph model and $f = f^{(0)} \in L_P^\infty(\mathcal{X}, \mathbb{R}^{d_0})$, $f^{(l+1)}$ is recursively computed by:

$$\forall x \in \mathcal{X} \quad f^{(l+1)}(x) = \mathcal{F}_P^{(l+1)}\left(f^{(l)}(x), \left(f^{(l)}, W(x, \cdot)\right)\right) \in \mathbb{R}^{d_{l+1}}. \tag{5.8}$$

Notice that $\mathcal{F}^{(l+1)}$ depends on the measure P. Considering the functions $f^{(l)}$ as signals on the vertex set $\mathcal{X}$, the update $f^{(l+1)}(x)$ of a node $x \in \mathcal{X}$ is calculated from the knowledge of its current representation $f^{(l)}(x)$ and all its "weighted neighborhood" $(f^{(l)}, W(x,\cdot))$. The latter being a short notation for the map $y \mapsto (f^{(l)}(y), W(x,y))$ at x fixed, which is the continuum equivalent of the multiset of pairs of weighted neighbors $\{\!\{(z_j^{(l)}, w_{i,j})\}\!\}_{v_j \in \mathcal{N}(v_i)}$ from (5.6). We denote $\Theta_{W,P}(f) = f^{(L)}$ the output.

$$\Theta_{W,P}(f) = f^{(L)} \in L_P^\infty(\mathcal{X}, \mathbb{R}^{d_L}). \tag{5.9}$$

In the following are some examples of cMPGNN. The reader will of course see the connection to the previous Examples 66, 67, 68 and 69.

Example 70 (Convolutional Message Passing, continuous counterpart of Example 66)**.** *The arithmetic mean becomes an integral over the probability space:*

$$f^{(l+1)}(x) = \int_{y \in \mathcal{X}} W(x,y) \psi^{(l+1)}\left(f^{(l)}(y)\right) dP(y)$$

Example 71 (Degree Normalized Convolutional Message Passing, continuous counterpart of Example 67)**.** *The continuous counterpart is:*

$$f^{(l+1)}(x) = \int_{y \in \mathcal{X}} \frac{W(x,y)}{\int_{t \in \mathcal{X}} W(x,t) dP(t)} \psi^{(l+1)}\left(f^{(l)}(y)\right) dP(y).$$

Example 72 (Attention based Message Passing, continuous counterpart of Example 68)**.** *The continuous counterpart is:*

$$f^{(l+1)}(x) = \int_{y \in \mathcal{X}} \frac{c^{(l+1)}\left(f^{(l)}(x), f^{(l)}(y), W(x,y)\right)}{\int_{t \in \mathcal{X}} c^{(l+1)}\left(f^{(l)}(x), f^{(l)}(t), W(x,t)\right) dP(t)} \psi\left(f^{(l)}(y)\right) dP(y).$$

Example 73 (Max Convolutional Message Passing, continuous counterpart of Example 69)**.** *The maximum becomes a component-wise essential supremum according to the probability measure P:*

$$f^{(l+1)}(x) = \operatorname*{ess\,sup}_{y \in \mathcal{X},\, P} W(x,y) \psi^{(l+1)}\left(f^{(l)}(y)\right)$$

Preliminary results. Let (W, P) be a random graph model, and $(G_n)_{n \geq 1}$ be a sequence of random graphs drawn from $\mathcal{G}_n(W, P)$. Consider a L-layer MPGNN $(F^{(l)})_{1 \leq l \leq L}$, and its continuous counterpart $(\mathcal{F}^{(l)})_{1 \leq l \leq L}$. Without delving into the numerous technical details, we have the following convergence result (submitted):

Theorem 74 (MPGNN convergence towards cMPGNN)**.** *Under a few technical assumptions, the MPGNN $(F^{(l)})_{1 \leq l \leq L}$ converges to its continuous counterpart $(\mathcal{F}^{(l)})_{1 \leq l \leq L}$, as n tends to infinity. The speed of convergence can be controlled and depends on the particular MPGNN model. Notably, all four examples given above converge to their continuous counterparts.*

Moving further. We are currently investigating the following research directions.

- The random graph model used here is a simplified version of the latent position model where there is no randomness in the edge appearance (graphs are complete). Extending our analysis to improved version of this model is a natural further way of research: for instance a random graph where each edge is a Bernoulli random variable and exists only with probability proportional to $W(X_i, X_j)$. We could thus explore more realistic sparse random graph.

- Now that we have properly defined cMPGGNNs, we envision to take advantage of this to study properties of these continuous objects. Most notably, we would like to obtain possible results on universality. In fact, a major concern in the field of GNNs is the one of *expressive power*, *i.e.*, what class of functions can GNNs approximate. This fundamental notion is related to the classical *Universal Approximation Theorem* [Kurt & Hornik 1991, Cybenko 1989] in deep learning. We envision to investigate this via our new cMPGNN lens.

5.1.3 A DPP point-of-view on Kirchhoff forests: new proofs of known results and new perspectives

One learns a lot by taking a resolute DPP point-of-view on Kirchhoff forests. As said earlier, USTs are a well-known example of DPP over the edges of a graph. It is however not as well known that KFs are a DPP as well. Exhibiting the marginal kernel of KFs can then automatically provide new proofs of known results on KF (on the root process's DPP kernel, on the probability of node i to be rooted in node j, *etc.*). This DPP point-of-view also enables to write marginals of KFs that have not been studied until now, such as the leaf process of KFs. Because KFs have mainly been studied from a probabilistic point-of-view in the past (and not from the more algebraic point-of-view of DPPs), this direction of research promises new results on KFs.

For the following, let us define another important matrix associated to a graph $\mathcal{G}$ with n nodes and $|\mathcal{E}|$ edges: its edge-incidence matrix $\mathsf{B} \in \mathbb{R}^{n\times|\mathcal{E}|}$. It is built by: i/ considering an arbitrary orientation of the edges, ii/ to the l-th (oriented) edge $e_l = (i, j)$ associating a vector $\boldsymbol{b}_l \in \mathbb{R}^n$ defined with $b_l(i) = -\sqrt{\omega_l}$, $b_l(j) = \sqrt{\omega_l}$, where $\omega_l > 0$ is the weight of edge l, and zero everywhere else, iii/ creating B by concatenating all such vectors: $\mathsf{B} = (\boldsymbol{b}_1|\boldsymbol{b}_2|\dots|\boldsymbol{b}_{|\mathcal{E}|})$. A well-known result states that, independently of the choice of orientation, the Laplacian matrix of $\mathcal{G}$ verifies $\mathsf{L} = \mathsf{B}\mathsf{B}^\top$ [Chung 1997]. Similarly, all results involving B in the following are valid regardless of the choice of orientation.

The following is a well-known result:

Proposition 75. *A UST is a projective DPP with kernel* $\mathsf{K} = \mathsf{B}^\top\mathsf{L}^\dagger\mathsf{B}$.

The bijection of Definition 13 between KFs on $\mathcal{G}$ and USTs on the extended graph $\mathcal{G}_{\text{ext}}$ is now useful to lift this result on the extended graph. In fact, Proposition 75 applied on the extended graph $\mathcal{G}_{\text{ext}}$ gives:

Proposition 76. *T_{ext}, a UST of $\mathcal{G}_{\text{ext}}$, is a projective DPP with marginal kernel:*

$$\mathsf{K}_{\text{ext}} = \begin{bmatrix} \mathsf{B}^\top \\ \sqrt{\mathsf{Q}} \end{bmatrix} (\mathsf{L}+\mathsf{Q})^{-1} \begin{bmatrix} \mathsf{B} & \sqrt{\mathsf{Q}} \end{bmatrix} \in \mathbb{R}^{(|\mathcal{E}|+n)\times(|\mathcal{E}|+n)}, \tag{5.10}$$

where the first $|\mathcal{E}|$ lines and columns index the edges in $\mathcal{E}$ and the last n lines and columns index the new edges connecting node Γ to $\mathcal{V}$ in the extended graph.

Now that we have established that KFs are DPPs over the set of edges and nodes of a graph and shown its particular marginal kernel, we now can move on to showing how this determinantal point-of-view enables to prove known results on KFs.

A first striking example is the root process: Theorem 49. This Theorem can be proven in one line using Proposition 76 combined to the fact that DPPs are stable under restriction, whereas it requires a long proof if observed from the lens of (loop-erased) random walks.

Interestingly, this DPP point-of-view also enables to somewhat easily study marginal probabilities of KFs that have never been studied before. We give here an example where we concentrate on the marginal probability that a given node is a leaf in a KF. A leaf in Φ is a node that has degree 1. If node i has degree one but is also a root then we do not consider it to be a leaf.

Theorem 77. *Let Φ be random spanning forest and denote by $\mathcal{L} \subseteq \mathcal{V}$ its set of leaves. The probability that node i is a leaf may be written:*

$$\mathbb{P}(i \in \mathcal{L}) = d_i \frac{\det\left(\mathsf{L}_{\mathcal{G}_{\neg i}} + \mathsf{Q}_{\neg i}\right)}{\det\left(\mathsf{L}+\mathsf{Q}\right)}, \tag{5.11}$$

where $d_i = \sum_{j\in\mathcal{V}} \omega_{ij}$ is the degree of node i, and $\mathsf{L}_{\mathcal{G}_{\neg i}}$ is the Laplacian associated to the graph to which node i, along with all its edges*, are removed.*

An interesting avenue of research here would be to understand this leaf probability as a sort of centrality measure, and compare it without the many other types of centrality that exist in the literature. In fact, intuitively, a node that has a high probability of being a leaf is probably on the "edge" of the graph, whereas a node with a low probability of being a leaf is more central to the graph.

Let us now recall the following result from Derezinski [Derezinski & Mahoney 2021] (but known since at least Maurer [Maurer 1976]):

Theorem 78. *Let $X \sim \texttt{DPP}(\mathsf{Q}\mathsf{Q}^\top)$ be a projective DPP of size k, with $\mathsf{Q} \in \mathbb{R}^{n\times k}$ and $\mathsf{Q}^\top\mathsf{Q} = \mathsf{I}_k$ ($n \geq k$). Let $\boldsymbol{x}$ be any vector in dimension n. Then:*

$$\mathbb{E}_X\left((\mathsf{Q}_{X,:})^{-1} \boldsymbol{x}_X \right) = \mathsf{Q}\top\boldsymbol{x} \tag{5.12}$$

where we recall that the notation $\boldsymbol{x}_X \in \mathbb{R}^k$ stands for the reduction of $\boldsymbol{x}$ on the elements indexed by X.

Applying this theorem to KFs, one obtains:

Corollary 79. *Let T_{ext} be a UST on $\mathcal{G}_{\text{ext}}$, $\boldsymbol{y}$ any vector in dimension $|\mathcal{E}|$ and $\boldsymbol{z}$ any vector in dimension n, then:*

$$\mathbb{E}_{T_{ext}}\left(\left(\begin{bmatrix}\mathsf{B}^\top \\ \mathsf{I}_n\end{bmatrix}_{T_{ext},:}\right)^{-1}\begin{bmatrix}\boldsymbol{y} \\ \boldsymbol{z}\end{bmatrix}_{T_{ext}}\right) = (\mathsf{L}+\mathsf{Q})^{-1}(\mathsf{B}\,\boldsymbol{y}+\mathsf{Q}\,\boldsymbol{z})\,. \tag{5.13}$$

This interesting corollary enables for instance to show in a few lines that the probability that node i is rooted in node j, in a KF, is K_{ij} (Theorem 51). It also enables to obtain new formulas on KFs. I am currently pursuing this direction of research by applying Corollary 79 to different choices of vectors $\boldsymbol{y}$ and $\boldsymbol{z}$, yielding new insights on properties of KFs that were until now unknown.

5.2 Longer term

5.2.1 What other graph property can we estimate via KFs?

The results we have obtained until now on KFs and what graph-property they are able to estimate, do not take into account the random walks' output in its entirety. For instance, the estimation of $\mathrm{Tr}\,(\mathsf{K})$ only counts the number of roots of KFs, and completely discards their structure in trees, the number of times a given node and/or edge was visited, the number and position of leaves in the forest, etc.

The goal of this first direction of research is to i/ identify the different graph-based Numerical Linear Algebra (NLA) tasks one can solve via Wilson's algorithm, ii/ design, study and implement the relevant KF-based algorithms for each task.

Novel results can surely be obtained in this direction. Of particular interest, I will investigate to what extent can KFs i/ estimate effective resistances, ii/ more generally, estimate parts of the spectrum of the Laplacian.

