Robust and Efficient Data Clustering with Signal Processing on Graphs

THÈSE N^{O} 8184 (2018)

PRÉSENTÉE LE 26 JANVIER 2018

À LA FACULTÉ DES SCIENCES ET TECHNIQUES DE L'INGÉNIEUR
LABORATOIRE DE TRAITEMENT DES SIGNAUX 2
PROGRAMME DOCTORAL EN INFORMATIQUE ET COMMUNICATIONS

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Lionel Jérémie MARTIN

acceptée sur proposition du jury:

Prof. R. Urbanke, président du jury
Prof. P. Vandergheynst, directeur de thèse
Prof. A. Jung, rapporteur
Dr P. Gonçalves, rapporteur
Dr O. Lévêque, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Suisse
2018

Though you can love what you do not master,
you cannot master what you do not love.
— Mokokoma Mokhonoana

Acknowledgments

While finishing to write this thesis, I'm feeling simultaneously proud and nostalgic. This thesis has been a long trip, strewn with pitfalls, enforcing me to adapt and evolve with the situation. However, I would not be where I stand today without the people around me along the way.

Indeed, the world of research is complicated to apprehend at first and sometimes felt unsuitable. It is in those moments that support and guidance are the most important as a student. I would like to thank first my thesis advisors who helped me significantly passing over the puzzling moments of my thesis. For that, I'm thanking cheerfully Pearl Pu and Pierre Vandergheynst. Pierre deserves an extra thanks and a very special one. I would never have ended my thesis without his support (, my insistence,) and his kindness. Thank you for opening your door when I needed it the most and allowing me to meet so many great people in the lab.

A particular thanks goes to Johann Paratte, Nathanaël Perraudin and Andreas Loukas with whom I shared most of my scientific interests and had the chance to collaborate on the different projects. Thank you also for proofreading my thesis at the last minute. I would like to thank also Max, Fabien, Yann, Gil, Alex, Nico and Basile with whom I shared my office. We had a lot of great times there, mixing long philosophical debates, karaoke sessions, unstoppable laughs with some attempts of productivity.

During my PhD, I've been working on many different things where I met a lot of great people, whether it was during teaching, while working on the scientific side projects, in the lab or elsewhere. I do not forget these people and the good times we had together! Many thanks to Vassilis, Naumann, Michaël, Rodrigo, Kostas, Kirell, Helena, Youngjoo, Volodymyr, Benjamin, Xavier, Sibylle, Virginie, Daniel, Julien, Mihailo, Yu, Valentina, Onur, Julien, Virginie, Laurène, Anaïs, Manu, Michaël Gastpar, Olivier Lévêque, Peter Wittwer and Jean-Cédric Chappelier.

I would also like to thank my mother for her uninterrupted love, her endless support and the many encouragements in all my projects. I'm not forgetting the rest of my family and friends neither. I've had my ups and downs during this journey but you were always here for me, thank you! Obviously, last but not least, I want to thank Charlotte for everything that we lived together since we met. These last years have been amazing and I can only hope to live many more like these.

Lausanne, November 2017 Lionel Martin

Abstract

Data is pervasive in today's world and has actually been for quite some time. With the increasing volume of data to process, there is a need for faster and at least as accurate techniques than what we already have. In particular, the last decade recorded the effervescence of social networks and ubiquitous sensing (through smartphones and the Internet of Things). These phenomena, including also the progresses in bioinformatics and traffic monitoring, pushed forward the research on graph analysis and called for more efficient techniques.

Clustering is an important field of machine learning because it belongs to the unsupervised techniques (i.e., one does not need to possess a ground truth about the data to start learning). With it, one can extract meaningful patterns from large data sources without requiring an expert to annotate a portion of the data, which can be very costly. However, the techniques of clustering designed so far all tend to be computationally demanding and have trouble scaling with the size of today's problems.

The emergence of Graph Signal Processing, attempting to apply traditional signal processing techniques on graphs instead of time, provided additional tools for efficient graph analysis. By considering the clustering assignment as a signal lying on the nodes of the graph, one may now apply the tools of GSP to the improvement of graph clustering and more generally data clustering at large.

In this thesis, we present several techniques using some of the latest developments of GSP in order to improve the scalability of clustering, while aiming for an accuracy resembling that of Spectral Clustering, a famous graph clustering technique that possess a solid mathematical intuition.

On the one hand, we explore the benefits of random signal filtering on a practical and theoretical aspect for the determination of the eigenvectors of the graph Laplacian. In practice, this attempt requires the design of polynomial approximations of the step function for which we provided an accelerated heuristic. We used this series of work in order to reduce the complexity of dynamic graphs clustering, the problem of defining a partition to a graph which is evolving in time at each snapshot. We also used them to propose a fast method for the determination of the subspace generated by the first eigenvectors of any symmetrical matrix. This element is useful for clustering as it serves in Spectral Clustering but it goes beyond that since it also serves in graph visualization

Abstract

(with Laplacian Eigenmaps) and data mining (with Principal Components Projection).

On the other hand, we were inspired by the latest works on graph filter localization in order to propose an extremely fast clustering technique. We tried to perform clustering by only using graph filtering and combining the results in order to obtain a partition of the nodes.

These different contributions are completed by experiments using both synthetic datasets and real-world problems. Since we think that research should be shared in order to progress, all the experiments made in this thesis are publicly available on my personal Github account.

Key words: graph signal processing, clustering, spectral graph theory, temporal graphs, evolutionary networks, graph sampling, active sampling, low-rank reconstruction, spectrum analysis, subspace approximation, visualization, graph diffusion, graph localization, partitioning, eigencount, graph filter design.

Résumé

Les données numériques sont omniprésentes dans le monde d'aujourd'hui et le sont en fait depuis quelque temps. Avec l'accumulation des données et l'augmentation du volume à traiter, il est nécessaire de continuer à chercher des méthodes de traitement de données qui soient toujours plus rapide et au moins aussi efficace que celles qui existent à l'heure actuelle. En particulier, la dernière décennie a vu exploser l'utilisation des réseaux sociaux et l'arrivée des objets connectés. Ces phénomènes, comme en témoignent aussi les progrès dans les domaines de la bio-informatique ou de la surveillance du trafic, ont poussé les recherches dans le domaine de l'analyse de graphes à se développer afin de trouver des méthodes encore plus efficaces.

Le clustering est un domaine important parmi les techniques d'apprentissage automatisées puisqu'il appartient à la classe des méthodes non supervisées (c.à.d. qui ne nécessitent pas de posséder de connaissances particulières pour commencer à faire apprendre le modèle). En effet, il est possible d'extraire des motifs à partir de grands ensembles de données sans avoir besoin d'un expert pour en annoter une partie. Malheureusement, les techniques de clustering qui sont le plus utilisés à présent ont toutes tendance à être coûteuses en temps de calcul et de ce fait rendent le traitement de très grands ensembles de données problématique.

Le développement du Traitement de Signal sur Graphes, qui tente d'appliquer les méthodes classiques de traitement du signal sur des graphes à la place de l'axe temporel, a apporté des outils efficaces pour l'analyse de données sur graphes. En considérant le résultat du clustering comme un signal sur les noeuds du graphe, il est possible d'appliquer les outils du TSG afin d'améliorer le clustering sur graphe et plus généralement l'analyse de données sur graphes.

Cette thèse présente plusieurs techniques qui utilisent certains de ces outils pour améliorer la mise à l'échelle du clustering, tout en visant à obtenir une qualité qui s'approche de celle du clustering spectral, une méthode de clustering sur graphe très célèbre qui bénéficie d'une intuition mathématique bien fondée.

Dans un premier temps, nous explorons les bénéfices du filtrage de signaux aléatoires d'une manière pratique et théorique pour déterminer les vecteurs propres du Laplacien. En pratique, cette tentative nécessite de toucher au design des filtres sur graphes qui approximent un passe-bas idéal avec des polynômes. Nous proposons une amélioration à ceux-ci en accélérant l'heuristique qui sert à déterminer la valeur de la fréquence de coupure du filtre. Nous utilisons cette série

de travaux afin de réduire la complexité du clustering des graphes dynamiques (problème qui cherche à partitioner un graphe qui évolue dans le temps pour chaque instant que nous observons). Nous les avons également utilisés pour proposer une méthode rapide à même de déterminer le sous-espace généré par les premiers vecteurs propres d'une matrice symétrique quelconque. Cet élément est utile pour le clustering puisqu'il fait partie des prérequis pour faire fonctionner le clustering spectral, mais son intérêt va au-delà puisqu'il est également utile dans le cadre de la visualisation de graphes (avec Laplacian Eigenmaps) et du traitement de données (avec la projection sur les composantes principales).

Dans un second temps, nous avons été inspirés par les récents travaux sur la localisation des filtres dans le but de proposer un algorithme de clustering extrêmement rapide. Notre contribution a été de réaliser la procédure complète pour le clustering en n'utilisant que des filtrages sur graphes et en combinant les résultats pour former un partitionnement des noeuds.

Les différents travaux présentés dans cette thèse s'accompagnent d'expériences qui utilisent à la fois des données synthétiques et des ensembles de données extraits de problèmes de la vie réelle. Puisque nous soutenons le fait que la recherche se doit d'être un domaine ouvert, qui se partage afin qu'elle avance, l'ensemble des expériences est publiquement disponible sur mon compte personnel sur Github.

Mots clefs: traitement du signal sur graphe, clustering, théorie spectrale des graphes, graphes temporels, réseaux évolutifs, échantillonnage sur graphe, échantillonnage actif, reconstruction de rang faible, analyse spectrale, approximation de sous-espaces, visualisation, diffusion sur graphe, localisation sur graphe, partitionnement, compte de valeurs propres, design de filtres sur graphes.

Contents

List of Figures

List of Tables

1 Introduction

Data is at the heart of the modern world. More or less every human activity relates to data. From purchase records, to social interactions, going through medical examinations, geo-positioning or internet browsing, our everyday activities generate tremendous amount of data. People use it to impact our daily life. For instance, purchase records might help target advertisements and suggesting interesting sales to the consumer, while aggregated medical records are expected to help understanding the causes of diseases. Even if the emergence of the internet in the last decades and, more recently, the global digitalization facilitated significantly the recording, storage and access to these pieces of information, data has been here since the very beginning.

Human beings have always been interested in preserving information from their past actions; experience is based on this previous knowledge accumulated over time. It is actually these very same data that allowed the understanding of our primitive History through the paintings of Lascau or the clay tablets found in Mesopotamia. However, while preserving information was reserved to a small number of people originally, it has been extremely facilitated in our modern world and one can now record and distribute any piece of information in real time across the world.

This has dramatically lowered our expectations of the importance of stored data and led us to the generation of amazing volumes of data of partial interest. Indeed, studies show that 90% of the data that we now possess has been generated in the last two years. This means that 10 times more data should arrive in the next two years at that pace to reach an expected data volume exceeding 40 Zettabytes ($4e^{22}$ bytes) in total by 2020. A significant part of it results from the recent development of smartphones, Internet of Things (IoT) and wearables. These new sensors are of particular interest because they allow to measure events for which no systematic source of data existed so far. Interesting use cases include public transportation sensing [57, 22], personalized healthcare systems [28, 142] or optimizing the power grid [87] to cite a few.

Moreover, with the new habit of storing everything, we noticed that all these tiny pieces of activity allowed to comprehend our world better. This trend made companies realize how important the data that they were producing could help them improve their services and profitability. With 80% of the information created and used by companies being unstructured, the necessity of data

analysis has never been as important.

A very famous, and quite old, example of classification problem is that of USPS (US postal service) who tried to recognize the digits on letters to automatically process the mail. They hand-labeled some of the instances of each of the ten categories (one category per digit) in order to illustrate the possible variations in the writing. Then, the goal was to classify all the incoming digits based on the knowledge base constructed with the hand-labeled samples. The initial database for the training contains 7'291 samples, which is already a lot to label by hand. However, in comparison with today's volumes this is nothing. Statistical inference and learning thus became a large field of computer science; computational learning helping automatize various tasks.

Figure 1.1 – Examples of hand-written digits from the USPS dataset. The task is to recognize the different digits and to separate all samples into ten classes.

More recently, one of the major problems became the high speed of data. Due to the lack of interest for it in a near future, we want to compute a lot very fast or we will not need the output when it will be ready. Consider it in the following way: as a shop owner, it would be interesting to use the purchasing data of a given store accumulated over time to predict the attendance and help design the schedule for the months to come but this will not be of any help if the computation time is so large that the event that we want to predict happens before the predictions are produced.

This imposes two important constraints in the design of new techniques: they must be able to manage very large datasets, on the one hand, and simultaneously, they must produce their results as quickly as possible. One must thus ensure that the complexity and the storage required to process the data are as low as possible. These two constraints are at the heart of this work and will be our primary focus when proposing new techniques for the analysis of data.

In most applications, hand-labeling is highly time consuming and expensive, even practically infeasible in some cases, because it requires expert knowledge, which is not always accessible. Thus, one of the tracks in automatized learning concerns only the unsupervised problems: those for which no ground truth is provided with the data. In this setup, one can expect to regroup similar data points and allow to efficiently characterize the incoming entries in the future. This is the goal of *data clustering*. For instance, assuming that USPS did not have the capability to hand label the digits in the mail, we could try to separate the different images into the ten categories based on their similarity.

Figure 1.2 – Clustering of the USPS dataset into ten classes.

The current situation points to a clear direction, that of improved data analysis techniques. More specifically, we focus in this thesis on the study of unsupervised methods and especially the task of data clustering. When applicable, we will extend the proposed method to the neighboring fields of data visualization and dimensionality reduction. In this context, the main question that I would like to contribute to in this thesis is the following:

What tools could we design to improve data clustering in order to reduce the computational time required to process the current volume of information?

There are no doubts that a large number of researchers already addressed the problem of efficiency in data analysis. Indeed data science has already been a strong study field for a couple of decades. Actually, the methods used to extract efficiently meaningful pieces of information from the growing corpora at hand are split around several topics of interest, which are closely related.

First is *data acquisition*, which we could consider as data preprocessing. A piece of information such as an image can be acquired with various levels of details impacting its quality for instance

and also its size. When we try to understand a phenomenon based on a dataset, what is commonly done in the field is to extract the characteristics of each data point, that we call *features*. Obviously, the more features we have, the more complicated will be the processing and thus the longer. We usually define the *feature space* as the huge set of all the potential items that could exist.

Sticking to the previous example with USPS data, we could characterize the images with their pixel colors generating a feature space of dimension proportional to the product of the width and the height of the image. On the extreme opposite, for instance under very tight space constraints, we could simply store the color of the average of the pixels and reduce the feature space to a one-dimensional space.

While this last example is an extreme case that has little practical purpose, we are interested in representing information with the smallest dimension that allows to solve the task. Very often, this will translate into a question of trade-offs between quality and speed. This is because the vast majority of embeddings that allow to reduce the dimensionality are discarding part of the information. This goes into a second category: *dimensionality reduction*.

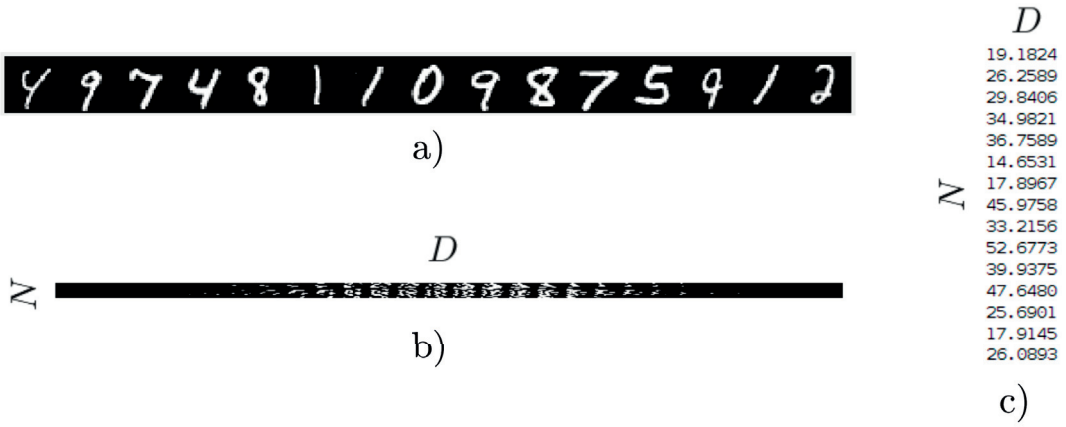


Figure 1.3 – Data acquisition of $N = 15$ samples from the entries of MNIST under various representations. (a) displays the input data in the format it was scanned. (b) is the $N \times D$ feature matrix associated with the entries above, where each column corresponds to a given pixel ($D = 784$). In both (a) and (b) values are stored as integers in $[0, 255]$ but are displayed in grayscale. (c) is an extreme application of dimension reduction where only the average of the pixels is kept and $D = 1$.

Ultimately, in order to visualize the data points one would reduce the dimensions down to two or three. Indeed, our comprehension of data also depends on our ability to represent it and get insights from it. Unfortunately, humans have difficulties to interpret data with many dimensions. Since we are not good for visualizing in more than three dimensions, a key challenge is to display the elements in such a space while they originally live in much larger spaces in order to simplify information while keeping the major differences on display. This specific reduction belongs to *data visualization*.

Overall, the field of dimensional reduction has been very active [134, 42] and at the same time the task of features selection has been clearly recognized through many research testing criteria to particular tasks [25, 117, 141]. Being able to accelerate even further the task of dimensionality reduction would help in the analysis of datasets, especially when the number of features is extremely large.

In the context of data clustering, a common practice of data preprocessing consists of constructing a graph from the high dimensionality features of the data points after their extraction. Graphs are useful because they allow to represent high dimensional objects with a single vertex and to model the similarity between the different entities with its links. They also contribute to the idea that the intrinsic dimensionality of a graph-based representation of a data set is actually lower than that produced by the feature extraction that we apply. Indeed, the construction of the graph would help capture only the intrinsic dimension, performing dimensionality reduction that would both fasten the following computations and help to explain the correlations between samples.

Moreover, social networks and virtual interactions opened new perspectives to social sciences allowing studies to overpass the geographic limitations and ease the observation of masses. In these applications, graphs are natural ways to represent the data of study. In particular, we witnessed, in the last decades, the development of social networks like Facebook or Twitter but it goes much further in the scientific communities. Generally speaking, networks are interesting because they provide a well-structured information layer on top of data. We use them commonly to connect different pieces of information together and not specifically when it comes to people. Network science has also been very useful for gene expression, vehicular traffic monitoring, influence detection, epidemics localization and many more. In this thesis, the goal is to address methods that can be applied both on raw data as well as network data.

Motivations

In the past, many accurate techniques have been introduced to tackle the questions of dimensionality reduction, clustering, and visualization. Mostly, they used the fact shared among those problems that high-dimensional data (in $\mathbb{R}^D$) often admits an accurate low-dimensional intrinsic representation. Finding this embedding alleviates the processing and storage constraints of further processing tasks by representing data points in a space of smaller dimensions $d \ll D$. In particular, it gave rise to the domain of low-rank representation and spectral graph theory that are theoretically well-founded and allow to solve the partitioning problem optimally. This enabled great improvements in the domain of data clustering and introduced the problem of graph eigendecomposition.

Eigendecomposition has been at the core of famous techniques used to extract low-dimensional embeddings from high-dimensional data by using the eigenvectors associated with specific eigenvalues. This has been used for partitioning (e.g., spectral clustering [98, 122]), data visualization (e.g., Laplacian eigenmaps [12]), but also simply as a dimensionality reduction

technique for preprocessing (e.g., principal components analysis [70]). The main drawback of all the aforementioned techniques is that they tend not to scale well as they have a rather high complexity. Indeed, a full eigendecomposition of the covariance matrix between feature costs $\mathcal{O}(D^3)$.

The recent development of Graph Signal Processing (GSP) provides new tools for the study of diagonalizable matrices using frequency analysis, similarly to what is done in Discrete Signal Processing. The reader is referred to Chapter 2 for a detailed introduction. Within this framework, I will propose a set of new techniques for dimensionality reduction and clustering applied in particular to network data. I will briefly recall a way to extend these techniques to any data analysis problem by constructing graphs based on the data similarity, using nearest neighbor graphs.

With these tools, I suggest taking a new look at spectral graph theory. We are now capable of avoiding the computation of the eigendecomposition while still obtaining the well-studied solution of partitioning. This allows to reduce drastically the complexity of the quite old method of Spectral Clustering (SC) with a loss of accuracy that is both small and adaptive to the task. One can indeed decide to set the threshold between speed and accuracy at her will. This new approach sets a whole new viewpoint and promises to go even further.

Well designed filters of a given kind will allow to solve the eigendecomposition problem while others tend to provide graph centrality measures efficiently. With the latter, I aim to replace the k-means algorithm, very often used for data partitioning and underlying in a lot of algorithms across various domains. Overall, the goal with GSP is definitely to help solve some of the central problems of data science, not necessarily with an improved accuracy but providing ways to apply them efficiently on large amount of data while achieving an accuracy similar to some of the most famous algorithms existing so far.

Contributions and thesis structure

I describe in more details the state-of-the-art in **Chapter 2**, together with the basic definitions and the notation used throughout this thesis. It contains an introduction to GSP, presenting the fundamental results on which the rest of the thesis is built. This chapter also serves to describe comparative methods in detail in order to present the benefits of the following works.

This thesis then focuses on the improvement of approximated methods mainly for the task of graph clustering using the latest developments in the field of GSP. In this regard, in **Chapter 3**, I recall the recent advances based on filtering of random signals and propose two improvements. On the one hand, I analyze theoretically the assignment quality guaranteed by the random filtering and establish a characterization of the approximation error to SC and a way to trade between accuracy and time. On the other hand, I introduce a natural extension for dynamic graphs where I consider that connections between the entities can vary over time and thus impact the partitioning

of the graph. We will see that one can reuse part of the information from the previous time step in order to approximate the result of SC with a reduced complexity.

Clustering, visualization and dimensionality reductions have an important common factor that is also studied in this thesis in **Chapter 4**. By contributing to the problem of efficient graph Fourier basis determination, we will see that we can improve the complexity for exact subspace recovery and thus the problems that use it such as visualization (with Laplacian eigenmaps) or graph clustering (with spectral clustering). Although not perfect for Principal Component Analysis (PCA), I show how to use this for Principal Component Projection (PCP).

Then, since graph clustering is at the heart of this thesis, we will also see how the localization operator defined on graphs can help partitioning graphs in **Chapter 5**. I will briefly talk about structural clustering where nodes are grouped based on their role in the graph, and not their similarity anymore before I move to use this tool for standard clustering. Since I present how to avoid the eigendecomposition in SC in the previous chapters, I now try to remove the k-means step using properties of graph signals.

Finally, **Chapter 6** summarizes the findings and concludes the work. It also presents future directions that are left as open problems and would benefit from additional research. A natural extension to this problem would be to consider multilayer clustering next. The primary concern is to combine several sources of information to cluster a single set of nodes. This problem is very interesting because we often possess several sources of information that we do not know how to combine and end up combining them to obtain a single adjacency matrix.

In the various chapters of this thesis, all the experiments are designed using the open-source Graph Signal Processing toolbox (GSPBox [107]) and the code is available on my personal Github account to promote open research and collaborations: https://github.com/lionel-martin/. Please note that since our methods use random processes, it is expected for the results to be slightly different in the details, but overall consistent with the presented figures and tables.

2 From Signal Processing to Clustering

This chapter will present the foundations of the works presented in the following of the thesis. It regroups all the pre-existing tools and techniques that are of principal interest for one or several of the following chapters together with the most famous and/or best performing methods of clustering with which this work should be compared.

It is organized into a couple of distinct parts. The first one is dedicated to the study of signal processing on graphs, starting from the formal definition of graphs to the latest development in the signal processing field around it. The second part is specifically going to present various data analytics methods and connects data clustering to graph clustering, our primary focus. Finally, a presentation of very famous clustering techniques, used in the latter parts as baselines for the evaluation of the quality of our proposed methods, gives a good opportunity to recall the mathematical properties behind it.

Notation

All along this thesis, we will use a consistent notation regarding the variable names.

- Sets are written with calligraphic letters (e.g., $\mathcal{A}, \mathcal{B}$) as well as the usual $\mathbb{N}, \mathbb{Z}$ and $\mathbb{R}$;
- Scalars are denoted with lowercase letters such as a or b;
- Vectors are defined as column vectors and are represented with boldface letters (e.g., $\mathbf{a}, \mathbf{b}$);
- Matrices and higher order tensors are bold uppercase letters (e.g., $\mathbf{A}, \mathbf{B}$). Their elements are referred to using comma-separated indices such as $\mathbf{A}_{i,j}$ for a matrix, defining the element in the i-th row and j-th column;
- Functions defined over matrices have to be interpreted as elementwise operations over diagonal matrices. For non-diagonal matrices, we should first diagonalize it and then apply the elementwise operator. For instance, if g is a univariate variable over $\mathbb{R}$, $g(\mathbf{L}) = \mathbf{U}g(\mathbf{\Lambda})\mathbf{U}^\top$ where $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ is its diagonalization.

2.1 Fundamentals of Graph Signal Processing

2.1.1 Graph theory

Graph theory is a field of discrete mathematics, whose origin dates back to several centuries ago, that is dedicated to the concept of *graphs*. Graphs are mathematical structures, generally used to model connections between two different entities. Although graph theory was initially used to solve puzzles such as the seven bridges of Königsberg [44] or the Knight tour problem [110], graphs' discrete structure made it a compelling field. Over the centuries, lots of theories involving graphs were proposed and various problems were solved such as perfect matching, coloring, routing, covering and more [18]. It thus became the ideal tool to observe networks in the recent developments of network science, given its core properties making it both a good data structure for the problems posed on networks and a well-studied tool for routing problems.

Graphs are at the center of the works presented in this thesis. We will start this section by an introduction of graphs and add concepts related to these objects in a step by step fashion. We define a graph $\mathcal{G}$ by a triplet $(\mathcal{V}, \mathcal{E}, \mathbf{W})$ where $\mathcal{V}$ is the set of vertices (also called nodes throughout the thesis with no distinction) and $\mathcal{E}$ the set of edges representing connections between nodes in $\mathcal{V}$. Additionally $\mathbf{W}$ weights the different edges, characterizing how the nodes relate to each other.

The vertices $v \in \mathcal{V}$ of the graph are ordered from v_1 to $v_N, N = |\mathcal{V}|$. In the most common setting, an edge is a direct connection between two nodes of the graph: $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$. Rarely, we will be considering hypergraphs instead that are constructed in the same way except edges will become abstractions connecting a subset of the nodes together: $\mathcal{E} \subset \mathcal{P}(\mathcal{V})$, the power set of $\mathcal{V}$. We characterize the graphs depending on the number of edges $M = |\mathcal{E}|$. We call *sparse graphs* those with $M = \mathcal{O}(N)$ and *dense graphs* when $M = \Theta(N^2)$. In between, the characterization is not well defined.

The matrix $\mathbf{W}$ is called the weighted adjacency matrix of the graph $\mathcal{G}$. This can be interpreted as a similarity measure between the nodes and is the most important part of the description of the graph. Although this might be an oversimplification in practice sometimes, a common assumption that we will also make throughout the thesis is that the connections are symmetrical, that is, the connection from a to b is as important as that from b to a. Such graphs are called *undirected graphs*.

The element $\mathbf{W}_{i,j}$ represents the weight of the edge between vertices v_i and v_j and a value of 0 means that the two vertices are not connected. The degree d_i of a node v_i is defined as the sum of the weights of all its edges

$$d_i = \sum_{j=1}^{N} \mathbf{W}_{i,j}. \tag{2.1}$$

The degree matrix $\mathbf{D}$ is the diagonal matrix having the degrees d_i on its diagonal.

Another characterization of the graph connectivity is represented through the incidence matrix of the graph, which we will be using in the following of this chapter.

Definition 2.1: Oriented Incidence Matrix.
The oriented incidence matrix $\mathbf{E}$ of a graph is a matrix of size $|\mathcal{V}| \times |\mathcal{E}|$ where each column describes an edge. In weighted directed graphs, the elements are populated with values such that:

$$\mathbf{E}_{i,j} = \begin{cases} -\sqrt{\mathbf{W}_{i,j}}, & \text{if } e_{i,j} \text{ leaves } v_i \\ \sqrt{\mathbf{W}_{i,j}}, & \text{if } e_{i,j} \text{ enters } v_i \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

The undirected case is treated similarly by selecting any extremity of each edge as the source, setting the remaining as sinks.

Through this thesis we will be referring to a small set of different synthetic graphs, useful to give insights on the new techniques that we will present. In the following, we present the properties of these various graphs.

Example 2.2: Nearest Neighbors Graph.
A Nearest Neighbors Graph (usually called NN-graph) is a graph whose edges are defined based on a distance metric between the vertices and where only the closest ones are connected to each other. Nodes are usually positioned beforehand into a dimensional space depending on their features. Often, the distance metric is defined to measure their similarity, usually as a kernelized version of the Euclidean norm (ℓ_2-norm): $d(u, v) = k\left(\left\|\boldsymbol{f}_u - \boldsymbol{f}_v\right\|_2\right)$.

There are two common classes of unweighted nearest neighbors. One is based on the number of neighbors, called k-NN graphs it restricts the connections to the k closest vertices of each source. The other one is named ϵ-NN graph and keeps only the neighbors at a distance of at most ϵ of their source independently of the number of close neighbors.

Additionally, the weights of the edges can either be binary (i.e., 1 when there is a connection and 0 otherwise) or be a function of the distance between the two vertices. We refer the latter model as the kernelized NN-graphs. Among the different kernels, the Gaussian kernel is prominently used and yields the following weighting

$$\mathbf{W}_{i,j} = \exp\left(-\frac{d(x_i, x_j)^2}{\sigma^2}\right), \tag{2.3}$$

where $d(\cdot,\cdot)$ is the distance metric of your choice and σ describes the speed of decay of the exponential. When not specified, we will be using the Euclidean distance for d and the average distance for σ.

Example 2.3: Sensor Graph.
A sensor graph is a k-NN graph whose nodes have been randomly placed in the unit square in $\mathbb{R}^2$. It resembles networks of measurements such as temperature networks or traffic networks, hence the name.

Example 2.4: Stochastic Block Model.
The Stochastic Block Model (SBM) is a probabilistic generative model of graphs designed specifically for clustering tasks. In this model, all the nodes are assigned a class between 1 and k at the beginning. Then, the graph is populated based on the class appertaining of the two extremities of the edge. Namely, if two nodes u and v belong to the same class, they have a probability q_1 of being connected. On the opposite, if they do not have the same class, the edge probability is q_2. We refer to q_1 as the intra-class probability while we call q_2 the inter-class probability. By definition, we must set $q_1 > q_2$ in order to emphasize the clusterability of the graph.

Abbe et al. [1] studied the regimes where we could almost surely detect a community structure. They deduced that if $q_1, q_2 \propto \frac{\log(N)}{N}$ and $\frac{q_1+q_2}{2} - \sqrt{q_1 q_2} > \frac{\log(N)}{N}$ then such community structure existed.

Alternatively, this model can be characterized entirely with the knowledge of the average degree in the graph $\overline{d_{\mathcal{G}}}$ and the ratio $\varepsilon = \frac{q_2}{q_1}$. Decelle et al. [36] suggested the existence of a threshold above which community detection is impossible:

$$\varepsilon_{\max} = \frac{\overline{d_{\mathcal{G}}} - \sqrt{\overline{d_{\mathcal{G}}}}}{\overline{d_{\mathcal{G}}} + (k-1)\sqrt{\overline{d_{\mathcal{G}}}}}. \tag{2.4}$$

For a given ratio ε, q_1 and q_2 can be determined assuming that the sizes of the classes are known. If we call them n_i for $i = 1, \dots, k$ we have

$$q_1 = \frac{\overline{d_{\mathcal{G}}} N}{\sum_{i=1}^{k} n_i \left(n_i - 1 + \varepsilon(N - n_i)\right)} \qquad \text{and} \qquad q_2 = q_1 \varepsilon. \tag{2.5}$$

If we assume the k classes to be of equal size, this yields

$$q_1 = \frac{\overline{d_{\mathcal{G}}} k}{N(k-1)\varepsilon + N - k} \qquad \text{and} \qquad q_2 = q_1 \varepsilon. \tag{2.6}$$

Example 2.5: Erdös-Rényi.
Named after the two scientists that made a breakthrough in the field of random graph theory [43], this model is a simplified version of the SBM. Here, no classes are assigned beforehand and all edges have the same probability p.

Additionally, the authors proved that if $p > \frac{(1+\varepsilon)\ln(N)}{N}$, then the graph will be almost surely connected.

We decided to make two remarks before we proceed. First, this model is important in graph clustering because it sometimes serves as the null hypothesis for clustering. Indeed, given its construction, it represents a set of graphs that do not possess any community structure. Then, the study of the discrepancies between a graph of interest and the statistics of the Erdös-Rényi points out the potential community structure of the given graph. This topic is detailed in Sec. 2.2.4.

Second, note that we can also reuse the property of connectivity to compute the threshold for q_1 and q_2 in the SBM to ensure connectivity. By computing the average probability in an SBM, assuming uniformly distributed clusters, this gives

$$\frac{\overline{d_{\mathcal{G}}}}{N} = \frac{q_1 + (k-1)q_2}{k} > \frac{(1+\varepsilon)\ln(N)}{N}. \tag{2.7}$$

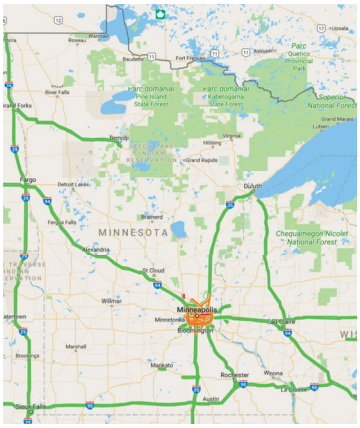
2.1.2 Graph signals

When the nodes of the graph represent locations (e.g., with sensor graphs or road networks), additional data is associated with the nodes. The reader is referred to Fig. 2.1 for an example of traffic data where the nodes are crossroads that possess the additional information of the jam at their position. The fluidity of traffic is a signal living on the graph. In practice, it frequently happens that the graph generated from a dataset represents the sampling of a continuous structure that might be known or not. In such a case, the assumption that the data is smoothly evolving on the graph is often used to model the behavior of the phenomenon under study. This can be measured by summing the square of the differences between each pair of connected nodes and this measure corresponds to the Dirichlet energy of the process. However, the signal can also be simply a categorization of the different parts of the network (e.g., for the study of epidemics or influence detection). We now introduce the mathematical definition of graph signals before to go back to Dirichlet energy and smoothness.

Definition 2.6: Graph signal.
A graph signal $\boldsymbol{f}$ is defined as an application $\boldsymbol{f} : \mathcal{V} \to \mathbb{R}$ and is represented as a vector of size N where the i-th component of the vector is the value of the signal at vertex v_i. Multidimensional signals can be defined by extension as a vector of value instead of scalars on the nodes, potentially describing temporally evolving behaviors.

From there, an important mathematical tool that has been defined is the Laplacian. There exist several definitions possessing each a slightly different interpretation but all of these relate to the continuous definition of the Laplace operator on Euclidean spaces, hence the name.

The link between the two has been presented in great details in [13] where they highlight the fact that it can be expressed as the divergence of the gradient over graphs. In this case the gradient



(a) Map data ©2017 Google

(b) Minnesota graph with traffic signal

Figure 2.1 – An example graph signal representing traffic data in Minnesota.

computes the derivative along all the edges of the graph based on the value at its endpoints

$$\left(\nabla_{\mathcal{G}} \boldsymbol{f}\right)_{e=(v_i,v_j)} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{e}}\right)_{e=(v_i,v_j)} = \sqrt{\mathbf{W}_{i,j}}\left(\boldsymbol{f}[j] - \boldsymbol{f}[i]\right) \tag{2.8}$$

and the divergence is its adjoint operator.

The gradient in this case can be represented by a multiplication with the incidence matrix of the graph. Indeed, we have

$$\begin{aligned}\left(\mathbf{E}^\top \boldsymbol{f}\right)_e &= \sqrt{\mathbf{W}_{i,j}}\left(\boldsymbol{f}[j] - \boldsymbol{f}[i]\right) \\ &= \left(\nabla_{\mathcal{G}} \boldsymbol{f}\right)_{e=(v_i,v_j)}.\end{aligned} \tag{2.9}$$

With this definition, the Dirichlet energy that was discussed above can be computed as

$$\boldsymbol{f}^\top \mathbf{E}\mathbf{E}^\top \boldsymbol{f} = \left(\nabla_{\mathcal{G}} \boldsymbol{f}\right)^\top \left(\nabla_{\mathcal{G}} \boldsymbol{f}\right) = \sum_{e=(v_i,v_j)} \mathbf{W}_{i,j}(f_i - f_j)^2. \tag{2.10}$$

This provides the most common definition of Graph Laplacian which is that of the combinatorial Laplacian $\mathbf{L}_C$ simply computed as

$$\mathbf{L}_C = \mathbf{E}\mathbf{E}^\top = \mathbf{D} - \mathbf{W}. \tag{2.11}$$

While possessing already useful properties for the analysis of the graph structure, there exist two additional famous versions of the Laplacian that we are going to present quickly before explaining with more details its properties.

First is the normalized Laplacian. Its name comes from the fact that we add a normalization

components in order to be unaffected by the local node degree. Its formulation is

$$\mathbf{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{L}_C\mathbf{D}^{-\frac{1}{2}}. \tag{2.12}$$

The second one is the random walk Laplacian:

$$\mathbf{L}_{\text{RW}} = \mathbf{D}^{-1}\mathbf{L}_C. \tag{2.13}$$

This one contains in its elements the probability distribution of the random walk over the graph following the edge weight as prior for the next movement.

In the last two cases, by convention, we set $\mathbf{D}_{i,i}^{-\frac{1}{2}} = 0$ when $d_i = 0$ and thus $\mathbf{L}_{i,i} = 0$ for isolated vertices v_i. For graphs with no isolated vertices, we can relate the Laplacians to the weight matrix as:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} \qquad \text{and} \qquad \mathbf{L}_{\text{RW}} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}. \tag{2.14}$$

2.1.3 Spectral theory

Now that signals are defined, signal processing is a natural way to proceed for its analysis. In traditional signal processing, signals represent the amplitude of temporal observations of an event (either continuous or discrete). Among the different transforms, the frequency content can be obtained via a Fourier transform of the signal. Frequency analysis allows to perform lots of different operations on the signal in order to improve it (e.g., denoising) or to ease its transmission or storage (e.g., compression) for instance.