KFs for effective resistance estimation. *The ideas in this paragraph are born from discussions with my ex-PhD student Yigit Pilavci (now in Lille).*

The effective resistance of edge (ij) is defined as $\mathsf{R}_{ij} = \mathsf{L}^\dagger_{ii} + \mathsf{L}^\dagger_{jj} - 2\mathsf{L}^\dagger_{ij}$, where $\mathsf{L}^\dagger$ is the Moore-Penrose pseudo-inverse of L (in fact, any generalized inverse of L could be used instead of $\mathsf{L}^\dagger$ [Ali *et al.* 2020]). Estimating effective resistances efficiently is an important task for many graph-based tasks. For instance, a famous and beautiful result from Spielman et al. [Spielman & Srivastava 2011] state that one can obtain spectral sparsifiers of graphs[1] by sampling iid with replacement $\mathcal{O}(n\log n)$ edges from a distribution proportional to effective resistances.

The state-of-the-art today to estimate effective resistances is via polynomial approximation and random projections.

However, effective resistances have some strong connections to USTs, and there is a chance that KFs can have an important role to play in the design of new estimation

[1] loosely speaking, a spectral sparsifier of a graph $\mathcal{G}$ is a sparsified version of $\mathcal{G}$ (of which only some edges are sampled) that preserves some spectral properties of (the Laplacian of) $\mathcal{G}$

algorithms. It is indeed a well-known fact that $w_{ij}\mathsf{R}_{ij}$ is the probability that edge (ij) is sampled in a UST, which yields a first natural Monte-Carlo estimator of R_{ij}: sample many USTs and count the proportion of trees that include (ij). This strategy cannot be competitive with random projections, especially in cases where $w_{ij}\mathsf{R}_{ij}$ is small. The question to address is how the versatility of KFs can help estimating R_{ij} more efficiently. Preliminary calculations show that by sampling KFs constrained to having exactly two roots (i and j), one can obtain useful marginals (see Section 4.2 of [Pilavci 2022]). These however need to be largely improved to have a chance to compete with the state-of-the-art.

KFs for spectrum estimation. *The ideas in this paragraph have already been discussed several times with my colleagues Simon Barthelmé (Grenoble), Luca Avena (Netherlands), Matteo Quattropani (Rome), Alexandre Gaudillière, Clothilde Mélot and Fabienne Castell (all three in Marseille).*

A more long-term goal that will drive my future research is (partial) estimation of the spectrum of L. Examples of questions could be the efficient estimation of the k-th eigenvalue (with k potentially large) – a difficult NLA problem nonetheless useful for machine learning applications such as in [Tremblay *et al.* 2016, Martin *et al.* 2018], the efficient estimation of the number of eigenvalues in a given interval [Napoli *et al.* 2016], the efficient estimation of the eigenvalues' distribution in order to adapt polynomial approximation techniques to the specific density at hand [Fan *et al.* 2020], etc.

A starting point linking KFs and the spectrum of L is the expected number of roots that we write here as a function of q: $g(q) = \mathbb{E}(|\rho(\Phi_q)|) = \mathrm{Tr}(\mathsf{K}) = \sum_i \frac{q}{q+\lambda_i} = 1+\sum_{i\geq 2} \frac{q}{q+\lambda_i}$ (the smallest eigenvalue of a graph is necessarily 0). This function g is a strictly increasing function defined on $\mathbb{R}^{+*}$, tends to 1 as q tends to 0^+, and to n as q tends to $+\infty$. Noting that all $\lambda_{i\geq 2}$ are strictly positive in a connected graph, and as $q > 0$, one can make the following changes of variables: $\forall i \geq 2, \exists\, \nu_i \in \mathbb{R} \mid \lambda_i = e^{\nu_i}$ and $\exists \eta \in \mathbb{R} \mid q = e^{\eta}$, yielding the following convolution formula:

$$\tilde{g}(\eta) = 1 + \sum_{i\geq 2} \frac{1}{1+e^{-(\eta-\nu_i)}} = 1 + \ell(\eta) \circledast \sum_{i\geq 2} \delta(\eta - \nu_i)$$

where $\ell(\eta) = 1/(1+e^{-\eta})$ is the standard logistic function and $\sum_{i\geq 2} \delta(\eta - \nu_i)$ is directly linked to the empirical eigenvalue distribution of L that we want to estimate. An interesting aspect of random forests that we can now leverage (and that we have not discussed yet in this manuscript) is that there exists a coupled forest algorithm [Avena & Gaudillière 2017] that can efficiently estimate $g(q)$ on an interval $[q_{\min}, q_{\max}]$ in time $\mathcal{O}(m/q_{\min})$, thus transforming the eigenvalue density's estimation into a deconvolution problem. Given the very flat nature of the logistic function, a straightforward deconvolution with standard signal processing tools will provide spectrum estimates with very large errors. However, looking at other KF observables than just the number of roots is possible and yields convolutions with narrower functions, that I intend to use.

5.2.2 Other DPPs overs graphs

The ideas in this section are born from discussions with my colleague Simon Barthelmé.

We know that KFs are a DPP with marginal kernel of the form

$$\begin{bmatrix} \mathsf{B}^\top \\ \sqrt{q}\mathsf{I} \end{bmatrix} (\mathsf{L} + q\mathsf{I})^{-1} \begin{bmatrix} \mathsf{B} & \sqrt{q}\mathsf{I} \end{bmatrix} \in \mathbb{R}^{(m+n)\times(m+n)} \tag{5.14}$$

where the first m lines and columns index the edges of the graph, and the last n lines and columns index the nodes of the graph.

However, this random process is but a very specific example of DPP over the links and/or the nodes of a graph. In fact, many other DPPs can be defined over graphs and µI will aim at designing new marginal kernels that i/ are useful to estimate standard Laplacian-based linear algebra quantities, ii/ remain efficient to sample. Even though finding sampling algorithms as fast as Wilson seems illusory in the generic case, recent advances in both DPP sampling schemes [Gillenwater *et al.* 2019, Derezinski *et al.* 2019, Gautier 2020b] and fast random walk samplers [Kelner & Madry 2009, Anari *et al.* 2020] pave the way to find, if not fully random walk based algorithms, hybrid schemes mixing fast random walks and partial standard linear algebra computations.

A starting point is to explore two directions generalizing the KF DPP. One of them consists in generalizing the edge-based random walks to higher-order random walks, that is, random walks on simplicial complexes [Kaufman & Oppenheim 2018, Alev & Lau 2020]. Another direction is to generalize the first-order differential operator B in Eq. (5.14) to higher order differential operators. Let us illustrate this idea with a simple example. Consider the path graph of size $n = 300$ and the root process associated with KFs. Denote by π_i the marginal probability of sampling i and define the following measure of repulsion: $\frac{\pi_{ij}}{\pi_i \pi_j}$, where π_{ij} is the joint probability of sampling i and j. Fixing j to be the central node of the path graph, Figure 5.1 (in red) shows $\frac{\pi_{ij}}{\pi_i \pi_j}$ as a function of i: it is null for $i = j$, and increases symmetrically as i is farther away from j. This increase is quite rapid and one would like, in order to design better graph estimation algorithms, sampling schemes that are more repulsive. The idea of using higher order differential operators in the definition of the DPP kernel is a natural way of increasing such repulsion. For instance, by adding L, the second order differential operator to the first (the edge-incidence matrix B) and zero (the identity matrix I) order operators to the definition of the kernel of Eq. (5.14), one can obtain[2] a DPP with more repulsion (see the curve in blue in Figure 5.1).

A trade-off is expected between the order of these generalizations (whether it be the simplicial or the differential generalization) and computation time: the higher the order, the more repulsion is added in the process (thus increasing the effectiveness of the associated DPPs for RandNLA applications), the more expensive are the associated algorithms.

[2] in practice here we show the result of sampling a DPP with kernel $\left(\mathsf{L}^2 + \mathsf{L} + q\mathsf{I}\right)^{-1} q\mathsf{I}$ instead of the usual $(\mathsf{L} + q\mathsf{I})^{-1} q\mathsf{I}$ of the root process

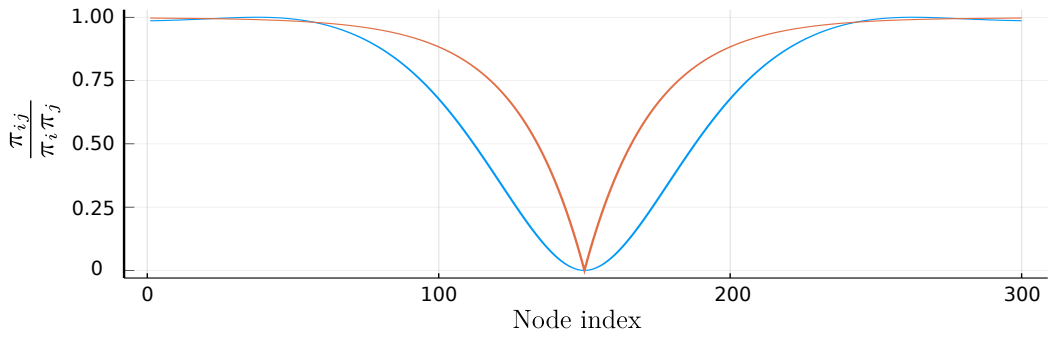

Figure 5.1: **Illustration of the difference in repulsion between the root DPP, and a generalized higher order version.** The underlying graph is the path graph of size 300. The index j is set to the middle node (index 150) and the normalized joint probability of sampling both i and j, $\frac{\pi_{ij}}{\pi_i \pi_j}$, is represented as a function of i for (in red) the first-order graph DPP (the root process of KFs) and (in blue) a second order DPP. As expected, the higher the order, the more repulsive the associated process.

5.2.3 Graph Neural Networks: towards random pooling layers for increased trustworthiness?

The ideas in this section are born from discussions with my colleagues Filippo Bianchi and Benjamin Ricaud (both at UiT, Norway).

This is a very prospective section that turns around one simple idea: what if we use *random* pooling layers in GNNs? *Pooling layers* are one of the building blocks of many neural networks, including GNNs. In a nutshell, pooling layers generate local summaries of the data to (i) enable faster computations by reducing the dimensionality; (ii) generate multiscale representations of the data by gradually extracting global properties. Existing pooling solutions in GNNs stem from a trial-and-error approach often driven by two practical and somewhat short-sighted objectives: computation efficiency and empirical predictive performance. It seems that there is room here for more principled approaches.

My aim in this direction of research is to incorporate *random methods* (mainly random sampling and random projections) into the design of pooling layers for GNNs. Randomized approaches exhibit three desirable properties. (i) It is often much easier to prove performance bounds of random methods compared to deterministic ones, bringing *reliability*. (ii) Randomness provides a measurable sense of uncertainty, bringing *interpretability*. (iii) Pooling based on random aggregation methods is a key privacy-preserving tool that keeps the important structural information while discarding the specific and sensitive data. Reliability, interpretability and privacy are core elements of trustworthiness and it is worth investigating if random pooling layers can help increase GNNs' trustworthiness.

Research for trustworthy GNNs has been largely conducted in the past few years

– the most recent review of which contains 247 references [Dai *et al.* 2022]. The topics of robustness, privacy, fairness, and explainability are thoroughly discussed and existing methods are categorized according to different possible definitions of these four core concepts. However, even if randomness is used in a few cases (*e.g.*, to create surrogate data to test for robustness), it is not used as a core principle and never for pooling layers.

Requirements for an ideal pooling layer. When it comes to pooling, all the methods in the GNN literature are driven by one or several of the nine following desiderata. A pooling layer should: (i) be computationally efficient since pooling must be done many times during training and in real-time during inference; (ii) account for both the topology and the node features, which carry additional information and are transformed by the GNN according to the downstream task at hand; (iii) be scalable and, possibly, distributable over mini-batches when the GNN is applied on large graphs; (iv) be differentiable in order to adapt its behavior based on the training objective; (v) generalize well to data not seen in training; (vi) be expressive, *i.e.*, it should produce different outputs for non-isomorphic graphs, (vii) be interpretable; (viii) respect privacy when needed; (ix) have provable invariants that are well-adapted to the learning task at hand. An invariant is a characteristic (the degree distribution, the distribution of the spectrum of the adjacency matrix, *etc.*) of the data $(\mathcal{G}, \mathbf{X})$ that can be reliably recovered from the summarized data $(\mathcal{G}_p, \mathbf{X}_p)$.

To date, all existing methods focus only on a subset of these desiderata. I want to investigate if random pooling methods have a chance to jointly address all nine points. In fact, all existing methods are deterministic, which are known to be difficult to study theoretically. To overcome this limitation, I intend to add randomness in the sampling design, inspired by statistical approaches. This will provide a strong theoretical framework to provably address points (vii) to (ix) of the ideal pooling layer. The main foreseeable challenge will be to adapt existing random methods from the theoretical literature to the highly demanding engineering environment of GNNs: verify points (i) to (vi) of the ideal layer while preserving the theoretical guarantees these random methods come with.