The signals defined on graphs cannot be interpreted in the Fourier domain following the traditional signal processing workflow. However, we would like an operator that would allow its frequency analysis. Noticing that in the time domain, the classical Fourier transforms compute the inner product of the signal with the eigenfunctions of the Laplace operator, a natural definition for the Graph Fourier Transform (GFT) is to take the inner product with the eigenvectors of the graph Laplacian. To this end, we first need to define the eigen-decomposition of the Laplacian.

Definition 2.7: Eigendecomposition of the Laplacian.
By application of the spectral theorem, we know that $\mathbf{L}$ can be decomposed into an orthonormal basis of eigenvectors noted $\{\boldsymbol{u}_\ell\}_{\ell=0,\dots,N-1}$. The ordering of the eigenvectors is given by the eigenvalues noted $\{\lambda_\ell\}_{\ell=0,\dots,N-1}$ sorted in ascending order $\lambda_0 \leq \lambda_1 \leq \dots \leq \lambda_{N-1} = \lambda_{\max}$.

The eigenvalues are computed by minimizing the following objective

$$\lambda_\ell = \min_{\boldsymbol{f} \notin S_{\ell-1}} \frac{\boldsymbol{f}^\top \mathbf{L} \boldsymbol{f}}{\boldsymbol{f}^\top \boldsymbol{f}}, \tag{2.15}$$

where S_ℓ corresponds to the span of the eigenvectors $\boldsymbol{u}_0, \dots, \boldsymbol{u}_\ell$; $S_{-1} = \emptyset$. The normalized version

of the minimizer of each eigenvalue λ_ℓ is the ℓ-th eigenvector associated with this eigenvalue $\boldsymbol{u}_\ell$.

In a matrix form we can write this decomposition as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*$ with $\mathbf{U} = \left(\boldsymbol{u}_0 \middle| \ldots \middle| \boldsymbol{u}_{N-1}\right)$ the matrix of eigenvectors and $\mathbf{\Lambda}$ the diagonal matrix containing the eigenvalues in ascending order.

It is called a Fourier transform by analogy to the continuous Laplacian whose spectral components are Fourier modes, and the matrix $\mathbf{U}$ is sometimes referred to as the graph Fourier matrix (see e.g., [33]). By the same analogy, the set $\left\{\sqrt{\lambda_\ell}\right\}_{\ell=0,\ldots,N-1}$ is often seen as the set of graph frequencies [126].

We now have the tools to represent the Fourier transform on graphs.

Definition 2.8: Graph Fourier Transform.
Given a signal $\boldsymbol{f}$ defined over the vertices of a graph $\mathcal{G}$, its GFT $\widehat{\boldsymbol{f}}$ is defined by the inner product of $\boldsymbol{f}$ with the eigenvectors of $\mathbf{L}$. We thus have

$$\widehat{\boldsymbol{f}}[\ell] = \langle \boldsymbol{f}, \boldsymbol{u}_\ell \rangle = \sum_{i=1}^{N} \boldsymbol{u}_\ell^*[i]\boldsymbol{f}[i] \quad \text{or equivalently in matrix form} \quad \widehat{\boldsymbol{f}} = \mathbf{U}^*\boldsymbol{f}. \tag{2.16}$$

Since $\mathbf{U}$ is an orthonormal basis, the inverse Fourier transform is simply

$$\boldsymbol{f} = \mathbf{U}\widehat{\boldsymbol{f}} \quad \text{which corresponds to} \quad \boldsymbol{f}[n] = \sum_{\ell=0}^{N-1} \boldsymbol{u}_\ell[n]\widehat{\boldsymbol{f}}[\ell]. \tag{2.17}$$

The Laplacian[1] is an interesting operator for graphs since it combines a complete characterization of the graph and useful algebraic properties. We will conclude this section by presenting some of them and refer interested readers to [33, Chap. 1] for a deeper analysis.

Properties 2.9: Graph Laplacian.
First, we present several properties that work for both the combinatorial and the normalized Laplacians. A very simple fact about those two is that they are symmetric matrices. Thus, the eigenvalues of their eigen-decomposition are all real and non-negative. On top of that, they both possess 0 as an eigenvalue. Indeed, if we take $\boldsymbol{u}_0 = \frac{\mathbf{1}}{\sqrt{\mathbf{N}}}$, then $\mathbf{L}_C\boldsymbol{u}_0 = \mathbf{0}$ and similarly with $\boldsymbol{f} = \mathbf{D}^{\frac{1}{2}}\boldsymbol{u}_0$ for the normalized Laplacian $\mathbf{L}_N$. Actually, this particular eigenvalue has a multiplicity that depends on the connectivity of the graph. Indeed, for a disconnected graph with c components of sizes $N_1, \ldots, N_c$, the c signals containing the constant value $\frac{1}{N_i}$ for the nodes of the i-th component and zeros elsewhere all return $\mathbf{L}_C\boldsymbol{f} = \mathbf{0}$ hence producing c orthonormal signals that are the c first eigenvectors of the graph Laplacian.

Next, note that the normalized Laplacian has a bounded spectrum, with $\lambda_{\max} \leq 2$. This is an equality if and only if the graph is bipartite, meaning that the graph can be separated into two

[1] We interchangeably mean the Laplacian matrix, and the operator defined by the multiplication of the Laplacian in the rest of the thesis.

disjoint subsets S and $\bar{S}$ and the edges all possess one end in S and the other one in $\bar{S}$. Regarding the spectrum of the normalized Laplacian, we also know that

$$\sum_{\ell} \lambda_\ell \leq N, \tag{2.18}$$

with equality if and only if the graph does not possess isolated vertices (vertices with degree 0). More generally, since $\mathrm{Tr}(\mathbf{L}_N) = \mathrm{Tr}(\mathbf{\Lambda})$, the sum of the eigenvalues corresponds to the number of non-isolated nodes in the graph.

2.1.4 Filtering graph signals

The frequency analysis of graph signals called for the definition of operations defined in the frequency domain. Continuing on the example of smooth signals on a discretized surface, we notice by definition that those must possess only low-frequency components in order to possess low Dirichlet energy. Thus, filtering out the high frequency components of a signal can help to generate smooth signals on a graph. It can also serve to compress complex signals into smooth approximations defined by only a few components.

One of the most important properties of Fourier in the traditional domain is that convolutions in the time domain are transformed into multiplications in the frequency domain. This property is preserved in graph analysis as we recall in the next definition.

Definition 2.10: Graph convolution.
Let us denote the GFT of a given signal $\boldsymbol{x}$ with $\widehat{\boldsymbol{x}} = \mathcal{F}\{\boldsymbol{x}\}$. If we now consider two given signals $\boldsymbol{x}$ and $\boldsymbol{y}$, the convolution between the two signals reads

$$\mathcal{F}\left\{\boldsymbol{x} * \boldsymbol{y}\right\}[\ell] = \widehat{\boldsymbol{x}}[\ell] \cdot \widehat{\boldsymbol{y}}[\ell] \quad \text{thus} \quad \left(\boldsymbol{x} * \boldsymbol{y}\right)[n] = \mathcal{F}^{-1}\left\{\widehat{\boldsymbol{x}} \cdot \widehat{\boldsymbol{y}}\right\}[n]. \tag{2.19}$$

Thus, filtering can be carried out by a pointwise multiplication in Fourier in the exact same way as in the traditional domain. To this end, we define a graph filter as a continuous function $g : \mathbb{R}_+ \to \mathbb{R}$ directly in the graph Fourier domain.

Definition 2.11: Graph filtering.
If we consider the filtering of a signal $\boldsymbol{f}$, whose GFT is written $\widehat{\boldsymbol{f}}$, by a filter g the operation in the spectral domain is a simple multiplication $\widehat{\boldsymbol{f}}'[\ell] = g(\lambda_\ell) \cdot \widehat{\boldsymbol{f}}[\ell]$, with $\boldsymbol{f}'$ and $\widehat{\boldsymbol{f}}'$ the filtered signal and its GFT respectively. Using the graph Fourier matrix to recover the vertex-based signals, we get the explicit matrix formulation for graph filtering:

$$\boldsymbol{f}' = \mathcal{F}^{-1}\left\{g(\lambda_\ell) \cdot \widehat{\boldsymbol{f}}[\ell]\right\} = \mathcal{F}^{-1}\left\{g(\mathbf{\Lambda})\mathcal{F}\left\{\boldsymbol{f}\right\}\right\}. \tag{2.20}$$

Combining equation (2.20) with the definitions of equations (2.16) and (2.17) provides a simple

vector-matrix formulation for the graph filtering:

$$\boldsymbol{f}' = \mathbf{U} g(\boldsymbol{\Lambda}) \mathbf{U}^* \boldsymbol{f}. \tag{2.21}$$

Using the definition of functions over matrices, $g(\mathbf{L}) := \mathbf{U} g(\boldsymbol{\Lambda}) \mathbf{U}^*$, this is most often shortened into

$$\boldsymbol{f}' = g(\mathbf{L}) \boldsymbol{f}. \tag{2.22}$$

Since the filtering equation defined above involves the knowledge of the full set of eigenvectors $\mathbf{U}$, it implies the diagonalization of the Laplacian $\mathbf{L}$ which can become intractable for large graphs. To circumvent this problem, a solution is to determine the value of $g(\mathbf{L})$ without the knowledge of $\mathbf{U}$, bypassing the computation of (2.21). A common method, introduced in [58], suggests to use the degree-n Chebyshev interpolation formula. The authors noticed that the recurrence relation used in the definition of the Chebyshev polynomials of the first kind allows to compute a faithful approximation of the filter iteratively and with a reduced cost.

Definition 2.12: First-kind Chebyshev polynomials.
The Chebyshev polynomials of the first-kind are defined with the recurrence relation

$$\mathcal{T}_0(x) = 1 \qquad \mathcal{T}_1(x) = x \qquad \mathcal{T}_n(x) = 2x\mathcal{T}_{n-1}(x) - \mathcal{T}_{n-2}(x). \tag{2.23}$$

Properties 2.13.
Among the different properties of the Chebyshev polynomials, we stress the fact that

1. $\mathcal{T}_n(x) = \cos(n \arccos(x)), \forall x \in [-1, 1]$
2. The roots of the polynomial are $x_j = \cos\left(\pi \frac{2j+1}{2(n+1)}\right), j = 0, \dots, n$. They are called the Chebyshev nodes and are often used in polynomial interpolation since they minimize the Runge's phenomenon (i.e., the fact that interpolation might increase the error with the polynomial order). Interpolating at these points provides an approximation close to the optimal solution under the maximum norm.

Definition 2.14: First-kind Chebyshev interpolation.
Given a continuous function $g : [-1, 1] \to \mathbb{R}$, the degree-$n$ Chebyshev interpolation is

$$g_n(x) := \sum_{m=0}^{n} c_m \mathcal{T}_m(x), \quad \text{where } c_m = \frac{2 - \delta(m)}{n+1} \sum_{j=0}^{n} g(x_j) \mathcal{T}_k(x_j). \tag{2.24}$$

In our case, since the spectrum is not contained in $[-1, 1]$, there is an extra step of renormalization of the domain. The mapping $\theta : x \to \frac{\lambda_{\max} x}{2} - 1$ will ensure that all the eigenvalues λ_ℓ will be

contained in the support. Finally, the Chebyshev approximation of order P_o of the graph filter is

$$g(\mathbf{L})\boldsymbol{f} \approx \sum_{m=0}^{P_o} c_m \mathcal{T}_m(\mathbf{L})\boldsymbol{f}, \quad \text{where } c_m = \frac{2-\delta(m)}{P_o+1} \sum_{j=0}^{P_o} g(\theta^{-1}(x_j))\mathcal{T}_k(x_j), \tag{2.25}$$
$$\text{and } \mathcal{T}_0(\mathbf{L})\boldsymbol{f} = \boldsymbol{f} \qquad \mathcal{T}_1(\mathbf{L})\boldsymbol{f} = \theta(\mathbf{L})\boldsymbol{f} \qquad \mathcal{T}_m(\mathbf{L})\boldsymbol{f} = 2\theta(\mathbf{L})\mathcal{T}_{m-1}(\mathbf{L})\boldsymbol{f} - \mathcal{T}_{m-2}(\mathbf{L})\boldsymbol{f}.$$

Apart from this frequently used approximation, recent works in the GSP community also used Lanczos method to approximate any filter [129] and Jackson-Chebyshev approximation alleviating the Gibbs oscillations of the standard Chebyshev approximation developed above [39].

In the particular case of approximating the sign function in $[-1,1]$, Allen-Zhu and Li [6] proposed an approximation g_n that converges even faster to the step function. It has been introduced jointly with mathematical guarantees that we recall quickly.

Properties 2.15: Estimation of the sign function with truncated polynomials.
The approximation of the sign function with these polynomial can be as precise as necessary. The polynomial is guaranteed to:

1. $|g_n - \mathrm{sgn}(x)| \leq \epsilon$ for all $x \in [-1,-\alpha] \cup [\alpha,1]$,
2. $g_n(x) \in [-1,0]$ for all $x \in [-\alpha,0]$ and $g_n(x) \in [0,1]$ for all $x \in [0,\alpha]$.

The choice of ϵ and α enforce a minimum degree such that $P_o \geq \frac{1}{\sqrt{2}\alpha}\log\left(\frac{3}{\epsilon\alpha^2}\right)$.

This polynomial is constructed in two steps. First using Chebyshev approximation, we can generate a polynomial approximation $q_{P_o} = \left(\frac{1+\kappa-x}{2}\right)^{-\frac{1}{2}}$ for $\kappa \in [0,1)$. Then g_n is computed simply as $g_n = x \cdot q_{P_o}(1+\kappa-2x^2)$. The value of κ has a direct impact on the approximation since $\kappa = 2\alpha^2$. Finally, the degree of g_n is $2P_o+1$ due to the composition of x^2 and $q_{P_o}(x)$.

All these methods scale with the number of edges M and reduce the complexity of a filtering operation to $\mathcal{O}(P_o M)$, which is especially advantageous in the case of sparse graphs. When the graphs are large, then $P_o \ll N$ and there is an undoubted gain to consider polynomial approximations instead of the $\mathcal{O}(N^3)$ computational cost induced by the eigendecomposition.

2.1.5 Localization operator

The localization of signals over their support is interesting in general. For discrete time signals, which are defined over a chain graph, translations, corresponding to time shifts, enable the localization of the signal. Translations are, moreover, particularly useful for stationarity analysis. Here, the support (i.e., the vertex set) is not ordered and a direct transposition of the concept to graphs is an ill-posed problem. However, in contrast to discrete time signals, all frequencies are not spread equally over the nodes (while it is the case with time in signal processing). In fact,

frequencies can be localized around certain nodes depending on the graph and the generalization of localization on graphs can be organized around this fact.

Several attempts were made to generalize the concept of localization from time signals to graph signals . However, the lack of order over the support alters the meaning of a signal translation and requires to find a correspondence in the spectral domain instead. Shuman et al. [126] as well as Perraudin et al. [106] proposed similar definitions for the graph translation. They started from the classical definition of the translation being the convolution between a signal and a Kronecker delta in time. Using (2.19) they obtain similar definitions, up to a normalization factor $\sqrt{N}$, which we state in the following definition.

Definition 2.16: Graph translation.
Given a signal $\boldsymbol{f} \in \mathbb{R}^N$ and a graph $\mathcal{G}$, the translation on graph reads

$$T_i \boldsymbol{f}[n] = \sum_{\ell=0}^{N-1} \widehat{\boldsymbol{f}}[\ell] u_\ell[i] u_\ell[n]. \tag{2.26}$$

Note that this definition of translation is very close to the definition of the Graph Wavelet Transform [58]. Moreover, we notice that this operator is kernelized and although any signal cannot be localized around a predetermined vertex, it is the case for filters defined in the spectral domain under certain assumptions. Since localization is an interesting property, that we will be using in the following chapters, we extend the definition of graph translation and call $\mathcal{T}_i g$ the localization operator of the filter g around vertex i:

$$\mathcal{T}_i g[n] = \sum_{\ell=0}^{N-1} g(\lambda_\ell) u_\ell[i] u_\ell[n]. \tag{2.27}$$

An isometric alternative to the definition of graph translation has been proposed in [50] but this one does not possess the localization properties of filters.

2.1.6 Other approaches to GSP

The field of signal processing on graphs emerged simultaneously from different research groups in the recent years. Although, a large community agreed on the definitions stated above, there is an alternative concept called $\mathrm{DSP_G}$, mainly used by Moura and its co-authors [119, 88, 29], that deserves to be quickly introduced as well.

The alternative approach, just as the one presented above, tries to extend the concepts of traditional signal processing to graphs. Similarly to the works on Graph Signal Processing, they also presented the ring graph (a cycle graph) as a convenient structure on which to start the comparisons but aim at defining different properties with this structure. Their corner stone is the definition of the equivalent of a time delay, denoted by z^{-1} in the classical domain. For this specific application,

they define the graph shift as the shift of the signal on the vertices. Namely $\tilde{s}[n] = s[n-1]$ with a circulant application on the borders of the domain ensuring that $\tilde{s}[1] = s[N]$. They generalize this with the notation $\tilde{s} = \mathcal{A}s$ where $\mathcal{A}$ is the adjacency matrix of any graph $\mathcal{G}$.

From that point on, they are able to characterize the class of linear shift-invariant graph filters and prove that it corresponds to the polynomials in $\mathcal{A}$: $\mathbf{H} = h_0\mathbf{I} + h_1\mathcal{A} + \cdots + h_L\mathcal{A}^L$. The form of $\mathcal{A}$ being more general than that of $\mathbf{L}$, there is no guarantee for $\mathcal{A}$ to be diagonalizable and thus the spectral decomposition associated with $\mathcal{A}$ is $\mathbf{VJV^{-1}}$, its Jordan Normal Form (JNF). For detailed explanations about its determination, readers are referred to [138]. In practice, however, this decomposition is untractable because the form is unstable (e.g., a ε-difference on the input can produce a very different output).

Although the application might not seem as obvious as spectral clustering in the case of GSP, the methods detailed in the following section could be extended to this definition of signal processing on graphs allowing potentially the computation of fast JNF. Despite the numerous applications that have been proposed following the two different approaches, the lack of complexity analysis in the DSP_G community makes it difficult to summarize the generalization in this section. We refer the reader to the works on denoising [123, 29], stationarity [106, 88] and big data analysis [78, 120] for a deeper analysis of the frameworks allowing a clearer comparison between them.

2.2 Clustering

Clustering is an important problem in Artificial Intelligence (AI) that has long been studied and tries to solve the following problem: based on data, the goal is to find a categorization of the entries that assign together those with common properties and separate the dissimilar data points. It differs from classification in the sense that a ground truth is not required in order to classify the data points. This is advantageous because it avoids the need of experts attributing labels on the samples (or a subset of them) which can be very expensive and time consuming. Data processing is then easier to automatize.

The fact that many methods are proposed in the literature can be explained by a lack of rigorous problem definition. Many techniques share similar approaches but have different formulations. In particular, the definition of similarity, and thus the solution to the problem often depends on the application and the subjectivity of the criteria involved. Nevertheless, clustering is a famous field of research among the AI community, attracting even more interest than before with the data deluge. Famous works of clustering allowed the segmentation of pixels in images [122], objects recognition [79], information retrieval [68, 27], text recognition [26], data mining [14, 2], protein-protein interactions [143] and others. The basics and main contributions are highlighted in this section. For a deeper analysis of this large field, readers are referred to the surveys of Jain et al. [67], Fortunato [45] and Porter et al. [111] as well as the book of Gan et al. [47].

2.2.1 Data similarity

A first important problem is to define the measure of similarity between the elements that we consider. It is, on the one hand, the metric of success of the algorithm under the hypotheses applied to our data, and, on the other hand, directly connected to the algorithm that will be employed.

Vector space model

A common model is the vector space model. Its basic idea is that entries are defined with a set of features that depend on the application but are known. Each feature defines a new dimension and the elements of the dataset are individually rated on all the dimensions. If we consider a corpus of text documents, the dimensions could be the words that appear in the whole corpus and a given document would be represented by a vector counting the number of occurrences of each word in the document. The vector space approach allows the definition of similarity measures based on the inner product between different pieces of the dataset. The cosine similarity is defined as:

$$d_{i,j} = \cos(x_i, x_j) = \frac{\langle x_i, x_j \rangle}{\|x_i\| \, \|x_j\|}. \tag{2.28}$$

Very often though, the similarity is computed by a distance metric representing the dissimilarity between two entries. In such cases, the Euclidean distance between the vector space representation of the data is proposed because it is very intuitive to represent distances in 2 and 3 dimensions. It simply reads:

$$d_{i,j} = \|x_i - x_j\|_2 . \tag{2.29}$$

This metric can be generalized for any $p \geq 1$ as the Minkowski metric, that is computed with:

$$d_{i,j} = \left(\sum |x_i - x_j|^p\right)^{\frac{1}{p}} = \|x_i - x_j\|_p . \tag{2.30}$$

However, one needs to be able to control the importance of the different dimensions with such distances. It requires that they should all impact the similarity in the same way and that the values on each dimension are in the same range. Otherwise, a preprocessing step of normalization is necessary, depending on the external knowledge that allowed the feature selection. The most common approaches for that are range normalization where the minimum is mapped to 0 and the maximum to 1, rescaling all data points with:

$$\widehat{x_i} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \tag{2.31}$$

and variance normalization where the entries are rescaled such that the sample mean μ and the sample variance σ become respectively 0 and 1:

$$\widehat{x_i} = \frac{x_i - \mu}{\sigma}. \tag{2.32}$$

Linear correlations between the dimensions can also impact the importance of each, in which case the squared Mahalanobis distance can alleviate the phenomenon by computing the inverse sample covariance matrix for the normalization:

$$d_{i,j} = \left(x_i - x_j\right) \Sigma^{-1} \left(x_i - x_j\right)^\top . \tag{2.33}$$

This has been reported as preprocessing in [85] for hyperellipsoidal clustering where the density of the data points is assumed to be generated by a multivariate Gaussian.

Mixed data types

Sometimes, and increasingly more often with the combination of data sources, datasets are formed with features of mixed types. Namely, some of the variables are defined with continuous values (or evenly discretized) such as age, number of occurrences, pixel value, while others are categorical variables such as music genres, color names or professions. In those cases, the application of the vector space model is complicated. Indeed, by replacing the nominal attributes by consecutive integers, we could be able to represent colors such as *black, blue, red, green, orange* with integers from 1 to 5. However, we cannot interpret the distance between blue and black as half of the distance between red and orange. This does not make sense and would alter the distance between the two instances unwillingly.

A suggestion to accommodate this data into the former model is to consider an ℓ_2-distance for the continuous variables and an ℓ_0-distance for the categorical ones. Weighting between the different dimensions can be added in order to circumvent the problem of importance of the dimensions [61, 4]. In this regard, two pairs of entries being part of different categories are equally distant irrespectively of the given categories. As such, if some categories are more similar than others, this information is lost in the similarity measure. This distance is called Heterogeneous Euclidean-Overlap Metric (HEOM) in [139], DHET in [49] or also IBL in [3].

Another possibility is to consider the Cartesian space model [64] where specific rules for union and intersection are defined depending on the kind of feature to be considered. The distance between two realizations is finally computed for each feature as:

$$\psi_{i,j}^{(k)} = \left|x_i^{(k)} \boxplus x_j^{(k)}\right| - \left|x_i^{(k)} \boxtimes x_j^{(k)}\right| + \gamma\left(2\left|x_i^{(k)} \boxtimes x_j^{(k)}\right| - \left|x_i^{(k)}\right| - \left|x_j^{(k)}\right|\right), \tag{2.34}$$

and a generalized Minkowski distance of order $p \geq 1$ is used to take all the dimensions into account together:

$$d_{i,j} = \left(\sum_k {\psi_{i,j}^{(k)}}^p\right)^{\frac{1}{p}} . \tag{2.35}$$

Alternatively, information-theoretical measures like minimal description length served as the basis for clustering measures of mixed-type datasets [17].

Data transformation

Data normalization in the vector space model has already been discussed above. However, it is not the only common transformation that can be applied to the data in order to prepare for clustering applications.

Principal Components Analysis (PCA) tries to preserve the majority of the variance between the samples together with a reduction of the dimensionality. Towards this goal, the data is projected in a different space of same dimensionality where the variables are uncorrelated and ordered such that the first components enclose most of the variations. To do so, one needs to know the covariance matrix (or compute the sample covariance of the data) and solve an optimization problem that was shown to correspond to the eigendecomposition of the covariance matrix [70].

2.2.2 Clustering techniques

The family of clustering methods is very vast, following the various needs in the different applications. This section separates the present approaches into two kinds, the partitioning methods and the hierarchical clustering techniques. Hierarchical clustering produces nested partitions with a varying granularity, while partitioning is providing a single answer with a predetermined number of classes to separate.

Hierarchical clustering

Many hierarchical clustering methods are based on the principle of agglomerative clustering. Starting from N points belonging each to their own cluster, points are grouped by pairs following a defined similarity metric among those described above. The procedure continues until all the nodes belong to the same cluster. The successive merging generate a tree characterizing the similarity between the entities, which is called a dendrogram. An example is presented in Figure 2.2. In the end, the dendrogram can be cut at any level to provide a clustering with different numbers of clusters.

Researchers defined two famous rules for the association of separation of clusters independently of the similarity measure of choice. These are called single-linkage [127] and complete-linkage [73]. More specifically, in the first steps, when the clusters to merge contain a single node each, the similarity score is computed via a simple application of the similarity measure to the two elements. However, once the clusters contain more than one node, the similarity metric must be applied on a representative of each class. In the case of single-linkage, one selects the two nodes that minimize the metric, while the complete-linkage takes the two nodes that maximize the metric.

$$sl(c_1, c_2) = \min_{x \in c_1, y \in c_2} d(x, y) \qquad \text{and} \qquad cl(c_1, c_2) = \max_{x \in c_1, y \in c_2} d(x, y). \tag{2.36}$$

Single-linkage can be useful to recognize concentric clusters or various specific shapes that are elongated because it aggregates nodes that are closest to the cluster already in place. However, since complete-linkage algorithms produce more compact clusters, it often outperforms single-linkage by proposing more useful hierarchies [66].

The opposite approach to agglomerative clustering, called divisive clustering, starts with a single cluster containing all the nodes and split the N nodes iteratively into several clusters based on the same similarity measure than before. Both agglomerative and divisive approaches are identical in terms of result as they operate very similarly for the large majority of metrics.

Hierarchical clustering is more versatile than the partitioning solution, however, it has a spatial and computational complexity that prevents it from being used with large datasets. Indeed, all combinations of the approaches described for the hierarchical clustering are $\mathcal{O}\left(N^2 \log(N)\right)$ in time and $\mathcal{O}\left(N^2\right)$ in space because it requires the computation of the full matrix of $N \times N$ entries with the similarity between all the pairs.

Partitioning

Partitioning, on its side, is usually cheaper to construct because the number of classes is predefined and one only needs to provide a partition for this level of detail. However, such algorithms always assume that the number of classes k is given as input, which is an important drawback when

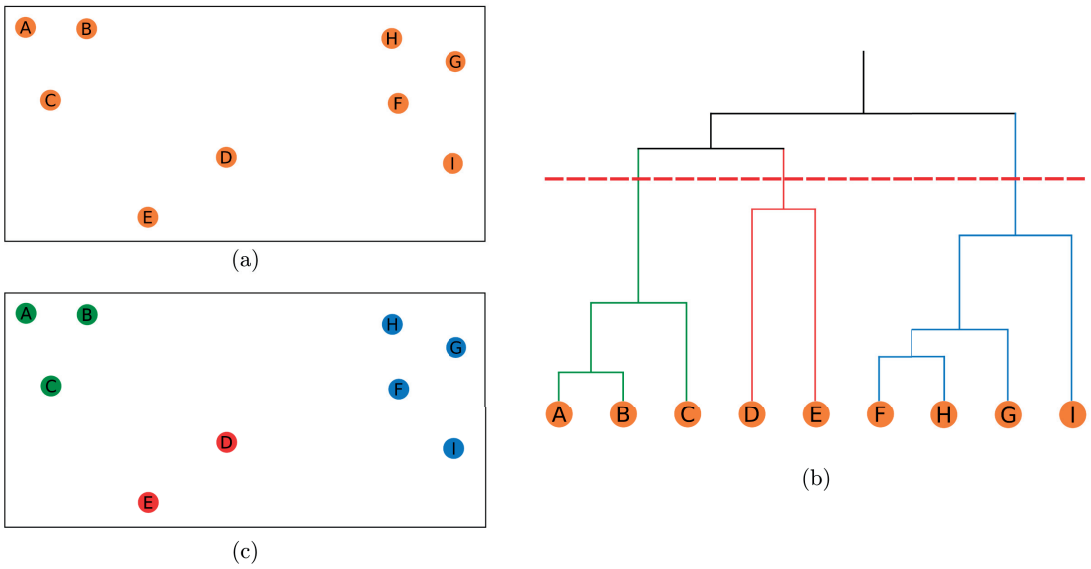


Figure 2.2 – Dendrogram representation of a dataset. (a) represents a toy example of 9 nodes in $\mathbb{R}^2$. (b) is the dendrogram linked to the example displayed in (a). The nodes are connected following the single-linkage association rule and the measure to connect two entities is the $\ell 2$-measure. For the example, a layer is chosen randomly for the separation of the nodes. (c) is the resulting clustering based on this cut.

we do not have any information about the dataset. When it is not possible to do otherwise, the algorithm will run several times with a different number of clusters in input picked among all the values ranging in a plausible search space that is defined by the user. Obviously, this is not efficient.

Here again, the favored metric for the similarity is the Euclidean norm in the vector space model because it is the most intuitive representation of distances. Thus, k-means [81, 91], the most famous method of partitioning is also the simplest of all. For k classes $j = 1, \ldots, k$, containing each n_j nodes, the objective function to minimize is

$$\sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\| x_i^{(j)} - c_j \right\|_2^2, \tag{2.37}$$

where $x_i^{(j)}$ is the ith element of the class j and c_j is the center of the jth cluster.

The algorithm starts with an initial assignment of k centers and iteratively adapts the partitions with two steps until convergence (either no modification of the assignment or a gain of the objective function smaller than a predefined threshold):

1. Assign each node to the class whose center is the closest in ℓ_2-norm
2. Recompute the position of the center for all the classes once a new assignment has been defined

The initial positioning (seeding) of the centers has been an active field of research for some time because the result of k-means is very sensitive to its initialization. The seeding mechanism proposed simultaneously and similarly in [9] and [100] has become the new standard since it provided naturally faster solutions (reducing the number of iterations) and more optimal with respect to the objective function. Overall this approach has a complexity of $\mathcal{O}(Nkdi)$ for d-dimensional data and where i is the number of iterations.

Many variants and generalizations of k-means were proposed to solve similar problems. The most famous are k-medians [66] with the mean being replaced by the median in the objective function, k-medoids [71] where the centers are selected among the data points and centers are recomputed as the closest point to the central position in the cluster, and fuzzy c-means [41, 15] when the partitioning is not binary anymore but each element has a probability of belonging to all of the classes that depend on its neighborhood.

Partitioning algorithms are further split into two categories: deterministic and stochastic methods. The methods presented so far are deterministic, except the variations of k-means using a probabilistic seeding. However, evolutionary algorithms such as genetic algorithms [54], simulated annealing [74, 75], or Tabu search [53, 5] also proved their usefulness for clustering purposes. Since the deterministic approaches are most often greedily optimizing the objective function, it might converge to a local optimum instead of the global one. These stochastic approaches

consider worst intermediate solutions with positive probability in order to escape from the nearest optimum.

2.2.3 Graph-based approaches and metrics

Network science brought the need of clustering graphs, which might be the only information available about a dataset. Fortunately, a graph is always related to an adjacency matrix that can be considered as the similarity matrix between the entities (the higher the weight in the adjacency matrix, the higher the similarity between two nodes). Thus the previously presented methods could theoretically be applied on such datasets as well. However, the determination of the distance between each pair of nodes is not contained in the graph for all pairs. Inferring the distance between the pairs of nodes that are not connected is an important effort. One way to solve this requires to infer a surface representing the dataset on which the nodes could be placed, based on the known distances. However, such approach is computationally very expensive.

Thus, researchers focused on graph clustering independently, trying to achieve partitioning of the nodes that reflects the structure of the graph. Namely, the goal here is to split the nodes such that few edges cross the boundaries of the classes and that the most similar nodes stay together. The methods that will be presented later all assume that graphs are sparse, i.e., the average degree is constant, $M = \mathcal{O}(N)$. This is simultaneously what makes the methods of data clustering inconvenient (due to the lack of lots of entries in the similarity matrix) and what allows the partitioning to separate the components with few connections. Otherwise, the graph clusterability would be questionable. When the number of edges becomes large, the diversity of weights could help preserving a community structure.

In order to get a good overview of graph clustering, different famous approaches are presented succinctly in the next section. These methods are structured around two criteria: either applying the framework of data clustering on graphs or using the properties of the network following graph theory principles.

These two directions allowed the development of methods achieving high quality assignments. An old method of graph clustering due to Kernighan and Lin [72], more recently turned into a post-processing heuristic, is used in practice to refine the output assignment of the most efficient algorithms. Its idea is to operate a constant number of swap operations between the partitions at each step, in order to optimize an objective function **Q**. Swapping must involve the same number of nodes in both communities (potentially only one) and some of the swaps decreasing the objective function are also added to the procedure to avoid local optimization. Since the result depends mostly on the initial partition, its use has been mostly interesting as a post-processing operation. However, the original work proposed this method with a running time of $\mathcal{O}\left(N^2 \log(N)\right)$ for the bisection problem with generalization increasing quickly with the number of classes. Thus, the idea of swapping nodes between communities was later included directly in some of the methods that we will present now in order to propose a more efficient approach.

2.2.4 Graph hierarchical methods

Edge-betweenness centrality

The hierarchical scheme defined above (Sec. 2.2.2) for data clustering can also be applied to graph clustering. Girvan and Newman [51, 97] proposed a divisive method, where the purpose is to identify the links between different communities and to remove these first. In this regard, they suggested the use of betweenness centrality measures to identify the weakest edges of the original graph. Edge-betweenness centrality is a measure that associates a scalar to an edge and assumes that one knows the shortest paths between all pairs of vertices in advance. This centrality measure counts the number of times an edge e is on the shortest path between u and v.

From the original graph, the algorithm starts by computing the edge betweenness centrality for all edges. This is already computationally expensive, the all-pairs shortest path being computed in $\mathcal{O}(NM)$ following the methods based on BFS proposed in [94, 21]. Then, the algorithm continues until all nodes are disconnected by alternating two steps:

1. Remove the edge whose edge centrality is maximum
2. Recompute the edge centrality of all edges in the new graph

Intuitively, the edge with maximal centrality is an edge that belongs to the shortest paths between vertices in different communities besides the intra-cluster connections that can be considered to be roughly the same for all edges at the very beginning. The creation of the full dendrogram and the different partitions of this hierarchical method has an overall cost of $\mathcal{O}\left(M^2N\right)$ operations in worst-case analysis. Some graphs with high clusterability, where the first removed edges disconnect the graph and produce several components, can benefit from an accelerated determination of

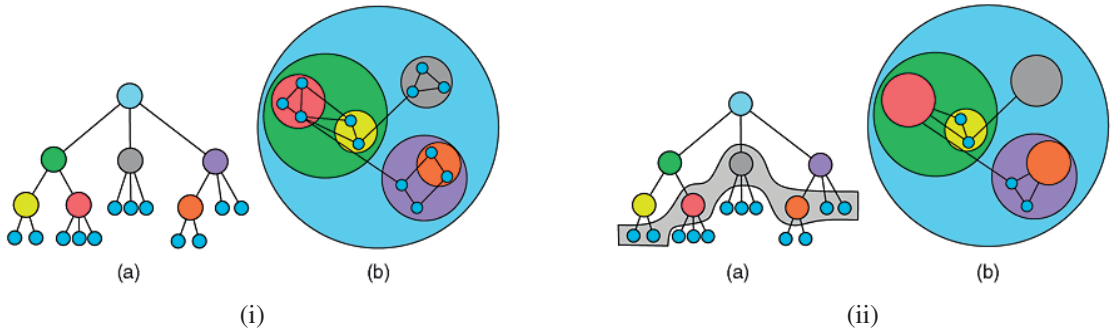


Figure 2.3 – An example of hierarchical clustering from [7]. (i) shows a hierarchical representation of the graph clustering, with the tree representation on the left and the nested assignments on the right. (ii) depicts a clustering for the same graph based on the hierarchy. Each nodes are selected either directly or via a node representative, higher in the tree, on the left in gray. The corresponding assignment and its nested upper layers are represented on the right.
Reproduced with the kind approval of the first author. ©2008 IEEE.

the hierarchy because the number of pairs to consider in the edge-betweenness computations is quickly reduced. However, there is no estimation of the gain in terms of complexity so far.

These works are important because they make a connection for the first time between community detection and statistical physics, which generated an important interest from this community and that of mathematics.

Modularity

The authors of the previous approach are also the inventors of a famous measure in graph clustering. Initially defined as a quality measure to determine which of the partitions defined in the dendrogram of the previously introduced method is the best one [97], the notion of *modularity* quickly became unavoidable.

Modularity is based on the concept that random graphs are not expected to have community structure and thus the structure only depends on the difference between the wiring of the actual graph and that of a random graph possessing the same characteristics, a null-model. The generic form for the modularity is then

$$Q = \frac{1}{\mathrm{Tr}\,\mathbf{D}} \sum_{i,j} (\mathbf{W}_{i,j} - P_{i,j}) \delta(C_i, C_j), \tag{2.38}$$

where $P_{i,j}$ is the expected weight between vertices i and j in the null model. Although several hypotheses have been proposed, the standard null-model takes into account the degree sequence of the nodes in the random connection process, thus $P_{i,j} = \frac{d_i d_j}{\mathrm{Tr}\,\mathbf{D}}$ for weighted graphs.

Despite its common usage as a quality measure, both for partition determination and for assignment comparison, modularity also serves as an objective function in several algorithms of community detection grouped into three categories: the greedy approaches detailed below that are approximations allowing the analysis of large datasets, the probabilistic procedures such as simulated annealing similarly to data clustering [56, 90, 92] that are extremely demanding in computations and extremal optimization heuristics applied to modularity optimization in [40] that produce good trade-offs between the two other kinds.

Newman [95] elaborated an agglomerative method where he computed at each step the modularity gain (or loss) – i.e., assuming that two clusters are merged ΔQ represents the variation between the previous modularity score and the new one. By greedily selecting the connections that maximize ΔQ at each step (potentially a negative value), the author obtained a dendrogram that can be cut at the level maximizing Q over all possibilities. This procedure costs $\mathcal{O}(MN)$ operations, and several improvements were proposed later. Clauset et al. [34] used max-heaps to reduce the complexity to $\mathcal{O}\big(Mh\log(N)\big)$ with h the height of the dendrogram, while others proposed to multiply the modularity gain with a consolidation ratio to avoid unbalanced communities [136] or to allow several pairs to be merged for each iteration [121]. Blondel et al. [16] suggested the

combination of several clusters into one at each step by associating vertex i to its neighbor's community that would produce the largest increase in modularity (as long as it is positive). Once there is no possible increase, a new supergraph is constructed and nodes associated to the same cluster are considered to be part of the same vertex. The process is repeated until there is no more merging. This procedure produces a dendrogram were nodes are not necessarily (and are most often not at all) connected by pairs. The height of this dendrogram is thus usually small compared to all the previously stated methods. The overall complexity of this method is claimed to be $\mathcal{O}\left(N\log(N)\right)$.[2]

Clustering coefficients

The definition of global clustering coefficients [60], defined to characterize graphs with a potential for partitioning by computing the number of triangles in the graph compared to the potential of triangles based on connected triplets, and the introduction of its local counterpart for network analysis [137] ultimately led Radicchi et al. [114] to propose a local metric for divisive hierarchical graph clustering. They define the edge-clustering coefficient where they focus on edges instead and measure the proportion of triangles formed with an edge with respect to the potential of this edge (based on the degree of its ends). Since many triangles exist inside classes while this is not often the case between them, edges with a small edge-clustering coefficient are prone to represent intercluster edges. Their divisive algorithm thus simply remove the edge with the smallest value at each step until complete disconnection of the graph. This is indeed a local measure since only the edges adjacent to the removed one need to be recomputed after each step. This allows the generation of a method faster than the previous ones, empirically compared against that of Girvan and Newman [51].

2.2.5 Graph partitioning

Clique percolation

Based on the observation that nodes in a community tend to be formed of small k-cliques (groups of k nodes connected to each of the $k-1$ other members) that share many of these k vertices with other cliques, the method of k-clique percolation arose [37, 101]. A k-clique community is composed of the union of all adjacent k-cliques. Two k-cliques are adjacent in the formal definition if they share $k-1$ of its nodes, but later extensions with only $k-\ell$ nodes in common were introduced. Assuming k is set, this method extracts all k-clique communities and provides them as an assignment. Two important points need to be made. First, note that setting k does not provide the number of clusters but the connectedness of the partitions and that not all the nodes will belong to communities depending on the value of k while some other nodes will belong to several communities. Second, the values of k should be chosen between 3 and 6 usually.

[2] this information comes from the authors' website, while they claim that the exact complexity is not known but that simulations tend to show a behavior close to linear in the number of edges in the paper.

Indeed, for $k = 2$, it corresponds exactly to the task of components extraction which is not going to provide insightful information about its communities. On the opposite, setting k too large will prevent the method from detecting communities efficiently. A potential exception or extension of this method concerns hypergraphs where the hyperedges could be considered as cliques between the nodes in an equivalent representation as a graph and where a higher value of k might produce relevant results. To date, this technique is among the most promising method for overlapping community detection even if its complexity remains prohibitive for large datasets.[3]

Spectral clustering

Spectral clustering [98, 122] (SC) is a standard algorithm for graph clustering. To determine the best node-to-cluster assignment, spectral clustering entails solving a k-means problem, with the eigenvectors of the graph Laplacian $\mathbf{L}$ as features.

Since the objective of graph clustering is to partition the graph into classes where well-connected nodes stay together, it appeared obvious to define the Cut function as a measure of quality of the partitioning:

$$\mathrm{Cut}(\mathrm{S}, \overline{\mathrm{S}}) = \sum_{\substack{i \in S, \\ j \in \overline{S}}} \mathbf{W}_{\mathrm{i,j}}. \tag{2.39}$$

However, optimizing this quantity was not producing meaningful results when there were some isolated nodes. Indeed, in such situation, it would be optimal to assign all the nodes to the same cluster but the isolated ones. Thus, a normalization was necessary to establish a trade-off between the size of the partitions and the number of edges in the cut. Volume and associativity were proposed to normalize the cut problem:

$$\mathrm{Vol}(\mathrm{S}) = \sum_{\substack{i \in S, \\ j \in \mathcal{V}}} \mathbf{W}_{\mathrm{i,j}} \qquad \text{and} \qquad \mathrm{Assoc}(\mathrm{S}) = \sum_{\substack{i \in S, \\ j \in S}} \mathbf{W}_{\mathrm{i,j}}. \tag{2.40}$$

The objective function thus became the normalized cut problem, defined as

$$\min_{\{S_i\}_{i=1}^k} \sum_{i=1}^{k} \frac{\mathrm{Cut}(\mathrm{S_i}, \mathrm{S_i^c})}{\mathrm{Vol}(\mathrm{S_i})} = \min_{\{S_i\}_{i=1}^k} \sum_{i=1}^{k} \frac{\mathrm{Vol}(\mathrm{S_i}) - \mathrm{Assoc}(\mathrm{S_i})}{\mathrm{Vol}(\mathrm{S_i})} = \max_{\{S_i\}_{i=1}^k} \sum_{i=1}^{k} \frac{\mathrm{Assoc}(\mathrm{S_i})}{\mathrm{Vol}(\mathrm{S_i})} - k. \tag{2.41}$$

It follows directly from the relaxation of this objective for k classes that the k eigenvectors associated with the smallest eigenvalues of $\mathbf{L}$ form the optimal solution.

[3] The authors present an experiment in [101] where their best fit is the function $t = aM^{b\log(M)}$ for some constants a and b.

Indeed, if we rewrite this optimization problem in matrix form, we obtain

$$\sum_{i=1}^{k} \frac{\text{Assoc}(S_i)}{\text{Vol}(S_i)} = \sum_{i=1}^{k} \frac{{Y_{\cdot i}}^{\top} \mathbf{W} Y_{\cdot i}}{{Y_{\cdot i}}^{\top} \mathbf{D} Y_{\cdot i}} = \sum_{i=1}^{k} \frac{{Y_{\cdot i}}^{\top}}{\left({Y_{\cdot i}}^{\top} \mathbf{D} Y_{\cdot i}\right)^{\frac{1}{2}}} \mathbf{W} \frac{Y_{\cdot i}}{\left({Y_{\cdot i}}^{\top} \mathbf{D} Y_{\cdot i}\right)^{\frac{1}{2}}}, \tag{2.42}$$

where $Y \in [0,1]^{N \times k}$ is the assignment matrix ($Y_{i,j} = 1$ if and only if node i belongs to cluster j).

Note that $Y^{\top}\mathbf{D}Y$ is a diagonal matrix and that $\left(Y^{\top}\mathbf{D}Y\right)^{-\frac{1}{2}} Y^{\top}\mathbf{D}Y \left(Y^{\top}\mathbf{D}Y\right)^{-\frac{1}{2}} = \mathbf{I}_k$. Then, we can show that eq. (2.42) equals

$$\text{Tr}\left(\left(Y^{\top}\mathbf{D}Y\right)^{-\frac{1}{2}} Y^{\top}\mathbf{W}Y \left(Y^{\top}\mathbf{D}Y\right)^{-\frac{1}{2}}\right) \qquad \text{such that} \quad \left(Y^{\top}\mathbf{D}Y\right)^{-\frac{1}{2}} Y^{\top}\mathbf{D}Y \left(Y^{\top}\mathbf{D}Y\right)^{-\frac{1}{2}} = \mathbf{I}_k. \tag{2.43}$$

Introducing $Z = \mathbf{D}^{1/2} Y \left(Y^{\top}\mathbf{D}Y\right)^{-\frac{1}{2}}$, this becomes equivalent to

$$\max_{Z} \text{Tr}\left(Z^{\top}\mathbf{D}^{-1/2} W \mathbf{D}^{-1/2} Z\right) \quad \text{such that} \quad Z^{\top}Z = \mathbf{I}_k, \tag{2.44}$$

whose solution is the same than

$$\min_{Z} \text{Tr}\left(Z^{\top}\mathbf{L}Z\right) \quad \text{such that} \quad Z^{\top}Z = \mathbf{I}_k. \tag{2.45}$$

Spectral clustering consists thus of computing the first k eigenvectors of $\mathbf{L}$ arranged in a matrix called $\mathbf{U}_k$ (i.e., $Z = \mathbf{U}_k$) and subsequently computing a k-means assignment of the N vectors of size k found in the rows of $\mathbf{U}_k$. Formally, if $\Phi \in \mathbb{R}^{N \times d}$ is the feature matrix (here $\Phi = \mathbf{U}_k$ and $d = k$), and k is a positive integer denoting the number of clusters, the k-means clustering problem finds the indicator matrix $\mathbf{X} \in \mathbb{R}^{N \times k}$ which satisfies

$$\mathbf{X}_{\Phi} = \underset{\mathbf{X} \in \mathcal{X}}{\arg\min} \|\Phi - \mathbf{X}\mathbf{X}^{\top}\Phi\|_F, \tag{2.46}$$

with associated cost $C_{\Phi} = \|\Phi - \mathbf{X}_{\Phi}\mathbf{X}_{\Phi}^{T}\Phi\|_F$. The symbol $\mathcal{X}$ denotes the set of all $N \times k$ indicator matrices $\mathbf{X}$. These matrices indicate the cluster membership of each data point by setting

$$\mathbf{X}_{i,j} = \begin{cases} \frac{1}{\sqrt{s_j}} & \text{if data point } i \text{ belongs to cluster } j \\ 0 & \text{otherwise} \end{cases}, \tag{2.47}$$

where s_j is the size of cluster j, and is also equal to the number of non-zero elements in column j.

The complexity of SC is its most important weakness since it requires the computation of the eigendecomposition of the graph Laplacian which is $\mathcal{O}\left(N^3\right)$ in general for the complete eigendecomposition and $\mathcal{O}\left(kN^2\right)$ for an approximation of the first k eigenvectors. As we will see later, we will be using the methods presented in Sec. 2.1 in order to accelerate SC.

Note that the cost described in eq. (2.46) is the square root of the more traditional definition

expressed with the distances to the cluster centers [35, Sec 2.3]. In the following of the thesis, we are interested in expressing the quality of an assignment with respect to spectral clustering. Towards this goal, we will express the cost of the optimal assignment of k-means using different types of features and compare them to discuss the quality of the clustering that we obtain. We refer the reader to the work by Boutsidis et al. [20] and its references for more details.

Other spectral methods

Eigendecomposition of other matrices was proposed while trying to optimize different objectives. Among them is the modularity matrix $\boldsymbol{B}$ whose elements are $\boldsymbol{B}_{i,j} = \mathbf{W}_{i,j} - \frac{d_i d_j}{2m}$. In the case of bipartitioning, we call the assignment vector $\boldsymbol{s}$ and assign it values in $\{-1, +1\}$. Modularity then reads:

$$Q = \frac{1}{4m} \boldsymbol{s}^\top \boldsymbol{B} \boldsymbol{s}. \tag{2.48}$$

The positive eigenvectors of $\boldsymbol{B}$ (i.e., those associated with positive eigenvalues) provide an assignment optimizing the modularity. Exactly like the previous method based on $\mathbf{L}$, a data matrix of size $N \times p$ is constructed from the positive eigenvectors and each node is associated with the corresponding row in this matrix. Then, it can be shown that merging nodes together increase the modularity if and only if the p-dimensional vectors form an angle smaller than $\frac{\pi}{2}$. In this case, the community is represented as a vector formed by the sum of its components [96].

On top of that, spectral approximations were introduced. Their goal is to improve the complexity of the spectral approaches with a minimal loss of quality. Fowlkes et al. [46] used an extension of the Nyström method. Instead of approximating an integral equation, they evaluate it in several points and compute a quadrature rule to derive the values for the rest of the elements. This still requires the computation of the eigendecomposition of a submatrix of smaller size. Time and space refinements of this method were proposed by computing directly a rank-k approximation of the submatrix formed by the column sampling [80]. These methods were often compared to the Implicitly Restarted Lanczos Method [23] that also produces an approximative computation of the first eigenvectors. This is what is used in the ARPACK framework used in several programming languages such as Matlab or Scipy (a python library for linear algebra).

From data to graphs

In the context of this thesis, I will be focusing on simple undirected graphs. I will distinguish two situations: when the data provides a network and we study this network, or when we must construct a graph out of the data in order to analyze the underlying properties of the dataset. In the latter case, I will always use k-NN graphs for the construction of the graph. Instead of using the full distance matrix between each pair of nodes, we will consider only the similarity between those that have the most in common. This procedure ensures the efficiency of graph clustering methods for use in data clustering domains.

The graph construction can be a computationally expensive operation since it requires the computation of all the $\frac{N(N-1)}{2}$ pairwise distances. Since we want our methods to be efficient and applicable to large graphs, we do not consider starting with $\mathcal{O}\left(N^2\right)$ operations for the initialization of the problem to solve. We thus choose to construct approximated k-NN graphs based on kd-trees as presented in [93]. This implementation generates the graph in $\mathcal{O}\left(N \log N\right)$, which is already significant compared to the cost of the methods presented in the following chapters.

3 Fast Approximation of Spectral Clustering for Dynamic Networks

3.1 Introduction

Spectral clustering (SC) is one of the most well-known methods for clustering multivariate data, with numerous applications in biology (e.g., protein-protein interactions, gene co-expression) and social sciences (e.g., call graphs, political study) among others (see Sec. 2.2.5). However, because of its inherent dependence on the spectrum of some large graph, SC is computationally expensive. We recall that clustering a graph takes $\mathcal{O}\left(N^3\right)$ operations if a full eigendecomposition is performed and $\mathcal{O}\left(kN^2\right)$ if the Lanczos method is used [11]. This has motivated a surge of research focusing in reducing its complexity, for example using matrix sketching methods [46, 80, 52] and more recently using compressive sampling techniques [115, 132], attaining a complexity reduction by almost a factor of N on sparse graphs.

Yet, computation is still problematic for dynamic graphs, where the edge set is a function of time. Temporal dynamics constitute an important aspect of many network datasets and should be taken into account in the algorithmic design and analysis. Unfortunately, SC is poorly suited to this setting as eigendecomposition –its main computational bottleneck– has to be recomputed from scratch whenever the graph is updated, or at least periodically [99]. This is a missed opportunity since the clustering assignments of many real networks change slowly with time, suggesting that successive algorithmic runs wastefully repeat similar computations.

Motivated by this observation, this chapter proposes an algorithm that reuses information of past cluster assignments to expedite computation. Different from previous work on dynamic clustering, our objective is *not* to maximize the clustering quality at all cost, for example by enforcing a temporal-smoothness hypothesis [24, 31] or by using tensor decompositions with one mode for the temporality [48, 133]. Similar to recent work by [38], we focus entirely on reducing the complexity while producing assignments that are provably close to those of SC.

Our work is inspired by the recent idea of sidestepping eigendecomposition by utilizing as features random signals that have been filtered over the graph [132]. Our main argument is

that, instead of computing the clustering assignment of a graph G_1 using d filtered signals as features, one may utilize a percentage of features of a different graph G_2 without significant loss in accuracy, as long as G_1 and G_2 are appropriately close. This leads to a natural clustering scheme for time-varying topologies: each new instance of the dynamic graph is clustered using pd signals computed previously and only $(1-p)d$ new filtered signals, where p is a percentage. Moreover, inspired by similar ideas we can also attain further complexity reductions with respect to the graph filter design, i.e., by identifying the k-th eigenvalue.

Concretely, we provide the following contributions:

1. In Section 3.3 we refine the analysis of compressive spectral clustering (CSC) presented in [132]. Our goal is to move from assertions about distance preservation (previously known) to guarantees about the quality of the solution of CSC itself. We prove that with high probability, the quality of the clustering assignments of CSC and SC differ by $\mathcal{O}\left(k/\sqrt{d}\right)$, and thus $d \propto k^2$ filtered vectors are sufficient to obtain a good approximation. Importantly, our analysis does not make restricting assumptions about the graph structure, such as assuming a stochastic block model [113].

2. In Section 3.5, we focus on dynamic graphs and propose dynamic CSC (dCSC), an algorithm that reuses information of past cluster assignments to expedite computation. We discover that the algorithm's ability to reuse features is inherently determined by a metric of spectral similarity ρ between consecutive graphs. Indeed, we prove that, when pd features are reused (i.e., each new instance of the dynamic graph is clustered using pd features of the previous graph and $(1-p)d$ new features, where $0 < p \leq 0.5$), with high probability the clustering quality of dCSC approximates that of CSC up to an additive term in the order of $p\rho$. We also connect the approximation error to the more practical similarity measure of edge difference between consecutive graphs.

3. We complement our analysis with a numerical evaluation in Sections 3.4 and 3.6. Our experiments first study the static version of CSC and the tightness of our theoretical bound in practice with synthetic examples. Then, it confirms that dCSC yields computational benefits when the graph dynamics are bounded, mainly due to the reusability of filters, while producing assignments with quality closely approximating that of SC. A case in point, we show that we can cluster 30'000 node graphs 3.9 times faster than SC and 1.5 times faster than CSC in average. Similar trends are also confirmed in a large-scale Arxiv citation dataset.

3.2 Compressive spectral clustering (CSC)

To reduce the $\mathcal{O}\left(N^3\right)$ cost of (naive) SC, Tremblay et al. [132] proposed to approximate $\mathbf{U}_k$ using a filtering of random signals (a similar idea was also examined by [20]). The former work also introduced the benefits of compressive sampling techniques reducing the total cost down to $\mathcal{O}\left(k^2\log^2(k) + cN(\log(N)+k)\right)$, where c is the order of the polynomial approximation. Their algorithm, summarized in Algorithm 1, consists of two concepts.

Step 1. Approximate features.

The targeted feature matrix $\Phi = \mathbf{U}_k$ being costly to compute, one can instead approximate it with the projection of random signals over the same subspace. In particular, let $\mathbf{R} \in \mathbb{R}^{N\times d} = (\mathbf{r_1}|\mathbf{r_2}|\ldots|\mathbf{r_d})$ be a random Gaussian matrix with centered i.i.d. entries, each having variance $\frac{1}{d}$. We can project $\mathbf{R}$ onto span$\{\mathbf{U}_k\}$ by filtering each one of its columns by a low-pass graph filter $g(\mathbf{L}) = \mathbf{H}$ defined as

$$\mathbf{H} = \mathbf{U}\begin{pmatrix} \mathbf{I}_k & 0 \\ 0 & 0 \end{pmatrix}\mathbf{U}^\top. \tag{3.1}$$

We thus expect to produce the ideal low-pass filter

$$g_{\lambda_k}(\ell) = \begin{cases} 1, & \ell \le \lambda_k \\ 0, & \ell > \lambda_k \end{cases} \tag{3.2}$$

It is then a simple consequence of the Johnshon-Lindenstrauss lemma that the rows $\boldsymbol{\psi}_i^\top$ of matrix $\Psi = \mathbf{HR}$ can act as a replacement of the features used in spectral clustering, i.e., the rows $\boldsymbol{\phi}_i^\top$ of $\Phi = \mathbf{U}_k$.

Theorem 3.1: Compressive Spectral Clustering, adapted from [132].
For every two nodes v_i and v_j the restricted isometry relation

$$(1-\varepsilon)\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 \le \left\|\boldsymbol{\psi}_i - \boldsymbol{\psi}_j\right\|_2 \le (1+\varepsilon)\left\|\boldsymbol{\phi}_i - \boldsymbol{\phi}_j\right\|_2 \tag{3.3}$$

holds with probability larger than $1 - N^{-\beta}$, as long as the dimension is

$$d > \frac{4+2\beta}{\varepsilon^2/2 - \varepsilon^3/3}\log(N). \tag{3.4}$$

We note that, even though Ψ is also expensive to compute, it can be approximated in $O(Mdc)$ number of operations using Chebyshev polynomials [123, 58] or rational graph filters [65], resulting in a small additive error that decreases with the polynomial order c.

Algorithm 1 Compressive Spectral Clustering from [132]

Input: $\mathcal{G}, d, k$

Output: $\mathbf{X}$

1: Determine h_k the filter approximation for $\mathcal{G}$ using bisection of eigencount
2: Generate d i.i.d. random signals $r_i \in \mathbb{R}^N$
3: Compute Ψ the filtering of the random signals with the filter h_k
4: Select a subset of the nodes at random and apply k-means to partition them
5: Interpolate the assignment for the full graph using graph filtering of the partial assignment

2. Compressive k-means.

The complexity is reduced further by computing the k-means step for only a subset of the nodes. The remaining cluster assignments are then inferred by solving a Tikhonov regularized interpolation problem involving k additional graph filtering operations, each with a cost linear in cm.

To guarantee a good approximation, it is sufficient to compute k-means on $\mathcal{O}\left(\nu_k^2 \log(k)\right)$ nodes uniformly at random, where $\nu_k = \sqrt{N} \max_i \left\|\boldsymbol{\phi}_i\right\|_2$ is the global cumulative coherence. However as shown by Puy et al. [112, Cor. 2.3 and Thm. 4.1], it is always sufficient to sample $\mathcal{O}\left(k \log(k)\right)$ nodes using variable density sampling where nodes are selected following the distribution $p_i = \frac{\left\|\mathbf{U}_k^\top \delta_i\right\|_2^2}{k}$.

For simplicity, in the following, we present our theoretical results with respect to the non-compressed version of their algorithm. The proofs can be generalized using identical arguments as in [112].

3.3 The approximation quality of static CSC

Before delving to the dynamic setting, we refine the analysis of CSC. Our objective is to move from assertions about distance preservation currently known (see Thm. 3.1) to guarantees about the quality of the solution of CSC itself. Formally, let

$$\mathbf{X}_\Psi = \underset{\mathbf{X} \in \mathcal{X}}{\arg\min} \left\|\Psi - \mathbf{X}\mathbf{X}^\top \Psi\right\|_F, \tag{3.5}$$

be the clustering assignment obtained from using k-means with Ψ as features (CSC assignment), where $\mathcal{X}$ is the set of assignment matrices as defined in eq. (2.47), and define the *CSC cost* C_Ψ as

$$C_\Psi = \left\|\Phi - \mathbf{X}_\Psi \mathbf{X}_\Psi^\top \Phi\right\|_F. \tag{3.6}$$

The question we ask is: *how close is C_Ψ to the cost C_Φ of the same problem, where the assignment has been computed using Φ as features, i.e., the SC cost corresponding to eq.* (2.46)*?* Note that, as in previous work [20], we express the approximation quality in terms of the difference of

clustering assignment costs and not of the distance between the assignments themselves. We are not aware of any analysis that would allow us to characterize (the perhaps more intuitive goal of) how well $\mathbf{X}_\Psi$ approximates $\mathbf{X}_\Phi$, which is a combinatorial objective. Yet, our approach exhibits the benefit of not penalizing approximation algorithms that choose alternative assignments of the same or similar quality.[1]

This section is devoted to the analysis of the quality of the assignments delivered by CSC compared to those of SC for the same graph. Our central theorem, stated below, asserts that with high probability the two costs are close.

Theorem 3.2.
The SC cost C_Φ and the CSC cost C_Ψ are related, with probability at least $1-\exp(-t^2/2)$, by

$$C_\Phi \leq C_\Psi \leq C_\Phi + 2\sqrt{\frac{k}{d}}(\sqrt{k}+t). \tag{3.7}$$

The result above emphasizes the importance of the number of filtered signals d and directly links it to the distance with the optimal assignment for the spectral features. Indeed, one can see that the difference between the two costs vanishes when d is sufficiently large. Importantly, setting $d \propto k^2$ guarantees a small error.

We propose as a side note a multiplicative expression for the error term. Since C_Φ^2 is the cost of the optimal k-means, we can write it as

$$C_\Phi = \sqrt{\sum_{c:\text{classes}} \sum_{x\in c} \left\| f_x - \mu_c \right\|_2^2} \geq \sqrt{k \min_c \sum_{x\in c} \left\| f_x - \mu_c \right\|_2^2} = \sqrt{ks}, \tag{3.8}$$

where s is the minimal spread of the data points in a class and k the number of classes. Using this fact, we can rewrite the result of Thm. 3.2 as follows

$$C_\Psi \leq C_\Phi \left(1 + \frac{2(\sqrt{k}+t)}{\sqrt{sd}}\right). \tag{3.9}$$

We remark here again that the error can be set arbitrarily small with increasing values of d but also that well clusterable graphs (with small spread s) are harder to bound. Keeping in mind that the complexity of CSC is $\mathcal{O}\left(k^2\log^2(k) + cN(\log(N)+k)\right)$, we see that our result implies that CSC is particularly suitable when the number of desired cluster is small, i.e., when k is $\mathcal{O}\left(\sqrt{N}\right)$.

3.3.1 The approximation quality of CSC

The first step in proving Thm. 3.2 is to establish the relation between C_Φ and C_Ψ. The following lemma relates the two costs by an additive error term that depends on the feature's differences

[1] The k-means objective is a non-convex objective and has multiple minima. For instance, any of the $k!$ relabelings of the optimal assignment are valid solutions with the same cost.

$\|\Psi - \Phi \mathbf{I}_{k\times d}\mathbf{Q}\|_F$.[2] Since Φ and Ψ have different sizes we introduced the multiplication by a unitary matrix $\mathbf{Q}$. We will first show that any unitary $\mathbf{Q}$ can be picked in Lem. 3.3 and then derive the optimal $\mathbf{Q}$, the one minimizing the additive term, in Thm. 3.4.

Lemma 3.3.
For any unitary matrix $\mathbf{Q} \in \mathbb{R}^{d\times d}$, the SC cost C_Φ and the CSC cost C_Ψ are related by

$$C_\Phi \leq C_\Psi \leq C_\Phi + 2\,\|\Psi - \Phi \mathbf{I}_{k\times d}\mathbf{Q}\|_F\,, \tag{3.10}$$

where, the matrix $\mathbf{I}_{\ell\times m}$ of size $\ell \times m$ above contains only ones on its diagonal and serves to resize matrices.

Being able to show that the additive term is small encompasses the result of Thm. 3.1, ensuring distance preservation. However, this statement is stronger than the previous one as our lemma is not necessarily true under distance preservation only.

Proof (Lemma 3.3)**.**
Let $\mathbf{X}_\Phi$ and $\mathbf{X}_\Psi$ be respectively the SC and CSC clustering assignments. Moreover, we denote for compactness the additive error term by $\mathbf{E} = \Psi - \Phi\mathbf{I}_{k\times d}\mathbf{Q}$. We have that

$$\begin{aligned}
C_\Psi &= \left\|\Phi - \mathbf{X}_\Psi\mathbf{X}_\Psi^\top\Phi\right\|_F \\
&= \left\|(\mathbf{I} - \mathbf{X}_\Psi\mathbf{X}_\Psi^\top)\Phi\mathbf{I}_{k\times d}\mathbf{Q}\right\|_F \\
&= \left\|(\mathbf{I} - \mathbf{X}_\Psi\mathbf{X}_\Psi^\top)(\Psi - \mathbf{E})\right\|_F \\
&\leq \left\|(\mathbf{I} - \mathbf{X}_\Psi\mathbf{X}_\Psi^\top)\Psi\right\|_F + \left\|(\mathbf{I} - \mathbf{X}_\Psi\mathbf{X}_\Psi^\top)\mathbf{E}\right\|_F \\
&\leq \left\|(\mathbf{I} - \mathbf{X}_\Psi\mathbf{X}_\Psi^\top)\Psi\right\|_F + \|\mathbf{E}\|_F \\
&\leq \left\|(\mathbf{I} - \mathbf{X}_\Phi\mathbf{X}_\Phi^\top)\Psi\right\|_F + \|\mathbf{E}\|_F \\
&= \left\|(\mathbf{I} - \mathbf{X}_\Phi\mathbf{X}_\Phi^\top)(\Phi\mathbf{I}_{k\times d}\mathbf{Q} + \mathbf{E})\right\|_F + \|\mathbf{E}\|_F \\
&\leq \left\|(\mathbf{I} - \mathbf{X}_\Phi\mathbf{X}_\Phi^\top)\Phi\mathbf{I}_{k\times d}\mathbf{Q}\right\|_F + 2\,\|\mathbf{E}\|_F \\
&= C_\Phi + 2\,\|\Psi - \Phi\mathbf{I}_{k\times d}\mathbf{Q}\|_F
\end{aligned} \tag{3.11}$$

The lower bound directly comes from the fact that $\mathbf{X}_\Phi$ in eq. (2.46) defines the argmin of our cost functions thus $C_\Phi \leq C_\Psi$. □

We will show that filtering Gaussian random signals with an ideal low pass filter of cut-off frequency λ_k provides a perturbed version of a matrix containing the k first eigenvectors. The remaining of this section is devoted to bounding the Frobenius error $\|\Psi - \Phi\mathbf{I}_{k\times d}\mathbf{Q}\|_F$ between the features of SC and CSC. In order to prove this result, we will first express our Frobenius norm exclusively in terms of the singular values of the random matrix $\mathbf{R}$ and then in a second step we

[2] We assume all along that $d \geq k$ but a similar result holds when $d < k$. In this case, we can consider the term $\left\|\Psi\mathbf{I}_{d\times k}\mathbf{Q} - \Phi\right\|_F$ and derive the optimal unitary $\mathbf{Q}$ in order to obtain the same result as Thm. 3.4. However there is little interest in practice since one cannot expect the recovery of k eigenvectors with less random filtered signals as shown in Sec. 4.2.1.

will study the distribution of these singular values.

Our next result, reveals that the achieved error $\mathbf{E} = \|\Psi - \Phi\mathbf{I}_{k\times d}\mathbf{Q}\|_F$ is exactly determined by how close a Gaussian matrix is to a unitary matrix.

Theorem 3.4.
Let $\mathbf{R} \in \mathbb{R}^{N\times d}$ be a matrix of d i.i.d. Gaussian random vectors of size N with independent components of zero-mean and variance $\frac{1}{d}$. Let $\mathbf{R}' = \mathbf{I}_{k\times N}\mathbf{R}$ denote a random matrix with the same properties than $\mathbf{R}$ but a different shape and Σ its singular values. For any unitary Φ and $\Psi = \Phi\Phi^\top\mathbf{R}$, there exists a $d \times d$ unitary matrix $\mathbf{Q}$, such that

$$\|\Psi - \Phi\mathbf{I}_{k\times d}\mathbf{Q}\|_F = \|\Sigma - \mathbf{I}_{k\times d}\|_F. \tag{3.12}$$

Before presenting the proof, let us observe that $\mathbf{R}'$ is an i.i.d. Gaussian random matrix of size $k \times d$ and its entries have zero mean and the same variance as that of $\mathbf{R}$. We use this fact in the following to control the error by appropriately selecting the number of random signals d.

From this result, we see that if the singular values of the matrix $\mathbf{R}'$ were all ones, one would have a perfect reconstruction of the first eigenvectors with a rotated version of the random filtering. However, the exact determination of these singular values is a difficult question. For i.i.d. Gaussian random variables, this problem has been extensively lectured and we will present probabilistic bounds in the following to approximate their values. In the meantime, we present the proof for the theorem above.

Proof (Theorem 3.4)**.**
We start by noting that, by the unitary invariance of the Frobenius norm, for any $k \times k$ matrix $\mathbf{M}$

$$\|\Phi\mathbf{M}\|_F = \|\mathbf{U}\mathbf{I}_{N\times k}\mathbf{M}\|_F = \|\mathbf{I}_{N\times k}\mathbf{M}\|_F = \|\mathbf{M}\|_F. \tag{3.13}$$

We can thus rewrite the feature error as

$$\begin{aligned}\|\Psi - \Phi\mathbf{I}_{k\times d}\mathbf{Q}\|_F &= \left\|\Phi\Phi^\top\mathbf{R} - \Phi\mathbf{I}_{k\times d}\mathbf{Q}\right\|_F \\ &= \left\|\Phi^\top\mathbf{R} - \mathbf{I}_{k\times d}\mathbf{Q}\right\|_F \\ &= \left\|\mathbf{I}_{k\times N}\mathbf{U}^\top\mathbf{R} - \mathbf{I}_{k\times d}\mathbf{Q}\right\|_F \\ &= \left\|\mathbf{R}' - \mathbf{I}_{k\times d}\mathbf{Q}\right\|_F. \end{aligned} \tag{3.14}$$

We claim that there is a unitary matrix $\mathbf{Q}$ that satisfies eq. (3.12). We describe this matrix as follows. Let $\mathbf{R}' = \mathbf{Q}_L\Sigma\mathbf{Q}_R^\top$ be the singular value decomposition of $\mathbf{R}'$ and set

$$\mathbf{Q} = \begin{pmatrix}\mathbf{Q}_L & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{d-k}\end{pmatrix}\mathbf{Q}_R^\top. \tag{3.15}$$

Substituting this to the feature error, we have that

$$\begin{aligned}\left\|\mathbf{R}' - \mathbf{I}_{k\times d}\mathbf{Q}\right\|_F &= \left\|\mathbf{Q}_L\Sigma\mathbf{Q}_R^\top - \mathbf{I}_{k\times d}\mathbf{Q}\right\|_F \\ &= \left\|\Sigma - \mathbf{Q}_L^\top\mathbf{I}_{k\times d}\mathbf{Q}\mathbf{Q}_R\right\|_F \\ &= \left\|\Sigma - \mathbf{Q}_L^\top\mathbf{I}_{k\times d}\begin{pmatrix}\mathbf{Q}_L & \mathbf{0}\\ \mathbf{0} & \mathbf{I}_{d-k}\end{pmatrix}\mathbf{Q}_R^\top\mathbf{Q}_R\right\|_F \\ &= \left\|\Sigma - \mathbf{Q}_L^\top\begin{pmatrix}\mathbf{Q}_L & \mathbf{0}\end{pmatrix}\right\|_F \\ &= \|\Sigma - \mathbf{I}_{k\times d}\|_F,\end{aligned} \tag{3.16}$$

which is the claimed result. □

To bound the feature error further, we will use the following result by Vershynin, whose proof is not reproduced.

Corollary 3.5: Singular values of random Gaussians, adapted from [135, Cor. 5.35].
Let $\mathbf{N}$ *be an* $d \times k$ *matrix whose entries are independent standard normal random variables. Then for every* $t, i \geq 0$*, with probability at least* $1 - \exp(-t^2/2)$ *one has*

$$\sigma_i(\mathbf{N}) - \sqrt{d} \leq \sqrt{k} + t, \tag{3.17}$$

where $\sigma_i(\mathbf{N})$ *is the* i*th singular value of* $\mathbf{N}$*.*

Exploiting this result, the following corollary of Thm. 3.4 reveals the relation of the feature error and the number of random signals d.

Corollary 3.6.
There exists a $d \times d$ *unitary matrix* $\mathbf{Q}$*, such that, for every* $t \geq 0$*, one has*

$$\|\Psi - \Phi\mathbf{I}_{k\times d}\mathbf{Q}\|_F \leq \sqrt{\frac{k}{d}}\,(\sqrt{k} + t), \tag{3.18}$$

with probability at least $1 - \exp(-t^2/2)$*.*

Proof (Corollary 3.6)**.**
To obtain the following extremal inequality for the singular values of $\mathbf{R}'$, we note that $\mathbf{R}'$ is composed of i.i.d. Gaussian random variables with zero mean and variance $1/d$, and thus use Cor. 3.5 setting $\mathbf{R}' = \mathbf{N}/d$ and thus for every i,

$$\begin{aligned}\sigma_i(\mathbf{R}') &= \sigma_i(\mathbf{N})/\sqrt{d} \\ &\leq \frac{\sqrt{d} + \sqrt{k} + t}{\sqrt{d}} = 1 + \frac{\sqrt{k} + t}{\sqrt{d}}.\end{aligned} \tag{3.19}$$

By simple algebraic manipulation, we then find that

$$\begin{aligned}\|\Sigma - \mathbf{I}_{k\times d}\|_F^2 &= \sum_{i=1}^{k} \left(\sigma_i(\mathbf{R}') - 1\right)^2 \\ &\leq k\left(\sigma_{\max}(\mathbf{R}') - 1\right)^2 \\ &\leq k\left(\frac{\sqrt{k}+t}{\sqrt{d}}\right)^2 = \frac{k}{d}(\sqrt{k}+t)^2, \end{aligned} \tag{3.20}$$

which, after taking a square root, matches the claim. □

Finally, Cor. 3.6 combined with Lem. 3.3 provide the direct proof of Thm. 3.2 that we introduced earlier.

Before proceeding, we would like to make some remarks about the tightness of the bound. First, guaranteeing that the feature error is small is a stronger condition than distance preservation (though necessary for a complete analysis of CSC). For this reason, the bound derived can be larger than that of Thm. 3.1. Nevertheless, we should stress it is tight: the only inequality in our analysis stems from bounding the k largest singular values of the random matrix by Vershynin's tight bound of the maximal singular value.

3.3.2 Practical aspects

The study presented above assumes the use of an ideal low-pass filter $\mathbf{H}$ of cut-off frequency λ_k, namely that it is a projector on the subspace spanned by $\mathbf{U}_k$, the first k eigenvectors of $\mathbf{L}$. In practice, however, to be computationally efficient, we opt to compute $\mathbf{H}$ by an application of a polynomial function on $\mathbf{L}$ using the inexpensive Chebyshev graph filters (see Sec. 2.1.4). In this case, the used filter takes the form $\tilde{\mathbf{H}}_k = \mathbf{U}h(\mathbf{\Lambda})\mathbf{U}^\top$, where $h(\cdot)$ is a polynomial function acting on the diagonal entries of $\mathbf{\Lambda}$. We choose more specifically those approximating the ideal low-pass responses using polynomials introduced by Allen-Zhu et al. [6]. Nevertheless, it is not difficult to see that, when the filter approximation is tight, the clustering quality is little affected.

In particular, letting $\tilde{\Psi} = \tilde{\mathbf{H}}_k\mathbf{R}$ the feature error becomes

$$\left\|\tilde{\Psi} - \Phi\mathbf{I}_{k\times d}\mathbf{Q}\right\|_F \leq \left\|\tilde{\Psi} - \Psi\right\|_F + \left\|\Psi - \Phi\mathbf{I}_{k\times d}\mathbf{Q}\right\|_F. \tag{3.21}$$

We recognize the second term that is exactly the result of Cor. 3.6 and focus thus on the first term.

$$\begin{aligned}\left\|\tilde{\Psi} - \Psi\right\|_F &\leq \left\|\mathbf{U}(h(\mathbf{\Lambda}) - \mathbf{I}_{N\times k}\mathbf{I}_{k\times N})\mathbf{U}^\top\mathbf{R}\right\|_F \\ &= \left\|(h(\mathbf{\Lambda}) - \mathbf{I}_{N\times k}\mathbf{I}_{k\times N})\,\mathbf{R}\right\|_F \\ &\leq \left\|h(\mathbf{\Lambda}) - \mathbf{I}_{N\times k}\mathbf{I}_{k\times N}\right\|_2 \left\|\mathbf{R}\right\|_F. \end{aligned} \tag{3.22}$$

An extension of Thm. 3.2, taking into account filter approximation, can thus be derived where

eq. (3.22) would read with probability as least $1 - \exp(-dt^2/2)$:

$$\left\|\tilde{\Psi} - \Psi\right\|_F \text{ is } \mathcal{O}\left(c^{-c}(\sqrt{N} + t)\right), \tag{3.23}$$

where c is the order of the polynomial, $\|h(\mathbf{\Lambda}) - \mathbf{I}_{N\times k}\mathbf{I}_{k\times N}\|_2$ reduces to the approximation error of a steep sigmoid that can be bounded using [124, Proposition 5] and $\|R\|_F$ is bounded in [77, Lemma 1].