5.3 A few last words

I cannot finish this manuscript without a more personal reflection on my work and how I relate to it. I take a lot of pleasure in *doing math*. It is an activity I often see as an art, in the way it asks for creativity and intuition. To the many non-scientific people asking me "aren't we done yet with research in math?" I like to answer that there are as many theorems out there to discover than there are poems to write. I also take a lot of pleasure in designing and implementing new algorithms, taking time to find the subtle ways of an acceleration, or the perfect trick to correctly represent the data. Algorithms are often treasures of ingenuity. Perhaps what I love the most about my work is the social part: chalk, black board, coffee, and collective brainstorming.

To these very positive aspects are opposed many challenges. On top of challenges I will not talk about here (the competition, the grant writing, the low SNR of the current scientific publications –to which I am partly responsible of), one of them stands out by its complexity and how much it affects me: ethical considerations. Q1: What is the impact of my research on society? Q2: Are these impacts fully in line with my own set of values?

A first reasonable answer to Q1 is, humbly, ε. However, this answer is an easy way out of the ethical uneasiness I sit with. I am in fact fully aware that I am part of a research system that is partly (if not solely) driven by economic considerations, and that the many incentives I have from all parts to be "innovative" are more for the possible economical growth it can lead to rather than for the sheer beauty of the theorem's proof.

As for Q2, the answer is clearly: no, the impacts of the research communities I belong to are not fully in line with my set of values. Or rather, they are only partly in line. When I go in an MRI scanner, I do feel proud to be part of the signal processing community that largely helped to improve those medical imaging machines. When I read or hear colleagues in mainstream media debunking conspiracy theories, or answering point-by-point to wacky arguments from a politician or another, I am proud to be part of an academic community that holds facts, reproducibility, critical thinking as a compass. However, when I am confronted to the environmental impact of research (the power consumption of large scale learning algorithms, the many conferences on the other side of the world inducing so many travels by plane) and the technologies that may result from it (Do we need 6G? Do we need high-resolution streaming on our small screen smartphones? Do we need generative models?), I start to stagger. When I am confronted to the worst possible applications of ML (say, face or emotion recognition, not even to mention autonomous weapons), I usually want to hide somewhere and stay there.

I have my own clear red lines I have never crossed, but those are easy to draw, in a sense. It is much harder to accept that I belong to research communities producing results that have (more or less indirectly) unethical applications. Should I stop research for this reason? A part of me says no: that I can try, at my local level, to steer research towards sobriety. On a personal, research level, find ways, *e.g.*, to efficiently estimate otherwise expensive calculations, find ways to improve the guarantees we can have over neural networks' results, *etc.* or otherwise work as much as possible on theoretical aspects, hoping they will never be applicable to anything and stay in the realm of Science for Science's sake. On a more political level, find ways to encourage collective organizations of research that are more sustainable and respectful of our planet, and ourselves. However, another part of me is at times afraid that this somewhat optimistic vision is nothing but an illusion and that I should follow Grothendieck's path to a much stricter sobriety, letting go of scientific research altogether. It is a relief for me to know that my heated inner debates on these subjects are taken seriously by many other researchers today: it is partly our responsibility to identify, estimate and communicate the risks of the applications stemming from our fundamental research.

Appendix A

Curriculum Vitae

I detail here my extended CV, presenting my academic career, the list of my past and current PhD students, the funded projects to which I participated, *etc.* I also present my responsibilities in the organization of international conferences or workshops, or in the administration of research. I finish with my full list of publications. To be precise, this appendix is organized as follows:

Contents

A.1 Academic positions and education

since 2016 chargé de Recherche CNRS, section 07, Gipsa-lab (Grenoble)

2015–2016 post-doc, INRIA (Rennes) and EPFL (Lausanne, Switzerland)

2014–2015 post-doc, ENS Lyon

2011–2014 PhD in signal processing, ENS Lyon

2006–2009 BsC and MsC in theoretical physics (minor in modelization of complex systems), ENS Lyon

A.2 Prizes and awards

I received the 2015 best PhD award from three French scientific societies (awarded jointly by GRETSI, GdR ISIS and club EEA): gretsi.fr/prix-de-these2015/resultats.php

A.3 Invited presentations

To access all my presentations, please visit my dedicated webpage: https://ntremblay.cnrs.fr/index.php?page=talks.

A.3.1 Invited speaker to workshops and conferences

2022 I was invited to talk at the **journées MAS 2022** (bi-annual French-speaking conference on statistics and probability) in Rouen, France

2022 I was invited to talk at the **DPP-fermions workshop** held at ENS Lyon, France

2018 I was invited to talk at the **journées MAS 2018** (bi-annual French-speaking conference on statistics and probability) in Dijon, France

2017 I was invited to talk at the **journée Horizon Maths 2017** entitled *Mathematics and networks* and organized by the IHP (Institut Henri Poincaré), Paris, France

2017 I was invited to talk at the **EURANDOM workshop on community detection and network reconstruction** in Eindhoven, Netherlands

2016 I was invited to talk at a **one-day conference on signal processing over networks**, Paris, France

2016 I was invited to talk at the **Data-driven approach to networks and language workshop** in Lyon, France

A.3.2 Ordinary research seminars

I gave around thirty invited ordinary research seminars since 2013 in diverse laboratories on my subjects of research, *e.g.,* at EPFL (Switzerland, Switzerland), at The University of Tokyo (Japan), at UiT (Tromsø, Norway), at Leiden University (Netherlands), at UCLouvain (Louvain-la-Neuve, Belgium), as well as at many laboratories in France: IMT (Toulouse), CRISTAL (Lille), LIP6 (Paris), I2M (Marseille), LJK (Grenoble), etc.

A.4 Institutional responsibilities

2021-2025 Elected member of the board of the CNRS' 7th scientific section (as a signal processing and statistical machine learning specialist). As such, I contribute to the evaluation of research on those topics at a national level in France and sit in the jury of the CNRS' yearly very competitive entrance exam. This responsibility, alone, takes 20% of my annual work time.

2020-2025 Elected member of the board of my laboratory (Gipsa-lab, Grenoble)

since 2017 Organizer of my department's scientific seminar

A.5 Funding received

A.5.1 Projects as PI (Principal Investigator)

- **European calls** (total: $\simeq$ 230 k€)

2025-2027 *Rand4TrustPool: Random sampling for trustworthy pooling layers in graph neural networks*, MSCA (Marie-Curie), $\simeq$ 230 k€ for a two-year mobility in Norway.

- **National calls** (total: $\simeq$ 210 k€)

2021-2024 *Determinantal point processes on graphs for randomized numerical linear algebra*, ANR JCJC, 199 k€ for a PhD funding (the PhD of Hugo Jaquard) + workshop organization

2019-2020 Extension of *Random walks for graph spectrum estimation*, CNRS PEPS, 3.5 k€.

2018-2019 *Random walks for graph spectrum estimation*, CNRS PEPS, 6 k€.

- **Local calls** (total: $\simeq$ 140 k€)

2021-2024 *Graph Neural Networks on Large Random Gaphs*, Labex Persyval, 115 k€ for a PhD funding (the PhD of Matthieu Cordonnier). I am co-PI.

2020-2021 *Random Forests for Graph Spectral Estimation*, Labex Persyval, 10 k€.

2018-2019 *Procedural memory: neuronal substrate, pathophysiology, modelisation and inference of involved networks*, Grenoble Alpes Data Institute, 6 k€.

2017-2018 *Determinantal processes for graph sampling*, Labex Persyval, 9 k€.

A.5.2 Projects as participant

- **National calls**

2018-2022 *Variational methods for graph signals*, ANR JCJC. PI: Samuel Vaiter (Nice, France). Contribution: I am leader of WP2.

- **Local calls**

since 2019 *"LargeData" Chair at the MIAI institute.* PI: Romain Couillet (Grenoble, France). Contribution: I am "core member" of the chair.

A.6 Organization of international conferences

2023 Member of the organizing committee of the week-long **GRETSI** conference (Grenoble, France). This is the bi-annual French-speaking signal processing conference ($\sim$ 500 participants).

2022 Co-organizer of the week-long international workshop **Machine Learning and Signal Processing on Graphs**. CIRM (Marseille, France). ($\sim$ 70 participants)

2020 Co-organizer of the 3-day international workshop **DPPs in the Alps** in Grenoble (France). Cancelled in the last minute (it was supposed to be in April) due to COVID. ($\sim$ 40 participants)

2019 Co-organizer of a conference day entitled **Signal processing over graphs, with a focus on neuroscience data**, in Paris (France). ($\sim$ 60 participants)

2019 Member of the scientific committee of the 3-day workshop **Graph signals : learning and optimization perspectives** organized in Montpellier (France). ($\sim$ 70 participants)

2018 Member of the organizing committee of the 3-day **Graph signal processing workshop** (EPFL, Lausanne, Switzerland). ($\sim$ 200 participants)

2017 Co-organizer of a **special session on graph signal processing** at ASILOMAR conference (California)

A.7 Supervising and mentoring

PhD students.

2021–2024 Director (thanks to a special authorization of the doctoral school) of Matthieu Cordonnier's PhD. Co-advisors: N. Keriven (IRISA, Rennes) and S. Vaiter (Dieudonné, Nice). This PhD studies GNN properties on large random graphs, and is funded on a grant of which I am co-PI. Participation to supervision: 33%.

2021–2024 Co-supervisor of Hugo Jaquard's PhD, with S. Barthelmé and P.-O. Amblard (both at Gipsa-lab, Grenoble), on KFs applied to graph synchronization. This PhD is funded on a grant of which I am PI. Participation to supervision: 40%.

2019–2022 Co-supervisor of Yusuf Yigit Pilavci's PhD, with S. Barthelmé and P.-O. Amblard, on Wilson's algorithm and its applications to numerical linear algebra. Participation to supervision: 40%. Yigit is now a post-doc in Lille, France.

2018–2021 Co-supervisor of Lorenzo Dall'Amico's PhD, with R. Couillet (LJK, Grenoble), on the Bethe-Hessian and its application to optimized spectral clustering in sparse graphs. Participation to supervision: 50%. Lorenzo is now a post-doc in Turin, Italy.

I participate or have participated in the supervision of two other PhDs (of which I am not officially in the supervising team):

2022–2025 the thesis of Lucas Chatelain, who is supervised by A. Gourrier and D. Rousseau, and who conducts his research in LiPhy (Grenoble). This is a biophysics thesis on the network of porosities in the dentin.

2017–2020 the thesis of Nagham Badreddine,who was supervised by E. Fino and S. Achard, and who conducted her research at the GIN (Grenoble Institute of Neuroscience). She studied the effect of learning simple tasks in the areas of the brain network responsible for procedural memory. She is now a post-doc in Marseille, France.

Other students. Since 2016, I cosupervised 5 M2 internships for their Master's theses (J. Vertaure at 50%, H. Ghanem at 50%, Y. Pilavci at 45%, H. Jaquard at 45 % and L. Chatelain at 20%), 1 M1 internship (S. el Bouch at 100%), and 1 L3 internship (V. Srivastava at 100%).

A.8 Teaching activities

My CNRS position is a 100% research position. I nevertheless have given more than 300h of lectures on diverse subjects since the start of my PhD in 2011.

2018-2020 I gave, with Sophie Achard, a lecture entitled *Graph theory and social networks* to Master's students, 30h per year. At Grenoble University, France.

2020 Co-organizer of an online *reading group on graph multi-scale analysis*, for PhD students from Grenoble as well as other places in France (to help provide scientific animation during COVID). I presented a mini introductory 4h lecture to open this reading group.

2019 I gave, with Simon Barthelmé, a 3h tutorial on DPPs at EUSIPCO (Spain).

2018 I gave, with Michaël Defferrard, a 6h practical course at the summer school GraphSIP.

2011-2014 During my PhD, I gave a total of 192 hours of lectures on various subjects ranging from statistical physics, to mathematics for physics or scientific English. At ENS Lyon, France.

A.9 Public outreach

2022 Seminar introducing the job of researcher, for Bachelor level students (*Classes Préparatoires*). Paris, France.

2016 Member of the organisation committee of the *Journée Sciences et Musique*: a day celebrating the links between Science and Music, for the general public (more than 500 visitors). Rennes, France.

2012-2014 Participation to the *Fête de la Science* (French annual Science event for the general public). Lyon, France.

2013 3 months of weekly interventions (2h/week) in a primary school on several scientific subjects. Via the organisation *la main à la pâte*. Lyon, France.