We notice that the cost of the approximation of ideal low-pass filter depends directly on the quality of the filter. Indeed, the overall error rises with the discrepancies with respect to the ideal filter as shown in eq. (3.22). Interestingly, the determination of λ_k is also very important because a correct approximation will reduce the number of non-zero eigenvalues and thus the effect of the approximated filter in the very last term of the same equation. Towards these goals, we refer the readers to [39, 5] and their respective eigencount techniques that allow to approximate the filter in $\mathcal{O}\left(scM\log(N)\right)$ operations where s is the number of required iterations.

3.4 Static experiments

3.4.1 Experimental setup

As is common practice, we use SBMs to evaluate the efficiency of this spectral clustering method [e.g., 55, 132]. The model is presented in Ex. 2.4. We set $N = 15'000, \overline{d_{\mathcal{G}}} = 60$ and $\varepsilon = \frac{\varepsilon_{\max}}{2}$ in the rest of this section. We change the value of k among the experiments to study the required number of signals and the impact of the compression on the assignment.

We compare the quality and complexity of the algorithm of Tremblay et al. (CSC) to an optimized spectral clustering [98] that uses the Lanczos algorithm to compute the first k-eigenvectors. This is significantly faster than doing the entire eigendecomposition while introducing negligible error.

We use relative error measures to compare the achieved clustering accuracy of CSC with that of SC (i.e., $(C_{CSC} - C_{SC})/C_{SC}$, where C_A is the cost of algorithm A —either SC or CSC). We considered two cost measures: the k-means cost (eq. (3.6)) and the normalized cut (ncut) cost. Since the obtained results were almost identical, we only report the results for ncut in the rest of this section. After all, the k-means cost of the spectral features is a relaxation of the ncut cost.

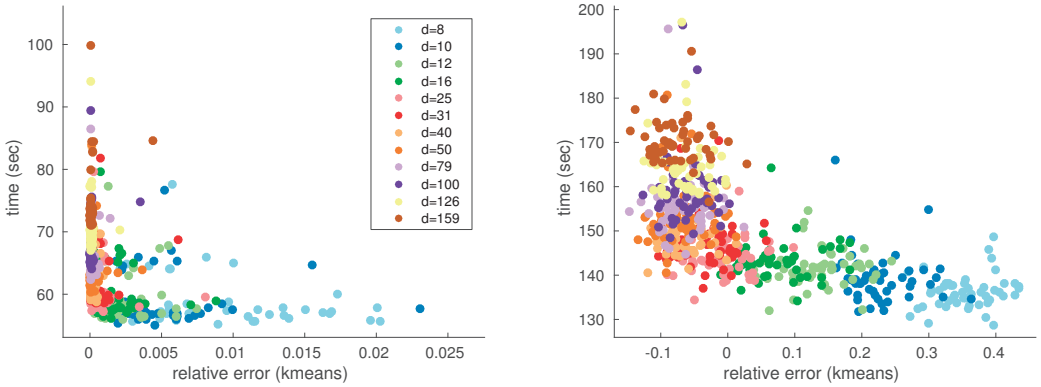
3.4.2 Required number of filtered signals

In a first experiment, we present the impact of d on the quality of the assignment. As we proved theoretically in the previous section, having more random signals filtered allow to construct an assignment which should be closer to the optimal one. However, this comes with an extra cost.

We fixed $N = 15'000, \overline{d_{\mathcal{G}}} = 60, \varepsilon = \varepsilon_{\max}/2$ and computed the assignment for 50 instances of an

SBM for several values of $k \in \{10, 16, 25, 40, 63, 100\}$ with the CSC algorithm. First, we observed that the compression step had an important impact if not correctly parametrized. In a first attempt with $n = 2k\log(k)$ nodes in the low-dimensional model (as suggested by the authors), we obtained results for $k = 10$ with a high relative error that was due to the lack of samples for the reconstruction of the complete assignment. Although asymptotically $\mathcal{O}\big(k\log(k)\big)$ samples are enough in theory, we encourage the practitioners to verify that the compression is not hampering the quality of the assignment especially when k is small.

Second, we noticed an important impact of the first filtered signals on the quality of the assignment for almost no impact on the complexity (see fig. 3.1). The comparison between the quality and the complexity can be optimized by selecting d in the center of the curve of the elbow drawn by the simulations. For $k = 10$ for instance, taking d around 25 optimizes the trade-off between quality and complexity. A similar elbow shape exists for the same plot with $k = 100$ on the right but the curve is straighter and the optimal choice is harder to see. We would recommend to pick values of d around 60 or above in this case.



(a) $k = 10$ (b) $k = 100$

Figure 3.1 – Impact of the number of filtered signals on the approximation of the assignment with the CSC algorithm for different values of k. In this experiment $N = 15'000, \overline{d_{\mathcal{G}}} = 60, \varepsilon = \varepsilon_{\max}/2$. A trade-off between the complexity and the quality is highlighted especially on the left, where the different simulations follow an elbow curve. The optimal choice for d is in the bottom left corner.

3.4.3 Evolution of the filtered signals with larger k

Interestingly, even if the k-means cost increases with k, the relative error increases faster for a fixed d, confirming our theoretical upper bound derived in the previous section. We display in figure 3.2a the experimental error difference between C_Ψ and C_Φ together with the upper bound of eq. (3.7). We used here the experiment with $k = 40$ to exemplify the phenomenon and set $t = 3$ for the theoretical bound (corresponding to the 99$^{\text{th}}$ percentile). The theoretical bound, albeit not tight in many situations, draw a representative limit of the quality of the assignment for given number of filtered signals. Some of the experiments generated from SBMs remain close to this

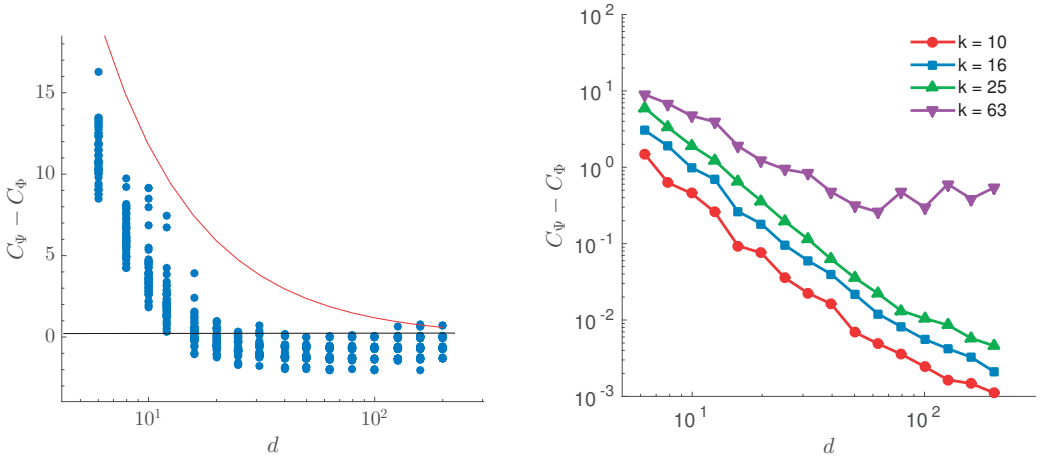
limit.

We conclude the static experiments with the study of the required number of filtered signals for different values of k (keeping all the other parameters the same). The number of random signals required to maintain a good quality of the assignment grows faster than the number of classes. In figure 3.2b, we plot the relative error at the 90$^{\text{th}}$ percentile and observe the similar shapes between the curves. This shows that up to a constant multiplicative factor that depends on k, the error difference is diminishing with the same speed for given number of signals.

3.5 Compressive clustering of dynamic graphs

Their method inspired our work on dynamic graph clustering. Embedding their technique, we aim to cluster nodes of a graph using graph filtering of random signals. We will use the information encoded in the filtering at one time step to reduce the search space at the next step and require fewer computations in the subsequent part of the graph assignment. In this respect, we introduce two additional matrices. On the one hand, $\mathbf{I}_{\ell\times m}$ of size $\ell \times m$ with ones on the diagonal will be used to resize matrices. Multiplications with this matrix allow to add or remove columns and rows. We note that $\mathbf{I}_{N\times k}\mathbf{I}_{k\times N} = \mathbf{S}_k^N$ is equal to the sub-identity matrix from (3.1) and thus $\mathbf{H} = \mathbf{U}\mathbf{I}_{N\times k}(\mathbf{U}\mathbf{I}_{N\times k})^\top = \mathbf{U}_k\mathbf{U}_k^\top$ where $\mathbf{U}_k$ represents the first k columns of $\mathbf{U}$. On the other hand, we have $\mathbf{Q} \in \mathbb{R}^{d\times d}$ that will define arbitrary orthogonal matrices of size d.

In this section, we consider the problem of spectral clustering a sequence of graphs. We focus on



(a) (b)

Figure 3.2 – Experimentations on the assignment quality with SBM for different k. In this experiment $N = 15'000, \overline{d_{\mathcal{G}}} = 60, \varepsilon = \varepsilon_{\max}/2$. (a) presents the quality of the theoretical bound for $k = 40, t = 3$ given the proximity of the worst experiments and the theoretical bound at the 99$^{\text{th}}$ percentile. (b) compare the number of signals required d to achieve a given error difference with several values of k the behavior is almost identical when k increase.

graphs $\mathcal{G}_t$ where $t \in \{1, \ldots, \tau\}$, composed of a static vertex set $\mathcal{V}$ and evolving edge sets $\mathcal{E}_t$.

Identifying each assignment from scratch (using SC or CSC) is in this context a computationally demanding task, as the complexity increases linearly with the number of time steps. In the following, we exploit two alternative metrics of similarity between graphs at consecutive time steps in order to reduce the computational cost of clustering.

Definition 3.7: Measures of graph similarity.
Two graphs $\mathcal{G}_{t-1}$ and $\mathcal{G}_t$ are:

- (ρ, k)**-spectrally similar** if the spaces spanned by their first k eigenvectors are almost aligned

$$\|\mathbf{H}_t - \mathbf{H}_{t-1}\|_F \leq \rho. \tag{3.24}$$

- ρ**-edge similar** if the edgewise difference of their Laplacians is less than ρ

$$\|\mathbf{L}_t - \mathbf{L}_{t-1}\|_F \leq \rho. \tag{3.25}$$

We argue that both metrics of similarity are relevant in the context of dynamic clustering. Two spectrally similar graphs might have very different connectivity in terms of their detailed structure, but possess similar clustering assignments. On the other hand, assuming that two graphs are edge similar is a stronger condition that postulates fine-grained similarities between them. It is, however, more intuitive and computationally inexpensive to ascertain.

3.5.1 Algorithm

We now present an accelerated method for the assignment of the nodes of an evolving graph. Without loss of generality, suppose that we need to compute the assignment for $\mathcal{G}_t$ while knowing already that of $\mathcal{G}_{t-1}$ and possessing the features that served to compute it. Our approach will be to provide an assignment for graph $\mathcal{G}_t$ that reuses (partially) the features Ψ_{t-1} computed at step $t-1$. Let p be a number between zero and one, and set $q = 1 - p$.[3] Instead of recomputing Ψ_t from scratch running a new CSC routine, we propose to construct a feature matrix Θ_t which consists of dq new features (corresponding to $\mathcal{G}_t$) and dp randomly selected features pertaining to graph $\mathcal{G}_{t-1}$:

$$\begin{aligned}\Theta_t &= \begin{pmatrix}\mathbf{H}_{t-1}\mathbf{R}_{dp} & \mathbf{H}_t\mathbf{R}_{dq}\end{pmatrix} \\ &= \Psi_{t-1}\mathbf{S}_{dp}^d + \Psi_t\overline{\mathbf{S}_{dp}^d}\end{aligned} \tag{3.26}$$

[3] Although in practice p can go up to 1, the analysis only considers the case when reused features are only from $\mathcal{G}_{t-1}$ (and not from earlier graphs).

Algorithm 2 Dynamic Compressive Spectral Clustering

Input: $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_\tau), p, d, k$
Output: $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_\tau)$

1: Determine h_k^1 the filter approximation for $\mathcal{G}_1$
2: Find an assignment $\mathbf{X}_1$ for $\mathcal{G}_1$ using CSC and h_k^1
3: **for** t from 2 to τ **do**
4: Randomly pick dp filtered signals generated on $\mathcal{G}_{t-1}$
5: Generate dq feature vectors by filtering as many random signals on $\mathcal{G}_t$ with h_k^{t-1}
6: Compute the eigencount on the features of step 5
7: Refine h_k^t if the eigencount is wrong, else keep h_k^{t-1}
8: Combine these two sets of features to find an assignment $\mathbf{X}_t$ using CSC and h_k^t
9: **end for**

Above, we use the sub-identity matrix $\mathbf{S}_{dp}^d = \mathbf{I}_{d\times dp}\mathbf{I}_{dp\times d}$ and its complement $\overline{\mathbf{S}_{dp}^d} = \mathbf{I}_{d\times d} - \mathbf{S}_{dp}^d$.

We noticed that an important part of the complexity of CSC stems from using the eigencount algorithm to estimate λ_k and construct the Chebyshev polynomials (step 1 of Algo. 1). We propose to benefit from the dynamic setting to avoid recomputing it at each step. We propose to admit that the previous value for λ_k is a good candidate for the filter at the next step, use it to filter the new random signals and validate whether it suits the new graph. Indeed, the eigencount method [39] requires exactly the result of the step 5 of our algorithm to determine if λ_k was correctly estimated. We thus compute the new filtered signals and proceed if the eigencount using the new signals is close enough to k. Otherwise, we suggest to use the knowledge of the previous result and perform a bisection (also called *dichotomy*) with this additional knowledge following [39]. The final set of features generated in the eigencount now serves as Ψ_t.

The method is sketched in Algo. 2. For simplicity, in the following we set $p \leq 0.5$ such that the reused features always correspond to $\mathcal{G}_{t-1}$ (and not to some previous time step).

Complexity analysis

We describe now the complexity of our method and compare it to that of CSC. For simplicity, we focus in a first step on the aspects that do not involve compression. Note that the first graph in the time series is computed following exactly the procedure of CSC. However, starting from the second graph, there are two steps where the complexity is reduced with respect to CSC.

First, the optimization proposed for the determination of λ_k avoids computing s steps of dichotomy for every graph. We claim that spectrally similar graphs must possess close spectrum, thus close values for λ_k. One could then expect to recompute λ_k from time to time only and that when doing so, benefit from a reduced number of iterations due to the proximity.[4] We call S the total number of steps that we gain. Since one step costs $\mathcal{O}\big(cM\log(N)\big)$ the total gain is $\mathcal{O}\big(cSM\log(N)\big)$.

[4] Though this trend has been confirmed by our numerical experiments, a formal proof remains elusive.

Second, since we reuse random filtered signals from one graph to the next, the total number of computed random signals will necessarily be reduced compared to the use of τ independent CSC calls. The gain here is $\mathcal{O}\left(cMdp\right)$ per time step.

Finally, all reductions applied through compression can also benefit to our dynamic method. Indeed, we theoretically showed that reusing features from the past can replace the creation of new random signals. Thus, sampling the combination of old and new signals can be applied exactly as defined in CSC. Then, the result of the sub-assignment can be interpolated also as defined in [132].

3.5.2 Analysis of dynamic CSC

Similarly to the static case, our objective is to provide probabilistic guarantees about the approximation quality of the proposed method. Let

$$\mathbf{X}_{\Theta_t} = \underset{\mathbf{X}\in\mathcal{X}}{\arg\min} \left\|\Theta_t - \mathbf{X}\mathbf{X}^\top\Theta_t\right\|_F. \tag{3.27}$$

be the clustering assignment obtained from using k-means with Θ_t as features, and define the *dynamic CSC cost* C_{Θ_t} as

$$C_{\Theta_t} = \left\|\Phi_t - \mathbf{X}_{\Theta_t}\mathbf{X}_{\Theta_t}^\top\Phi_t\right\|_F. \tag{3.28}$$

As the following theorem claims, the temporal evolution of the graph introduces an additional error term that is a function of the graph similarity (spectral- or edge-wise).

Theorem 3.8.
At time t, the dynamic CSC cost C_{Θ_t} and the SC cost C_{Φ_t} are related by

$$C_{\Phi_t} \le C_{\Theta_t} \le C_{\Phi_t} + 2\sqrt{\frac{k}{d}}(\sqrt{k}+s) + (1+\delta)p\gamma, \tag{3.29}$$

with probability at least

$$1 - \exp(-s^2/2) - N^{-\beta}, \tag{3.30}$$

under the constraint

$$dp \ge \frac{4+2\beta}{\frac{\delta^2}{2}-\frac{\delta^3}{3}}\log(N), \tag{3.31}$$

where $0 < \delta \le 1$. Above, γ depends only on the similarity of the graphs in question. Moreover, if graphs $\mathcal{G}_{t-1}$ and $\mathcal{G}_t$ are

- *(ρ, k)-spectrally similar, then $\gamma = \rho$,*

- *ρ-edge similar, then $\gamma = (\sqrt{2}\,\rho)/\alpha$, where $\alpha = \min\{\lambda_k^t, \lambda_{k+1}^{(t-1)} - \lambda_k^t\}$ is the Laplacian eigengap.*

Proof (Theorem 3.8)**.**
Let $\mathbf{X}_{\Phi_t}$ and $\mathbf{X}_{\Theta_t}$ be respectively the optimal SC and dynamic CSC clustering assignments at time t, and denote $\mathbf{E} = \Theta_t - \Phi_t \mathbf{I}_{k\times d}\mathbf{Q}$. We have that,

$$C_{\Theta_t} \leq C_\Phi + 2\,\|\Theta_t - \Phi_t \mathbf{I}_{k\times d}\mathbf{Q}\|_F, \tag{3.32}$$

following the exact same steps as eq. (3.11).
By completing the matrices containing the filtering of both graphs, we can see that the error term can be rewritten as

$$\begin{aligned}\|\mathbf{E}\|_F &= \left\|\Psi_{t-1}\mathbf{S}_{dp}^d + \Psi_t\overline{\mathbf{S}_{dp}^d} - \Phi_t\mathbf{I}_{k\times d}\mathbf{Q}\right\|_F \\ &= \left\|(\Psi_{t-1} - \Psi_t)\mathbf{S}_{dp}^d + \Psi_t - \Phi_t\mathbf{I}_{k\times d}\mathbf{Q}\right\|_F \\ &\leq \left\|(\Psi_t - \Psi_{t-1})\mathbf{S}_{dp}^d\right\|_F + \left\|\Psi_t - \Phi_t\mathbf{I}_{k\times d}\mathbf{Q}\right\|_F. \end{aligned}\tag{3.33}$$

The rightmost term of eq. (3.33) corresponds to the effects of random filtering and has been studied in depth in Thm. 3.4 and Cor. 3.6. The rest of the proof is devoted to studying the leftmost term using the Johnson-Lindenstrauss lemma (see [69]).

Setting $\mathbf{R}' = \frac{1}{\sqrt{p}}\mathbf{R}\mathbf{I}_{d\times dp}$, we have that

$$\begin{aligned}\left\|(\Psi_t - \Psi_{t-1})\mathbf{S}_{dp}^d\right\|_F^2 &= \left\|(\mathbf{H}_t - \mathbf{H}_{t-1})\mathbf{R}\mathbf{I}_{d\times dp}\right\|_F^2 \\ &= p\left\|(\mathbf{H}_t - \mathbf{H}_{t-1})\mathbf{R}'\right\|_F^2 \\ &= p\sum_{i=1}^{n}\left\|\mathbf{R}'^{\top}(\mathbf{H}_t - \mathbf{H}_{t-1})^{\top}\delta_i\right\|_2^2. \end{aligned}\tag{3.34}$$

Since the matrix $\mathbf{R}'$ has $n \times dp$ Gaussian i.i.d. entries with zero-mean and variance $\frac{1}{dp}$, it follows from the Johnson-Lindenstrauss lemma that

$$\begin{aligned}\left\|(\Psi_t - \Psi_{t-1})\mathbf{S}_{dp}^d\right\|_F^2 &\leq p(1+\delta)\sum_{i=1}^{n}\left\|(\mathbf{H}_t - \mathbf{H}_{t-1})^{\top}\delta_i\right\|_2^2 \\ &\leq p(1+\delta)\,\|\mathbf{H}_t - \mathbf{H}_{t-1}\|_F^2, \end{aligned}\tag{3.35}$$

with probability at least $1 - N^{-\beta}$ and for $dp \geq \frac{4+2\beta}{\delta^2(\frac{1}{2}-\frac{\delta}{3})}\log(N)$.

This concludes the part of the proof concerning spectrally similar graphs. The result for edgewise similarity follows from Cor. 3.9 below. □

Corollary 3.9: Link between the similarity measures, adapted from [63, Cor. 4].
Let $\mathbf{H}_{t-1}$ *and* $\mathbf{H}_t$ *be the orthogonal projection on to the span of* $[\mathbf{U}_k]_{t-1} (= \Phi_{t-1})$ *and* $[\mathbf{U}_k]_t (= \Phi_t)$. *If there exists an* $\alpha > 0$ *such that* $\alpha \leq \lambda_{k+1}^{(t-1)} - \lambda_k^t$ *and* $\alpha \leq \lambda_k^t$, *then,*

$$\|\mathbf{H}_t - \mathbf{H}_{t-1}\|_F \leq \frac{\sqrt{2}}{\alpha} \|\mathbf{L}_t - \mathbf{L}_{t-1}\|_F . \tag{3.36}$$

Note that the bounds on α *are those described in [63, Thm. 3].*

Optimality of theorem 3.8

We will now consider a different application of the Johnson-Lindenstrauss lemma on a problem of different size. Just like the one chosen in the proof of the theorem above, this one provides a bound for the leftmost term of eq. (3.33) that we are interested in. The final result stems from the combination of this note with the previous theorem.

We also have an instance of the lemma with $\mathbf{R}$ that says that

$$\begin{aligned} \left\| (\Psi_t - \Psi_{t-1}) \mathbf{S}_{dp}^d \right\|_F^2 &\leq \|(\Psi_t - \Psi_{t-1})\|_F^2 \\ &\leq (1+\varepsilon) \|\mathbf{H}_t - \mathbf{H}_{t-1}\|_F^2 . \end{aligned} \tag{3.37}$$

with probability at least $1 - N^{-\beta}$ as long as $d \geq \frac{4+2\beta}{\varepsilon^2(\frac{1}{2} - \frac{\varepsilon}{3})} \log(N)$.

Since eq. (3.35) and (3.37) are both true simultaneously with probability at least $1 - N^{-\beta}$ under different conditions, we introduce μ and restate Thm. 3.8 with more details.

Definition 3.10.
Let $0 < \delta, \varepsilon < 1$ describe approximation parameters. We define μ as

$$\mu = \min\left\{ p(1+\delta), 1+\varepsilon \right\} \tag{3.38}$$

under the constraints

$$d \geq \frac{4+2\beta}{\varepsilon^2/2 - \varepsilon^3/3} \log(N), \qquad dp \geq \frac{4+2\beta}{\delta^2/2 - \delta^3/3} \log(N). \tag{3.39}$$

An analysis of the constrained minimization problem is contained in Appendix A.1.

Corollary 3.11: Tighter version of Thm. 3.8.
At time t, *the dynamic CSC cost* C_{Θ_t} *and the SC cost* C_{Φ_t} *are related by*

$$C_{\Phi_t} \leq C_{\Theta_t} \leq C_{\Phi_t} + 2\sqrt{\frac{k}{d}}(\sqrt{k} + s) + \mu p \gamma, \tag{3.40}$$

with probability at least $1 - \exp(-s^2/2) - N^{-\beta}$.

3.5.3 Managing the approximation of the assignment

With the previous theorem, one can now decide of a fixed budget for the error and adapt the quantity of reused signals depending on the evolution of the graph. For instance, if the graph did not evolve much between the last two steps, more signals could be reused than otherwise. For that, one needs to be able to approximate the graph similarity without computing the spectral basis.

This can be quite easily done for the edge similarity since it simply requires a matrix difference. However the estimation of spectral similarity is less straightforward.

Proposition 3.12: Estimator of $\|\mathbf{H}_t - \mathbf{H}_{t-1}\|_F$.
We propose $\hat{\rho} := \left\| g(\mathbf{L}_t)\mathbf{R} - g(\mathbf{L}_{t-1})\mathbf{R} \right\|_F$ *an unbiased estimator of the spectral similarity between two graphs.*

$$\begin{aligned} \mathbb{E}\left[\|\mathbf{H}_t\mathbf{R} - \mathbf{H}_{t-1}\mathbf{R}\|_F\right] &= \mathbb{E}\left[\|\mathbf{H}_t - \mathbf{H}_{t-1}\|_F \|\mathbf{R}\|_2\right] \\ &= \|\mathbf{H}_t - \mathbf{H}_{t-1}\|_F \mathbb{E}\left[\|\mathbf{R}\|_2\right] \\ &= \|\mathbf{H}_t - \mathbf{H}_{t-1}\|_F = \rho. \end{aligned} \tag{3.41}$$

Then, we need to modify slightly our previous algorithm to include the estimation of ρ and adapt the value of p accordingly. Since Cor. 3.11 proved an additive error of at most $\mu p \gamma$, we have $p = \frac{\varepsilon}{\mu\gamma}$. The modifications required to include the estimation of ρ allowed to predetermine part of the features required to estimate the assignment on the new graph. Indeed, since we forced $p \leq \frac{1}{2}$ we needed at least $\frac{d}{2}$ new filterings on the new graph. We now compute these mandatory feature vectors at the very beginning and use these to compute $\hat{\rho}$. Later, we process the missing number of random signals and combine everything in order to feed CSC with enough features.

The detailed technique is summarized in Algo. 3. Note that the random signals[5] $\mathbf{R}$ selected in step 4 can correspond to either those of step 4 at the previous time step or the random signals generated at step 11 previously if any but not from step 10.

3.6 Experiments

This section complements the theoretical results described in Section 3.5. First, we present a model of synthetic graph perturbations, and study its impact on the ρ-spectral similarity. From there, we present the results of our dynamic clustering algorithm on graphs of different sizes and connectivity.

[5] The reader is warned that we are talking about the random signals $\mathbf{R}$ and not the random filterings $\mathbf{HR}$ here.

Algorithm 3 Dynamic Compressive Spectral Clustering with variable reusing rate

Input: $(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_\tau), d, k, \beta$
Output: $(\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_\tau)$

1: Determine h_k^1 the filter approximation for $\mathcal{G}_1$
2: Find an assignment $\mathbf{X}_1$ for $\mathcal{G}_1$ using CSC with h_k^1 and $\mathbf{R}$ and store $\mathbf{H}_1\mathbf{R}$ for later
3: **for** t from 2 to τ **do**
4: Pick $\frac{d}{2}$ random signals $\mathbf{R}$ from the filterings on $\mathcal{G}_{t-1}$
5: Filter the features of step 4 with h_k^{t-1} and compute the eigencount
6: Refine h_k^t if the eigencount is wrong, else keep h_k^{t-1}
7: Recompute step 5 iff the filter changed
8: Estimate ρ with $\hat{\rho}$ using the filtering of the same random signals on two different graphs
9: Derive p based on the budget as $\min\left(\frac{1}{2}, \varepsilon(\mu\hat{\rho})^{-1}\right)$
10: Randomly pick dp filtered signals generated on $\mathcal{G}_{t-1}$
11: Generate $d(\frac{1}{2} - p)$ feature vectors by filtering as many random signals on $\mathcal{G}_t$ with h_k^t
12: Combine together the filtered signals of steps 7, 10 and 11
13: Find an assignment $\mathbf{X}_t$ using compressive k-means of CSC with the features of step 12
14: **end for**

Here again, we compare the quality and complexity of our dynamic method (dCSC) against the algorithm of Tremblay et al. (CSC) and the spectral clustering algorithm by Ng et al. [98] (SC). The details about the evaluation measures and SBM construction can be found in Sec. 3.4.1.

3.6.1 Spectral similarity

Our theoretical approach highlights the importance of the spectral similarity between two consecutive steps of the graph. We thus start this section by describing how much the graph can change between two assignments. Starting from an SBM, we perform two types of perturbations: *edge redrawing* and *node reassignment*. Edge redrawing consists of removing some edges at random from the original graph and then adding the same number following the probabilities defined by the graph model (using q_1 and q_2). In node reassignment, one selects nodes, removes all edges that share at least one end with the nodes previously picked, reassigns those nodes to any other class at random and reconnects these nodes with new edges using again the same probabilities q_1 and q_2.

Figure 3.3 shows the similarity of graphs under various perturbation models tested with many different parameters. Interested by the impact of the rightmost term of eq. 3.29 on the assignment cost, we analyzed the ratio $\rho/\mathcal{C}_\Phi$ for graphs of different shapes. We tried $N = 1'000, 5'000, k = 6, 25, \overline{d_\mathcal{G}} = 30, 60$ with 10 replicates each. Figures 3.3a and 3.3b illustrate the impact of the two aforementioned perturbation models combined. In 3.3a, each curve maps to a perturbation of x% of the edges plus the perturbation of the nodes defined by the x-axis where $x \in [0, 5]$. Similarly, in 3.3b each curve represents a different perturbation of the nodes. In both scenarios, the similarity is altered proportionally to the perturbation. As one could expect, redrawing a given percentage of the nodes has more impact than the same percentage of edges redrawing. This is due to the

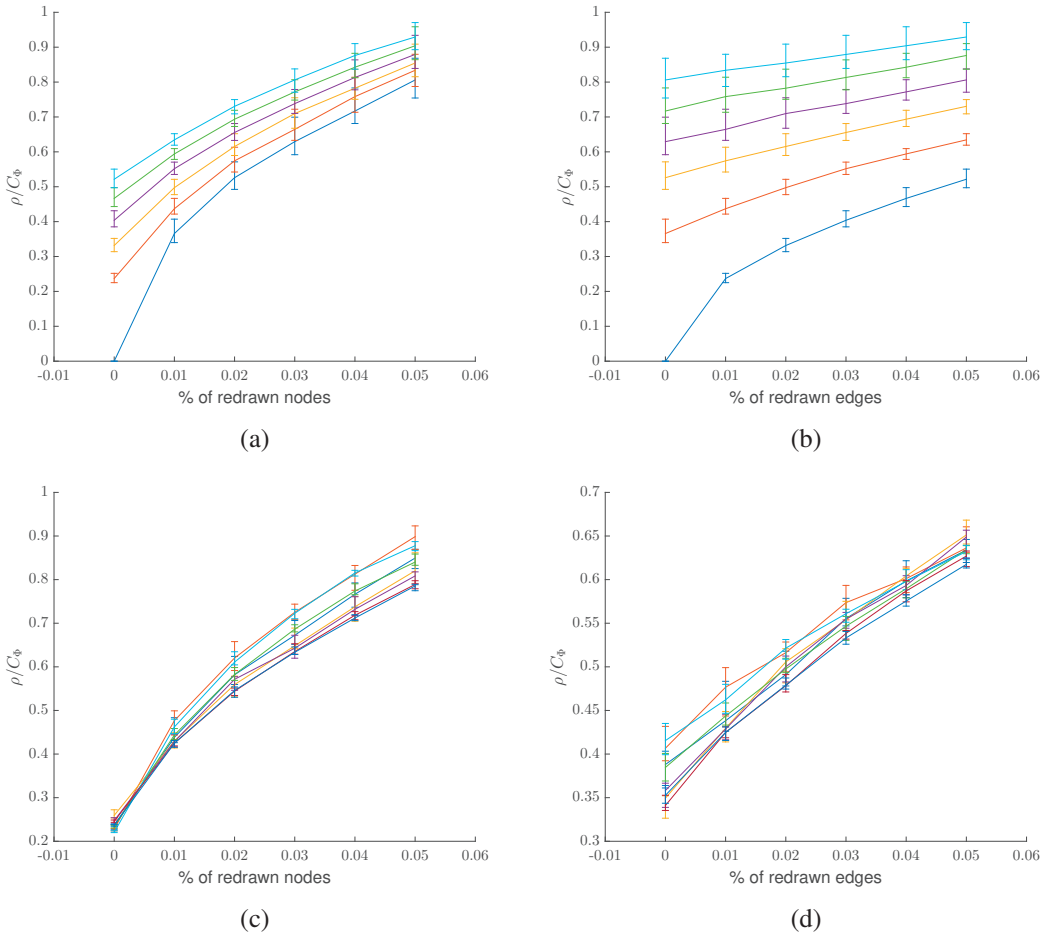


Figure 3.3 – Study of the different perturbation models and their impact on the normalized graph spectral similarity. Combined effects of the redrawing of nodes and edges are shown in figures (a) and (b). The different curves correspond to the variable that is not perturbed on the x-axis (from 0% to 5% with increments of 1%). In figures (c) and (d) one of the two perturbation models is fixed to 1% and different values of $N, k, \overline{d_{\mathcal{G}}}$ are compared in the different curves with no significant impact ($N \in \{1'000, 5'000\}, k \in \{6, 25\}, \overline{d_{\mathcal{G}}} \in \{30, 60\}$).

fact that x% of the nodes are connected to more than x% of the edges since they are redrawn as soon as one of the two extremities is redrawn.

Figures 3.3c and 3.3d show in the different lines the combinations of N, k and $\overline{d_{\mathcal{G}}}$ with one fixed parameter for the perturbation and the second varying on the x-axis. We were not able to detect any significant difference between the combinations. Overall, the use of this combined perturbation model has the advantage to produce an error proportional to the modification and not to the graph of study. We will thus be able to compare graphs of different sizes or number of classes in the later experiments.

3.6.2 Dynamic clustering of SBM

We proceed to evaluate the efficiency of dynamic CSC. Both perturbations are combined in the synthetic graph that we are studying next. We replicate the construction of 100 different SBM based on the same parameters, then we alter each of them with 1% of node reassignment and 1% of edges modifications. The modified graph is used for the evaluation of all methods.

When does reusing features pay off?

The latter methods, being designed for static graphs, they are run once on each of the two graphs. Timing takes into account only the part required to compute the assignment of the second graph. We first study the error-complexity trade-off achieved by the compressive clustering methods as a function of d. We set $N = 15000, k = 25, \overline{d_{\mathcal{G}}} = 60$. Each point in Figure 3.4 corresponds to a single graph being clustered. For each of the two methods, there are 1600 points resulting from 100 repetitions when the number of features is $d \in [6, 200]$ with logarithmic increments. To comprehend the results, it is helpful to consider each of the six sextants in the figure separately. The top-middle sextant shows that when d is large enough (left side), the relative error of CSC and dCSC is close to zero. Increasing d reduces the error but increases the time required for the computation, following the elbow from right to left. The top-right and top-left sextants occur because Lloyd's algorithm (despite being rerun 100 times) sometimes fails to retrieve the optimal solution to the k-means problem: the top-left (resp. right) sextant corresponds to cases when Lloyd's algorithm produces a suboptimal assignment for SC (resp. CSC/dCSC). The bottom three sextants correspond to cases when dCSC did not have to recompute λ_k (step 7 of Algo. 2). In these cases, dCSC is up to twice faster than CSC. Though the frequency of this phenomenon depends on many factors, such as the size of the eigengap and the spectral similarity of consecutive graphs, we report that in our experiment dCSC could avoid recomputing λ_k, roughly 50% of the times.

In summary, reusing the vectors produces a clear computational benefit with a very reasonable loss of accuracy. Most of this benefit comes from the step of estimation of λ_k that can be avoided as long as the spectral similarity between the two graphs is small, especially for well-clusterable graphs (where the gap $\lambda_{k+1} - \lambda_k$ is large). Indeed, we can highlight two regimes in the figure 3.6 in red with the same behavior with respect to the error but an additional constant time separates the two.

What proportion of signals to reuse?

Figure 3.5 displays the results of our clustering for different proportions of previous signals reused in terms of two important metrics: time and accuracy. The error displayed on the figures is the multiplicative error of the k-means cost defined in Thm. 3.8, namely $\frac{C_{\Theta_t} - C_{\Phi_t}}{C_{\Phi_t}}$. Since this quantity requires the computation of SC, we are forced to consider problems where N stays in the order of thousands due to its important complexity. Figures 3.5a and 3.5b illustrate the benefits of

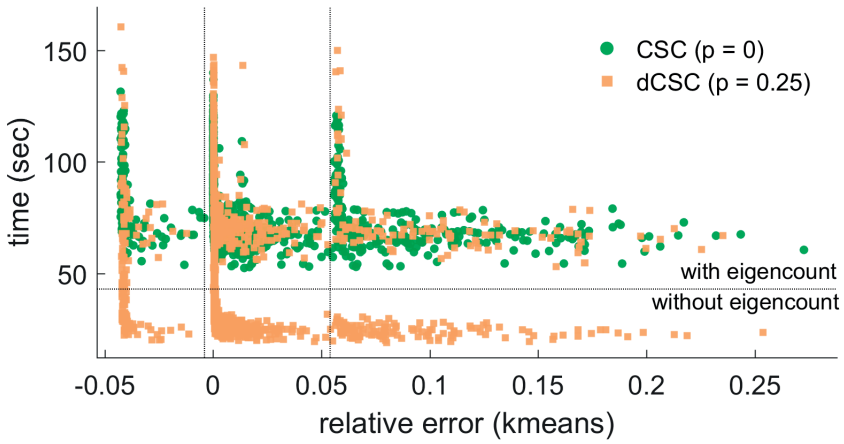
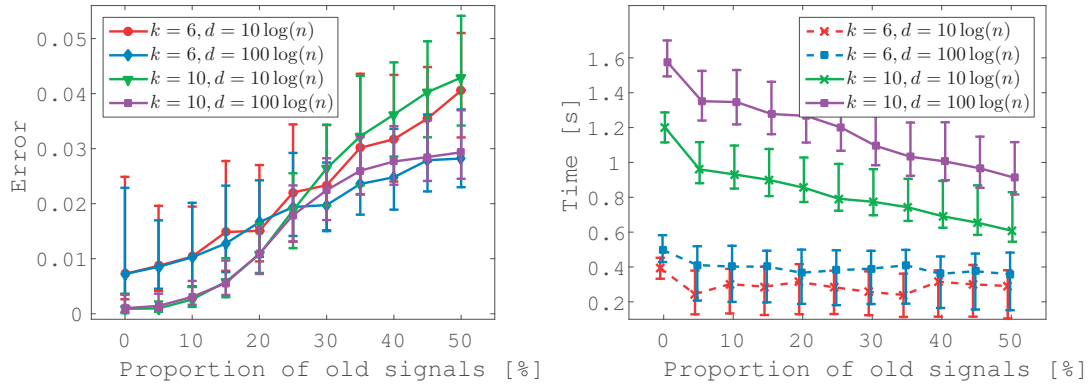


Figure 3.4 – Error-complexity trade-off achieved by compressive clustering methods. The points correspond to different number of features d and repetitions. Three behaviors are highlighted: the majority of the runs tend to the same quality than SC with provided enough features; the rest is relatively close (1% deviation) and can be of better or worse quality; dCSC benefits from a significant complexity reduction when λ_k is not recomputed (up to a factor 3).

reusing large parts of the previously computed features on graphs with $N = 1'000, \overline{d_{\mathcal{G}}} = 25, \varepsilon = \frac{1}{6}$.



(a) (b)

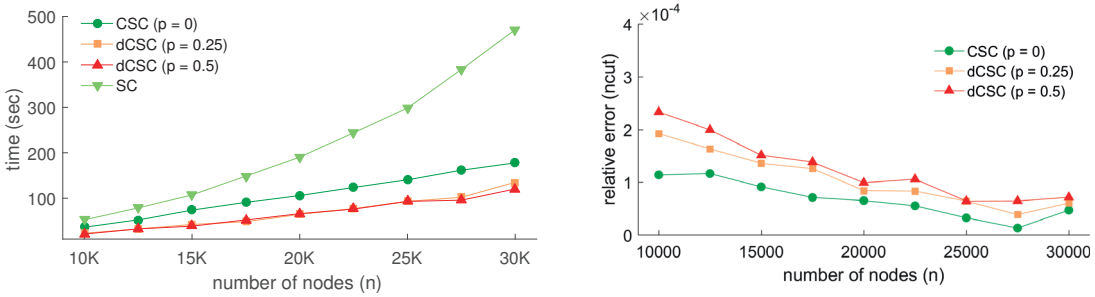
Figure 3.5 – Performances of our algorithm for dynamic graph clustering on synthetic data. The experiment is conducted on SBM with $N = 1'000$. The number of classes and the number of filtered signals are chosen among several values to observe their dependence. The relative error $\frac{C_{\Theta_t} - C_{\Phi_t}}{C_{\Phi_t}}$ is computed in subfigure (a) where we observe that both k and d have an impact on the quality of the approximation. The right figure presents the computational benefit of features reusing.

First, it is important to notice that, as it could be expected, the error that we observe is slightly increasing as p grows, up to 3% of the SC cost when reusing 50% of the previously computed signals. This is very encouraging since in practice, such proportional error is not significant. Finally, we observe a computational benefit by looking at the time gained by more use of the

previous features, as shown in Fig. 3.5b. We emphasize that the improvement in terms of time can attain 25% of the total time in the most extreme cases depicted in this figure.

3.6.3 Comparison with state-of-the-art

To evaluate the efficiency of dCSC, we varied the number of nodes N (while fixing $k = 25, d = 50, \overline{d_{\mathcal{G}}} = 60$). Figure 3.6 shows the results. As expected, the difference of complexity between spectral clustering using the partial eigen-decomposition and dCSC is clearly visible. Increasing p from 0.25 to 0.5 incurs only a small computational benefit, that becomes non-negligible only for larger N (14.5 seconds when N =30'000, corresponding to a 12% improvement). We also report that the achieved relative error for both methods remained consistently below 0.1% and did not grow as N increased. We do not present values of N above 30'000 as, for such cases, SC took too long to complete. To get an idea, with 64Gb of RAM, SC took one hour to process the graph and return an assignment when $N = 50'000$. For the same graph, dCSC runs in 6.5 minutes producing a result of similar k-means cost.



(a) (b)

Figure 3.6 – Scaling capabilities of SC and its approximations. dCSC consistently outperforms CSC in terms of complexity. Their almost linear complexity makes them good candidates for big data analysis compared to SC. When reusing signals, the assignment quality converges towards CSC when N grows, sending another positive signal for big datasets.

Table 3.1 further compares our proposed method to SC, CSC and IASC, the state-of-the-art method for spectral clustering suitable for dynamic graphs [38]. We can see that dCSC achieves a significant improvement in timing when N is large enough. Note that IASC results were obtained by running the optimized and parallel implementation kindly provided by the original authors.[6]

3.6.4 Real-world dataset

We conclude this experiment section with the study of a dynamic dataset constructed for the occasion. Based on Arxiv submissions of papers between 1992 and 2013, we constructed a series of graphs representing the collaborations at each end of the month. The nodes represent the authors and two authors are connected if they cited each other more than 20 times in the last

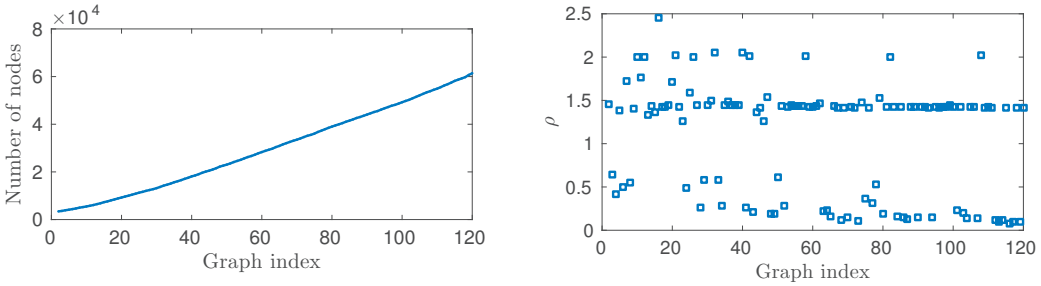
[6] Available at https://github.com/charanpald/sandbox

		SC (Lanczos)	IASC	CSC (p=0)	dCSC (p=0.25)	dCSC (p=0.5)
N = 1'000	time (sec)	**1.27** (±0.09)	1.37 (±0.20)	2.81 (±0.56)	2.53 (±0.54)	2.86 (±0.56)
	k-means	**5.42** (±0.06)	6.19 (±0.28)	6.16 (±6.40)	5.49 (±1.35)	5.48 (±5.02)
	ncut	**18.96** (±0.01)	19.20 (±0.08)	19.18 (±2.00)	18.98 (±0.44)	18.99 (±1.39)
N = 10'000	time (sec)	48.29 (±5.41)	806.29 (±72.91)	41.21 (±3.11)	23.33 (±9.29)	**21.98** (±11.33)
	k-means	**6.24** (±0.01)	6.61 (±0.31)	6.31 (±0.61)	6.28 (±0.40)	6.26 (±0.40)
	ncut	**18.80** (±0.00)	18.91 (±0.10)	18.82 (±0.21)	18.81 (±0.13)	18.81 (±0.13)

Table 3.1 – Timing and accuracy comparison for various sizes, $k = 25, d = 50, \overline{d_{\mathcal{G}}} = 60, \varepsilon = \frac{\varepsilon_c}{2}$. SC: Spectral Clustering using the Lanczos method. IASC: Incremental Approximate Spectral Clustering [38]. CSC: Compressive Spectral Clustering [132]. dCSC: our method (Alg. 2).

three years. We generated an undirected graph with no self-loops where the edge weights are the sum of the citations in both ways. The first graph contains the connections related to the papers published between 1992 and 1994. Then, the second one considers the papers between February 1992 and January 1995. Each following graph is constructed by considering the publications of the next month and omitting those of the first month in the previous graph. We consider only the largest connected component for each graph and tolerates nodes addition and deletion. Newly added nodes only possess new filtered signals the first time that they appear. The graph size evolves with time between 3'140 nodes in 1994 and 61'405 in 2004. We displayed in Figure 3.7 the evolution of the number of nodes and the spectral similarity between the consecutive snapshots.

For the evaluation, we computed the Ncut measure as before but also the accuracy compared with a ground truth based on the most frequent section of Arxiv in which the author published between 'cs', 'math', 'cond-mat', 'physics', and 'astro-ph'. We ran the experiment with and without the compressive k-means but the quality of the assignment was significantly lower when performed



(a) (b)

Figure 3.7 – Statistics of our Arxiv dataset. The evolution of the number of nodes is summarized on the left. The spectral graph similarity between the consecutive snapshots is displayed on the right.

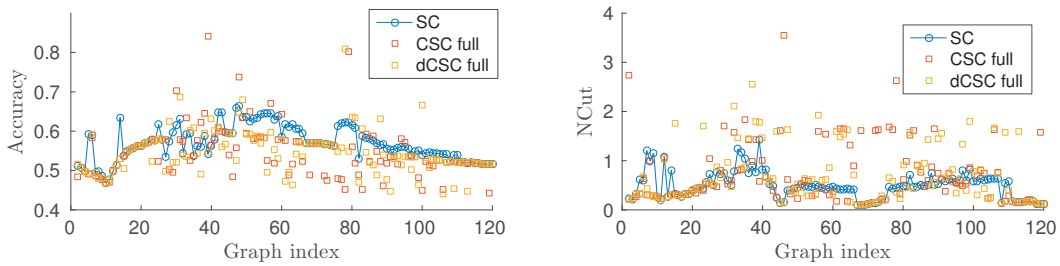
with the compressive k-means. We decided to report only the results without the compression in Figure 3.8 to ease the readability. Accuracy and ncut are often exactly approximated with the dynCSC method in this experiment($d = 100, p = 0.15$ here).

3.7 Conclusion and Future Work

The major contribution of this chapter is the presentation of a fast clustering algorithm for dynamic graphs that achieves similar quality than SC. We proved theoretically how much the graph can change before losing information for a given computational budget.

Recent advances in graph clustering using efficient methods, including this approach, can provide a huge complexity gain. Nevertheless, practitioners must pay attention because these works require a proper setup. In particular, N and k must be large enough for the approximated method to make sense. Otherwise, the computational benefit is not significant (if existing at all). Moreover, highly clusterable graphs are necessary for the approximated methods to work best. Specifically when working with dynamic networks, the spectral similarity is very important and must remain properly bounded at all times. For instance, we tested compressive methods on a citation network based on the public Arxiv dataset on which we did not see any computational gain although the quality of the obtained assignment was similar to that of SC.

We highlighted in this chapter several open directions of research for the future. It appears clearly in the experiments that the majority of the remaining complexity lies in the estimation of $\mathbf{H}$ and more precisely the capability to reuse the previous determination of λ_k. Moreover, the compression step of CSC tend to hamper slightly the quality of the assignment and a proper theoretical analysis including this step would be a nice improvement. Finally, expressing the approximation error in function of the assignment instead of Ncut could produce a more insightful explanation of the impact of the various factors.



(a) (b)

Figure 3.8 – Comparison of DynCSC with SC and CSC on the evolutive graph of Arxiv citations between 1992 and 2004. In this experiment, the k-means is run on the full graph. The left figure shows the proportion of correctly classified authors with respect to the ground truth based on the principal domain of publication of the authors. On the right, we computed the ncut for each graph and compared the results between the three methods.

4 Approaching Laplacian Eigenspaces with Random Signals

Eigendecomposition has been at the core of famous techniques used to extract low-dimensional embeddings from high-dimensional data by using the eigenvectors associated with specific eigenvalues. This has been used for partitioning (e.g., SC [98, 122]), data visualization (e.g., Laplacian eigenmaps [12]), but also simply as a dimensionality reduction technique for preprocessing (e.g., PCA [70]). The main drawback of all the aforementioned techniques is that they tend not to scale well as they have a rather high complexity since a partial eigendecomposition costs already $\mathcal{O}\left(kN^2\right)$ operations.

This chapter is the result of a work realized with Johann Paratte aiming to improve the solution of two problems at once: clustering and visualization. The common denominator in the two approaches is the knowledge of the eigenvectors of the Laplacian, which is also the most computationally demanding task in the algorithms that we study. The recent developments in the domain of random filtering made us wonder if there was a way to go further than the distance preservation property proved by Tremblay et al. [132] with a similar technique (see Sec. 3.2). Our primary goal was finally set as the determination of the eigenspace generated by the first eigenvectors since it was enough for the applications that we tried to improve (visualization in low dimensions and spectral clustering). Going even further with the exact determination of the eigenvectors (up to the sign) remains an open problem despite our numerous attempts.

4.1 Related works

The classical way to recover the eigenspace of a symmetric matrix $\mathbf{L}$ is to diagonalize it as $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^*$, with $\mathbf{U}$ being the matrix of eigenvectors and $\mathbf{\Lambda}$ the matrix of eigenvalues. Often, one then takes only the first k columns of $\mathbf{U}$ for the subsequent processing. The diagonalization is typically done using a Singular Value Decomposition (SVD) of a symmetric matrix of size N in $\mathcal{O}\left(N^3\right)$ operations, which is intractable even for medium scale N. A great deal of work has been done attempting to compute eigenvalues and eigenvectors of $\mathbf{L}$ efficiently (see [11] for a review). The fastest methods are variants of Arnoldi or Lanczos iteration methods ([8] and [76]

respectively) such as Implicitly Restarted Arnoldi Method (IRAM) [128] or Implicitly Restarted Lanczos Method (IRLM) [23]. The preferred method for graph Laplacians is the IRLM since the matrix is symmetric and sparse most of the time. The IRLM has a worst case complexity of $\mathcal{O}\left(i(kM + k^2N + k^3)\right)$, with i the number of iterations to reach convergence and assuming there are $\mathcal{O}(k)$ extra Lanczos steps [11]. If we consider sparse graphs with $M \in \mathcal{O}(N)$ and a fixed k independent of the value of N, the complexity of the IRLM is bounded by the term $\mathcal{O}\left(k^2N\right)$.

Since the exact computation of the eigenspace proves to be expensive, several angles were considered to approximate the result. Physicists came up with a solution to the problem of eigenspace determination, using contour integration techniques for the reduction of the matrices on which to apply the eigendecomposition [109], that allows improving the complexity with almost no loss of precision. Meanwhile, with the additional constraint that the matrix should contain a subset of the columns of the original matrix, Boutsidis et al. [19] proposed a fast method to approximate low-rank matrix reconstruction (whose optimal solution is the eigenspace generated by the first eigenvectors). Some works, such as [84], focused on their side on the determination of the first non-trivial eigenvector only. Finally, Bai [10] proposed a solution for the approximation of eigenvectors using tridiagonalization of sparse matrices that requires *efficiently sparse* matrices as input.[1] Although this might not necessarily apply in practice depending on the data set at hand, it proved to be efficient in various problems involving modeling physical phenomena with strong locality properties. More related works are also introduced in Sec. 2.2.5.

Instead of computing the eigenspace as features of the data points in the new space, distance preservation can be considered sufficient depending on the application. Indeed, for tasks such as clustering, assuming an algorithm such as k-means is performed to get the final assignment step, the preprocessing for dimensionality reduction only requires pairwise distances between points to be preserved in the new space. In this mind, [115] presents a clustering algorithm that avoids the computation of an SVD by computing fast filtering of random signals and using the Johnson-Linderstrauss lemma.

On the same track, the authors of [20] show that the power method (computing powers of the normalized weight matrix) gives a good approximation of the eigenvectors for distance preservation. They give a bound on the exponentiation required to obtain a good approximation of the clustering. This is among the first works, to my knowledge, to use random signal multiplied by powers of the weight matrix.

Even more recently, Tremblay et al. [132] proposed a fast algorithm for graph clustering that is provably almost as good as spectral clustering. The first half of their work uses random signal filtering and provides a result similar to the one presented in [20]. Moreover, they additionally showed that only a subset of the nodes must be assigned with k-means and that the rest could be inferred from the graph structure by solving an optimization problem. They stated bounds on the number of signals required and the number of nodes to label with k-means achieving a theoretical complexity of no more than $\mathcal{O}\left(k^2 \log^2(k) + MN(\log(N) + k)\right)$.

[1] See the paper for a definition of efficiently sparse matrices

In this chapter, we present a new algorithm for Fast Eigenspace Approximation using Random Signals (FEARS) to estimate the k eigenvectors of the Laplacian associated with the smallest eigenvalues, using random signal filtering techniques that were already used in the works on distance preservation. This time, however, we do not simply find a mapping for distance preservation but we are able to obtain the partial eigenspace, with a total complexity inferior to the previous works.

In this context, this chapter proposes various improvements to the field, whose main contributions are as follows:

1. *In Section 4.2 we propose a computationally efficient scheme for the estimation of Laplacian eigenspaces using filtering of random graph signals, called FEARS.* We assess theoretical guarantees about the approximation of the subspace and prove a tight bound for the number of random signals needed for perfect recovery. We also extend our analysis with the study of the random filterings prior to the reorthogonalization.

2. *Algorithms and implementations with practical considerations regarding filter design, fast filtering, and numerical stability are addressed in Section 4.3 together with a complexity analysis.* We also propose in that section an accelerated method for the count of eigenvalues in a given range based on the eigencount method introduced in [39].

3. *Applications of our proposed algorithm are presented in Section 4.4 and compared to state-of-the-art methods.* We also support the theoretical analyses in the different sections with synthetic experiments reproducing the expected behavior of the different theoretical parts as and when they are introduced.

4.2 Fast eigenspace approximation using random signals

The goal of our method (FEARS) is to get the best estimation of the subspace of the graph Laplacian $\mathbf{L}$, denoted $\mathbf{U}_k$, for the lowest computational cost. In a similar approach to [132] and [20], we consider the filtering of random signals. We chose an ideal low-pass filter $g(\mathbf{L}) = \mathbf{U}_k\mathbf{U}_k^\top$ to achieve this goal.[2] Throughout this section, we prove the following theorem, one of our main results:

Theorem 4.1.
Let $g(\lambda)$ be an ideal low-pass filter of cut-off frequency λ_k, let $\mathbf{R} \in \mathbb{R}^{N\times d}$ a random matrix formed of entry-wise independent and identically distributed Gaussian random variables $\sim \mathcal{N}(0, \frac{1}{d})$. Let $\mathbf{L}$ be the Laplacian of any undirected graph $\mathcal{G}$.

For any $d \geq k$, performing a QR decomposition on the result of the filtering of $\mathbf{R}$ by g provides

[2] We consider in this chapter that $\mathbf{L}$ is real symmetrical and hence that $\mathbf{U}$ is orthogonal.

the first k eigenvectors of $\mathbf{L}$ *altered only by a rotation in* $\mathbb{R}^k$*. Mathematically we have:*

$$\mathbf{BX} = g(\mathbf{L})\mathbf{R} \quad \textit{and} \quad \mathbf{B}\mathbf{I}_{N\times k} = \mathbf{U}_k\mathbf{Q} \quad \textit{a.s.}, \tag{4.1}$$

where $\mathbf{I}_{N\times k}$ *corresponds to a* $N \times k$ *matrix with ones on the diagonal and* $\mathbf{Q} \in \mathbb{R}^{k\times k}$ *defines arbitrary orthogonal matrices of size k, and where* $\mathbf{BX}$ *represents the QR decomposition with* $\mathbf{B} \in \mathbb{R}^{N\times N}$ *and* $\mathbf{X} \in \mathbb{R}^{N\times d}$*.*

4.2.1 Exact eigenspace recovery with random signals

Assuming we pack d Gaussian random signals with i.i.d. entries $\sim \mathcal{N}(0, \frac{1}{d})$ in a Gaussian random matrix $\mathbf{R} \in \mathbb{R}^{N\times d}$, the result of the filtering using the filter g can be written as $\Psi = \mathbf{U}_k\mathbf{U}_k^\top\mathbf{R} = \mathbf{U}_k\mathbf{R}_k$. We will first state a result regarding $\mathbf{R}_k$ and then use $\mathbf{R}_k$ directly to compute the projection.

Lemma 4.2.
Let $\mathbf{V}$ *be an orthonormal basis and denote* $\mathbf{V}_k$ *a sampling of any k rows.*

The projection of a Gaussian random matrix $\mathbf{R} \sim \mathcal{N}(0, \sigma^2 I)$ *onto* $\mathbf{V}_k$ *preserves all the Gaussian properties of* $\mathbf{R}$*.*

Proof.
The multiplication of a Gaussian random matrix by a basis such as $\mathbf{V}$ preserves all the properties of the initial random matrix (Gaussianity, entry-wise independence, identical mean, variance, and size). The result is called $\mathbf{R}' = \mathbf{VR}$ and the proof can be found in Appendix B.1.

Selecting any subset of the columns of $\mathbf{V}$ changes the size but preserves the orthonormal properties over the rows. Indeed, without loss of generality on the selected rows, we have

$$\begin{pmatrix}\mathbf{I}_k & \mathbf{0}\end{pmatrix}\mathbf{VR} = \begin{pmatrix}\mathbf{I}_k & \mathbf{0}\end{pmatrix}\mathbf{R}' = \mathbf{R}_k \tag{4.2}$$

Thus, only the size will be altered compared to a multiplication by the full matrix $\mathbf{V}$. This concludes the proof. □

With lemma 4.2, setting $\mathbf{V}_k = \mathbf{U}_k^\top$,we have that $\mathbf{R}_k \in \mathbb{R}^{k\times d}$ is i.i.d. Gaussian of zero mean and variance $\frac{1}{d}$. The next step is to show that $\mathbf{R}_k$ is full rank.

Lemma 4.3.
Let $\mathbf{R}_k \in \mathbb{R}^{k\times d}, d \geq k$ *be a Gaussian random matrix of entry-wise i.i.d.* $\sim \mathcal{N}(0, \sigma^2)$*.*

$\mathbf{R}_k$ *is a full rank matrix with probability 1. That is,* $\text{rank}\{\mathbf{R}_k\} = k$ *since* $d \geq k$*.*

Proof.
Let us consider the limit case $d = k$. In this case we have to show that the square $(k \times k)$ matrix $\mathbf{R}_k$ is non-singular. Indeed, the set of singular Gaussian random matrices $\mathcal{R}_s = \{\mathbf{R}_k : \det(\mathbf{R}_k) = 0\}$ is of dimension $k-1$ since it is generated by the zeros of a polynomial of order k. Moreover,

since the complete set $\mathcal{R} = \{\mathbf{R}_k\}$ has dimension k, the codimension of $\mathcal{R}_s$ is 1. Thus, the set $\mathcal{R}_s$ is a null set, which means that picking a matrix at random from the set $\mathcal{R}$ returns a matrix from $\mathcal{R}_s$ with probability 0. Hence, $\mathbf{R}_k$ is non-singular with probability 1.

If we consider the case where $d > k$, any square matrix formed of k of the columns of $\mathbf{R}_k$ has rank k following the proof provided above for the square case. Now, adding columns to this matrix cannot modify the rank since it cannot reduce it and the matrix is full rank already. □

In practice, numerical approximations can alter the result of the previous lemma since the singular value decomposition could consider columns to be linearly dependent if the singular value is below the machine precision of the value 0. We prove in Appendix B.2 that this will not happen in high probability.

Now that we confirmed that $\mathbf{R}_k$ is full rank, even considering numerical approximations, we analyze the final projection $\Psi = \mathbf{U}_k\mathbf{R}_k$.

Lemma 4.4.
Let $\Psi = \mathbf{U}_k\mathbf{R}_k$ a matrix of size $N \times d$, with $\mathbf{U}_k$ and $\mathbf{R}_k$ as defined above. The two following statements are correct:

$$\forall x \in \mathbb{R}^k, \exists y \in \mathbb{R}^d : \mathbf{U}_k x = \Psi y. \tag{4.3}$$

$$\forall y \in \mathbb{R}^d, \exists x \in \mathbb{R}^k : \mathbf{U}_k x = \Psi y. \tag{4.4}$$

That is Ψ and $\mathbf{U}_k$ share the same column space.

Proof.
Since $\mathbf{R}_k$ is full rank, its span is able to generate any matrix in $\mathbb{R}^{k\times d}$. Then, the projection of this full space onto $\mathbf{U}_k$ can form any matrix generated by the span of $\mathbf{U}_k$. □

Note that, although all the lemmas above assume $d \geq k$, we suggest using $d = k$ in practice since this is the minimal value for which the result holds and thus the one that will require the least computation. We are now able to prove the theorem introduced at the beginning of this section.

Proof (Proof of theorem 4.1)**.**
From Ψ, we can find a set of k orthonormal vectors $\mathbf{B} = \{\mathbf{b_1}|\mathbf{b_2}|\ldots|\mathbf{b_k}\}$, e.g., by applying an SVD. We obtain a decomposition such as $\mathbf{M} = \mathbf{B}\Sigma\mathbf{V}^\top$, with Σ a diagonal matrix and $\mathbf{V}$ an orthogonal matrix. This gives the following equality:

$$\mathbf{U}_k\mathbf{R}_k = \Psi = \mathbf{B}\Sigma\mathbf{V}^\top \tag{4.5}$$

and thus $\mathbf{U}_k$ and $\mathbf{B}$ have the same column space by definition. But since $\mathbf{B}$ and $\mathbf{U}_k$ also have the same shape and orthonormal columns, they necessarily relate to each other as $\mathbf{B} = \mathbf{U}_k\mathbf{Q}$, for some rotation matrix $\mathbf{Q} \in \mathbb{R}^{k\times k}$. □

Before moving on to the following of the chapter, we would like to stress the fact that the theory described here does not use any assumption made on $\mathbf{L}$. Thus, the statements we make are also true for any matrix for which there exists a spectral decomposition.[3] However, the sparsity of this matrix is key to a fast implementation using graph filtering as we will show next.

4.2.2 Ψ as an approximation of $\mathbf{U}_k$

The matrix $\mathbf{B}$ has been shown to approximate $\mathbf{U}_k$ up to a rotation, which is perfectly fine for all common applications (e.g., embedding, spectral clustering, etc.). In the following lines, we wanted to present the quality of Ψ as a direct approximation of $\mathbf{U}_k$. In the discussion below, we show that it could be enough in some situations to stop the procedure before the SVD step and reduce the complexity of the algorithm by doing so.

Recall that Ψ and $\mathbf{B}$ share the same column space (i.e., $\text{span}\{\mathbf{U}_k\}$) as we proved in Thm. 4.1 and have the same shape. The major difference between the two is that only the latter is composed of normalized columns. However, the distribution of the singular values of Ψ is well-known: it is the same as that of $\mathbf{R}_k$ since $\mathbf{U}_k$ has unitary columns. Moreover, the works of Marchenko and Pastur [86] contain lots of results regarding the study of Gaussian ensemble and Wishart matrices. They showed, among other things, that the distribution of the eigenvalues of Wishart matrices follows a quarter circle law, which means that the distribution of any singular value of Ψ is a normalized quarter circle of support $[0;2]$ when $d = k$. On top of that, they proved that the expected value and the standard deviation of those eigenvalues tend to 1 as N becomes large. This means that in average, even with $d = k$, Ψ is a very good candidate for the approximation of the subspace. The problem is that with the variance on the eigenvalue distribution, random samples hardly benefit from the expectation.

Meanwhile, the Johnson-Linderstrauss lemma says that with $d \in \mathcal{O}\big(\log(N)\big)$, the distances between rows of $\mathbf{U}_k$ and rows of Ψ are almost preserved (i.e., up to a $(1+\varepsilon)$ multiplicative factor) with high probability. Thus, it seems intuitive that picking more random signals would improve the distribution of the eigenvalues between 0 and 2 and concentrate around the mean. In fact, from the definition of the Marchenko-Pastur distribution, we have the following result:

Corollary 4.5: Corollary 5.35 from [135].
Let A be an $N \times n$ matrix whose entries are independent standard normal random variables. Then for every $t \geq 0$, with probability at least $1 - 2\exp(-t^2/2)$ one has $\sqrt{N} - \sqrt{n} - t \leq s_{\min}(A) \leq s_{\max}(A) \leq \sqrt{N} + \sqrt{n} + t$.

[3] Note that bounds on the spectrum must be known or computed in order to design a polynomial approximation of the filter that usually has $[-1,1]$ as domain. This does not prevent the technique to be used but add an extra step in the algorithm that we do not consider in the following.

In our case, the entries are Gaussians of variance $\frac{1}{N}$ and the result becomes:

$$1 - \sqrt{\frac{k}{d}} - \frac{t}{\sqrt{d}} \leq s_{\min}(\mathbf{R}_k) \leq s_{\max}(\mathbf{R}_k) \leq 1 + \sqrt{\frac{k}{d}} + \frac{t}{\sqrt{d}} \tag{4.6}$$

We conclude that the more the matrix $\mathbf{R}_k$ is flat (i.e., $d > k$), the more its eigenvalues are concentrated around 1 in good probability, which confirms our intuition.

4.2.3 Quality of approximation for various graphs

In this section, we measure the accuracy of our FEARS algorithm with different classes of graphs and for different values of k and N. In particular, we wish to evaluate two things: on the one hand, the actual eigenvectors $\mathbf{b}_i$ with respect to the true $\boldsymbol{u}_i$ of the eigenspace $\mathbf{U}_k$, and, on the other hand, the quality of the approximation of the eigenspace $\mathbf{U}_k$ itself with Algo. 5.

The graphs chosen for this experiment are all well-known classes in the field and have various spectral properties. Here is a list of all graphs with short descriptions:

- **Sensor network**: A graph of a synthetic sensor network, which represents randomly positioned sensors connected in a k-NN fashion.
- **SBM**: Stochastic Block Model graphs model social networks or community graphs and are known to be clusterable (and thus possess a large eigengap).
- **Swissroll**: This graph is a k-NN graph of the famous Swissroll manifold, a point cloud drawn from a rolled 2D surface in 3D.
- **Bunny**: This graph is the k-NN graph constructed from the 3D point cloud of the Stanford bunny.
- **Image graph**: This graph is created by connecting the pixels of an image using similarity of patches. The image of interest is the grayscale image of Barbara, a natural image often used in image processing.
- **Road network**: This graph represents the Minnesota road network (originally from the MatlabBGL library).

In order to measure the quality of the approximated eigenspace (up to a rotation), we introduce a measure of the amount of energy which is preserved when the approximated eigenspace is projected on the real eigenspace computed with exact methods. If we note the approximated eigenspace as $\mathbf{B}_k$ and the exact eigenspace $\mathbf{U}_k$, the normalized energy kept by the projection is:

$$E(\mathbf{B}_k, \mathbf{U}_k) = \frac{1}{k} \|\mathbf{B}_k^T \mathbf{U}_k\|_F^2. \tag{4.7}$$

We chose to use the normalized energy to score the quality of the estimated eigenspace as it gives a number between 0 and 1 where higher values mean better approximation.

Table 4.1 aggregates the results for 50 replicates of the experiment with the different graphs. Here we set $k = 25$ and we observe as expected that with the exact computation of $\Psi = \mathbf{U}_k\mathbf{U}_k^\top\mathbf{R}$ the eigenspace is exactly preserved. The difference between Ψ and the result of its QR-decomposition cannot be observed with this measure but we insist on the fact that Ψ does not possess orthogonal columns (we however renormalized the columns at a negligible complexity cost). This perfect recovery is highly interesting. However it requires the knowledge of $\mathbf{U}_k$ itself to determine it. The interest of the method is thus to replace Ψ with an approximation that would be faster to compute.

In the bottom part of the table, the computation of Ψ is replaced with $g(\mathbf{L})\mathbf{R}$ where g is a polynomial approximation of the ideal low-pass filter (see Sec. 2.1.4 and 4.3.1). We tried to recover $\mathbf{U}_k$ with different polynomial orders and observed improvements depending on the quality of the filter. These results are discussed further in the following of this chapter.

Next, we used the sensor graph and the SBM and computed the second eigenvector (Fig. 4.1). We displayed it for the three subspaces $\mathbf{U}_k, \mathbf{B}_k$ and Ψ. Although not identical (especially in terms of scaling and sign), the resemblance with $\mathbf{U}_k$ is very clear. On the bottom row, the nodes are assigned coordinates based on the first two eigenvectors of each of the decomposition and a signal corresponding to the cluster assignment. With the three methods, we see that the clusters are linearly separable in $\mathbb{R}^2$ with different cuts.

4.3 Computational aspects of subspace approximation

Now that our main theoretical result is established (Thm. 4.1), we look into its practical implementation, while focusing on efficient solutions. First, we present our choice for the actual filter design using polynomials enabling fast filtering operations while limiting the problems caused by

	Sensor network $N = 10'000$	SBM $N = 10'000$	Swiss-roll $N = 10'000$	Bunny $N = 2'503$	Image $N = 16'384$	Road network $N = 2'642$	
svd($\mathbf{U}_k\mathbf{U}_k^\top\mathbf{R}$)	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00
$\mathbf{U}_k\mathbf{U}_k^\top\mathbf{R}$	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00
Approx $c = 100$	0.53 ± 0.02	0.99 ± 0.02	0.78 ± 0.01	0.53 ± 0.02	0.89 ± 0.01	0.67 ± 0.01	0.73
Approx $c = 500$	0.85 ± 0.02	1.00 ± 0.01	0.92 ± 0.01	0.85 ± 0.02	0.98 ± 0.01	0.89 ± 0.01	0.92

Table 4.1 – Quality of the subspace approximation in theory and practice. Theoretically, we expect to recover the exact eigensubspace with FEARS and even without the QR decomposition (top rows). However in practice, this would require the knowledge of $\mathbf{U}_k$ and thus would be unnecessary computations. We compare it with fast filtered version of $\mathbf{U}_k\mathbf{U}_k^\top\mathbf{R}$ in the bottom part of the table using Jackson-Chebyshev approximation of the step function.

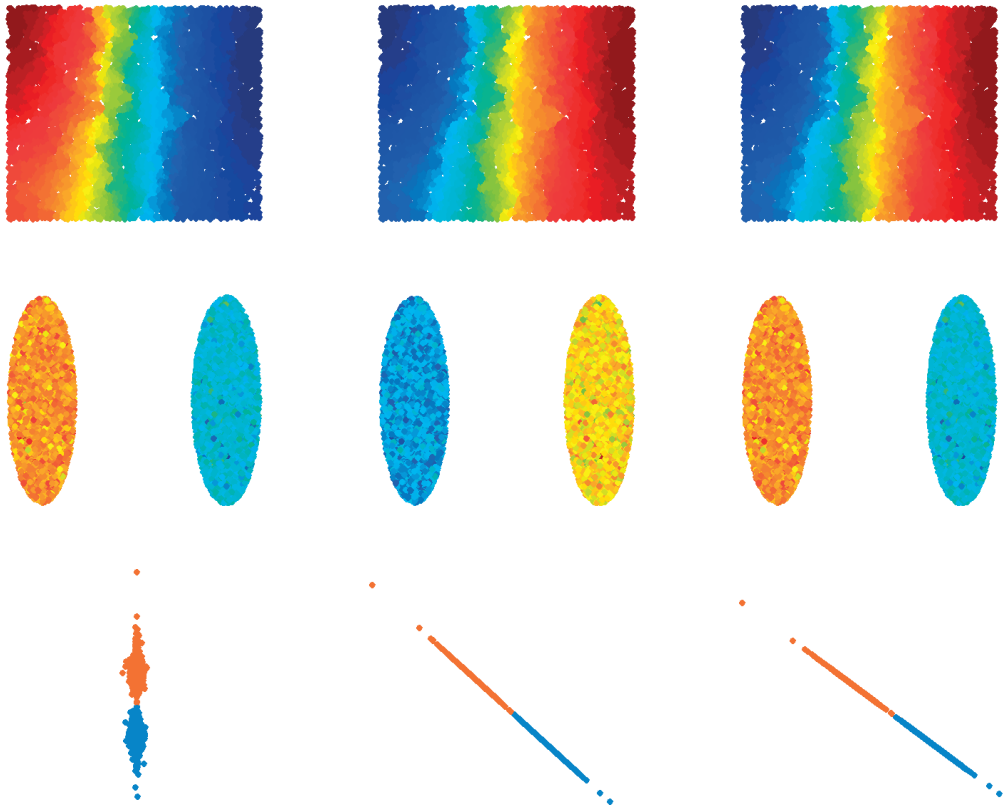


Figure 4.1 – Quality of the second eigenvector estimation with FEARS ($\mathbf{B}_k$) and its non-orthogonalized version (Ψ). The first column is $\mathbf{U}_k$, the second one is $\mathbf{B}_k$, the last one is Ψ. The first eigenvector is not drawn because $\boldsymbol{u}_1$ is the constant vector. Each row represents a different case. The top one is performed on a $10'000$ vertices sensor network, the middle one on a $10'000$ vertices SBM and the bottom one is a visualization of the clustering of the same SBM than before. The color represents the value of the signal from its minimum (dark blue) to its maximum (dark red). On the bottom line, orange and blue represent the two classes.

the approximation. Then, we propose a solution to find the cut-off eigenvalue λ_k required by the polynomial approximation of the filter. Finally, we analyze the computational complexity of our algorithm.

4.3.1 Acceleration using fast filtering

The construction of the matrix Ψ in the previous section requires the knowledge of the first k eigenvectors of the graph Laplacian. This knowledge is very costly for large graphs (N large) since it requires a partial SVD of a $N \times N$ matrix, which we try to avoid in the first place. Fortunately, as we explained before, the product $\mathbf{U}_k\mathbf{U}_k^\top$ corresponds to a graph filtering with $g(\mathbf{L})$,

g being an ideal low-pass filter:

$$g(\lambda) = \begin{cases} 1 & \lambda \leq \lambda_k \\ 0 & \lambda > \lambda_k \end{cases} \tag{4.8}$$

Since we cannot afford the cost of exact filtering, we use a polynomial approximation of the filter $g(\mathbf{L})$. There exist several methods using powers of the Laplacian provide an approximation for such filters using polynomials (Chebyshev [58, 6], Jackson-Chebyshev [39], Lanczos [129] polynomials). For the task at hand, the Jackson-Chebyshev polynomial approximation is the best suited to approximate the step function of $g(\mathbf{L})$ since it avoids the Gibbs effect of Chebyshev polynomials as can be seen in Fig. 4.2a on a random Sensor Graph with $N = 500$ nodes.

However, a Chebyshev-based approximation of the sign function proposed recently by Allen-Zhu and Li [6] seems to achieve similar results than Jackson-Chebyshev polynomials. A thorough study would be required to determine which of the two is best fitted to the problem since our first observations showed differences in our experiments with no systematic winner. Our guess would be that with enough parameter tuning, Allen-Zhu's method is more appropriate in a large variety of setups.

We computed the norm $\left\| \widehat{g}(\Lambda) - g(\Lambda) \right\|_2$ where $\Lambda \in \mathbb{R}^{N \times N}$ is the diagonal matrix of eigenvalues for random sensor graphs. This measure compares the maximum eigenvalues of the filtering differences and is of interest for theoretical approximations such as that of eq. (3.22). We reported the results in table 4.2 below. We observe that depending on the order of the approximation, each method performs best for some values of c and both methods achieve similar results overall.