A.10 Reviewing and Edition

Reviewing. I have a regular reviewing activity in signal processing journals such as TSP, TSIPN, TIP, ACHA, OJSP, IMAJNA, SIIMS, TKDE, SP, SPL, JSPS, and also Proba/Stats/Machine Learning journals such as JMLR, JASA, VMSTA, Applied Probability, Bioinformatics, Constructive Approximation or PLOS One. I also have a regular reviewing activity for signal processing conferences such as ICASSP, GLOBALSIP, CAMSAP, EUSIPCO, GRETSI or other conferences from other fields such as ISIT, ESA, Neurips ou ICLR.

Edition. I was co-editor, with my colleague Andreas Loukas, of a "special issue" of the journal **Algorithms** entitled *Efficient Graph Algorithms in Machine Learning*. This was before I was made aware of the predatory practices of MDPI...

Participation to jurys of PhD theses. I was examinator of two PhD theses: M. Alexis GALLAND, ENS Paris (2020) and Mme. Yiye JIANG, IMB Bordeaux (2022).

A.11 Publication list

A.11.1 Some information on the ordering of the list of authors

- publications with a green code, such as [J14]: these are publications from my students (are concerned PhD students: Cordonnier, Dall'Amico, Jaquard, Pilavci and Badreddine; and a MsC student: Ghanem). In these cases, the student is always first in the list and the advisors are in alphabetical order. The participation to papers is proportional to the supervision participation detailed in Section A.7. [J17] and [NC6] are exceptions to this rule: I only participated to the data processing steps of these papers.
- publications with a red code, such as [J18]: there is no student in the list and the ordering of authors reflects the effective participation to the paper.

A.11.2 International journals with review committee

[J20] Simon Barthelmé, Pierre-Olivier Amblard, Nicolas Tremblay, Konstantin Usevich. *Gaussian Process Regression in the Flat Limit.* In **Annals of Statistics**, 2023. [PDF]

[J19] Nicolas Tremblay, Simon Barthelmé, Konstantin Usevich, Pierre-Olivier Amblard. *Extended L-ensembles: a new representation for Determinantal Point Processes.* In **Annals of Applied Probability**, 2023. [PDF]

[J18] Simon Barthelmé, Nicolas Tremblay, Konstantin Usevich, Pierre-Olivier Amblard. *Determinantal Point Processes in the Flat Limit.* In **Bernoulli**, 2023. [PDF]

[J17] Nagham Badreddine, Gisela Zalcman, Florence Appaix, Guillaume Becq, Nicolas Tremblay, Frédéric Saudou, Sophie Achard and Elodie Fino. *Spatiotemporal reorganization of corticostriatal network dynamics encodes motor skill learning.* In **Cell Reports**, 2022. [PDF]

[J16] Lorenzo Dall'Amico, Romain Couillet, Nicolas Tremblay. *Nishimori meets Bethe: a spectral method for node classification in sparse weighted graphs.* In **Journal of Statistical Mechanics: Theory and Experiment**, 2021. [PDF] [Code]

[J15] Lorenzo Dall'Amico, Romain Couillet, Nicolas Tremblay. *A unified framework for spectral clustering in sparse graphs.* In **Journal in Machine Learning Research**, 2021. [PDF] [Code]

[J14] Yusuf Y. Pilavci, Pierre-Olivier Amblard, Simon Barthelmé, Nicolas Tremblay. *Graph Tikhonov regularization and interpolation via random spanning forests.* In **Transactions on Signal and Information Processing over Networks**, 2021. [PDF] [Code]

[J13] Nicolas Tremblay, Simon Barthelmé, Pierre-Olivier Amblard. *Determinantal Point Processes for Coresets.* In **Journal of Machine Learning Research**, 2019. [PDF] [Code]

[J12] Benjamin Ricaud, Pierre Borgnat, Nicolas Tremblay, Paulo Goncalves, Pierre Vandergheynst. *Fourier could be a data scientist: From graph Fourier transform to signal processing on graphs.* In **CRAS (Compte-Rendus de l'Académie des Sciences)**, 2019. [PDF]

[J11] Simon Barthelmé, Pierre-Olivier Amblard, Nicolas Tremblay. *Asymptotic Equivalence of Fixed-size and Varying-size Determinantal Point Processes.* In **Bernoulli**, 2019. [PDF] [Code]

[J10] Nicolas Tremblay. *Independent reanalysis of alleged mind-matter interaction in double-slit experimental data.* In **PLOS One**, 2019. [PDF] [Code]

[J9] Luc Le Magoarou, Rémi Gribonval, Nicolas Tremblay. *Approximate Fast Graph Fourier Transforms via multi-layer sparse approximations.* In **Transactions on Signal and Information Processing over Networks**, 2018. [PDF] [Code]

[J8] Gilles Puy, Nicolas Tremblay, Rémi Gribonval, Pierre Vandergheynst. *Random sampling of bandlimited signals on graphs.* In **Applied and Computational Harmonic Analysis**, 2018. [PDF] [Code]

[J7] Rasha Boulos, Nicolas Tremblay, Alain Arneodo, Pierre Borgnat, Benjamin Audit. *Multi-scale structural community organisation of the human genome.* In **Bioinformatics**, 2017. [PDF]

[J6] Nicolas Tremblay, Pierre Borgnat. *Subgraph-based Filterbanks for Graph Signals.* In **Transactions on Signal Processing**, 2016. [PDF] [Code]

[J5] Patrice Abry, Stéphane G. Roux, Herwig Wendt, Paul Messier, Andrew G. Klein, Nicolas Tremblay, Pierre Borgnat, Stéphane Jaffard, Béatrice Vedel, Jim Coddington, and Lee Ann Daffner. *Multiscale Anisotropic Texture Analysis and Classification of Photographic Prints.* In **Signal Processing Magazine**, 2015. [PDF]

[J4] Nicolas Tremblay, Pierre Borgnat. *Graph Wavelets for Multiscale Community Mining.* In **Transactions on Signal Processing**, 2014. [PDF] [Code]

[J3] Nicolas Tremblay, Alain Barrat, Cary Forest, Mark Nornberg, Jean-François Pinton and Pierre Borgnat. *Bootstrapping under constraint for the assessment of group behavior in human contact networks.* In **Phys. Rev. E**, 2013. [PDF]

[J2] Kévin Tse-Ve-Koon, Nicolas Tremblay, Doru Constantin and Eric Freyssingeas. *Structure, thermodynamics and dynamics of the isotropic phase of spherical non-ionic surfactant micelles.* In **Journal of Colloid and Interface Science**, 2013. [PDF]

[J1] Nicolas Tremblay, Eric Larose and Vincent Rossetto. *Probing slow dynamics of consolidated granular multicomposite materials by Diffuse Acoustic Wave Spectroscopy (DAWS).* In **Journal of the Acoustical Society of America**, 2010. [PDF]

A.11.3 Book chapters

[CH3] Nicolas Tremblay, Andreas Loukas. *Approximating Spectral Clustering via Sampling: a Review.* In book **Sampling Techniques for Supervised or Unsupervised Tasks**, 2020. [PDF]

[CH2] Nicolas Tremblay, Paulo Gonçalves, Pierre Borgnat. *Design of graph filters and filterbanks.* In book **Cooperative and Graph Signal Processing**, 2018. [PDF]

[CH1] Pierre Borgnat, Céline Robardet, Patrice Abry, Patrick Flandrin, Jean-Baptiste Rouquier, Nicolas Tremblay. *A Dynamical Network View of Lyon's Vélo'v Shared Bicycle System.* In book **Dynamics On and Of Complex Networks**, 2013. [PDF]

A.11.4 International conferences with review committee

[IC24] Simon Barthelmé, Nicolas Tremblay, Pierre-Olivier Amblard. *A Faster Sampler for Discrete Determinantal Point Processes.* In **AISTATS**, 2023. [PDF] [Code]

[IC23] Hugo Jaquard, Michaël Fanuel, Pierre-Olivier Amblard, Rémi Bardenet, Simon Barthelmé, Nicolas Tremblay. *Smoothing complex-valued signals on Graphs with Monte-Carlo.* In **ICASSP**, 2023. [PDF] [Code]

[IC22] Yusuf Y. Pilavci, Pierre-Olivier Amblard, Simon Barthelmé, Nicolas Tremblay. *Variance Reduction in Stochastic Methods for Large-scale Regularised Least-square Problems.* In **EUSIPCO**, 2022. [PDF] [Code]

[IC21] Hashem Ghanem, Nicolas Keriven, Nicolas Tremblay. *Fast graph kernel with optical random features.* In **ICASSP**, 2021. [PDF] [Code]

[IC20] Lorenzo Dall'Amico, Romain Couillet, Nicolas Tremblay. *Community detection in sparse time-evolving graphs with a dynamical Bethe-Hessian.* In **Neurips**, 2020. [PDF] [Code]

[IC19] Yusuf Y. Pilavci, Pierre-Olivier Amblard, Simon Barthelmé, Nicolas Tremblay. *Smoothing graph signals via random spanning forests.* In **ICASSP**, 2020. [PDF] [Code]

[IC18] Lorenzo Dall'Amico, Romain Couillet, Nicolas Tremblay. *Optimal Laplacian Regularization for Sparse Spectral Community Detection.* In **ICASSP**, 2020. [PDF]

[IC17] Lorenzo Dall'Amico, Romain Couillet, Nicolas Tremblay. *Revisiting the Bethe-Hessian: Improved Community Detection in Sparse Heterogeneous Graphs.* In **Neurips**, 2019. [PDF] [Code]

[IC16] Pierre-Olivier Amblard, Simon Barthelmé, Nicolas Tremblay. *Subsampling with k determinantal point processes for estimating statistics in large data sets.* In **SSP**, 2018.

[IC15] Luc Le Magoarou, Nicolas Tremblay, Rémi Gribonval. *Analyzing the Approximation Error of the Fast Graph Fourier Transform.* In **ASILOMAR**, 2017. [PDF] [Code]

[IC14] Nicolas Tremblay, Pierre-Olivier Amblard, Simon Barthelmé. *Graph Sampling with Determinantal Processes.* In **EUSIPCO**, 2017. [PDF]

[IC13] Nicolas Keriven, Nicolas Tremblay, Yann Traonmilin, Rémi Gribonval. *Compressive K-means.* In **ICASSP**, 2017. [PDF] [Code]

[IC12] Nicolas Tremblay, Gilles Puy, Rémi Gribonval, Pierre Vandergheynst. *Compressive Spectral Clustering.* In **ICML**, 2016. [PDF] [Code]

[IC11] Nicolas Tremblay, Gilles Puy, Pierre Borgnat, Rémi Gribonval, Pierre Vandergheynst. *Accelerated Spectral Clustering Using Graph Filtering of Random Signals.* In **ICASSP**, 2016. [PDF]

[IC10] Nicolas Tremblay, Pierre Borgnat. *Joint Filtering of Graph and Graph-Signals.* In **ASILOMAR**, 2015. [PDF]

[IC9] Pierre Borgnat, Paulo Gonçalves, Nicolas Tremblay, Nathanaël Willaime-Angonin. *Community Mining with Graph Filters for Correlation Matrices.* In **ASILOMAR**, 2015. [PDF]

[IC8] Stéphane Roux, Nicolas Tremblay, Pierre Borgnat, Patrice Abry, Herwig Wendt, Paul Messier. *Multiscale Anisotropic Texture Unsupervised Clustering for Photographic Paper.* In **WIFS**, 2015. [PDF]

[IC7] Nicolas Tremblay, Pierre Borgnat and Patrick Flandrin. *Graph Empirical Mode Decomposition.* In **EUSIPCO**, 2014. [PDF]

[IC6] Nicolas Tremblay, Pierre Borgnat. *Multiscale Community Mining in Networks Using the Graph Wavelet Transform of Random Vectors.* In **GlobalSIP**, 2013. [PDF]

[IC5] Nicolas Tremblay, Pierre Borgnat. *Multiscale Detection of Stable Communities Using Wavelets on Networks.* In **European Conference on Complex Systems**, 2013. [PDF]

[IC4] Nicolas Tremblay, Pierre Borgnat. *Multiscale Community Mining in Networks Using Spectral Graph Wavelets.* In **EUSIPCO**, 2013. [PDF]

[IC3] Romain Fontugne, Nicolas Tremblay, Pierre Borgnat, Patrick Flandrin et Hiroshi Esaki. *Mining anomalous electricity consumption using ensemble empirical mode decomposition.* In **ICASSP**, 2013. [PDF]

[IC2] Romain Fontugne, Jorge Ortiz, Nicolas Tremblay, Pierre Borgnat, Patrick Flandrin, Kensuke Fukuda, David Culler and Hiroshi Esaki. *Strip, Bind, and Search: A Method for Identifying Abnormal Energy Consumption in Buildings.* In **IPSN**, 2013. [PDF]

[IC1] Nicolas Tremblay, Alain Barrat, Cary Forest, Mark Nornberg, Jean-François Pinton et Pierre Borgnat. *European Conference on Complex Systems.* In **European Conference on Complex Systems**, 2012. [PDF]

A.11.5 National conferences with review committee

[NC10] Matthieu Cordonnier, Nicolas Keriven, Nicolas Tremblay, Samuel Vaiter. *Convergence of Graph Neural Networks with generic aggregation functions on random graphs.* In **GRETSI**, 2023. [PDF]

[NC9] Yusuf Y. Pilavci, Pierre-Olivier Amblard, Simon Barthelmé, Nicolas Tremblay. *Variance Reduction for Inverse Trace Estimation via Random Spanning Forests.* In **GRETSI**, 2022. [PDF]

[NC8] Pierre-Olivier Amblard, Simon Barthelmé, Nicolas Tremblay, Konstantin Usevich. *Mesures d'indépendance dans des rkHs en limite plate.* In **GRETSI**, 2022. [PDF]

[NC7] Simon Barthelmé, Nicolas Tremblay, Alexandre Gaudillière, Luca Avena, Pierre-Olivier Amblard. *Estimating the inverse trace using random forests on graphs.* In **GRETSI**, 2019. [PDF]

[NC6] Guillaume Becq, Nagham Badreddine, Nicolas Tremblay, Florence Appaix, Gisela Zalcman, Elodie Fino, Sophie Achard. *Classification de types de neurones à partir de signaux calciques.* In **GRETSI**, 2019. [PDF]

[NC5] Lorenzo Dall'Amico, Romain Couillet, Nicolas Tremblay. *Classification spectrale par la laplacienne déformée dans des graphes réalistes.* In **GRETSI**, 2019. [PDF]

[NC4] Nicolas Tremblay, Simon Barthelmé, Pierre-Olivier Amblard. *Échantillonnage de signaux sur graphes via des processus déterminantaux.* In **GRETSI**, 2017. [PDF]

[NC3] Nicolas Tremblay, Stéphane Roux, Pierre Borgnat, Patrice Abry, Herwig Wendt, Paul Messier. *Texture classification of photographic papers: improving spectral clustering using filterbanks on graphs.* In **GRETSI**, 2015. [PDF]

[NC2] Rasha Boulos, Nicolas Tremblay, Alain Arnéodo, Pierre Borgnat, Benjamin Audit. *Applications des ondelettes sur graphe en génomique.* In **GRETSI**, 2015. [PDF]

[NC1] Nicolas Tremblay, Pierre Borgnat. *Partitionnement multi-échelle d'un graphe en communautés : détection des échelles pertinentes.* In **GRETSI**, 2013. [PDF]

A.11.6 Codes

Many of my publications are associated to code libraries, that I put online for the community, both on my website and on specialized websites. To access these codes, the simplest is to click on the "[Code]" links when they appear in the above list of publications, or visit my dedicated webpage on my website.

These codes enable in general to reproduce the figures presented in the papers and are written in Matlab (2011-2017), Python (2017-2021), or in Julia (2020-now).

Bibliography

[Abramowitz & Stegun 1964] Milton Abramowitz and Irene A Stegun. Handbook of mathematical functions with formulas, graphs, and mathematical tables, volume 55. US Government printing office, 1964. (Cited on page 51.)

[Ahmed *et al.* 2020] Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Keaton Hamm, Mohammad Javad Latifi Jebelli, Stephen Kobourov and Richard Spence. *Graph spanners: A tutorial review.* Computer Science Review, vol. 37, page 100253, 2020. Publisher: Elsevier. (Cited on page 6.)

[Alev & Lau 2020] Vedat Levi Alev and Lap Chi Lau. *Improved Analysis of Higher Order Random Walks and Applications.* arXiv:2001.02827, February 2020. (Cited on page 106.)

[Ali *et al.* 2020] Patrick Ali, Fouzul Atik and Ravindra B Bapat. *Identities for minors of the Laplacian, resistance and distance matrices of graphs with arbitrary weights.* Linear and Multilinear Algebra, vol. 68, no. 2, pages 323–336, 2020. Publisher: Taylor & Francis. (Cited on page 104.)

[Amini *et al.* 2013] Arash A. Amini, Aiyou Chen, Peter J. Bickel and Elizaveta Levina. *Pseudo-likelihood methods for community detection in large sparse networks.* The Annals of Statistics, vol. 41, no. 4, pages 2097–2122, 2013. (Cited on page 20.)

[Anari *et al.* 2020] Nima Anari, Kuikui Liu, Shayan Oveis Gharan and Cynthia Vinzant. *Log-Concave Polynomials IV: Exchange Properties, Tight Mixing Times, and Faster Sampling of Spanning Trees.* arXiv:2004.07220, April 2020. (Cited on page 106.)

[Anis *et al.* 2016] A. Anis, A. Gadde and A. Ortega. *Efficient Sampling Set Selection for Bandlimited Graph Signals Using Graph Spectral Proxies.* IEEE Transactions on Signal Processing, vol. 64, no. 14, pages 3775–3789, July 2016. (Cited on pages 29 and 81.)

[Avena & Gaudillière 2013] Luca Avena and Alexandre Gaudillière. *Random spanning forests, Markov matrix spectra and well distributed points.* arXiv:1310.1723, October 2013. (Cited on page 78.)

[Avena & Gaudillière 2017] L. Avena and A. Gaudillière. *Two Applications of Random Spanning Forests.* Journal of Theoretical Probability, July 2017. (Cited on pages 42, 78, 79 and 105.)

[Avena *et al.* 2017] Luca Avena, Fabienne Castell, Alexandre Gaudillière and Clothilde Melot. *Approximate and exact solutions of intertwining equations through random spanning forests.* arXiv:1702.05992 [math], February 2017. arXiv: 1702.05992. (Cited on page 57.)

[Avron & Toledo 2011] Haim Avron and Sivan Toledo. *Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix.* Journal of the ACM, vol. 58, no. 2, pages 1–34, April 2011. (Cited on page 89.)

[Bachem *et al.* 2017] Olivier Bachem, Mario Lucic and Andreas Krause. *Practical Coreset Constructions for Machine Learning.* arXiv:1703.06476, March 2017. (Cited on pages 61, 64 and 66.)

[Badreddine *et al.* 2022] Nagham Badreddine, Gisela Zalcman, Florence Appaix, Guillaume Becq, Nicolas Tremblay, Frédéric Saudou, Sophie Achard and Elodie Fino. *Spatiotemporal reorganization of corticostriatal networks encodes motor skill learning.* Cell reports, vol. 39, no. 1, 2022. (Cited on page 2.)

[Bai *et al.* 2000] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe and Henk van der Vorst. Templates for the solution of algebraic eigenvalue problems: a practical guide. SIAM, 2000. (Cited on page 17.)

[Balcan *et al.* 2013] Maria-Florina F Balcan, Steven Ehrlich and Yingyu Liang. *Distributed k-means and k-median Clustering on General Topologies.* In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 26, pages 1995–2003. Curran Associates, Inc., 2013. (Cited on page 64.)

[Baraud 2010] Yannick Baraud. *A Bernstein-type inequality for suprema of random processes with applications to model selection in non-Gaussian regression.* Bernoulli, vol. 16, no. 4, pages 1064–1085, 2010. (Cited on page 67.)

[Bardenet & Hardy 2020] Rémi Bardenet and Adrien Hardy. *Monte Carlo with determinantal point processes.* The Annals of Applied Probability, vol. 30, no. 1, pages 368 – 417, 2020. Publisher: Institute of Mathematical Statistics. (Cited on pages 43 and 54.)

[Bardenet 2022] Rémi Bardenet. *Turning repulsive point processes into subsampling algorithms.* Habilitation à diriger des recherches, Université de Lille, December 2022. (Cited on page 43.)

[Barthelmé *et al.* 2019] Simon Barthelmé, Pierre-Olivier Amblard and Nicolas Tremblay. *Asymptotic Equivalence of Fixed-size and Varying-size Determinantal Point Processes.* Bernoulli, vol. 25, no. 4B, pages 3555–3589, 2019. (Cited on pages 46, 47 and 48.)

[Barthelme *et al.* 2023a] Simon Barthelme, Nicolas Tremblay and Pierre-Olivier Amblard. *A Faster Sampler for Discrete Determinantal Point Processes.* In International Conference on Artificial Intelligence and Statistics, pages 5582–5592. PMLR, 2023. (Cited on pages 49 and 53.)

[Barthelmé *et al.* 2023b] Simon Barthelmé, Nicolas Tremblay, Konstantin Usevich and Pierre-Olivier Amblard. *Determinantal point processes in the flat limit.* Bernoulli, vol. 29, no. 2, pages 957–983, 2023. (Cited on page 60.)

[Barthelmé *et al.* 2019] Simon Barthelmé, Nicolas Tremblay, Alexandre Gaudillière, Luca Avena and Pierre-Olivier Amblard. *Estimating the inverse trace using random forests on graphs.* In Proc. GRETSI Symposium Signal and Image Processing, Lille, France, 2019. (Cited on page 85.)

[Bassett & Sporns 2017] Danielle S Bassett and Olaf Sporns. *Network neuroscience.* Nature neuroscience, vol. 20, no. 3, pages 353–364, 2017. (Cited on page 1.)

[Becq *et al.* 2019] Guillaume Becq, Nagham Badreddine, Nicolas Tremblay, Florence Appaix, Gisela Zalcman, Elodie Fino and Sophie Achard. *Classification de types de neurones à partir de signaux calciques.* In 27ème Colloque sur le traitement du signal et des images, pages p. 749–752, Lille, August 2019. GRETSI - Groupe de Recherche en Traitement du Signal et des Images. (Cited on page 2.)

[Belhadji *et al.* 2020] Ayoub Belhadji, Rémi Bardenet and Pierre Chainais. *A determinantal point process for column subset selection.* Journal of Machine Learning Research, vol. 21, no. 197, pages 1–62, 2020. (Cited on page 43.)

[Benczúr & Karger 1996] András A Benczúr and David R Karger. *Approximating st minimum cuts in $O(n^2)$ time.* In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pages 47–55, 1996. (Cited on page 6.)

[Blackwell 1947] David Blackwell. *Conditional Expectation and Unbiased Sequential Estimation.* Ann. Math. Stat., vol. 18, no. 1, pages 105–110, mar 1947. (Cited on page 92.)

[Blondel *et al.* 2008] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte and Etienne Lefebvre. *Fast unfolding of communities in large networks.* Journal of Statistical Mechanics: Theory and Experiment, vol. 2008, no. 10, page P10008, 2008. (Cited on page 24.)

[Borodin & Rains 2005] Alexei Borodin and Eric M Rains. *Eynard–Mehta theorem, Schur process, and their Pfaffian analogs.* Journal of statistical physics, vol. 121, no. 3-4, pages 291–317, 2005. (Cited on page 44.)

[Boumal 2016] Nicolas Boumal. *Nonconvex phase synchronization.* SIAM Journal on Optimization, vol. 26, no. 4, pages 2355–2377, 2016. (Cited on page 96.)

[Boutsidis *et al.* 2015] Christos Boutsidis, Prabhanjan Kambadur and Alex Gittens. *Spectral Clustering via the Power Method-Provably.* In Proceedings of the 32nd International Conference on Machine Learning (ICML-15), pages 40–48, 2015. (Cited on pages 31 and 32.)

[Braverman *et al.* 2016] Vladimir Braverman, Dan Feldman, Harry Lang, Adiel Statman and Samson Zhou. *New frameworks for offline and streaming coreset constructions*. arXiv preprint arXiv:1612.00889, 2016. (Cited on page 61.)

[Bronstein *et al.* 2017] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam and P. Vandergheynst. *Geometric Deep Learning: Going beyond Euclidean data*. IEEE Signal Processing Magazine, vol. 34, no. 4, pages 18–42, July 2017. (Cited on page 6.)

[Burton & Pemantle 1993] Robert Burton and Robin Pemantle. *Local characteristics, entropy and limit theorems for spanning trees and domino tilings via transfer-impedances*. The Annals of Probability, pages 1329–1371, 1993. Publisher: JSTOR. (Cited on page 42.)

[Calvetti *et al.* 1994] Daniela Calvetti, L Reichel and Danny Chris Sorensen. *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*. Electronic Transactions on Numerical Analysis, vol. 2, no. 1, page 21, 1994. (Cited on page 17.)

[Casazza & Kutyniok 2012] Peter G Casazza and Gitta Kutyniok. Finite frames: Theory and applications. Springer, 2012. (Cited on page 69.)

[Chamon & Ribeiro 2018] L. F. O. Chamon and A. Ribeiro. *Greedy Sampling of Graph Signals*. IEEE Transactions on Signal Processing, vol. 66, no. 1, pages 34–47, January 2018. (Cited on page 28.)