	$c = 50$	$c = 100$	$c = 200$	$c = 400$
Jackson-Chebyshev	**0.4266**	0.3580	0.2342	0.0694
Allen-Zhu	0.4288	**0.3270**	**0.2231**	**0.0444**

Table 4.2 – Measure of the approximation of the ideal low-pass filter with polynomial approximations of different degrees c using $\left\| \widehat{g}(\Lambda) - g(\Lambda) \right\|_2$. Lower values represent approximations of better quality and 0 would mean that the filter takes the correct values for all the eigenvalues of the graph. The study is performed on random sensor networks with $N = 500$. When the degree is high enough, Allen-Zhu seems to outperform Jackson-Chebyshev.

The quality of the approximation is based on the order of the polynomial, directly related to the number of coefficients to compute. If we define c as the highest degree of the polynomial, we can show that the error of approximation decreases as c increases. This effect is shown in Fig. 4.2b and 4.2c where we can see the convergence to the ideal low-pass with an increasing value of c. But since the complexity of the filtering increases linearly with c one cannot let it become too large. In particular, we cannot let c be proportional to N since it would have a huge impact on the overall complexity.

Let us remind here that the filter approximation needs to be correct only on the discrete values

given by the eigenvalues. Indeed, the approximation does not need to fit closely g as long as the discrete values that the filter takes on the eigenvalues are correct. In our case, since we only want to approximate a step function, we need the value of the filter to be equal to 1 for $\lambda_0, \lambda_1, \ldots, \lambda_{k-1}$ and 0 for $\lambda_k, \lambda_{k+1}, \ldots, \lambda_{N-1}$. Two situations could lead to the non-respect of this condition. The estimated cut-off eigenvalue can be wrong or the order of the polynomial can be too small.

If the order c is too small, then, as can be seen in Fig. 4.2 for $c = 100$, the filter will be below 1 for a few eigenvalues below λ_{k-1}, and above 0 for a few eigenvalues after λ_k. If the estimated cut-off eigenvalue is a bit off, a similar situation will happen, with a shift towards lower or higher frequencies. In both cases, the value of the filter will still be 1 up to some eigenvalue λ_j, then monotonically decreasing to 0 up to some eigenvalue λ_l and 0 up to λ_{N-1}, with $\lambda_j < \lambda_k < \lambda_l$. In such a case, the filter will have non-zero coefficients in the range $[\lambda_k, \lambda_l]$ and thus, Ψ will be contaminated by some elements of the space $\mathbf{U}_{[k+1,l+1]}$. However, these contributions will

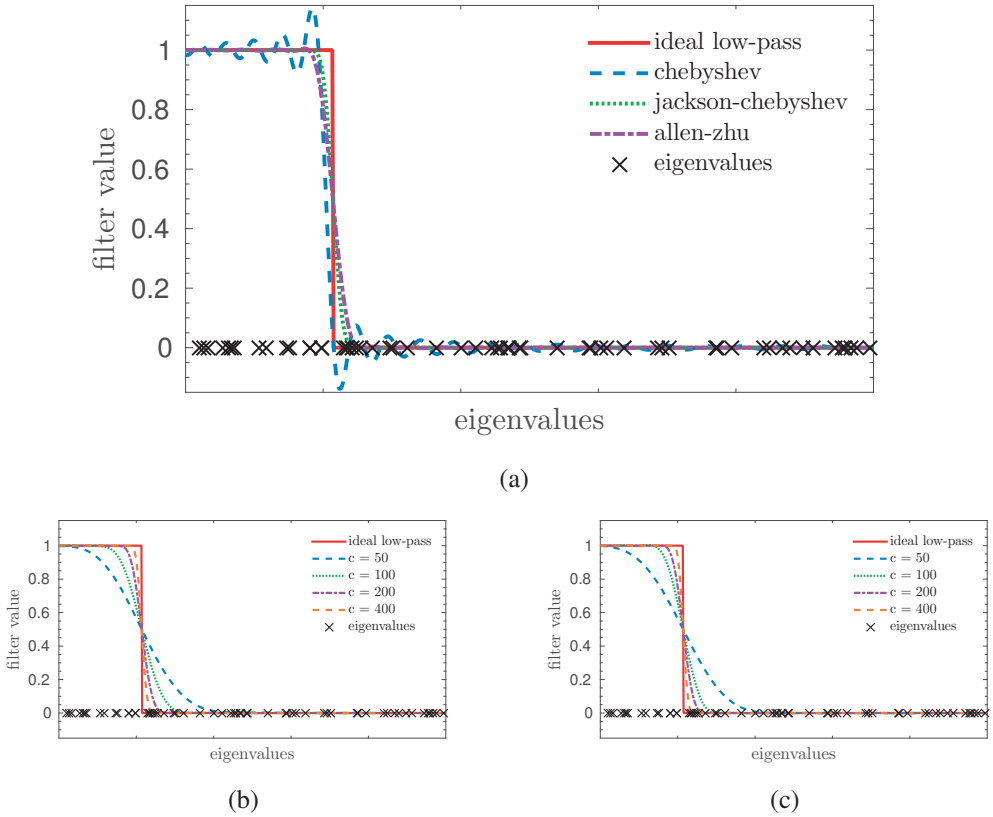


Figure 4.2 – The effect of approximating a step function with polynomials. The solid red line is the ideal step function. The black crosses represent the eigenvalues. The approximation using Jackson-Chebyshev polynomials (dotted line) is compared with Chebyshev polynomial approximation (dashed line) and the Allen-Zhu proposal (dash-dot line) of same order $c = 200$ in (a). Jackson-Chebyshev approximations with different orders c are compared in (b). Allen-Zhu approximations with the same orders c are compared in (c).

not appear too much in the energy of Ψ since the coefficients of the filter for the eigenvalues bigger than λ_k are smaller than all coefficients for the range $[\lambda_0, \lambda_{k-1}]$. Indeed, since our final approximation $\mathbf{B}_k$ is done using an SVD of Ψ, then $\mathbf{B}_k$ will be the best rank k approximation of Ψ by minimizing the energy of the residuals. Overall, as one can verify in the experiments of Sections 4.2.3 and 4.4, the polynomial order is important to the quality of the subspace determination but $\mathbf{B}_k$ will provide features that remain very good for the various applications that we develop, even with a low polynomial order.

4.3.2 Estimation of λ_k

The computation of Ψ described above depends on the quality of the filter g and the determination of its cut-off frequency λ_k which is not known a priori. A standard method is to use eigencount techniques such as the one proposed in [39]. In this work, the authors used the fact that the energy retained by an ideal low-pass filtering of random signals with cut-off frequency λ_k, called g_{λ_k}, is proportional to the number of eigenvalues that are smaller than λ_k. Mathematically, we have:

$$\mathbb{E}[\|g_{\lambda_k}(\mathbf{L})\mathbf{R}\|_F^2] = |\{\lambda : \lambda \leq \lambda_k\}|\,. \tag{4.9}$$

Thus, by dichotomy, one could approximate the desired threshold value λ_k for our filter since we want it to capture exactly k eigenvalues and we know that $\lambda_{\max} \leq 2$ for normalized Laplacians.[4] Unfortunately, each step of the dichotomy requires d filterings and the dichotomy must be applied $\mathcal{O}\big(\log(N)\big)$ times, without making strong assumptions on the distribution of the eigenvalues over the spectrum. In [39, Lem. 2.1], the authors proved with high probability an upper bound on the estimation error of the eigencount which depends on d and combined it with experiments showing that $d \propto k$ is enough. Thus, the estimation of λ_k such as defined and used in [132] is $\mathcal{O}\big(cMk\log(N)\big)$, which is above the complexity of all the rest of our problem.

We propose now an accelerated version of the eigencount technique for the determination of the threshold λ_k that will not increase the complexity of the overall algorithm. We first assume that the eigenvalues are distributed evenly over the spectrum (between 0 and $\lambda_{\max}$). Thus, on average, the k^{th} eigenvalue should be $\mathbb{E}(\lambda_k) = \frac{k}{N}\lambda_{\max}$. However, one will not find the exact count systematically on the first guess, due to the randomness of the process and the non-uniformity of the eigenvalue distribution in practice. We suggest thus to iterate with the assumption of local uniformity of the distribution of the eigenvalues until the goal is reached. In practice, this means that after picking $\lambda(0) = \frac{k}{N}\lambda_{\max}$, one should apply the eigencount technique to compute the approximation of the real number of eigenvalues below $\lambda(t)$ in the graph of study, called $n_\lambda(t)$, and iterate with $\lambda(t+1) = \frac{k}{n_\lambda(t)}\lambda(t)$ until the targeted count is achieved with good precision (see Algo. 4 for details). As the number of iterations does not depend on N but only of the local eigenvalue distribution, a good precision can be achieved with a constant number of iterations. The cost in number of operations of this accelerated version is thus $\mathcal{O}(cMk)$ which is acceptable

[4] One can efficiently bound $\lambda_{\max}$ for the combinatorial Laplacian as well by running IRLM for $k = 1$ in $\mathcal{O}(M)$ operations.

since it is of the same order than the remaining of our method.

Algorithm 4 presents in detail the strategy described in Section 4.3.2 for the accelerated estimation of λ_k. The main assumption here is that the distribution of the eigenvalues is uniform by part over the spectrum. We thus try to reach such segment of the spectrum where uniformity applies to fasten the discovery of the value of λ_k. Since some parts of the spectrum can be empty due to eigengaps for some classes of graphs, we implemented a dichotomic step to get a broad spectrum distribution estimate if the search does not progress.

We consider the setting presented in Section 4.2.3 and compare the quality of approximation of the eigenspace $\mathbf{U}_k$ without the knowledge of λ_k this time. In order to construct the graph filter, our accelerated eigencount method is used and compared against the reference method used in [132]. We summarize the results in Table 4.3.

In order to compare our accelerated eigencount method with the reference dichotomy implementation of [132] (abbreviated fast and standard respectively in the table), we used two measures. First, the number of iterations required until convergence, which is adequate since the workload

Algorithm 4 Estimation of λ_k

Input: $k, \lambda_{\max}$ and $\mathbf{L}$
Output: λ_{est} (the cut-off frequency hopefully in $[\lambda_k, \lambda_{k+1}]$)

Initialize: $\lambda_{lb}, c_{lb}, \text{iter}, c_{est} \leftarrow 0$
$\lambda_{ub} \leftarrow \lambda_{\max}, c_{ub} \leftarrow N, \lambda_{est} \leftarrow k\frac{\lambda_{\max}}{N}$
Generate $\mathbf{R}$ with $d = k$
while $c_{est} \neq k$ and $\text{iter} < \max_{\text{iter}}$ **do**
    Compute approximated graph filter g_λ with $\lambda = \lambda_{\text{est}}$
    $c_{\text{est}} \leftarrow \|g_\lambda(\mathbf{L})\mathbf{R}\|_F^2$
    if $c_{est} < k$ **then**
        $\lambda_{lb} \leftarrow \lambda_{\text{est}}$
    else
        $\lambda_{ub} \leftarrow \lambda_{\text{est}}$
    end if
    if $c_{lb} = c_{\text{est}}$ or $c_{ub} = c_{\text{est}}$ **then**
        $\lambda_{\text{est}} \leftarrow \frac{\lambda_{lb}+\lambda_{ub}}{2}$
    else
        if $c_{\text{est}} < k$ **then**
            $c_{lb} \leftarrow c_{\text{est}}$
        else
            $c_{ub} \leftarrow c_{\text{est}}$
        end if
        $\lambda_{\text{est}} \leftarrow \lambda_{lb} + (k - c_{lb})\frac{\lambda_{ub}-\lambda_{lb}}{c_{ub}-c_{lb}}$
    end if
end while
return λ_{est}

per iteration is the same in the two algorithms. Second, we measure how close to the actual value of k have the algorithm converged by computing the mean squared deviation between our estimator and k. This last measure is useful to state if the method was able to converge with respect to the current random matrix used for estimation, and not with respect to the actual value of λ_k.

The results of all measures for the various graphs described above are reported in Table 4.3. Due to the randomness of the methods we evaluate, all experiments are averaged over 50 realizations and the standard deviation is indicated for all measures.

If we first focus on the upper part of Table 4.3 we can see that the measure of the energy (ME, see eq. (4.7)) using the true cut-off λ_k (exact) shows an average above 90% of precision over all graphs with a perfect score for very clusterable graphs (such as SBM) and lower values for more difficult graphs (such as Sensor network). The trend is similar using estimated values for λ_k both with the standard and fast methods. Using the approximated cut-off lowers the score of about 5%. Using the fast method leads to marginally better results. One very interesting fact regarding these results is that both the λ_k estimation step and the eigenspace approximation contribute to the lost energy in approximately equal amounts. This tends to indicate that it is important to balance the computational effort between the two steps and not favoring one against the other.

On the middle part of Table 4.3, we can see the first measure reported for the eigencount evaluation. The number of iterations needed to compute λ_k (IT) is lower with the fast method for all but the SBM graph. On average, the fast method is 2.5 times faster than the standard method. For the SBM graph, fast is close to its maximum number of iterations meaning that the

		Sensor network $N = 10'000$	SBM $N = 10'000$	Swiss-roll $N = 10'000$	Bunny $N = 2'503$	Image $N = 16'384$	Road network $N = 2'642$	
ME	exact	0.86 ± 0.01	1.00 ± 0.01	0.86 ± 0.02	0.99 ± 0.01	0.91 ± 0.01	0.93 ± 0.01	0.92
	standard	0.80 ± 0.03	0.95 ± 0.05	0.79 ± 0.03	0.94 ± 0.05	0.86 ± 0.04	0.90 ± 0.05	0.87
	fast	0.80 ± 0.03	0.96 ± 0.04	0.79 ± 0.03	0.95 ± 0.04	0.86 ± 0.04	0.90 ± 0.04	0.88
IT	standard	14.62 ± 0.90	**5.32** ± 1.58	4.68 ± 0.62	8.74 ± 1.77	13.06 ± 1.58	11.34 ± 1.22	11.29
	fast	**3.02** ± 0.71	9.36 ± 1.06	**2.86** ± 0.70	**4.48** ± 1.25	**3.12** ± 0.75	**3.06** ± 0.51	**4.31**
KD	standard	0.60 ± 0.53	2.46 ± 4.92	0.52 ± 0.58	0.36 ± 0.53	0.34 ± 0.48	0.36 ± 0.48	0.79
	fast	**0.00** ± 0.00	**1.00** ± 1.01	**0.00** ± 0.00	**0.00** ± 0.00	**0.00** ± 0.00	**0.00** ± 0.00	**0.17**

Table 4.3 – Quality evaluation of our proposed methods for eigenspace estimation and λ_k estimation. For all experiments, the following parameters were used: the order of the polynomial approximation $c = 500$, $k = 25$, $\varepsilon = 10^{-1}$ for the standard eigencount method and the maximum number of iterations in fast is 10. Bold face numbers are the best score between two eigencount methods. The last column is the average over all graphs. The table summarizes results of mean energy conservation (ME), average number of iterations of the eigencount (IT) and eigencount difference with the target (KD).

eigencount rarely converged. This result can be easily explained by the fact that the eigenvalue distribution for SBM is known to be highly non-uniform, especially for low frequencies, which is partly incompatible with the local uniformity hypothesis assumed by the fast method.

On the lower part of Table 4.3 the precision of the estimated k (KD) is reported. We compute simply here the difference between the expected k (i.e., 25 in this experiment) and the resulting k given the approximated threshold λ_k. Both the fast and standard method converge most of the time, with a better overall convergence of the former which converges exactly to the true value except for SBM. This could be expected from the high number of iterations needed for this specific graph.

From those results, we can see that the quality of the estimated subspace computed using our proposed method is decent, while not perfect. The imprecision coming both from the approximation in the filter design and cut-off eigenvalue estimation. Our scheme for accelerated λ_k estimation is faster than the reference method and provides very good results.

4.3.3 Complexity analysis

Algorithm 5 summarizes the steps of our method to approximate the Laplacian eigenspace $\mathbf{U}_k$ from data points. If a graph is not provided with the data, a k-NN graph can be constructed and its associated Laplacian computed beforehand. The algorithm takes a graph and a number k as input and outputs a set of k approximated eigenvectors of the graph Laplacian.

Steps 1 and 3 of Algo. 5 are nonsignificant in the analysis of the overall complexity. We focus here on steps 2, 4 and 5 for which the number of operations is studied in detail. Using fast filtering operations, applying our method consists of k graph filtering operations at step 4, which requires $\mathcal{O}(cMk)$ operations, with c the order of the polynomial approximation of the filter. The SVD performed in step 5 has an additional cost of $\mathcal{O}\left(k^2N + k^3\right)$ for a tall matrix of size N by k like here. Finally, step 2 takes $\mathcal{O}(cMk)$ if we consider the accelerated method proposed in Section 4.3.2. Thus, the overall complexity of our method is $\mathcal{O}\left(cMk + k^2N + k^3\right)$.

Comparision with IRLM [23]

As reminded above, the complexity of IRLM is $\mathcal{O}\left(h(Mk + k^2N + k^3)\right)$ with h a convergence factor. Thus, assuming h and c have similar orders, the IRLM needs at least $\mathcal{O}\left((h-1)(k^2N + k^3)\right)$ more operations than our method. In any reasonable application, we will have either $k < N$ or $k \ll N$, thus, the term $\mathcal{O}\left(hk^2N\right)$ will be larger than the term $\mathcal{O}\left(hk^3\right)$.

Comparison with CSC [132]

Although the method presented in CSC is not directly an eigenspace estimation method, both CSC and FEARS might be used to approximate SC. We study now if it is computationally more

efficient to use FEARS and k-means than CSC for this task.
For CSC, the number of filtering needed is d, which has to be larger than a threshold given by results presented in Thm. 3.2 and 3.4 of [132]. To simplify, we can say that $d = \gamma \log(\alpha k \log(k))$ where γ and α are influenced by the precision of the distance preservation and the probability that the distance is preserved. Note that even with medium precision (e.g., in the order of 10^{-1}), the constants γ and α will be large (i.e., in the order of 10^3). This means that the overall complexity for the features estimation will cost $\mathcal{O}(cMd)$ operations. To this cost, one needs to add $\mathcal{O}\big(k\log(N)\big)$ filterings required to estimate λ_k have an added cost of $\mathcal{O}\big(cMk\log(N)\big)$.

If we compare the complexity of our proposed method with CSC for clustering, we need to consider the k-means computation on the different features. Using the compressive scheme defined in 3.2 for both, we have to add $\mathcal{O}\big(ik^3\alpha\log(k) + cMk\big)$ operations for FEARS and $\mathcal{O}\big(ik^2d\alpha\log(k) + cMk\big)$ for CSC, where i is the number of iterations of k-means. We now estimate the difference of number of operations, neglecting the impact of the constants in the Landau notation:

$$\begin{aligned}\Delta &= cMk + k^2N + k^3 + ik^3\alpha\log(k) + cMk - cM(d + k\log(N)) - ik^2d\alpha\log(k) - cMk \\ &= cM(k - d - k\log(N)) + k^2N + k^3 + ik^2\alpha\log(k)(k-d) \\ &= cM(k - \gamma\log(\alpha k\log(k)) - k\log(N)) + k^2N + k^3 + ik^2\alpha\log(k)(k - \gamma\log(\alpha k\log(k)))\end{aligned} \tag{4.10}$$

For sparse graphs we can assume $M = \overline{d_{\mathcal{G}}}N$, with $\overline{d_{\mathcal{G}}}$ the average node degree, which gives:

$$\Delta = c\overline{d_{\mathcal{G}}}N\Big(k - \gamma\log\big(\alpha k\log(k)\big) - k\log(N)\Big) + k^2N + k^3 + ik^2\alpha\log(k)\Big(k - \gamma\log\big(\alpha k\log(k)\big)\Big). \tag{4.11}$$

In order to finish the comparison, we now need to make hypotheses on the relation between k and N.

If we assume that $k \propto \log(N)$, then, for N large

$$\begin{aligned}\Delta &= c\overline{d_{\mathcal{G}}}N\Big(\log(N) - \gamma\log\big(\alpha k\log(k)\big)\Big) + N\log^2(N)\left(1 - c\overline{d_{\mathcal{G}}}\right) + o(N) \\ &< 0,\end{aligned} \tag{4.12}$$

with the last step following from the fact that $c\overline{d_{\mathcal{G}}} > 1$ and $\mathcal{O}\big(\log(N)\big) \subset \mathcal{O}\big(\log^2(N)\big)$. This means that for this regime, our method is cheaper than CSC, for large N.

If we assume that $k \propto \sqrt{N}$, then, for N large

$$\begin{aligned}\Delta &= c\overline{d_{\mathcal{G}}}N(\sqrt{N} - \gamma\log(\alpha N^{\frac{1}{2}}\log(N^{\frac{1}{2}})) - \sqrt{N}\log(N)) + N^2 + o\left(N^{\frac{3}{2}}\right) \\ &> 0,\end{aligned} \tag{4.13}$$

with the last step simply following from $\mathcal{O}\left(N^{\frac{3}{2}}\right) \subset \mathcal{O}\big(N^2\big)$. This means that for this regime CSC will be cheaper than our method for large enough N surpassing the impact of γ and α which

Algorithm 5 Fast Eigenspace Approximation using Random Signals (FEARS)

1: Generate $\mathbf{R}$ with $d = k$ cf. Section 4.2.1
2: Estimate λ_k cf. Algorithm 4
3: Compute the approximated graph filter g cf. Section 4.3.1
4: Apply filtering: $\Psi = g(\mathbf{L})\mathbf{R}$
5: Compute an economic SVD: $\mathbf{BSV} = \mathrm{SVD}(\Psi)$
6: Return the left singular vectors $\mathbf{B}$

are hard to apprehend. A real-world experiment comparing the timing of CSC and FEARS for clustering is summarized in Table 4.4.

From the two cases described above we can assess that if $k \in \mathcal{O}\left(\log(N)\right)$ our method is cheaper and if $k \in \Omega\left(\sqrt{N}\right)$ then CSC is cheaper. Note that in both cases the order of the filter c was kept constant, but that it benefits to our method to pick a large polynomial degree (e.g., $\Delta < 0$ when $c \propto N$).

4.3.4 Experimental analysis of timing

Since the complexity analysis in Section 4.3.3 only covers asymptotically large N, it is also interesting to look at the cost of the algorithms for actual implementations and realistic values of N and k. In addition to the eigenspace estimation with IRLM (eigs) and the k-dimensional spectral features of Compressive Spectral Clustering (CSC) mentioned in Section 4.3.3, we consider the power method described in [20] (power).

The data on which the different methods are evaluated consists of N points of small intrinsic dimension which are randomly drawn. In addition, a k-NN graph with 10 neighbors is constructed from the data points. Each method is run with fixed parameters and the time is measured in total CPU time to completion. The results of the experiments can be seen in Fig. 4.3.

Fig. 4.3a shows the time needed in function of k with N fixed and for small values of k. The first note is that the power method does not scale well with k and is exceedingly time-consuming for everything other than very small values of k for which it performs well. Since it is order of magnitudes slower for the parameters used in the other experiments, it is not displayed in the remaining figures to keep readability. Fig. 4.3b is the same as Fig. 4.3a for larger values of k. We see that, as expected in accordance with the complexity analysis, above a threshold corresponding to $\sqrt{N}$ (i.e., 100), our method performs better than eigs and worse than CSC.

Fig. 4.3c shows the results for an exponentially growing N and $k = \log(N)$. In this regime, our method outperforms both eigs and CSC for all values. The regime $k = \sqrt{N}$ is presented in Fig. 4.3d where we can see that our method performs best up to $N = 10^6$. Above this value, CSC is faster. Note that results above $N = 10^6$ for this regime are not shown due to memory limitations for eigs.

Combined, those results confirm the conclusions drawn from the complexity analysis of Section 4.3.3. First, except for very small values of k, IRLM is the most time-consuming method, even though it benefits from very optimized implementations. Second, for the $\log(N)$ regime, our method performs best for all values of N. For the $\sqrt{N}$ regime, our method is cheaper than CSC for $N < 10^6$. Above the limit $k = \sqrt{N}$, CSC is the fastest method. As a final remark on these results, we need to point out that, contrarily to the other methods considered in this experiment, CSC does not compute an eigensubspace *per se* but only k-dimensional features allowing good pairwise distance measurements between data points.

As a last remark on timing, we want to call attention to the fact that when filtering multiple random signals, all filtering operations are independent. Indeed, the signals are independent by definition and both the polynomial coefficients of the filter and the Laplacian are unaltered by the successive filtering operations. Our algorithms could thus easily benefit from a parallel implementation.

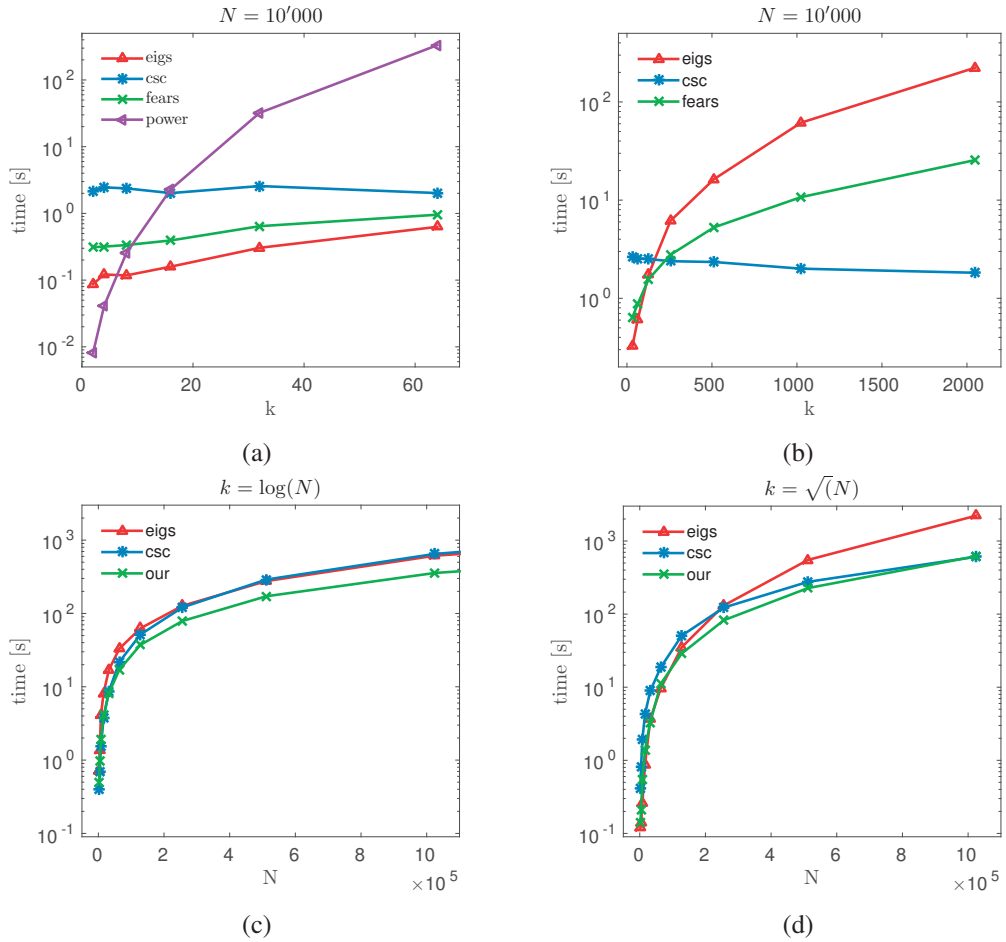


Figure 4.3 – Comparison of CPU time needed between different methods for the estimation of an eigensubspace of dimension k. In (a) and (b) N is fixed and k increases. In (c) and (d) k varies in function of N in two regimes ($k = \log(N)$ and $k = \sqrt{N}$ respectively). Time axes are in log-scale.

4.4 Applications of fast eigenspace estimation

In this section, we provide experiments whose objective is to show how our proposed methods behave in practice. We want to ensure that our proposed algorithms do fulfill their goals, i.e., that they provide accurate enough results and do so efficiently. Both as illustrations and practical applications, we show the performance of our eigenspace approximation method on typical clustering and visualization tasks.

4.4.1 Clustering

This experiment proves the capability of our filtered signals (FEARS) to produce an assignment for the data points. We will compare the results obtained by our method to Spectral Clustering (SC) (Sec. 2.2.5, [122]) and Compressive Spectral Clustering (CSC) (Sec. 3.2, [132]). We will also see that the compressive step of the latter can be used with k filtered signals instead of d.

Synthetic case with the Stochastic Block Model

For this experiment, we use an SBM with $N = 5'000$ nodes and $k = 20$ clusters. We set the average degree of the nodes to $\overline{d_{\mathcal{G}}} = 16$ and the nodes are associated at random with a particular class (the ground truth for the assignment). We generate several graphs with different ratios $\varepsilon = \frac{q_2}{q_1}$ (the larger ε, the harder the community detection) to evaluate our clustering capabilities in the task.

The evaluation of the presented methods is performed using the adjusted Rand similarity index [62] between the SBM ground truth and the resulting assignments, following the methodology used by Tremblay et al. [132]. All results presented here are averaged over 50 realizations in each setup. By looking at Fig. 4.4 we can first observe that our method is the one that best approximates the results of SC. It is not necessarily the method achieving the best rand index as ε increases but the ground truth is set before the edges are created. Thus, for relatively large values of ε, it might not make sense to keep this assignment for clustering purposes. In our view, spectral clustering is the target to fit at best. Moreover, notice that the order of the polynomial approximations alters the result of the clustering in both our method and CSC. Finally, CFEARS represents the result of our features assigned with the compressive step of CSC instead of the full k-means. We see that k-means is more faithful to spectral clustering than the regularized label diffusion on the graph.

Real-world example: Amazon co-purchasing network

In addition to the synthetic SBM graphs, we want to go further and show that FEARS also works well for real-world data sets. To this end, we consider the problem of clustering the Amazon co-purchasing network [140] that has also been evaluated for the study of CSC. The

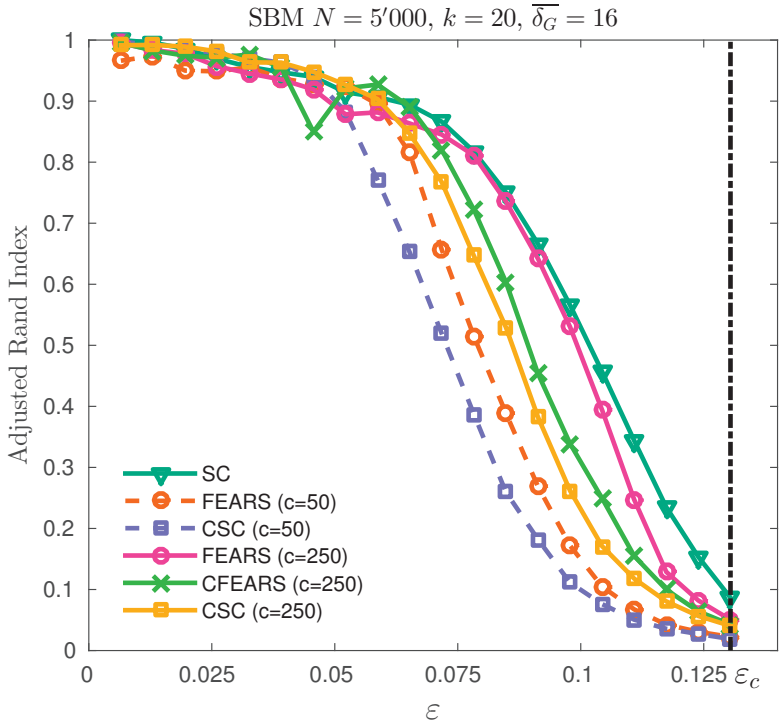


Figure 4.4 – Study of the clusterability of Stochastic Block Models for various values of ε, representing how well the graph can be split into clusters. Our method is the best to approximate the result of spectral clustering.

graph is composed of 334'863 nodes and 925'872 edges.[5] No clear ground truth can be used for comparison since the provided labels are the belonging of products to categories with overlaps. We decided to reproduce the experiment published in [132], adding our method to the benchmark. We measured the resulting assignments with two measures: the modularity score [97], used to determine whether a given partition is separating the network efficiently (see Sec. 2.2.4), and the adjusted Rand similarity index compared to the result of SC, used to identify the resemblance of the two assignments. This experiment is among the very few that have not been repeated several times due to its computational load.

In Table 4.4 we first show the performance of the different algorithms with three different cluster sizes: 250, 500 and 1'000. We split the timing into two parts, one for the feature extraction process and the other for the assignment based on these features. We see that consistently the features extracted using random signal filtering are faster to compute than those requiring partial eigendecomposition. We also notice that until $k = 500$, k-means is an efficient method for the assignment of the points to the clusters, it is even 5 times faster than the compressive assignment for $k = 250$ in our experiment. However, when k becomes large, using the compressive method of CSC (also applied in CFEARS) is helping greatly to reduce the overall time of the computation,

[5] Available at http://snap.stanford.edu/data/com-Amazon.html

earning a factor 2 speedup between FEARS and CFEARS. Intuitively, we would expect the assignment (k-means) step of SC and FEARS to take the same amount of time since their complexity is the same while CSC and CFEARS should also have similar timings for the assignment, with CFEARS slightly faster because the number of features is smaller. All the variations that do not follow these trends are due to the convergence speed of the given instances with the random initialization that happened at the time of the experiment.

Next, we consider the efficiency of the clustering reported in Table 4.5, where two important observations stand out. On the one hand, the best modularity is achieved using CSC and we see that our method, with the use of the compressive step, tends to similar results with increasing k. This is explained by the diffusion step being performed at the very end of the compression. Since some nodes get their assignment based on the graph structure, the compressive assignment is probably more robust than a full k-means. On the other hand, the adjusted Rand similarity index clearly shows that our method is the one assigning the nodes the most similarly to SC. This is an expected behavior since the goal of our method is to reconstruct the set of the k first eigenvectors used as features in SC.

	$k = 250$	$k = 500$	$k = 1\,000$
SC	14.37min + **2.13h**	25.09min + **14.96h**	55.63min + 106.87h
FEARS	**0.12min** + 2.55h	**0.19min** + 22.75h	**0.52min** + 104.82h
CFEARS	**0.12min** + 11.36h	**0.19min** + 17.22h	**0.52min** + 58.46h
CSC	2.34min + 9.74h	3.73min + 21.07h	2.61min + **35.47h**

Table 4.4 – Timing of clustering for Amazon data set. Values are averages over several experiments and the order of the polynomial approximation is $c = 500$. Each experiment is split into two steps: features extraction (in minutes), and the assignment from the features to a cluster (in hours).

	SC	FEARS		CFEARS		CSC	
	mod	mod	rand	mod	rand	mod	rand
$k = 250$	0.344	0.387	**0.884**	0.588	0.711	**0.764**	0.509
$k = 500$	0.507	0.605	**0.818**	0.759	0.677	**0.818**	0.586
$k = 1'000$	0.663	0.638	**0.851**	**0.815**	0.780	0.798	0.749

Table 4.5 – Evaluation of clustering for Amazon data set. Values are averages over several experiments and the order of the polynomial approximation is $c = 500$. FEARS mimics SC the best (high rand score) while compressive k-means helps raising the modularity.

4.4.2 Visualization

In this last experiment, we show how our method can be used in the context of visualizing high-dimensional data, since eigenspaces are commonly used for dimensionality reduction in this context. We wish to see how our proposed method behaves first in a very simple synthetic

example and second for real-world data sets of larger size. For this task we compare the following visualization algorithms:

Laplacian eigenmaps Belkin and Niyogi [12] proposed to solve the generalized eigenvalue problem $\mathbf{L}\mathbf{y} = \lambda D\mathbf{y}$ where y is called the Laplacian eigenmaps. This method is interesting to validate the fact that our method finds a good approximation of $\mathbf{U}_k$ because it finds the eigenspace of the random walk Laplacian. Indeed, if we define the random walk Laplacian as $P = D^{-1}\mathbf{L}$ then the equation above can be rewritten as $P\mathbf{y} = \lambda\mathbf{y}$. Thus, Laplacian eigenmaps aims at finding the eigenspace of P and use it as an embedding for visualization. We implemented the method in Matlab with the eigs eigensolver which uses the IRLM algorithm.

t-SNE [83] a famous state-of-the-art technique for visualization which enhanced the Stochastic Neighbor Embedding method [59]. The use of a heavy-tail distribution for the embedded points probabilistic model allows avoiding the crowding effect and at the same time gives rise to an easier optimization problem. The original implementation having an $\mathcal{O}\left(N^2\right)$ complexity, the Barnes-Hut accelerated version is often used for large data sets since it has a $\mathcal{O}\left(N\log(N)\right)$ complexity. We used the C++ implementation of the Barnes-Hut t-SNE for our experiments.[6]

LargeVis [130] a recent technique based on graph visualization which aims at solving the scalability problems of state-of-the-art methods such as t-SNE. Its first contribution is to accelerate the graph construction step by using an approximated k-NN graph construction method. Second, it formulates the embedding problem as a probabilistic model which keeps similar vertices close to each other and dissimilar vertices apart. Inspired by negative sampling techniques they propose to optimize the probabilistic model using independent stochastic gradient descent steps. The C++ implementation of the algorithm was used for the experiments.[7]

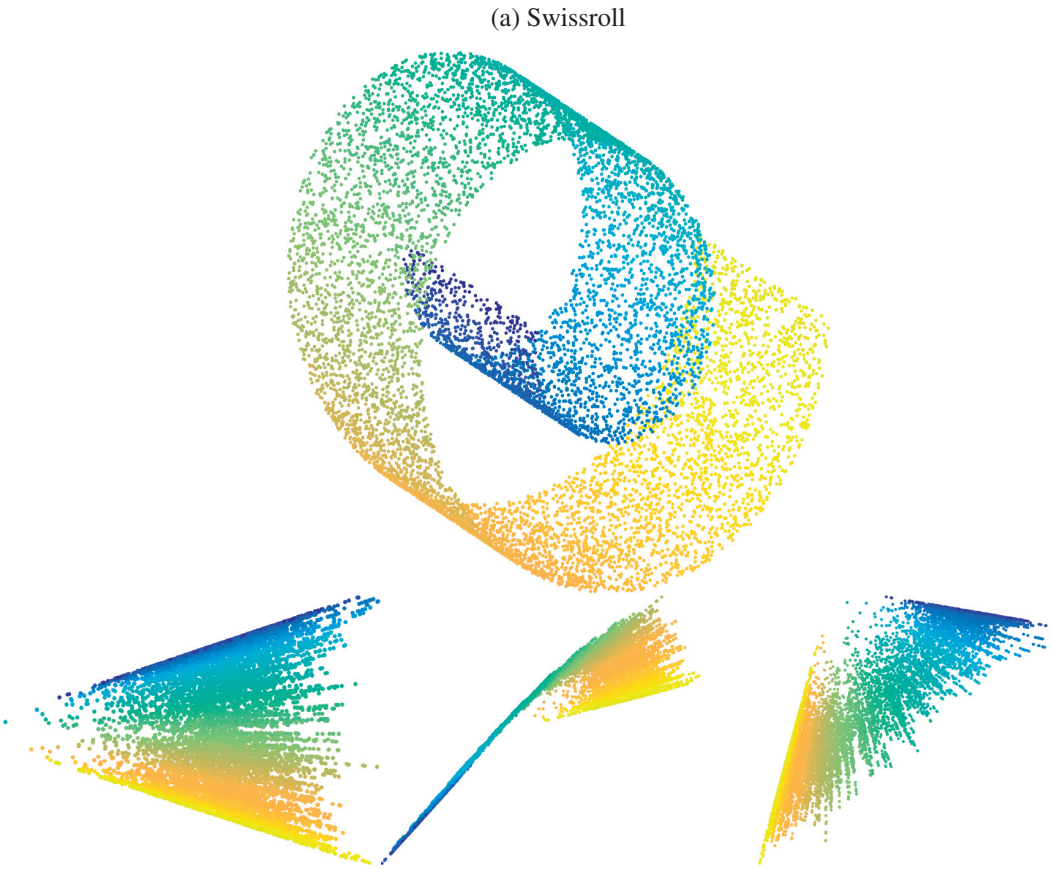
Toy example: the Swissroll

In this first small experiment, we wish to assess the validity of using our proposed method of eigenspace estimation for visualization on a simple toy example. We will compare the results obtained by our method only with Laplacian eigenmaps as we would like to verify that we get similar results.