[Chan 1987] Tony F Chan. *Rank revealing QR factorizations*. Linear algebra and its applications, vol. 88, pages 67–82, 1987. (Cited on page 52.)

[Chatelain *et al.* 2023] Lucas Chatelain, Elsa Vennat, Nicolas Tremblay, David Rousseau and Aurélien Gourrier. *Graph modeling of cellular porosity in dentin*. In 12th International Conference on Complex Networks and their Applications, Menton, November 2023. (Cited on page 3.)

[Chen *et al.* 2008] Yanqing Chen, Timothy A Davis, William W Hager and Sivasankaran Rajamanickam. *Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate*. ACM Transactions on Mathematical Software (TOMS), vol. 35, no. 3, pages 1–14, 2008. (Cited on page 86.)

[Chen *et al.* 2015] Siheng Chen, Rohan Varma, Aliaksei Sandryhaila and Jelena Kovacevic. *Discrete Signal Processing on Graphs: Sampling Theory*. CoRR, vol. abs/1503.05432, 2015. (Cited on page 28.)

[Chen 2009] Ke Chen. *On Coresets for k -Median and k -Means Clustering in Metric and Euclidean Spaces and Their Applications*. SIAM Journal on Computing, vol. 39, no. 3, pages 923–947, January 2009. (Cited on page 61.)

[Chung 1997] F.R.K. Chung. Spectral graph theory. Number 92. Amer Mathematical Society, 1997. (Cited on pages 30, 58 and 102.)

[Coeurjolly *et al.* 2021] Jean-François Coeurjolly, Adrien Mazoyer and Pierre-Olivier Amblard. *Monte Carlo integration of non-differentiable functions on [0, 1] ι, ι= 1,. . . , d, using a single determinantal point pattern defined on [0, 1] d.* Electronic Journal of Statistics, vol. 15, no. 2, pages 6228–6280, 2021. (Cited on pages 43 and 54.)

[Coifman & Maggioni 2006] Ronald R. Coifman and Mauro Maggioni. *Diffusion wavelets.* Applied and Computational Harmonic Analysis, vol. 21, no. 1, pages 53–94, July 2006. (Cited on page 15.)

[Colosi *et al.* 2022] Elisabetta Colosi, Giulia Bassignana, Diego Andrés Contreras, Canelle Poirier, Pierre-Yves Boëlle, Simon Cauchemez, Yazdan Yazdanpanah, Bruno Lina, Arnaud Fontanet, Alain Barrat*et al. Screening and vaccination against COVID-19 to minimise school closure: a modelling study.* The Lancet Infectious Diseases, vol. 22, no. 7, pages 977–989, 2022. (Cited on page 4.)

[Copenhaver *et al.* 2014] Martin S. Copenhaver, Yeon Hyang Kim, Cortney Logan, Kyanne Mayfield, Sivaram K. Narayan, Matthew J. Petro and Jonathan Sheperd. *Diagram vectors and tight frame scaling in finite dimensions.* Operators and Matrices, no. 1, pages 73–88, 2014. (Cited on page 67.)

[Coste & Zhu 2021] Simon Coste and Yizhe Zhu. *Eigenvalues of the non-backtracking operator detached from the bulk.* Random Matrices: Theory and Applications, vol. 10, no. 03, page 2150028, 2021. (Cited on pages 37 and 39.)

[Crovella & Kolaczyk 2003] M. Crovella and E. Kolaczyk. *Graph wavelets for spatial traffic analysis.* In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies, volume 3, pages 1848–1857, 2003. (Cited on page 15.)

[Cybenko 1989] G. Cybenko. *Approximation by superpositions of a sigmoidal function.* Mathematics of Control, Signals, and Systems (MCSS), vol. 2, no. 4, pages 303–314, December 1989. (Cited on page 102.)

[Dai *et al.* 2022] Enyan Dai, Tianxiang Zhao, Huaisheng Zhu, Junjie Xu, Zhimeng Guo, Hui Liu, Jiliang Tang and Suhang Wang. *A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability.* arXiv preprint arXiv:2204.08570, 2022. (Cited on page 108.)

[Dall'Amico *et al.* 2019] Lorenzo Dall'Amico, Romain Couillet and Nicolas Tremblay. *Revisiting the Bethe-Hessian: Improved Community Detection in Sparse Heterogeneous Graphs.* In Advances in Neural Information Processing Systems (NeurIPS). 2019. (Cited on page 37.)

[Dall'Amico *et al.* 2021] Lorenzo Dall'Amico, Romain Couillet and Nicolas Tremblay. *A unified framework for spectral clustering in sparse graphs.* The Journal of Machine Learning Research, vol. 22, no. 1, pages 9859–9914, 2021. (Cited on pages 35 and 37.)

[Davis & Rabinowitz 2007] Philip J Davis and Philip Rabinowitz. Methods of numerical integration. Courier Corporation, 2007. (Cited on page 5.)

[De Vico Fallani *et al.* 2014] Fabrizio De Vico Fallani, Jonas Richiardi, Mario Chavez and Sophie Achard. *Graph analysis of functional brain networks: practical issues in translational neuroscience.* Philosophical Transactions of the Royal Society B: Biological Sciences, vol. 369, no. 1653, page 20130521, 2014. (Cited on page 1.)

[Decelle *et al.* 2011] Aurelien Decelle, Florent Krzakala, Cristopher Moore and Lenka Zdeborová. *Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications.* Phys. Rev. E, vol. 84, no. 6, page 066106, December 2011. (Cited on page 18.)

[Defferrard *et al.* 2016] Michaël Defferrard, Xavier Bresson and Pierre Vandergheynst. *Convolutional neural networks on graphs with fast localized spectral filtering.* In Advances in Neural Information Processing Systems, pages 3837–3845, 2016. (Cited on pages 99 and 100.)

[Derezinski & Mahoney 2021] Micha\l Derezinski and Michael W Mahoney. *Determinantal point processes in randomized numerical linear algebra.* Notices of the American Mathematical Society, vol. 68, no. 1, 2021. (Cited on pages 43 and 103.)

[Derezinski & Warmuth 2017] Michal Derezinski and Manfred K. K Warmuth. *Unbiased estimates for linear regression via volume sampling.* In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. (Cited on page 81.)

[Derezinski *et al.* 2019] Michal Derezinski, Daniele Calandriello and Michal Valko. *Exact sampling of determinantal point processes with sublinear time preprocessing.* In Advances in Neural Information Processing Systems (NeurIPS), 2019. (Cited on pages 53 and 106.)

[DeVore & Temlyakov 1996] Ronald A DeVore and Vladimir N Temlyakov. *Some remarks on greedy algorithms.* Advances in computational Mathematics, vol. 5, pages 173–187, 1996. (Cited on page 8.)

[Dhillon *et al.* 2007] I.S. Dhillon, Yuqiang Guan and B. Kulis. *Weighted Graph Cuts without Eigenvectors A Multilevel Approach.* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 29, no. 11, pages 1944–1957, November 2007. (Cited on page 6.)

[di Lorenzo *et al.* 2018] Paolo di Lorenzo, Sergio Barbarossa and Paolo Banelli. *Sampling and recovery of graph signals.* In Cooperative and Graph Signal Processing, pages 261–282. Elsevier, 2018. (Cited on page 27.)

[Drineas & Mahoney 2016] Petros Drineas and Michael W. Mahoney. *RandNLA: randomized numerical linear algebra.* Communications of the ACM, vol. 59, no. 6, pages 80–90, May 2016. (Cited on pages 5 and 29.)

[Drineas & Mahoney 2018] Petros Drineas and Michael W Mahoney. *Lectures on randomized numerical linear algebra.* The Mathematics of Data, vol. 25, page 1, 2018. (Cited on page 63.)

[Drineas *et al.* 2012] Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney and David P Woodruff. *Fast approximation of matrix coherence and statistical leverage.* The Journal of Machine Learning Research, vol. 13, no. 1, pages 3475–3506, 2012. (Cited on page 63.)

[Ekambaram *et al.* 2013] V. N. Ekambaram, G. C. Fanti, B. Ayazifar and K. Ramchandran. *Circulant structures and graph signal processing.* In 2013 IEEE International Conference on Image Processing, pages 834–838, September 2013. (Cited on page 23.)

[Fan *et al.* 2020] Tiffany Fan, David I Shuman, Shashanka Ubaru and Yousef Saad. *Spectrum-Adapted Polynomial Approximation for Matrix Functions with Applications in Graph Signal Processing.* Algorithms, vol. 13, no. 11, page 295, 2020. Publisher: Multidisciplinary Digital Publishing Institute. (Cited on pages 84 and 105.)

[Feldman & Langberg 2011] Dan Feldman and Michael Langberg. *A unified framework for approximating and clustering data.* In Proceedings of the forty-third annual ACM symposium on Theory of computing, pages 569–578. ACM, 2011. (Cited on page 64.)

[Fiedler 1973] Miroslav Fiedler. *Algebraic connectivity of graphs.* Czechoslovak mathematical journal, vol. 23, no. 2, pages 298–305, 1973. (Cited on page 32.)

[Fortunato & Hric 2016] Santo Fortunato and Darko Hric. *Community detection in networks: A user guide.* Physics Reports, vol. 659, pages 1 – 44, 2016. (Cited on page 6.)

[Fortunato 2010] S. Fortunato. *Community detection in graphs.* Physics Reports, vol. 486, no. 3-5, pages 75–174, 2010. (Cited on page 6.)

[Gallotti & Barthelemy 2015] Riccardo Gallotti and Marc Barthelemy. *The multilayer temporal network of public transport in Great Britain.* Scientific data, vol. 2, no. 1, pages 1–8, 2015. (Cited on page 3.)

[Gautier 2020a] Guillaume Gautier. *On sampling determinantal point processes.* Ph.d. thesis, Ecole Centrale de Lille, 2020. (Cited on page 54.)

[Gautier 2020b] Guillaume Gautier. *On sampling determinantal point processes.* Ph.D. thesis, Ecole Centrale de Lille, 2020. (Cited on page 106.)

[Gavish *et al.* 2010] Matan Gavish, Boaz Nadler and Ronald R Coifman. *Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning.* In International Conference on Machine Learning (ICML), 2010. (Cited on pages 6 and 15.)

[Gillenwater *et al.* 2019] Jennifer Gillenwater, Alex Kulesza, Zelda Mariet and Sergei Vassilvtiskii. *A Tree-Based Method for Fast Repeated Sampling of Determinantal Point Processes.* In International Conference on Machine Learning (ICML), 2019. (Cited on page 106.)

[Gilmer *et al.* 2017] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals and George E. Dahl. *Neural Message Passing for Quantum Chemistry.* arXiv:1704.01212 [cs], June 2017. arXiv: 1704.01212. (Cited on page 99.)

[Girard 1987] Didier Girard. *Un algorithme simple et rapide pour la validation croisée généralisée sur des problèmes de grande taille.* Technical report, 1987. (Cited on page 85.)

[Gremban 1996] Keith D Gremban. *Combinatorial preconditioners for sparse, symmetric, diagonally dominant linear systems.* PhD Thesis, Carnegie Mellon University, 1996. (Cited on page 88.)

[Gulikers *et al.* 2016] Lennart Gulikers, Marc Lelarge and Laurent Massoulié. *Non-backtracking spectrum of degree-corrected stochastic block models.* arXiv preprint arXiv:1609.02487, 2016. (Cited on page 34.)

[Gulikers *et al.* 2017] Lennart Gulikers, Marc Lelarge and Laurent Massoulié. *A spectral method for community detection in moderately sparse degree-corrected stochastic block models.* Advances in Applied Probability, vol. 49, no. 3, pages 686–721, 2017. (Cited on page 37.)

[Gulikers *et al.* 2018] Lennart Gulikers, Marc Lelarge and Laurent Massoulié. *An impossibility result for reconstruction in the degree-corrected stochastic block model.* The Annals of Applied Probability, vol. 28, no. 5, pages 3002 – 3027, 2018. (Cited on page 34.)

[Hamilton *et al.* 2017] Will Hamilton, Zhitao Ying and Jure Leskovec. *Inductive Representation Learning on Large Graphs.* In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. (Cited on page 100.)

[Hammond *et al.* 2011] D.K. Hammond, P. Vandergheynst and R. Gribonval. *Wavelets on graphs via spectral graph theory.* Applied and Computational Harmonic Analysis, vol. 30, no. 2, pages 129–150, 2011. (Cited on page 15.)

[Han *et al.* 2021] Insu Han, Mike Gartrell, Jennifer Gillenwater, Elvis Dohmatob*et al. Scalable Sampling for Nonsymmetric Determinantal Point Processes.* In International Conference on Learning Representations, 2021. (Cited on page 44.)

[Hastie *et al.* 2005] T. Hastie, R. Tibshirani, J. Friedman and J. Franklin. *The elements of statistical learning: data mining, inference and prediction.* The Mathematical Intelligencer, vol. 27, no. 2, pages 83–85, 2005. (Cited on page 85.)

[Hough *et al.* 2006] J. Ben Hough, Manjunath Krishnapur, Yuval Peres and Bálint Virág. *Determinantal Processes and Independence.* Probability Surveys, vol. 3, no. 0, pages 206–229, 2006. (Cited on pages 45 and 47.)

[Hunter *et al.* 2014] Timothy Hunter, Ahmed El Alaoui and Alexandre Bayen. *Computing the log-determinant of symmetric, diagonally dominant matrices in near-linear time.* arXiv preprint arXiv:1408.1693, 2014. (Cited on page 88.)

[Hutchinson 1989] Michael F Hutchinson. *A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines.* Communications in Statistics-Simulation and Computation, vol. 18, no. 3, pages 1059–1076, 1989. Publisher: Taylor & Francis. (Cited on page 85.)

[Isella *et al.* 2011] Lorenzo Isella, Juliette Stehlé, Alain Barrat, Ciro Cattuto, Jean-François Pinton and Wouter Van den Broeck. *What's in a crowd? Analysis of face-to-face behavioral networks.* Journal of theoretical biology, vol. 271, no. 1, pages 166–180, 2011. (Cited on page 4.)

[Jegelka 2022] Stefanie Jegelka. *Theory of Graph Neural Networks: Representation and Learning*, April 2022. arXiv:2204.07697 [cs, stat]. (Cited on page 99.)

[Joseph & Yu 2016] Antony Joseph and Bin Yu. *Impact of regularization on spectral clustering.* The Annals of Statistics, vol. 44, no. 4, pages 1765–1791, August 2016. (Cited on page 20.)

[Jubran *et al.* 2021] Ibrahim Jubran, Alaa Maalouf and Dan Feldman. *Overview of accurate coresets.* Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, vol. 11, no. 6, page e1429, 2021. (Cited on page 5.)

[Karger 1999] David R Karger. *Random sampling in cut, flow, and network design problems.* Mathematics of Operations Research, vol. 24, no. 2, pages 383–413, 1999. Publisher: INFORMS. (Cited on page 6.)

[Karrer & Newman 2011] Brian Karrer and Mark EJ Newman. *Stochastic blockmodels and community structure in networks.* Physical review E, vol. 83, no. 1, page 016107, 2011. (Cited on pages 18 and 33.)

[Kaufman & Oppenheim 2018] Tali Kaufman and Izhar Oppenheim. *High order random walks: Beyond spectral gap.* In Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2018. (Cited on page 106.)

[Kelner & Madry 2009] J. A. Kelner and A. Madry. *Faster Generation of Random Spanning Trees.* In 2009 50th Annual IEEE Symposium on Foundations of Computer Science, pages 13–21, October 2009. ISSN: 0272-5428. (Cited on page 106.)

[Kelner *et al.* 2013] Jonathan A Kelner, Lorenzo Orecchia, Aaron Sidford and Zeyuan Allen Zhu. *A simple, combinatorial algorithm for solving SDD systems in nearly-linear time.* In Proceedings of the forty-fifth annual ACM symposium on Theory of computing, pages 911–920. ACM, 2013. (Cited on page 88.)

[Keriven *et al.* 2020] Nicolas Keriven, Alberto Bietti and Samuel Vaiter. *Convergence and Stability of Graph Convolutional Networks on Large Random Graphs.* page 12, 2020. (Cited on page 98.)

[Kipf & Welling 2017] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks.* In Proceedings of the 5th International Conference on Learning Representations, ICLR '17, 2017. (Cited on pages 99 and 100.)

[Klein & Randić 1993] Douglas J Klein and Milan Randić. *Resistance distance.* Journal of mathematical chemistry, vol. 12, no. 1, pages 81–95, 1993. (Cited on page 60.)

[Kotzagiannidis & Dragotti 2016] Madeleine S. Kotzagiannidis and Pier Luigi Dragotti. *Splines and Wavelets on Circulant Graphs.* CoRR, vol. abs/1603.04917, 2016. (Cited on page 23.)

[Krzakala *et al.* 2013] Florent Krzakala, Cristopher Moore, Elchanan Mossel, Joe Neeman, Allan Sly, Lenka Zdeborová and Pan Zhang. *Spectral redemption in clustering sparse networks.* Proceedings of the National Academy of Sciences, vol. 110, no. 52, pages 20935–20940, 2013. (Cited on pages 19, 36 and 37.)

[Kulesza & Taskar 2011] Alex Kulesza and Ben Taskar. *k-dpps: Fixed-size determinantal point processes.* In Proceedings of the 28th International Conference on Machine Learning (ICML-11), pages 1193–1200, 2011. (Cited on pages 43 and 45.)

[Kulesza & Taskar 2012] Alex Kulesza and Ben Taskar. *Determinantal Point Processes for Machine Learning*. Foundations and Trends® in Machine Learning, vol. 5, no. 2–3, pages 123–286, 2012. (Cited on pages 8, 43, 44, 48 and 57.)

[Kurt & Hornik 1991] Kurt and Hornik. *Approximation capabilities of multilayer feedforward networks*. Neural Networks, vol. 4, no. 2, pages 251–257, 1991. (Cited on page 102.)

[Langberg & Schulman 2010] Michael Langberg and Leonard J. Schulman. *Universal ϵ-approximators for integrals*. In Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms, pages 598–607. SIAM, 2010. (Cited on pages 61, 63 and 64.)

[Le *et al.* 2018] Can M Le, Elizaveta Levina and Roman Vershynin. *Concentration of random graphs and application to community detection*. In Proceedings of the International Congress of Mathematicians: Rio de Janeiro 2018, pages 2925–2943. World Scientific, 2018. (Cited on page 20.)

[Lei & Rinaldo 2015] J. Lei and A. Rinaldo. *Consistency of spectral clustering in stochastic block models*. The Annals of Statistics, vol. 43, no. 1, pages 215–237, 2015. (Cited on pages 17 and 18.)

[Li *et al.* 2001] Yi Li, Philip M. Long and Aravind Srinivasan. *Improved Bounds on the Sample Complexity of Learning*. Journal of Computer and System Sciences, vol. 62, no. 3, pages 516 – 527, 2001. (Cited on page 66.)

[Lloyd 1982] S. Lloyd. *Least squares quantization in PCM*. IEEE Transactions on Information Theory, vol. 28, no. 2, pages 129–137, March 1982. (Cited on page 17.)

[Loukas 2019] Andreas Loukas. *Graph Reduction with Spectral and Cut Guarantees*. Journal of Machine Learning Research, vol. 20, no. 116, pages 1–42, 2019. (Cited on page 6.)

[Lucic *et al.* 2016] Mario Lucic, Olivier Bachem and Andreas Krause. *Linear-time Outlier Detection via Sensitivity*. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, pages 1795–1801. AAAI Press, 2016. event-place: New York, New York, USA. (Cited on page 63.)

[Macchi 1975] Odile Macchi. *The coincidence approach to stochastic point processes*. Advances in Applied Probability, vol. 7, no. 1, pages 83–122, 1975. (Cited on pages 8, 42 and 54.)

[Mahoney 2011] Michael W. Mahoney. *Randomized Algorithms for Matrices and Data*. Foundations and Trends® in Machine Learning, vol. 3, no. 2, pages 123–224, 2011. (Cited on page 63.)

[Makarychev *et al.* 2016] Konstantin Makarychev, Yury Makarychev, Maxim Sviridenko and Justin Ward. *A Bi-Criteria Approximation Algorithm for k-Means.* In Klaus Jansen, Claire Mathieu, José D. P. Rolim and Chris Umans, editors, Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016), volume 60 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 14:1–14:20, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. (Cited on page 64.)

[Marchal & others 2000] Philippe Marchal and others. *Loop-erased random walks, spanning trees and Hamiltonian cycles.* Electronic Communications in Probability, vol. 5, pages 39–50, 2000. Publisher: The Institute of Mathematical Statistics and the Bernoulli Society. (Cited on page 74.)

[Martin *et al.* 2018] Lionel Martin, Andreas Loukas and Pierre Vandergheynst. *Fast approximate spectral clustering for dynamic networks.* In International Conference on Machine Learning, pages 3423–3432. PMLR, 2018. (Cited on page 105.)

[Martinsson & Tropp 2020] Per-Gunnar Martinsson and Joel A Tropp. *Randomized numerical linear algebra: Foundations and algorithms.* Acta Numerica, vol. 29, pages 403–572, 2020. (Cited on page 52.)

[Maskey *et al.* 2022] Sohir Maskey, Ron Levie, Yunseok Lee and Gitta Kutyniok. *Generalization Analysis of Message Passing Neural Networks on Large Random Graphs.* In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 4805–4817. Curran Associates, Inc., 2022. (Cited on pages 99 and 100.)

[Massoulié 2014] Laurent Massoulié. *Community detection thresholds and the weak Ramanujan property.* In Proceedings of the 46th Annual ACM Symposium on Theory of Computing, pages 694–703. ACM, 2014. (Cited on pages 18 and 19.)

[Maurer 1976] Stephen B Maurer. *Matrix generalizations of some theorems on trees, cycles and cocycles in graphs.* SIAM Journal on Applied Mathematics, vol. 30, no. 1, pages 143–148, 1976. Publisher: SIAM. (Cited on page 103.)

[Micchelli 1986] Charles A Micchelli. *Interpolation of scattered data: Distance matrices and conditionally positive definite functions.* Constructive Approximation, vol. 2, no. 1, pages 11–22, 1986. (Cited on page 55.)

[Mossel *et al.* 2017] Elchanan Mossel, Joe Neeman and Allan Sly. *A proof of the block model threshold conjecture.* Combinatorica, August 2017. (Cited on page 18.)

[Munteanu & Schwiegelshohn 2017] Alexander Munteanu and Chris Schwiegelshohn. *Coresets-Methods and History: A Theoreticians Design Pattern for Approximation and Streaming Algorithms.* KI - Künstliche Intelligenz, December 2017. (Cited on page 61.)

[Napoli *et al.* 2016] Edoardo Di Napoli, Eric Polizzi and Yousef Saad. *Efficient estimation of eigenvalue counts in an interval.* Numerical Linear Algebra with Applications, vol. 23, no. 4, pages 674–692, 2016. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nla.2048. (Cited on page 105.)

[Narang & Ortega 2012] S.K. Narang and A. Ortega. *Perfect Reconstruction Two-Channel Wavelet Filter Banks for Graph Structured Data.* Signal Processing, IEEE Transactions on, vol. 60, no. 6, pages 2786–2799, June 2012. (Cited on pages 15 and 23.)

[Newman & Girvan 2004] M. E. J. Newman and M. Girvan. *Finding and evaluating community structure in networks.* Physical Review E, vol. 69, no. 2, February 2004. (Cited on page 24.)

[O'Leary 1980] Dianne P O'Leary. *The block conjugate gradient algorithm and related methods.* Linear algebra and its applications, vol. 29, pages 293–322, 1980. (Cited on page 86.)

[Pastor-Satorras & Vespignani 2004] Romualdo Pastor-Satorras and Alessandro Vespignani. Evolution and structure of the internet: A statistical physics approach. Cambridge University Press, 2004. (Cited on page 3.)

[Peleg & Schäffer 1989] David Peleg and Alejandro A Schäffer. *Graph spanners.* Journal of graph theory, vol. 13, no. 1, pages 99–116, 1989. Publisher: Wiley Online Library. (Cited on page 6.)

[Pemantle & Peres 2014] Robin Pemantle and Yuval Peres. *Concentration of Lipschitz Functionals of Determinantal and Other Strong Rayleigh Measures.* Combinatorics, Probability and Computing, vol. 23, no. 1, pages 140–160, 2014. (Cited on pages 43 and 67.)

[Pemantle 2004] Robin Pemantle. *Uniform random spanning trees.* arXiv preprint math/0404099, 2004. (Cited on page 71.)

[Pesenson 2009] Isaac Pesenson. *Variational splines and Paley–Wiener spaces on combinatorial graphs.* Constructive Approximation, vol. 29, no. 1, pages 1–21, 2009. Publisher: Springer. (Cited on page 91.)

[Pilavci *et al.* 2021] Yusuf Yigit Pilavci, Pierre-Olivier Amblard, Simon Barthelmé and Nicolas Tremblay. *Graph Tikhonov Regularization and Interpolation Via Random Spanning Forests.* IEEE Transactions on Signal and Information Processing over Networks, vol. 7, pages 359–374, 2021. (Cited on page 93.)