For this experiment, we use a classical Swissroll graph for which we compute a 2 dimensional embedding. The Swissroll is computed by sampling its continuous manifold in the following way: given a set of randomly drawn angles θ in $[a\pi, b\pi]$ the coordinates are set as $x = \theta\cos(\theta)$, y drawn uniformly in $[0, 1]$ and $z = \theta\sin(\theta)$. A k-NN graph with 10 neighbors is constructed from the data points. For this experiment, the normalized Laplacian was used for all methods.

The resulting embeddings are shown in Fig. 4.5. The color map is a linear function of θ. The first

[6] Available at https://github.com/ninjin/barnes-hut-sne
[7] Available at https://github.com/lferry007/LargeVis



(a) Swissroll

(b) Laplacian eigenmaps (c) FEARS with Ψ (d) FEARS with $\mathbf{B}_k$

Figure 4.5 – The Swissroll point cloud (a) with $10'000$ nodes and its 2D embeddings using Laplacian eigenmaps (b), our proposed fast eigenspace estimation method prior to the SVD step (c), and after the SVD step (d).

thing to notice is that all embeddings are very smooth with respect to θ. The second interesting observation is that $\mathbf{B}_k$ indeed seems to be a good approximation of the Laplacian eigenmaps up to a rotation because they display very similar shapes. This tends to validate that the method indeed provides a good approximation of $\mathbf{U}_k$. In addition, in this specific example, while embedding with Ψ gives a smooth result, the normalization step provided by the SVD enhances significantly the quality of the visualization. This observation makes sense since few random signals are used to compute Ψ for the visualization ($d = 2$ or 3), which, as discussed in Section 4.2.2, is not sufficient to have an expectation effect smoothing the variance on the eigenvalues. This scaling is normalized by the final SVD step, which is not costly for visualization tasks since k is very small.

Real-world data sets

In this second half, we consider large scale real-world examples and compare our method with existing approaches. We use the two following data sets:

- **MNIST:** a well-known data set of handwritten digit images, from which we take all $70'000$ data points .[8] All points are labeled with the digit it represents.

- **LiveJournal:** a dataset from the LiveJournal social network. The graph used in the experiment is the largest connected component of the complete graph which has $3'997'962$ nodes.[9]

In Fig. 4.6 we can see the visualizations of the MNIST data set, where the color map is derived from the labels. The first observation is that both Laplacian eigenmaps and our proposed method yield similar results. Both do not achieve a very good separation of the classes and suffer from a concentration around the origin (i.e., the crowding problem). Our method seems to do a slightly better job at separating the classes in the middle than Laplacian eigenmaps. The embeddings provided by both t-SNE and LargeVis are of much greater quality with respect to class separation even if they leave outliers. Also, both methods find 11 clusters instead of 10 as they split one class into two clusters. LiveJournal was not displayed due to the very large number of nodes to represent.

In Table 4.6 we report the time needed to compute the embeddings using the methods above on the two datasets. On MNIST, our method has the lowest CPU time, closely followed by Laplacian eigenmaps. Our method is one order of magnitude faster than t-SNE, which is twice slower than LargeVis. On LiveJournal, our method is still the fastest and one order of magnitude faster than t-SNE. LargeVis, while being slower than our method, performs rather well. Laplacian eigenmaps exceeded the available memory and did not complete.

	Eigenmaps	t-SNE	LargeVis	FEARS
MNIST	0.06	0.46	0.26	**0.04**
LiveJournal	OoM	41.70	0.91	**0.26**

Table 4.6 – Timing of 2D visualization of real-world datasets comparing FEARS with state-of-the-art methods. Times are given in hours, the bold face numbers highlight the fastest method. Eigenmaps on LiveJournal did not terminate and exceeded the maximum memory available (128 GB).

From these results we can say that our method is valid for visualization but cannot achieve a quality close to state-of-the-art methods such as t-SNE or LargeVis. However, it has the advantage to be fast and scales well even using a non-optimized mono-thread implementation.

4.5 Conclusion

In this chapter, we have presented a theoretical way to recover the set of k smallest eigenvectors of a graph Laplacian. We have shown an accelerated algorithm for the approximation of the

[8] Available at http://yann.lecun.com/exdb/mnist/

[9] Available at http://snap.stanford.edu/data/com-LiveJournal.html

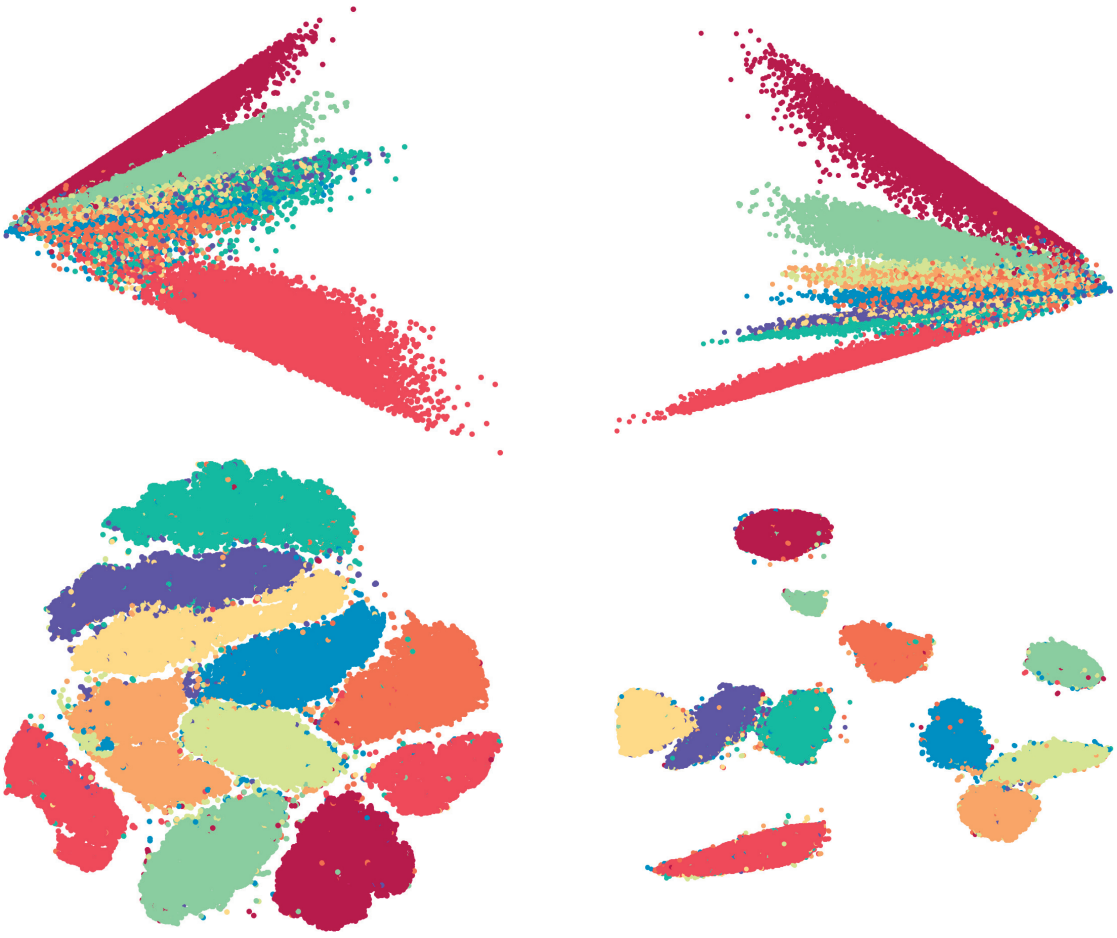


Figure 4.6 – Visualizations of MNIST using (a) Laplacian eigenmaps, (b) our method, (c) t-SNE (with Barnes-Hut implementation) and (d) LargeVis. The colors correspond to the different categories (i.e., numbers from 0 to 9).

eigenspace of the Laplacian $\mathbf{L}$ solely based on Gaussian random signals filtering. We proved the bound on the number of signals to be as tight as ever possible. In addition, we proposed an accelerated eigenvalue estimation algorithm based on eigencount techniques. We presented different applications and compared the efficiency against the state of the art, showing the ability for our method to scale with very large N.

This is an interesting result for the field of graph signal processing and many further questions arise in this context. Among them, the design of the filter could be reconsidered. Could we gain even more efficiency by using a naturally polynomial function for the filter instead of the approximation of an ideal low-pass filter? We suggest using exponentially decreasing kernels, which are low-pass and infinitely differentiable and will assign to the eigenvalues an energy proportional to its position in the spectrum. One could wonder whether such design could allow stopping the computation before the SVD step.

Regarding the practical aspects, the eigencount method introduced by Di Napoli et al. [39]

suggested the use of either polynomial approximations with Chebyshev approximations or rational functions. The latter showed to be more accurate for the estimation of the number of eigenvalues in an interval during their experiments. One could also study the impact of rational filters every time $g(\mathbf{L})$ is required. Following the works proposing ARMA graph filters [82], one could benefit from an alternative construction of the filters that is more involved but converges better to the ideal solution.

5 Filter Localization for Graph Clustering

We have studied deeply the positive outcomes of random filtering in the previous chapters and observed how efficient it can be to estimate the assignments of SC. After focusing on the determination of the features of SC, we aim now at presenting an alternative to the computation of k-means. As we saw previously in Chapter 3, k-means is an approximated algorithm for the resolution of the optimization problem looking for k ideal representatives among N points one wants to separate, i.e., those minimizing the sum of distances from any point to the closest representative. There, we highlighted the lack of reliability of k-means by the fact that CSC is able to produce an assignment of better quality than SC with respect to the k-means cost, which is simply impossible in theory if we assume that k-means returns the optimal assignment.

The proposed method makes use of $\mathcal{T}_i g$, the localization operator defined in Sec. 2.1.5. Our approach is to consider the diffusion process over the graph to transport assignment information on the graph making use of a very sparse sampling scheme. The relation between the diffusion and the localization operator will be presented, along with the other previous related works. In particular, we recall the concepts of graph spectrogram and active graph sampling that are two corner stones of the proposed method for *light clustering*, our proposed method.

This chapter proposes the following contributions to the field:

1. *We suggest in a first place that diffusion can allow to propagate the assignment from the centroids of k-medoids to the rest of the graph.* We experiment with this intuition and relate our diffusion method to the optimization problem of k-medoids. For some graphs, we show that we are even able to perform better than k-medoids with 100 replications.

2. *Then, we present two different methods in order to find the most central nodes of the graph.* We additionally prove the relation between node centrality and graph localization, establishing an equality for some specific graph filters.

3. *Finally, we propose a new algorithm for clustering, which is computationally cheap, based on the graph localization operator.* The idea is to combine a fast procedure to extract the

most central nodes with the propagation of their label to the rest of the graph. Both steps are performed using the computation of the same k graph filterings.

5.1 Related works

5.1.1 Sampling clustering

This chapter focuses on the methods for graph clustering based on sketching and sampling. The idea of approximating the distances between points in order to lower the dimensionality of the problem has been extensively researched and found its most recent development with the theory of *compressive sensing*. Using this theory, Park [104] proposed to reduce the data dimensionality while preserving the information (i.e., ensuring perfect reconstruction of the original data points) with the application of a sampling function. Her goal is to perform the clustering step in the sampled feature space to reduce the computations and transpose the solution on the original feature space at the very end.

Ruta et al. [116] also proposed a method based on compressive sampling where they use the data points' features as a dictionary and alter the representation vectors based on the classes of the data points. Following Lloyd's algorithm they iteratively improve their assignment until convergence similarly to Park.

These works also resemble [131] where the authors randomly sampled features of interest in order to reduce the complexity of k-means. Overall, the goal is always to reduce the dimensionality and transfer the problem into one of smaller size to apply k-means.

The same techniques were experienced with graph clustering by additionally computing spectral features before the k-means step. Among them is CSC of Tremblay et al. [132] and the random sketching algorithm of Gittens et al. [52]. Both use random projections on the graph matrices to approximate the first eigenvectors and compute spectral clustering. On top of this technique, Sakai et al. [118] as well as Hunter et al. [63] proposed simultaneously two methods where the reduction is performed on the data points before the construction of an affinity matrix between samples, respectively the graph adjacency matrix. The latter paper features a comprehensive theoretical analysis of the impact of the compression pipeline on spectral clustering.

Despite all the efforts made to reduce the feature space's dimensionality, another item could be compressed in the procedure: the vertices. Indeed, if one could solve the problem on a subpart of the entities and extend the result to the full problem, the complexity would be significantly reduced. In [30], subsets of the nodes are sampled and combined to determine the assignment. Unfortunately, the author showed that a very high number of samples are required to determine the partition correctly.

Finally, close to our approach, the work of Chung et al. [32] proposes to revisit Lloyd's algorithm, replacing the Euclidean space with personalized PageRank vectors. Under certain assumptions,

they define the k centroids among $\mathcal{O}\big(\log(N)\big)$ candidate sets and assign a class to each node as in k-means using the PageRank distance instead of the $\ell 2$-norm.

5.1.2 Applications of the localization operator

The localization operator introduced in Sec. 2.1.5 served various applications in recent years. First, Shuman et al. [126] used it to emphasize the local differences between vertices in the graph. Using various filters of different natures, they filtered δ_i for all nodes i and compute the norm $\left\|\mathcal{T}_i g\right\|_2^2$. This quantity allows to highlight the role of the node in the graph. Actually, contrary to the traditional Fourier transform where eigenvectors are un-localized, here all frequencies are not shared equally among the nodes and some eigenvectors may be concentrated around some of the nodes (i.e., they can have non-null energy for a small subset of the nodes only) [108]. This also implies that each node has its own profile when it comes to eigenvectors' energy repartition. Filtering Kronecker delta signals with a sharp band-pass filter allows to stress the presence of these given frequencies in this node's spectral components. Reproducing these filterings with different filters covering the full spectrum overall, they propose to run a vertex-frequency analysis of the graphs. Interestingly, the spectrogram can be able to identify clear structure in the graph.

Later, Perraudin [105] continued to study their benefit for graph clustering in his thesis. In particular, highly structured networks possess few frequency profiles shared by nodes with the same context. Although it depends significantly on the degree of the nodes, the local context required to identify the prominent frequencies goes beyond degree. Indeed, in his report, Perraudin used a 3-regular tree where all nodes (except the root) have the same direct neighborhood. However, nodes of each layer of the tree have different profiles while all nodes in the same layer are identical in that respect. His structural clustering algorithm emerged from these observations. In the process, the author improved the running time of spectrogram construction by proposing an approximated method estimating $\left\|\mathcal{T}_i g_k\right\|_2^2$ for all nodes with $\frac{1}{T}\sum_{t=1}^{T} g_k(\mathbf{L})\boldsymbol{w_t}$ where $\boldsymbol{w_t}$ are i.i.d. random centered Bernoulli vectors with $p = 0.5$. Overall the construction of a spectrogram reduced from $\mathcal{O}(KNMO_c)$ to $\mathcal{O}(KTMO_c)$ where K is the number of filters, T the number of filterings and O_c the order of the polynomial approximation.

Interestingly, an adaptive sampling strategy using the generalized localization operator $\mathcal{T}_i g$ has recently been proposed [103]. The authors suggested to cover the graph with a minimal quantity of energy at every node by sampling iteratively at different positions and summing the contributions of the $\mathcal{T}_i g$ for these nodes i. The purpose was to define a distance metric between key points of interest instead of computing the full matrix of $\mathcal{O}\big(N^2\big)$ distances. For transductive learning purposes (learning the full signal while possessing information for some of the nodes only), Paratte [102] also introduced the *localized kernel distance* between two nodes as $\mathrm{LKD}(i,j) = 1 - \frac{\mathcal{T}_i g^2[j]}{\|\mathcal{T}_i g\|\|\mathcal{T}_j g\|}$ helping him to interpolate the missing values.

5.2 Avoiding k-means with filter localization

The goal of this chapter is to present a new clustering algorithm combining the benefits of the localization operator and the sampling technique proposed by Puy et al. in [112]. The reunion of these two objectives produced a technique similar to that of Chung et al. [32] mentioned above. In fact, the idea is to replace k-means whose two components are the determination of the centers and the propagation of the information in the neighborhood of these centers. It seemed judicious to use the theory of Puy for the diffusion of the information, replacing the former component of k-means, but the connections between $\mathcal{T}_i g$ and the diffusion distance are also very promising. For the latter aspect, our findings based on the spectrogram works highlighted an interesting property of $\mathcal{T}_i g$ for the characterization of core nodes in a network. In this section, we dive into the details of our propositions for each of these two aspects.

5.2.1 Propagating clustering information from key nodes

Localizing a filter around a given node (see eq. (2.41)) can allow the propagation of the information from that node in its vicinity when computed with an appropriate filter. Low-pass filters such as heat kernels were prominently pushed forward for this application for at least two reasons. First, from a physical point of view it represents the heat diffusion on the continuous manifold sampled with the graph. Second, mathematically, the convolution of the heat kernel with diracs produces only positive values and, moreover, $\left\|\mathcal{T}_i g\right\|_1 = 1$ which facilitates computations in some

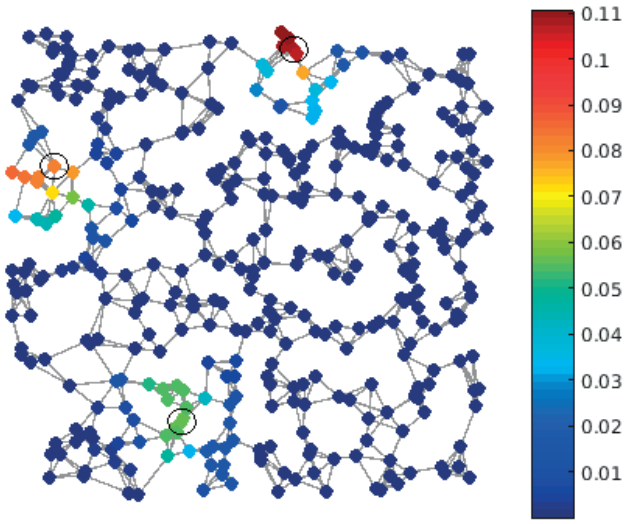


Figure 5.1 – Information diffusion using heat kernel on a Sensor graph. The signal represents the filtering of three Kronecker deltas. The propagation in the vicinity of central nodes (surrounded in black) is the corner stone of our efficient method of clustering.

experiments (see [102, 105] for the details).

Knowing the class representatives, we argue that the diffused labels accurately estimate the result of the assignment step of k-means. We admit that the complexity of the graph diffusion by itself is similar to k-means but it serves a bigger picture that we will introduce step by step. Computing k-medoids on a graph, we use the centroids as an initial subset of nodes from which to propagate information, we then filter the k Kronecker delta vectors on the full graph and obtain k diffusion processes that should cover the whole graph. Finally, we assign to each node the class of the diffusion process that propagated the most information at this node. Note that this assignment has the advantage of being easily transformed into a fuzzy assignment (where nodes are assigned to each class with a probability). For this matter, one simply needs to renormalize the matrix of the k filtered signals so that each row sums to 1 independently.

Reproducing this experiment on different graphs, we observed results extremely close to that of k-medoids.[1] We conducted the experiment using two kinds of random graphs (sensor and SBM), constructed 100 realizations for each, both with $N = 10'000$ and $k = 5$. We extracted the first eigenvectors and ran k-medoids on it (i.e., the exact method). Then, we used the centers of k-medoids as the initialization of the diffusion process described above (i.e., approx). Table 5.1 shows that our method performs slightly better than the k-medoids approach with respect to both modularity and normalized cut measures. For SBM the gain is non-significant but for sensor graphs, it reduces almost by half the normalized cut score in average. Overall, since our goal was to verify that this method was decent at approximating k-medoids, we are rather satisfied with these observations.

Theoretically, this can be explained as follows. The k-medoids problem endeavors solving an optimization problem that reads

$$\operatorname{class}(v_i) = \operatorname*{argmin}_{\mu_c \in \mathcal{C}} \sum_{\ell=0}^{k-1} \big(\boldsymbol{u}_\ell[i] - \boldsymbol{u}_\ell[\mu_c]\big)^2 \tag{5.1}$$

$$\mu_c = \operatorname*{argmin}_{v_i \in \mathcal{V}:\operatorname{class}(v_i)=c} \sum_{\substack{j \neq i \\ \operatorname{class}(v_j)=c}} \sum_{\ell=0}^{k-1} \big(\boldsymbol{u}_\ell[i] - \boldsymbol{u}_\ell[j]\big)^2, \tag{5.2}$$

where $\mathcal{C}$ is the set of medoids defined by the second equation.

Let's assume for a moment that the class representatives are given and focus on the first equation, we have

$$\begin{aligned} \operatorname*{argmin}_{\mu_c \in \mathcal{C}} \sum_{\ell=1}^{k} \big(\boldsymbol{u}_\ell[i] - \boldsymbol{u}_\ell[\mu_c]\big)^2 &= \operatorname*{argmin}_{\mu_c \in \mathcal{C}} \sum_{\ell=1}^{k} \boldsymbol{u}_\ell^2[\mu_c] - 2\boldsymbol{u}_\ell[i]\boldsymbol{u}_\ell[\mu_c] + \sum_{\ell=1}^{k} \boldsymbol{u}_\ell^2[i] \\ &= \operatorname*{argmax}_{\mu_c \in \mathcal{C}} \sum_{\ell=1}^{k} \boldsymbol{u}_\ell[i]\boldsymbol{u}_\ell[\mu_c] - \frac{1}{2}\boldsymbol{u}_\ell^2[\mu_c]. \end{aligned} \tag{5.3}$$

[1] k-medoids was favored over k-means here because it had to be run in order to obtain the input of our method independently of the comparison.

On the other hand, the method that we propose is selecting the class following

$$\begin{aligned}\widehat{\text{class}}(i) &= \underset{\mu_c \in C}{\arg\max}\, \mathcal{T}_{\mu_c} g[i] \\ &= \underset{\mu_c \in C}{\arg\max} \sum_{\ell=0}^{N-1} g(\lambda_\ell) \boldsymbol{u}_\ell[\mu_c] \boldsymbol{u}_\ell[i] \\ &= \underset{\mu_c \in C}{\arg\max} \sum_{\ell=0}^{k-1} \boldsymbol{u}_\ell[\mu_c] \boldsymbol{u}_\ell[i],\end{aligned} \tag{5.4}$$

where the last step comes from the fact that we used an ideal low-pass filter $g(\lambda_\ell) = \begin{cases} 1 & \lambda_\ell < \lambda_k \\ 0 & \lambda_\ell \geq \lambda_k \end{cases}$.

We easily relate the two objectives and notice that the only difference is the term $\frac{1}{2}\sum_{\ell=1}^{k} \boldsymbol{u}_\ell^2(\mu_c)$. We expect the results of eq. (5.3) and (5.4) to provide identical solutions in most situations.

Figure 5.2 summarizes the behavior on a sensor graph of $N = 500$ nodes that we want to separate into three classes. First we observe that the difference of assignment between the two methods is limited. Most of the nodes belong to the same class in the two subfigures 5.2c and 5.2d. The main differences appear at the interface between the three partitions in the center of the plot. The differences can be observed in figures 5.2a and 5.2b that respectively represent the measure of eq. (5.3) and (5.4). The term $\boldsymbol{u}_\ell^2(\mu_c)$ which is the only to differ between the two equations does not impact the result most of the time. Its impact is only visible at the boundaries between the classes where the curves cross.

5.2.2 Defining appropriate initial vertices

Now that we have presented a way to propagate the assignment to the full graph, we need to be able to determine the initial centroids efficiently and accurately. In this section, we will first share a few experimental observations, explain them theoretically and propose an algorithm for the centroids selection.

	Modularity		Normalized cut	
	exact	approx	exact	approx
Sensor	0.909 (±0.000)	**0.921** (±0.000)	0.251 (±0.002)	**0.140** (±0.000)
SBM	0.248 (±0.000)	**0.248** (±0.000)	11.017 (±0.000)	**11.015** (±0.00)

Table 5.1 – Quality evaluation of the approximation of k-medoids with kernel diffusion centered at the representatives of k-medoids. The experiment has been performed on 100 random sensors and SBM graphs of size $N = 10'000$. The approximated version performs better than k-medoids with respect to modularity and normalized cut.

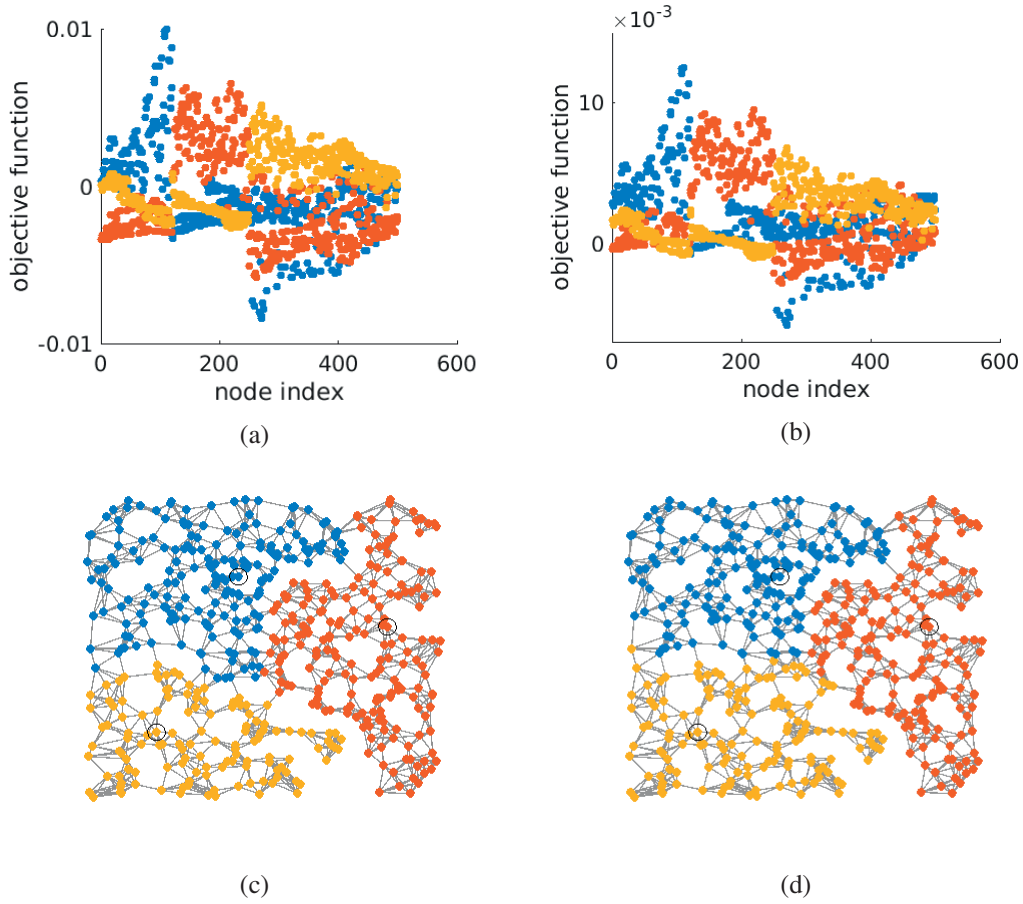


Figure 5.2 – Approximation of k-medoids with diffusion of Kronecker deltas with an ideal low-pass filter. Application on a sensor graph of $N = 500$ nodes. The top plots show the objective function for the assignment of the nodes with the clusters and the bottom ones the respective assignment. Left plots are computed using k-medoids for the assignment, while the right ones use the $\mathcal{T}_i g$ method.

Node centrality approximation

Experimentally, we noticed that central nodes react less to $\|\mathcal{T}_i g\|$ with low-pass filters g than the rest of the nodes. This is interesting and could help to define the nodes from which to start the diffusion. This idea is corroborated directly by the results of structural clustering. Since the nodes with similar $\mathcal{T}_i g$ in terms of Wasserstein distance possess similar roles in the network, finding the class of central nodes and interpreting the desired frequencies would also allow to cross-check this result.

Mathematically, we can relate the centrality and the norms of $\mathcal{T}_i g$. We first recall the kernelized diffusion distance from [102, eq. (2.19)] that compare the $\mathcal{T}_i g$ located at two different nodes for

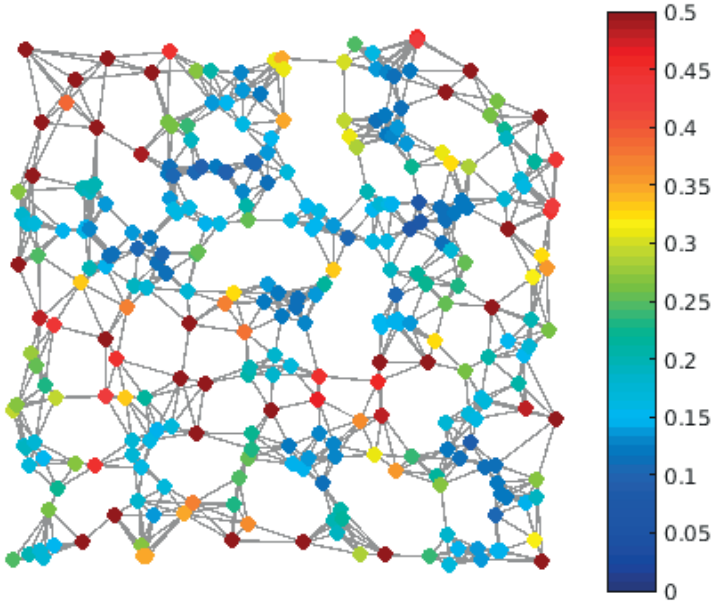


Figure 5.3 – Computation of the $\|\mathcal{T}_i g\|_2^2$ for all the vertices on a sensor graph with $N = 300$ applied with a low-pass filter. The most connected nodes are those with the lowest norm, while the isolated nodes get the highest values. A theoretical explanation is given later in this section.

a given kernel

$$\begin{aligned}\|\mathcal{T}_i g - \mathcal{T}_j g\|_2^2 &= \sum_{n=1}^{N} \left(\sum_{\ell=0}^{N-1} g(\lambda_\ell) \boldsymbol{u}_\ell[n] \left(\boldsymbol{u}_\ell[i] - \boldsymbol{u}_\ell[j] \right) \right)^2 \\ &= \sum_{\ell=0}^{N-1} \sum_{\ell'=0}^{N-1} g(\lambda_\ell) g(\lambda_{\ell'}) \left(\boldsymbol{u}_\ell[i] - \boldsymbol{u}_\ell[j] \right) \left(\boldsymbol{u}_{\ell'}[i] - \boldsymbol{u}_{\ell'}[j] \right) \underbrace{\sum_{n=1}^{N} \boldsymbol{u}_{\ell'}[n] \boldsymbol{u}_\ell[n]}_{=\delta(\ell-\ell')} \\ &= \sum_{\ell=0}^{N-1} g^2(\lambda_\ell) \left(\boldsymbol{u}_\ell[i] - \boldsymbol{u}_\ell[j] \right)^2. \end{aligned} \tag{5.5}$$

With this distance measure, we can compute the node centrality as the average diffusion distance to all other nodes in the graph

$$\begin{aligned}\frac{1}{N} \sum_{j=1}^{N} \|\mathcal{T}_i g - \mathcal{T}_j g\|_2^2 &= \frac{1}{N} \sum_{j=1}^{N} \sum_{\ell=0}^{N-1} g^2(\lambda_\ell) \left(\boldsymbol{u}_\ell[i] - \boldsymbol{u}_\ell[j] \right)^2 \\ &= \frac{1}{N} \sum_{\ell=0}^{N-1} g^2(\lambda_\ell) \sum_{j=1}^{N} \left(\boldsymbol{u}_\ell^2[j] + \boldsymbol{u}_\ell^2[i] - 2\boldsymbol{u}_\ell[i] \boldsymbol{u}_\ell[j] \right) \\ &= \frac{1}{N} \sum_{\ell=0}^{N-1} g^2(\lambda_\ell) \left(1 + N\boldsymbol{u}_\ell^2[i] - 2\boldsymbol{u}_\ell[i] \sum_{j=1}^{N} \boldsymbol{u}_\ell[j] \right). \end{aligned} \tag{5.6}$$

We know that $\boldsymbol{u}_0(n) = \frac{1}{\sqrt{N}}$ and $\langle \boldsymbol{u}_\ell, \boldsymbol{u}_m \rangle = \delta(\ell - m)$. Then if we set $m = 0$,

$$\begin{aligned} \langle \boldsymbol{u}_\ell, \boldsymbol{u}_0 \rangle &= \frac{1}{\sqrt{N}} \sum_{n=1}^{N} \boldsymbol{u}_\ell[n] = \delta(\ell) \\ \Rightarrow \sum_{n=1}^{N} \boldsymbol{u}_\ell[n] &= \sqrt{N}\delta(\ell) = \begin{cases} \sqrt{N} & , \ell = 0 \\ 0 & , \ell > 0 \end{cases}. \end{aligned} \tag{5.7}$$

We can thus express the kernel-diffusion centrality as

$$\begin{aligned} \frac{1}{N} \sum_{j=1}^{N} \left\| \mathcal{T}_i g - \mathcal{T}_j g \right\|_2^2 &= \frac{1}{N} \sum_{\ell=0}^{N-1} g^2(\lambda_\ell) \left(1 + N\boldsymbol{u}_\ell^2[i] - 2\sqrt{N}\boldsymbol{u}_\ell[i]\delta(\ell) \right) \\ &= \sum_{\ell=0}^{N-1} g^2(\lambda_\ell)\boldsymbol{u}_\ell^2[i] + \frac{1}{N} \sum_{\ell=0}^{N-1} g^2(\lambda_\ell) - \frac{2g^2(0)}{N} \\ &= \left\| \mathcal{T}_i g \right\|_2^2 + \frac{1}{N} \left(\sum_{\ell=0}^{N-1} g^2(\lambda_\ell) - 2g^2(0) \right). \end{aligned} \tag{5.8}$$

First we note that for all the nodes, the rightmost term will be constant for a given graph filter and always be independent of the node index i. Then, we can conclude that for graphs with a combinatorial Laplacian (required for the second step of eq. (5.8)) $\left\| \mathcal{T}_i g \right\|_2^2$ is an estimator of the kernelized centrality measure. Moreover, when using a low-pass filter g set to cancel this rightmost term, our estimator exactly computes the kernelized distance for each node v_i. We could then expect the nodes with smallest $\left\| \mathcal{T}_i g \right\|_2^2$ to be the most central ones.

We decided then to follow this approximation of the centrality of the nodes using $\mathcal{T}_i g$ for our light clustering algorithm. The single question that remains concerns the procedure for the determination of the centroids. Indeed, looking among the nodes with a low response to low-pass $\mathcal{T}_i g$ does not mean that picking the k largest values generates the best set of centroids. The reader is referred to Fig. 5.4 for an illustration. Here, the center of each cluster contains several nodes with a low $\left\| \mathcal{T}_i g \right\|$ in low frequency but the lowest values are all concentrated in a single cluster. We need then to select the nodes sequentially and penalize the nodes in the neighborhood of previously picked centroids to ensure a cover of the entire graph.

Selecting centroids

Two methods are proposed for the selection of the centroids. The first one is iterative and use mostly the concept of active sampling recalled in the related works section. The k centroids are selected one after the other. When a new centroid is chosen, a $\mathcal{T}_i g$ centered at this vertex is computed. We accumulate in each node the energy of the $\mathcal{T}_i g$s at each iteration and use this vector to define the position of the next centroid. We pick systematically the node with the lowest energy. This implies that after some point, all the nodes will have a centroid in a reasonably close neighborhood. The first node is picked using the centrality estimation described above. We call this method *energy-based determination* of the centroids.

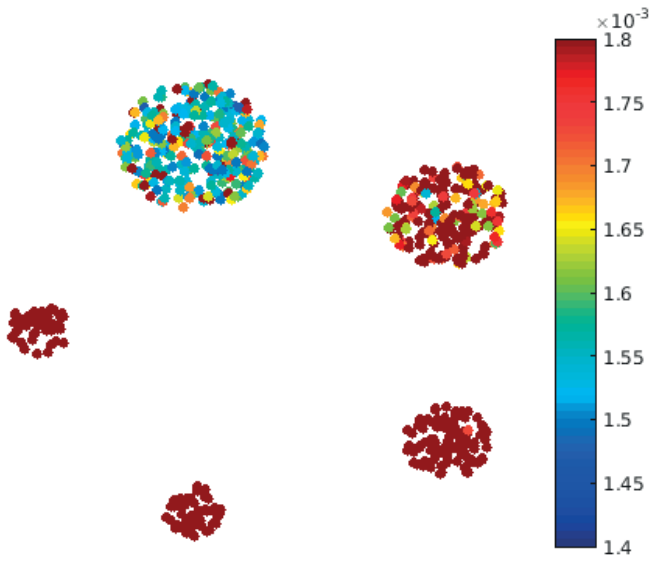


Figure 5.4 – Experimentation of $\|\mathcal{T}_i g\|$ to determine the class centers with unbalanced classes in an SBM graph. The nodes with lowest centrality are all gathered in the same class preventing the previously proposed method to work accurately.

On the other hand, we propose an alternative heuristic using only the estimation of the centrality. Since we observed that the lowest values of $\|\mathcal{T}_i g\|$ were not systematically those defining the best centroids, we decided to pick d nodes with the lowest norm ($d > k$, experimentally $d \propto k\log(k)$ seems good) and refine the set of k among d with a merging procedure. Iteratively, we reduce the set of centroids by an agglomerative strategy that assign the nodes belonging to the two centroids with the most similar spectral components together. The similarity between the centers is based on the measure $\|\mathcal{T}_i g - \mathcal{T}_j g\|_2$. This method is called *agglomerative selection* in the following.