[Pilavcı *et al.* 2022a] Yusuf Yigit Pilavcı, Pierre-Olivier Amblard, Simon Barthelmé and Nicolas Tremblay. *Variance Reduction in Stochastic Methods for Large-Scale Regularized Least-Squares Problems.* In 2022 30th European Signal Processing Conference (EUSIPCO), pages 1771–1775. IEEE, 2022. (Cited on pages 85 and 86.)

[Pilavci *et al.* 2022b] Yusuf Yigit Pilavci, Pierre-Olivier Amblard, simon barthelmé and Nicolas Tremblay. *Variance Reduction for Inverse Trace Estimation via Random Spanning Forests.* In 28ème Colloque sur le traitement du signal et des images, pages p. 525–528. GRETSI - Groupe de Recherche en Traitement du Signal et des Images, September 2022. Issue: 001-0131. (Cited on pages 85 and 86.)

[Pilavci 2022] Yusuf Yigit Pilavci. *Algorithme de Wilson pour l'Algèbre Linéaire Randomisée.* PhD thesis, 2022. 2022GRALT081. (Cited on pages 94 and 105.)

[Puy *et al.* 2016] Gilles Puy, Nicolas Tremblay, Rémi Gribonval and Pierre Vandergheynst. *Random sampling of bandlimited signals on graphs.* Applied and Computational Harmonic Analysis, pages –, 2016. (Cited on pages 28 and 29.)

[Qin & Rohe 2013] Tai Qin and Karl Rohe. *Regularized spectral clustering under the degree-corrected stochastic blockmodel.* Advances in neural information processing systems, vol. 26, 2013. (Cited on pages 20 and 38.)

[Rao 1992] C Radhakrishna Rao. *Information and the accuracy attainable in the estimation of statistical parameters.* In Breakthroughs in statistics, pages 235–247. Springer, 1992. (Cited on page 92.)

[Ruge & Stüben 1987] John W Ruge and Klaus Stüben. *Algebraic multigrid.* In Multigrid methods, pages 73–130. SIAM, 1987. (Cited on page 85.)

[Ruge & Stüben 1987] John W Ruge and Klaus Stüben. *Algebraic multigrid.* In Multigrid methods, pages 73–130. SIAM, 1987. (Cited on page 6.)

[Saad 2011] Yousef Saad. Numerical Methods for Large Eigenvalue Problems. Classics in Applied Mathematics 66. SIAM, 2nd. édition, 2011. (Cited on page 97.)

[Saade *et al.* 2014] Alaa Saade, Florent Krzakala and Lenka Zdeborová. *Spectral clustering of graphs with the bethe hessian.* Advances in neural information processing systems, vol. 27, 2014. (Cited on pages 19, 35, 37 and 38.)

[Saade 2016] Alaa Saade. *Spectral inference methods on sparse graphs : theory and applications.* PhD thesis, 2016. Thèse de doctorat dirigée par Krzakala, Florent Physique théorique Paris Sciences et Lettres (ComUE) 2016. (Cited on page 34.)

[Sandryhaila & Moura 2013] A. Sandryhaila and J.M.F. Moura. *Discrete signal processing on graphs: Graph fourier transform.* In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 6167–6170, May 2013. (Cited on page 13.)

[Shannon 1949] C.E. Shannon. *Communication in the Presence of Noise.* Proceedings of the IRE, vol. 37, no. 1, pages 10–21, January 1949. (Cited on page 4.)

[Sharp *et al.* 2020] Nicholas Sharp, Yousuf Soliman and Keenan Crane. *The Vector Heat Method*, July 2020. arXiv:1805.09170 [cs]. (Cited on page 98.)

[Shewchuk *et al.* 1994] Jonathan Richard Shewchuk*et al.* *An introduction to the conjugate gradient method without the agonizing pain*, 1994. (Cited on page 85.)

[Shuman *et al.* 2013] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega and P. Vandergheynst. *The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains.* Signal Processing Magazine, IEEE, vol. 30, no. 3, pages 83–98, May 2013. (Cited on page 13.)

[Sigal *et al.* 1984] MJ Sigal, S Pitaru, JE Aubin and AR Ten Cate. *A combined scanning electron microscopy and immunofluorescence study demonstrating that the odontoblast process extends to the dentinoenamel junction in human teeth.* The Anatomical Record, vol. 210, no. 3, pages 453–462, 1984. (Cited on page 3.)

[Skorski 2021] Maciej Skorski. *Modern Analysis of Hutchinson's Trace Estimator.* In 2021 55th Annual Conference on Information Sciences and Systems (CISS), pages 1–5. IEEE, 2021. (Cited on page 89.)

[Spielman & Srivastava 2011] Daniel A. Spielman and Nikhil Srivastava. *Graph Sparsification by Effective Resistances.* SIAM Journal on Computing, vol. 40, no. 6, pages 1913–1926, 2011. (Cited on pages 6 and 104.)

[Spielman & Teng 2011] Daniel A Spielman and Shang-Hua Teng. *Spectral sparsification of graphs.* SIAM Journal on Computing, vol. 40, no. 4, pages 981–1025, 2011. Publisher: SIAM. (Cited on page 6.)

[Spielman & Teng 2014] Daniel A Spielman and Shang-Hua Teng. *Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems.* SIAM Journal on Matrix Analysis and Applications, vol. 35, no. 3, pages 835–885, 2014. Publisher: SIAM. (Cited on page 6.)

[Sridharan 2002] Karthik Sridharan. *A gentle introduction to concentration inequalities.* Dept. Comput. Sci., Cornell Univ., Tech. Rep, 2002. (Cited on page 8.)

[Stella 2009] X Yu Stella. *Angular embedding: from jarring intensity differences to perceived luminance*. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 2302–2309. IEEE, 2009. (Cited on page 96.)

[Strang & Nguyen 1996] Gilbert Strang and Truong Nguyen. Wavelets and filter banks. SIAM, 1996. (Cited on pages 15, 20, 21 and 22.)

[Strano *et al.* 2012] Emanuele Strano, Vincenzo Nicosia, Vito Latora, Sergio Porta and Marc Barthélemy. *Elementary processes governing the evolution of road networks*. Scientific reports, vol. 2, no. 1, page 296, 2012. (Cited on page 3.)

[Teke & Vaidyanathan 2017] O. Teke and P. P. Vaidyanathan. *Extending Classical Multirate Signal Processing Theory to Graphs #8212;Part I: Fundamentals*. IEEE Transactions on Signal Processing, vol. 65, no. 2, pages 409–422, January 2017. (Cited on page 23.)

[Terras 2010] Audrey Terras. Zeta functions of graphs: a stroll through the garden, volume 128. Cambridge University Press, 2010. (Cited on page 36.)

[Traag 2015] V. A. Traag. *Faster unfolding of communities: Speeding up the Louvain algorithm*. Phys. Rev. E, vol. 92, no. 3, page 032801, September 2015. (Cited on page 24.)

[Tremblay & Loukas 2020] Nicolas Tremblay and Andreas Loukas. *Approximating Spectral Clustering via Sampling: A Review*. In Frédéric Ros and Serge Guillaume, editors, Sampling Techniques for Supervised or Unsupervised Tasks, pages 129–183. Springer International Publishing, Cham, 2020. (Cited on page 17.)

[Tremblay *et al.* 2016] Nicolas Tremblay, Gilles Puy, Rémi Gribonval and Pierre Vandergheynst. *Compressive spectral clustering*. In International Conference on Machine Learning, pages 1002–1011, 2016. (Cited on pages 30 and 105.)

[Tremblay *et al.* 2017] N. Tremblay, P. O. Amblard and S. Barthelmé. *Graph sampling with determinantal processes*. In 2017 25th European Signal Processing Conference (EUSIPCO), pages 1674–1678, August 2017. (Cited on pages 28 and 80.)

[Tremblay *et al.* 2018] Nicolas Tremblay, Paulo Gonçalves and Pierre Borgnat. *Design of graph filters and filterbanks*. In Cooperative and Graph Signal Processing, pages 299–324. Elsevier, 2018. (Cited on pages 13 and 24.)

[Tremblay *et al.* 2019] Nicolas Tremblay, Simon Barthelmé and Pierre-Olivier Amblard. *Determinantal Point Processes for Coresets*. Journal of Machine Learning Research, vol. 20, no. 168, pages 1–70, 2019. (Cited on pages 54, 61, 63 and 66.)

[Tremblay *et al.* 2023] Nicolas Tremblay, Simon Barthelmé, Konstantin Usevich and Pierre-Olivier Amblard. *Extended L-ensembles: a new representation for Determinantal Point Processes.* The Annals of Applied Probability, vol. 33, no. 1, pages 613–640, 2023. (Cited on pages 43, 55, 56, 57 and 60.)

[Tropp *et al.* 2004] J.A. Tropp, I.S. Dhillon and R.W. Heath. *Finite-Step Algorithms for Constructing Optimal CDMA Signature Sequences.* IEEE Transactions on Information Theory, vol. 50, no. 11, pages 2916–2921, November 2004. (Cited on page 69.)

[Tropp 2015] Joel A. Tropp. *An Introduction to Matrix Concentration Inequalities.* Foundations and Trends® in Machine Learning, vol. 8, no. 1-2, pages 1–230, 2015. (Cited on page 8.)

[Tsitsvero *et al.* 2016] Mikhail Tsitsvero, Sergio Barbarossa and Paolo Di Lorenzo. *Signals on Graphs: Uncertainty Principle and Sampling.* IEEE Transactions on Signal Processing, vol. 64, no. 18, pages 4845–4860, September 2016. (Cited on pages 28 and 81.)

[Tu 2000] Yuhai Tu. *How robust is the Internet?* Nature, vol. 406, no. 6794, pages 353–354, 2000. (Cited on page 3.)

[Veličković *et al.* 2017] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò and Yoshua Bengio. *Graph Attention Networks.* 6th International Conference on Learning Representations, 2017. (Cited on page 100.)

[Vennat *et al.* 2017] Elsa Vennat, Wenlong Wang, Rachel Genthial, Bertrand David, Elisabeth Dursun and Aurélien Gourrier. *Mesoscale porosity at the dentin-enamel junction could affect the biomechanical properties of teeth.* Acta biomaterialia, vol. 51, pages 418–432, 2017. (Cited on page 3.)

[Vertaure 2021] Jordan Vertaure. *Réorganisation neuronale dans le striatum à différents stades d'apprentissage d'une tâche motrice chez la souris : étude statistique de certains aspects.* Msc thesis, University Grenoble Alpes, 2021. (Cited on page 2.)

[Vishnoi 2013] Nisheeth K. Vishnoi. $Lx = b$. Foundations and Trends® in Theoretical Computer Science, vol. 8, no. 1-2, pages 1–141, 2013. (Cited on page 6.)

[Vladymyrov & Carreira-Perpiñán 2017] M. Vladymyrov and M. Á Carreira-Perpiñán. *Fast, accurate spectral clustering using locally linear landmarks.* In 2017 International Joint Conference on Neural Networks (IJCNN), pages 3870–3879, May 2017. (Cited on page 5.)

[Von Luxburg *et al.* 2008] Ulrike Von Luxburg, Mikhail Belkin and Olivier Bousquet. *Consistency of spectral clustering.* The Annals of Statistics, pages 555–586, 2008. (Cited on page 17.)

[von Luxburg 2007] U. von Luxburg. *A tutorial on spectral clustering.* Statistics and Computing, vol. 17, no. 4, pages 395–416, 2007. (Cited on page 17.)

[Walshaw 2006] Chris Walshaw. *A multilevel algorithm for force-directed graph-drawing.* Journal of Graph Algorithms and Applications, vol. 7, no. 3, pages 253–285, 2006. (Cited on page 6.)

[Wathen & Zhu 2015] Andrew J. Wathen and Shengxin Zhu. *On spectral distribution of kernel matrices related to radial basis functions.* Numerical Algorithms, vol. 70, no. 4, pages 709–726, Dec 2015. (Cited on page 52.)

[Wilson 1996] David Bruce Wilson. *Generating random spanning trees more quickly than the cover time.* In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pages 296–303. ACM, 1996. (Cited on pages 72 and 73.)

[Wu *et al.* 2021] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang and Philip S. Yu. *A Comprehensive Survey on Graph Neural Networks.* IEEE Trans. Neural Netw. Learning Syst., vol. 32, no. 1, pages 4–24, January 2021. arXiv: 1901.00596. (Cited on pages 6 and 100.)

[Xu & Zikatanov 2017] Jinchao Xu and Ludmil Zikatanov. *Algebraic multigrid methods.* Acta Numerica, vol. 26, pages 591–721, 2017. Publisher: Cambridge University Press. (Cited on page 6.)

[Zachary 1977] W. Zachary. *An information flow model for conflict and fission in small groups.* Journal of anthropological research, vol. 33, no. 4, pages 452–473, 1977. (Cited on page 15.)