5.3 Practical considerations

The study of these two components permitted us to construct a new algorithm which has the benefit of being extremely fast. In this section we will develop the construction of the method considering the two heuristics for the selection of the centroids defined above. We will also study its complexity and emphasize the impact of the most important parameters in the method.

5.3.1 Algorithm

We start with the clustering algorithm using the energy-based determination of the centroids, detailed in Algo. 6. Here, the step with largest computational complexity is step 3. Since it only serves to break ties and determine the first centroid, it might be worth removing it (as well as step 1) and replace it with a random selection of the ties and of the first centroid. Step 3 has a

complexity of $\mathcal{O}(tMc)$ where c is the order of the polynomial approximation of the graph filter. Then computing k times step 8 requires $\mathcal{O}(kMc)$ operations. Finally, step 11 is $\mathcal{O}(kN)$. Overall, this algorithm has a complexity of $\mathcal{O}(Mc(k+t)+kN)$ which is very low for graph clustering compared to the state of the art.

Algorithm 6 Light clustering (energy-based determination)

1: Generate $\mathbf{R} \in \{-1,1\}^{N\times t}$ composed of i.i.d. Bernoulli($p=\frac{1}{2}$)
2: Compute an approximated ideal low-pass filter g
3: Apply filtering: $\Psi = g(\mathbf{L})\mathbf{R}$
4: Estimate $\left\|\widehat{\mathcal{T}_i g}\right\| = \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{\psi}_i$
5: Initialize the energy vector to $\mathbf{en} = \mathbf{0}$
6: **for** i = 1,..., k **do**
7: Pick $\operatorname{argmin}_n \mathbf{en}[n]$ as the next centroid, break ties with lowest $\left\|\widehat{\mathcal{T}_i g}\right\|$
8: Compute $\mathcal{T}_i g$ centered at the last centroid
9: Update the total energy with $\mathbf{en}[n] = \mathbf{en}[n] + \left(\mathcal{T}_i g[n]\right)^2$ for all nodes
10: **end for**
11: Associate each node n with $\operatorname{argmax}_{i\in C} \mathcal{T}_i g[n]$

Our second proposition reuse the same scheme but replaces the centroid selection and is described in Algo. 7. We then have again $\mathcal{O}(tMc)$ operations for step 3 but now the selection of the centroids takes $\mathcal{O}\left(d^2N + d^3\right)$ because we must compute $\left\|\mathcal{T}_i g - \mathcal{T}_j g\right\|$ the first time and find the minimum among the remaining rows and columns at each step of the loop. Since $d < N$, we can omit the term d^3 in the following. Finally, adding step 11 just like the other algorithm we obtain a total complexity of $\mathcal{O}\left(tMc + d^2N\right)$ since $Nk < d^2N$. We conclude that the first algorithm is more efficient in terms of computations. We will see in section 5.4 how they perform on clustering tasks.

Algorithm 7 Light clustering (agglomerative selection)

1: Generate $\mathbf{R} \in \{-1,1\}^{N\times t}$ composed of i.i.d. Bernoulli($p=\frac{1}{2}$)
2: Compute an approximated ideal low-pass filter g
3: Apply filtering: $\Psi = g(\mathbf{L})\mathbf{R}$
4: Estimate $\left\|\widehat{\mathcal{T}_i g}\right\| = \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{\psi}_i$ around all the nodes at once
5: Randomly sample d nodes using a normalized version of $\frac{1}{\left\|\widehat{\mathcal{T}_i g}\right\|}$
6: Compute the $\mathcal{T}_i g$ centered at each of the d candidates
7: **while** more than k centroids remain **do**
8: Compute the distances $\left\|\mathcal{T}_i g - \mathcal{T}_j g\right\|$ for all pairs of remaining candidates
9: Merge the two centers with smallest difference
10: **end while**
11: Associate each node n with $\operatorname{argmax}_{i\in C} \mathcal{T}_i g[n]$

5.3.2 Impact of the graph kernel

The quality of the method that we proposed lies on the graph filter that we choose. In total, there are three different kernels that seem to be of interest based on the preliminary works.

Since the goal is to propagate the information from the centroids to the rest of the graph, the graph filter equivalent to the Tikhonov regularization problem is a potential candidate (called Tik in the experiments). It reads $g(\lambda_\ell) = \frac{1}{1+\gamma\lambda_\ell}$ (see example 2 in [125]). The parameter γ is the one used in the optimization formulation $\widehat{y} = \arg\min_x \|y - x\|_2 + \gamma x^\top \mathbf{L} x$. Since y is very sparse in our case, we will need to weight the regularization term generously for smoothness.

Second, one of the most famous diffusion process is that of heat. Moreover, as we said before, heat is particularly well suited to illustrate the diffusion process. Then, the heat kernel seems an appropriate low-pass filter for this task. Its equation is $g(\lambda_\ell) = \exp(\frac{-\alpha\lambda_\ell}{\lambda_{\max}})$. It is parametrized with α that controls how quickly the filter decays. Smaller values of α will retain more frequencies and would be advised when the graph of interest presents a large eigengap.

Finally, since we showed that the best approximation of k-medoids diffusion was performed with the ideal low-pass filter of cut-off frequency λ_k, we consider its polynomial approximation with parameter c to tweak the polynomial order. We chose to use the Jackson-Chebyshev polynomial approximation but the version of Allen-Zhu et al. [6] should perform similarly. The determination of λ_k is done once again with the eigencount technique of Di Napoli et al. [39].

We compared the different filters in an experiment where we tried to cluster 4 different graphs: a sensor graph, an SBM, a sampled manifold (the Stanford Bunny) and an image graph (reusing the image of Barbara described in Sec. 4.2.3 with $N = 4'096$). The results are summarized in tables 5.4 to 5.9 at the end of this chapter.

The three filters described above are each used with two values for their parameters. Heat and Tik are, moreover, approximated with Chebyshev polynomials of order $c = 100$. For each graph and filter, we reproduced the experiment 50 times, generating a new sensor and SBM each time (the bunny and the image patch are deterministic). We computed the time, the modularity score and the normalized cut in all realizations. We also compared our two different heuristics for centroid selection. In the energy-based determination, we did not compute the $\|\widehat{\mathcal{T}_i g}\|$ and rather broke the ties at random as suggested in the previous section.

First we observed that the energy-based method is significantly faster than the agglomerative selection as we were expecting. The agglomerative selection has a complexity that grows quickly with N which is not a good characteristic for large-scale problem solving. In both cases, the time required mostly depends on the polynomial order and the graph size. The energy-based approach is up to one order of magnitude faster than the Spectral Clustering against which we compare.

Then, regarding the clustering quality, Spectral Clustering is the best method in this experiment in all situations (with both measures and for the four graphs). However, our technique using

the approximated step function performs almost as well which is encouraging. The polynomial order has only little impact on the quality. In fact, half of the cases even give $c = 50$ in front of $c = 150$. Tikhonov overall did not reach the quality that we were expecting with both $\gamma = 1e^{-2}$ and $\gamma = 1e^{-1}$. Moreover, the reported quality measures are varying depending on the graph, even with fixed parameters. Finally, the heat kernel is almost as good as the step function except with the stochastic block model. We expect that the parameter α is unsuitable for this kind of graph spectrum while the step function follows the k^{th} eigenvalue.

5.4 Experiments

We conclude this chapter with several experiments presenting the capabilities of the method that we introduced. Following the previous section, we kept only the energy-based determination of the centroids and the approximated ideal low-pass filter.

First, we study the scalability of our proposed method in an experiment where we constructed several sensor networks of different sizes ($N = 1'000, 5'000, 10'000, 20'000, 30'000, 50'000$). We decided to split it into 10 classes. We compared our method against Spectral Clustering (using the IRLM approach) and Newman's algorithm with the ncut and the modularity measures. We expect Newman to optimize the modularity and SC the ncut measure. We replicated the experiment 25 times and reported the results in Table 5.2.

Next, we notice that our method and spectral clustering have almost the same runtime. When digging into the reasons of such complexity for our method, we found that the filter design was executed during the large majority of the running time. Hence, if we were able to generate the filter design quicker, we would be able to gain a lot of time .[2] On its side, Newman's algorithm scales badly with N. We were unable to report the time for more than 20'000 nodes, in which case it already took several hours. This method is thus impractical for large datasets.

Regarding the quality of the partitioning, as expected Newman's algorithm is the best when focusing on the modularity score. Our method performed slightly better than SC in this case when N started to be large enough. Interestingly, except for $N = 1'000$, our method was the one achieving the smallest ncut score in average. The difference with Spectral Clustering is impressive knowing that SC is optimizing the relaxed version of this objective.

Then, we also wanted to verify if this method was robust. We decided to cluster Stochastic Block Models of $N = 5'000$ nodes formed of $k = 15$ classes. We set $\overline{d_{\mathcal{G}}} = 40$, computed $\varepsilon_{\max}$ and decided to try to cluster graphs of all clusterability between 0 and $\varepsilon_{\max}$. Surprisingly, we noticed that the quality of our method was as good as Spectral Clustering for the small values of ε and slightly less good when the graph became harder to cluster (see Fig. 5.5). Both the modularity and the ncut measure highlight the same behavior. Moreover, when ε reaches $\varepsilon_{\max}$, our method is again performing similarly to Spectral Clustering. Moreover, timing of SC is impacted by the

[2] Designing a kernel of fixed parameter takes a negligible amount of time and does not depend on N.

		$N = 1'000$	$N = 5'000$	$N = 10'000$	$N = 20'000$	$N = 30'000$	$N = 50'000$
Time (sec)	Filter design	0.4020	3.1517	6.1486	14.014	22.672	48.824
	Light clustering	0.4679	3.2702	6.6880	15.498	24.786	51.033
	SC	0.6058	2.1562	5.3909	13.733	17.062	42.910
	Newman	4.6880	682.68	6'011.6	51'199	-	-
Modularity	Light clustering	0.8372	0.8663	0.8709	0.8741	**0.8756**	**0.8747**
	SC	0.8490	0.8628	0.8692	0.8714	0.8701	0.8726
	Newman	**0.9464**	**0.9734**	**0.9804**	**0.9853**	-	-
Ncut	Light clustering	0.2442	**0.1260**	**0.1006**	**0.0679**	**0.0609**	**0.0427**
	SC	**0.1878**	0.1720	0.1814	0.1816	0.1911	0.1880
	Newman	1.0976	1.5118	1.5997	1.6313	-	-

Table 5.2 – Scaling capabilities of the light clustering method compared to SC. Our method is using the energy-based determination with a polynomial approximation of the ideal low-pass filter (Algo. 6). The determination of λ_k is accounting for most of our computational time. Newman timed out for graphs of more than 20'000 nodes were it took already 51'199 sec.

complexity of the graph while our method does not react to this modification.

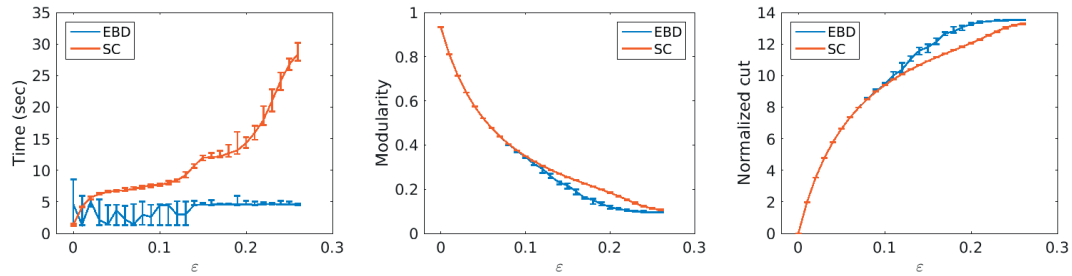


Figure 5.5 – Clustering quality of the light clustering method compared to SC on SBM of different clusterability. Our method is using the energy-based determination of the centroids (Algo. 6). Highly clusterable graphs (with low ε) are assigned similarly by SC and our method. SC performs better in the situations where the graph is hard to cluster.

Finally, we also used our light clustering technique to partition the MNIST dataset (see 4.4.2 for details). We reused the graph constructed from the 70'000 samples of digits and tried to partition it into the ten classes corresponding to the different digits. Once again, the approximated version with $\mathcal{T}_i g$ is very fast compared to SC and performs almost as well if we consider modularity or the rand index.

	Time	Ncut	Modularity	Rand index
SC	2h38mn	**0.9322**	**0.7961**	**0.9086**
Light clustering	**12mn30**	1.8753	0.7797	0.8946

Table 5.3 – Comparison of light clustering with SC on the MNIST dataset clustering task. The dataset contains 70'000 nodes and is clustered into 10 classes. The light method is run with an approximation of the ideal low-pass filter of polynomial order $c = 100$.

5.5 Conclusion

In this chapter, we proposed a clustering method based on the two steps of k-means and that makes use of the localization operator $\mathcal{T}_i g$. It looks similar to SC in the sense that it propagates the cluster information of the centroids to the rest of the graph, using the first eigenvectors as features. However, we proved our method to be very fast, with a complexity of $\mathcal{O}(Mck + kN)$ where c is the polynomial order of the graph filter approximation. The experiment section at the end of this chapter shows the potential of our technique by comparing to state-of-the-art reference algorithms on clustering tasks with different kinds of graphs, of different sizes and evaluated with two different measures (the modularity and ncut).

As we highlighted in Section 5.3.2, the choice of the filter is very important for the proposed method to be efficient. At the moment, this is the step that is the most computationally demanding, when ran with an approximated ideal low-pass filter. Since it reuses the eigencount technique of Di Napoli et al., improving this heuristic would also improve this algorithm in general.

		Sensor $N = 1'000$	SBM $N = 1'000$	Bunny	ImagePatch
Tik	$\gamma = 1e^{-2}$	0.0516 (±0.0002)	0.0667 (±0.0002)	0.0919 (±0.0001)	0.1504 (±0.0003)
	$\gamma = 1e^{-1}$	0.0460 (±0.0001)	0.0628 (±0.0001)	0.0884 (±0.0001)	0.1585 (±0.0005)
Heat	$\alpha = 10$	0.0442 (±0.0000)	0.0617 (±0.0001)	0.0878 (±0.0002)	0.1543 (±0.0003)
	$\alpha = 50$	0.0454 (±0.0001)	0.0633 (±0.0001)	0.0916 (±0.0002)	0.1587 (±0.0003)
Step	$c = 50$	0.1712 (±0.0004)	0.1550 (±0.0015)	0.4644 (±0.0068)	0.8029 (±0.0153)
	$c = 150$	0.4712 (±0.0028)	0.4691 (±0.0187)	1.4011 (±0.0252)	2.4361 (±0.0371)
Spectral Clustering		0.5471 (±0.0049)	0.4133 (±0.0012)	0.8847 (±0.0154)	1.8838 (±0.1065)

Table 5.4 – Timing with the energy-based determination method for different graph kernels and parameters. Tik and heat are approximated with Chebyshev polynomials of order $c = 100$. The timing depends almost entirely on the polynomial order and the graph size. Our method can be up to one order of magnitude faster than SC in this experiment.

		Sensor $N = 1'000$	SBM $N = 1'000$	Bunny	ImagePatch
Tik	$\gamma = 1e^{-2}$	1.5598 (±0.0036)	3.2309 (±0.0110)	12.2699 (±0.0400)	30.8258 (±0.0535)
	$\gamma = 1e^{-1}$	1.5950 (±0.0041)	3.2515 (±0.0126)	12.2068 (±0.0392)	30.8187 (±0.0703)
Heat	$\alpha = 10$	1.5817 (±0.0029)	3.2162 (±0.0148)	12.2118 (±0.0462)	30.7377 (±0.0580)
	$\alpha = 50$	1.5978 (±0.0037)	3.2200 (±0.0109)	12.2603 (±0.0315)	30.8291 (±0.0587)
Step	$c = 50$	0.9394 (±0.0013)	1.7528 (±0.0066)	6.5646 (±0.0297)	16.2020 (±0.0640)
	$c = 150$	2.7551 (±0.0056)	5.2224 (±0.0366)	19.6362 (±0.0826)	48.4117 (±0.2100)
Spectral Clustering		0.5471 (±0.0049)	0.4133 (±0.0012)	0.8847 (±0.0154)	1.8838 (±0.1065)

Table 5.5 – Timing with the agglomerative selection method for different graph kernels and parameters. Tik and heat are approximated with Chebyshev polynomials of order $c = 100$. The timing depends almost entirely on the polynomial order and the graph size.

		Sensor $N = 1'000$	SBM $N = 1'000$	Bunny	ImagePatch
Tik	$\gamma = 1e^{-2}$	0.6172 (±0.0007)	0.2409 (±0.0007)	0.4401 (±0.0003)	0.3356 (±0.0003)
	$\gamma = 1e^{-1}$	0.7674 (±0.0005)	0.2515 (±0.0005)	0.6975 (±0.0007)	0.5280 (±0.0010)
Heat	$\alpha = 10$	0.8077 (±0.0001)	0.2535 (±0.0136)	0.8042 (±0.0001)	0.8024 (±0.0001)
	$\alpha = 50$	0.8181 (±0.0001)	0.0000 (±0.0000)	0.8084 (±0.0000)	0.8000 (±0.0001)
Step	$c = 50$	0.8227 (±0.0002)	0.4781 (±0.0003)	0.8063 (±0.0001)	0.8046 (±0.0002)
	$c = 150$	0.8130 (±0.0006)	0.4625 (±0.0050)	0.8026 (±0.0002)	0.8097 (±0.0002)
Spectral Clustering		0.8210 (±0.0005)	0.4871 (±0.0000)	0.8052 (±0.0000)	0.8051 (±0.0000)

Table 5.6 – Quality evaluation using modularity with the energy-based determination method for different graph kernels and parameters. The highest score represents the best clustering with respect to this measure. Spectral clustering is consistently above the rest of the measures but the step function and the heat kernel achieve very close results except for the SBM graph.

		Sensor $N = 1'000$	SBM $N = 1'000$	Bunny	ImagePatch
Tik	$\gamma = 1e^{-2}$	0.6047 (±0.0016)	0.1802 (±0.0008)	0.3952 (±0.0011)	0.3283 (±0.0005)
	$\gamma = 1e^{-1}$	0.7227 (±0.0005)	0.1802 (±0.0007)	0.5883 (±0.0043)	0.4891 (±0.0025)
Heat	$\alpha = 10$	0.7845 (±0.0005)	0.3082 (±0.0031)	0.7576 (±0.0012)	0.7283 (±0.0034)
	$\alpha = 50$	0.7914 (±0.0020)	0.0228 (±0.0039)	0.7813 (±0.0003)	0.7589 (±0.0025)
Step	$c = 50$	0.7960 (±0.0007)	0.3306 (±0.0032)	0.7701 (±0.0005)	0.7790 (±0.0007)
	$c = 150$	0.7908 (±0.0007)	0.3319 (±0.0057)	0.7666 (±0.0003)	0.7794 (±0.0007)
Spectral Clustering		0.8210 (±0.0005)	0.4871 (±0.0000)	0.8052 (±0.0000)	0.8051 (±0.0000)

Table 5.7 – Quality evaluation using modularity with the agglomerative selection method for different graph kernels and parameters. The highest score represents the best clustering with respect to this measure. Spectral clustering is consistently above the rest of the measures but the step function and the heat kernel come close except for the SBM graph.

		Sensor $N = 1'000$	SBM $N = 1'000$	Bunny	ImagePatch
Tik	$\gamma = 1e^{-2}$	2.2457 (±0.0547)	5.8234 (±0.0603)	3.8524 (±0.0275)	4.9089 (±0.0300)
	$\gamma = 1e^{-1}$	0.8616 (±0.0388)	5.7360 (±0.0447)	1.5182 (±0.0492)	3.1411 (±0.0938)
Heat	$\alpha = 10$	0.4198 (±0.0057)	5.4403 (±0.6006)	0.5110 (±0.0014)	0.4002 (±0.0017)
	$\alpha = 50$	0.2999 (±0.0029)	NaN	0.4762 (±0.0009)	0.3625 (±0.0020)
Step	$c = 50$	0.2136 (±0.0033)	3.6393 (±0.0287)	0.5008 (±0.0119)	0.3872 (±0.0184)
	$c = 150$	0.1954 (±0.0231)	3.7571 (±0.4275)	0.4524 (±0.0109)	0.3082 (±0.0063)
Spectral Clustering		0.1456 (±0.0018)	3.5520 (±0.0000)	0.3729 (±0.0000)	0.2554 (±0.0000)

Table 5.8 – Quality evaluation using the ncut objective with the energy-based determination method for different graph kernels and parameters. The lowest score represents the best clustering with respect to this measure. Spectral clustering is consistently below the rest of the measures. Here none of the measures is really satisfactory compared to SC.

		Sensor $N = 1'000$	SBM $N = 1'000$	Bunny	ImagePatch
Tik	$\gamma = 1e^{-2}$	2.4305 (±0.1716)	6.4089 (±0.0608)	4.3466 (±0.1002)	4.9817 (±0.0454)
	$\gamma = 1e^{-1}$	1.2649 (±0.0507)	6.4092 (±0.0533)	2.6165 (±0.4126)	3.5535 (±0.2469)
Heat	$\alpha = 10$	0.5757 (±0.0273)	5.4126 (±0.2298)	0.8492 (±0.0681)	1.1888 (±0.3275)
	$\alpha = 50$	0.3702 (±0.0133)	NaN	0.6576 (±0.0109)	0.5623 (±0.0205)
Step	$c = 50$	0.3478 (±0.0456)	5.0637 (±0.5294)	0.9146 (±0.0964)	0.6161 (±0.0549)
	$c = 150$	0.3853 (±0.0971)	5.1057 (±0.7297)	0.8834 (±0.1045)	0.5994 (±0.1033)
Spectral Clustering		0.1456 (±0.0018)	3.5520 (±0.0000)	0.3729 (±0.0000)	0.2554 (±0.0000)

Table 5.9 – Quality evaluation using the ncut objective with the agglomerative selection method for different graph kernels and parameters. The lowest score represents the best clustering with respect to this measure. Spectral clustering is consistently below the rest of the measures. Here none of the measures is really satisfactory compared to SC.

6 Discussion

6.1 Summary of findings

Data analysis requires the development of robust and efficient methods in order to process the overwhelming quantity of information that we face today. This thesis offers a series of propositions in that matter, using Graph Signal Processing, under different shapes.

First, applying the technique of random signal filtering, we designed an algorithm capable of extracting the eigensubspace of any symmetric matrix in $\mathcal{O}\left(cMk + k^2N + k^3\right)$ operations. Compared to the existing methods that output the set of first eigenvectors, the complexity gain is of the order of $\mathcal{O}\left(hk^2N + hk^3\right)$. Additional research would be required to drop the multiplication by an orthonormal matrix $\mathbf{Q}$ with low cost. In its current stage, our proposed method already enables to approximate Spectral Clustering but also the Laplacian eigenmap visualization method or PCP problems since those three techniques do not require the knowledge of the eigenvectors but simply the subspace that they span in higher dimensions. While our technique theoretically provides an exact method for the recovery of the subspace, the efficient implementation requires the use of polynomial graph filters that approximate the step function.

Second, we studied the work of Tremblay et al. [132] and worked on theoretical guarantees concerning the use of random filtered signals for the approximation of the SC assignment. Moreover, this method is further accelerated since it combines random filtering with graph compression. They proposed a technique to reduce the cost of the final k-means operation of SC, computing it only on a subset of the nodes and propagating the information to the nodes with missing values using graph regularization. We constructed a clustering method for dynamic graphs (i.e., where the graph evolves over time) expanding on their works. We also provided theoretical guarantees about the quality of the assignment by reusing as much information as possible enclosed in the filtered signals of the previous steps.

Since the efficient methods for eigenvector determination that we encountered always involved filtering with an ideal low-pass filter, we studied and proposed alternatives for the approximation

of such filter. On the one hand, the application of the approximation of the sign function proposed by Allen-Zhu et al. [6] was used for the first time as a graph filter in this thesis. On the other hand, the different approximations require the knowledge of λ_k, the threshold of the ideal low-pass. We elaborated on the works of Di Napoli et al. [39] a procedure for the count of eigenvalues in an interval that was not using dichotomy but rather the assumption that locally, the eigenvalues are spread uniformly on the spectrum.

Finally, we also built on top of the graph localization operator. We proposed an extremely efficient, although slightly less accurate method for graph clustering that uses filtering of Kronecker deltas and the idea of propagation of the solution used in the compressive k-means of Tremblay in $\mathcal{O}(cMk + kN)$. The method depends on the selection of an appropriate graph filter and we established that approximating ideal low-pass filters was again among the best choices we could make. For that aspect $\mathcal{O}(cMd)$ extra operations with $d > k$ are required as of now but we expect this problem to be dug in the near future given the constant need for an improved technique on this side in all our works.

6.2 Future works

Even though the thesis reaches its end, the research does not. The field of Graph Signal Processing is already an important one for data analysis but important problems remain to be solved.

For instance, this thesis only considers problems where the graph is undirected. In practice, most of the real-world networks have different weights on the edges between two nodes. The $\text{DSP}_{\mathcal{G}}$ framework of Moura et al. (see Sec. 2.1.6) encompasses this class of adjacency matrices, however, all their experiments are performed on undirected networks, most probably because of the lack of stability of the Jordan decomposition.

Moreover, in parallel of the structural analysis of networks, many works study the importance of the content for machine learning applications. This calls for the development of methods aggregating the different sources of information together and performing data analysis on multilayer graphs (one per source). An important progress to make in that direction is to show a significant improvement when combining several sources independently compared to a linear combination of the features and the application of the single-layer methods that we already know.

Another interesting field that is progressing very fast is that of deep learning on graphs. Deep learning has shaken the machine learning community with the efficiency of the newly proposed methods. However the theoretical guarantees remain vague. GSP has lately shown to be useful to improve the techniques of deep learning on graphs. We encourage the GSP community to study this field and improve even more the quality of data analysis. In terms of efficient and robust machine learning, deep learning is interesting to consider for the comparison.

Finally, and more specifically related to the progress directly made in this thesis, we encourage the interested readers to consider the different key steps of SC and find workarounds to reduce

the complexity or improve the quality even further. In our opinion, two aspects are resisting and even if we addressed both of them we expect that there is room for improvements. The first one concerns the works on the approximation of the ideal low-pass filter and the determination of the threshold λ_k for any graph as discussed above. The second one is the k-means step that must be computed at the end. This step is currently solved by repeating the computations a large number of times to hope for an initialization that would provide the optimal solution. The progress with graph diffusion used by Tremblay et al. and applied in Chapter 5 could indicate a potential breakthrough in the near future.

The last words are for the reader, if you are reading this, thank you for your interest in this work.

A Fast Approximation of Spectral Clustering for Dynamic Networks

A.1 Study of the JL constraints of Cor. 3.11

We study here the behavior of the functions $p(1+\delta)$ and $1+\varepsilon$ with $p,\delta,\varepsilon \in (0,1)$ and under the constraints defined by equation (3.39). First, we start this appendix with the analysis of a particular polynomial of degree 3, which will show useful in the continuation of this section at several occasions:

$$\mathcal{A}(x,d) := ax^3 + bx^2 + cx + d = x^3 - \frac{3}{2}x^2 + d. \tag{A.1}$$

The computation of its discriminant gives

$$\Delta_{\mathcal{A}} = -4b^3 d - 27a^2 d^2 = 27d\left(\frac{1}{2} - d\right), \tag{A.2}$$

while its first derivative with respect to x is

$$\frac{\partial \mathcal{A}}{\partial x} = 3x^2 - 3x = 3x(x-1). \tag{A.3}$$

The first derivative indicates two changes of monotony in $x = 0$ and $x = 1$. We can easily compute that $\mathcal{A}(0,d) = d$ and $\mathcal{A}(1,d) = d - \frac{1}{2}$. Hence, we know that $\mathcal{A}(x,d)$ is increasing on $x \in (-\infty, 0]$, decreasing on $x \in [0,1]$ and increasing on $x \in [1,+\infty)$. Together with the previous computation, we notice that its number of real roots depends on the value of d.

Let us now come back to the study of the discriminant and evaluate all possibilities.
If $\Delta_{\mathcal{A}} < 0$, $\mathcal{A}$ must possess one real and two complex conjugate roots. This can be the case either if $d < 0$ or if $d > \frac{1}{2}$. Since $\mathcal{A}$ is monotonically increasing in the negatives up to 0 where it is worth d, it must cross the horizontal axis for a value $x < 0$ if $d > \frac{1}{2}$. On the opposite, if $d < 0$, then $\mathcal{A}$ remains negative until it reaches its first (and only) root somewhere in $[1,+\infty)$.

If we assume instead that $\Delta_{\mathcal{A}} = 0$, $\mathcal{A}$ possesses two real roots (one of the two with a double multiplicity). This means from equation (A.2) that $d = 0$ or $d = \frac{1}{2}$. If $d = 0$, $\mathcal{A}(x,0) = x^2(x - \frac{3}{2})$

so 0 is a root of double multiplicity and the second root is $\frac{3}{2}$. Conversely, if $d = \frac{1}{2}$, the root of double multiplicity is at 1 and the second root is $x = -\frac{1}{2}$.

Finally, if $\Delta_{\mathcal{A}} > 0$, $\mathcal{A}$ must have three distinct real roots but due to the two roots of the derivative, there must be one root of $\mathcal{A}$ in $(0,1)$, labeled r. This corresponds to $0 < d < \frac{1}{2}$ and this is the only situation where one of the roots belongs to the interval $x \in (0,1)$.

The easiest way to discover the roots of a polynomial of degree 3 is to substitute it such that $b = 0$. To do so, we exploit the change of variable $y = x + \frac{b}{3a} = x - \frac{1}{2}$. Then, $\mathcal{A}(x,d) = y^3 - \frac{3}{4}y + d - \frac{1}{4}$.

From here, the three roots with respect to y can be written in a trigonometric form as

$$y_k = \cos\left(\frac{\arccos(1-4d)}{3} - \frac{2k\pi}{3}\right), \quad k = 0,1,2. \tag{A.4}$$

Knowing that $y_0 \geq y_1 \geq y_2$, we deduce that the root of x that we are looking for must be

$$r = y_1 + \frac{1}{2} = \frac{1}{2} + \cos\left(\frac{\arccos(1-4d) - 2\pi}{3}\right). \tag{A.5}$$

We come back now to the study of the constrained minimization problem introduced in Def. 3.10.

On the one hand, we have $d \geq \frac{4+2\beta}{\varepsilon^2/2 - \varepsilon^3/3}\log(N)$ which can be expressed as $\varepsilon^3 - \frac{3}{2}\varepsilon^2 + \frac{12+6\beta}{d}\log(N) \leq 0$ or equivalently $\mathcal{A}\left(\varepsilon, \frac{12+6\beta}{d}\log(N)\right) \leq 0$.
From the previous study, we know that there exist a value for ε if and only if $0 < \frac{12+6\beta}{d}\log(N) < \frac{1}{2}$. In other words if $\beta < \frac{d}{12\log(N)} - 2$. Following this theory, it is not possible to assess the results with a higher probability with given N and d.
In such case, $\mathcal{A}$ is negative for $\varepsilon \in [r_\varepsilon, 1)$ where $r_\varepsilon = \frac{1}{2} + \cos\left(\frac{1}{3}\arccos\left(1 - \frac{48+24\beta}{d}\log(N)\right) - \frac{2\pi}{3}\right)$.

On the other hand, the constraint $dp \geq \frac{4+2\beta}{\varepsilon^2/2 - \varepsilon^3/3}\log(N)$ can be studied similarly and as long as $\beta < \frac{dp}{12\log(N)} - 2$, there are values satisfying the constraint in $\delta \in [r_\delta, 1)$ where $r_\delta = \frac{1}{2} + \cos\left(\frac{1}{3}\arccos\left(1 - \frac{48+24\beta}{dp}\log(N)\right) - \frac{2\pi}{3}\right)$.

Thus, the constrained minimum can be simplified into

$$\begin{aligned}
\delta &= \frac{1}{2} + \cos\left(\frac{1}{3}\arccos\left(1 - \frac{48+24\beta}{dp}\log(N)\right) - \frac{2\pi}{3}\right) \\
\varepsilon &= \frac{1}{2} + \cos\left(\frac{1}{3}\arccos\left(1 - \frac{48+24\beta}{d}\log(N)\right) - \frac{2\pi}{3}\right) \\
\mu &= \begin{cases} 1+\varepsilon, & \frac{dp}{12\log(N)} - 2 < \beta < \frac{d}{12\log(N)} - 2 \\ \min\{p(1+\delta), 1+\varepsilon\}, & \beta < \frac{dp}{12\log(N)} - 2 \end{cases}
\end{aligned} \tag{A.6}$$

B Approaching Laplacian Eigenspace with Random Signals

B.1 Properties of projected Gaussians

We stated in section 4.2.1 that a Gaussian random matrix projected over a basis remains a Gaussian of identical mean and variance. We will demonstrate the different properties in this appendix.

Let $\mathbf{U} \in \mathbb{R}^{N\times N}$ describe a basis of N orthonormal vectors and $\mathbf{R} \in \mathbb{R}^{N\times d}$ be a Gaussian random matrix with i.i.d. entries $\sim \mathcal{N}(0, \sigma^2)$.

Mathematically,

$$\forall i, j : (\mathbf{UR})_{i,j} = \langle u_{i-1}, r_j \rangle = \sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j}, \tag{B.1}$$

is a linear transformation of the elements of $\mathbf{R}$. Thus, there are Gaussians. Moreover, we already knew that the size of the product is a $N \times d$ matrix. Next, we will evaluate the two first moments of all those entries.

$$\mathbb{E}\left[\sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j}\right] = \sum_{\ell=1}^{N} u_{i-1}(\ell)\mathbb{E}[r_{\ell,j}] = 0 \tag{B.2}$$

$$\begin{aligned}\mathrm{Var}\left(\sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j}\right) &= \sum_{\ell=1}^{N} u_{i-1}^2(\ell)\,\mathrm{Var}(r_{\ell,j}) \\ &= \sigma^2 \sum_{\ell=1}^{N} u_{i-1}^2(\ell) = \sigma^2\end{aligned} \tag{B.3}$$

This shows that all entries of $\mathbf{UR}$ are identically distributed. Then we can compute the covariance

between any two entries $\big((\mathbf{UR})_{i,j}$ and $(\mathbf{UR})_{n,m}\big)$ to ensure independence:

$$\begin{aligned}
\mathrm{Cov}(\mathbf{UR}) &= \mathbb{E}\left[\sum_{\ell=1}^{N} u_{i-1}(\ell) r_{\ell,j} \sum_{\ell'=1}^{N} u_{n-1}(\ell') r_{\ell',m}\right] \\
&= \sum_{\ell=1}^{N}\sum_{\ell'=1}^{N} u_{i-1}(\ell) u_{n-1}(\ell') \mathbb{E}[r_{\ell,j} r_{\ell',m}] \\
&= \mathbb{1}_{\{m=j\}} \sum_{\ell=1}^{N} u_{i-1}(\ell) u_{n-1}(\ell) \mathbb{E}[r_{\ell,m}^2] \\
&= \sigma^2 \mathbb{1}_{\{m=j\}} \langle u_{i-1}, u_{n-1}\rangle \\
&= \sigma^2 \mathbb{1}_{\{m=j\}} \mathbb{1}_{\{n=i\}},
\end{aligned} \tag{B.4}$$

which shows that any two entries in $\mathbf{UR}$ are independent. Combining the last two shows that the entries of $\mathbf{UR}$ are i.i.d. Gaussian random samples with pdf $\sim \mathcal{N}(0, \sigma^2)$ just like $\mathbf{R}$.

B.2 Numerical limits of rank approximation

As lemma 4.3 is critical to the proof of theorem 4.1, we make a slight digression and discuss its numerical approximation. Indeed, we proved that the matrix $\mathbf{R}_k$ is full rank and this means that the smallest singular value of $\mathbf{R}_k$ is strictly positive. However, while computing singular value decomposition, numerical approximations are performed and the singular values below a given threshold are assimilated to linearly dependent columns. In other words, we need to make a stronger statement and ensure that the smallest singular value stays above a numerical precision threshold in good probability.

To this end, we recall the result of Martinsson et al.:

Lemma B.1: Bound on the singular values of Gaussian matrices [89, Lem. 3.15].
Suppose that k and ℓ are positive integers with $k \leq \ell$. Suppose further that G is a real $\ell \times k$ matrix whose entries are i.i.d. Gaussian random variables of zero mean and unit variance, and β is a positive real number, such that

$$1 - \frac{1}{\sqrt{2\pi(\ell - k + 1)}} \left(\frac{e}{(\ell - k + 1)\beta} \right)^{\ell - k + 1} \tag{B.5}$$

is nonnegative.

Then, the least (that is, the kth greatest) singular value of G is at least $\frac{1}{\sqrt{\ell}\beta}$ with probability not less than the amount in (B.5).

Since $\mathbf{R}_k$ in our case is a Gaussian random matrix of size $k \times k$, zero mean and variance $\frac{1}{k}$, then $s_{\min}$, the smallest singular value of $\mathbf{R}_k$, equals $\frac{\lambda_{\min}}{\sqrt{k}}$ with $\lambda_{\min}$ the smallest singular value of a matrix whose entries match the lemma above. Thus from this result, we can state that the cumulative density function of $s_{\min}$ is:

$$\mathbb{P}(s_{\min} < \frac{1}{\beta}) < \frac{e}{\beta\sqrt{2\pi}} \tag{B.6}$$

In practice, we need to ensure that the minimal singular value is above the predefined threshold of the rank estimate, which is usually around 10^{-13}. Knowing that the probability of $s_{\min}$ being below the numerical threshold is less than $\frac{e}{\sqrt{2\pi}} 10^{-13} \approx 10^{-13}$, we can conclude that the claim we made theoretically for the rank still holds in practice with very high probability.

List of publications

During my thesis I authored or co-authored the following papers and technical reports:

Conference papers

[1] L. Martin, A. Loukas, and P. Vandergheynst. Fast Approximate Spectral Clustering for Dynamic Networks. *arXiv preprint arXiv:1706.03591. Submitted to the 21st Int. Conf. on Artificial Intelligence and Statistics*, 2018.

[2] S. Fallet, Y. Schoenenberger, L. Martin, V. Moser, and J.-M. Vesin. Imaging Photoplethysmography: A Real-time Signal Quality Index. *Computing in Cardiology*, 2017.

[3] L. Martin and P. Pu. Prediction of helpful reviews using emotions extraction. *Proc. of the 28th AAAI Conf. on Artificial Intelligence (AAAI-14)*, 2014.

[4] L. Martin, V. Sintsova, and P. Pu. Are influential writers more objective?: an analysis of emotionality in review comments. *Proc. of the 23rd Int. Conf. on World Wide Web*, 2014.

Technical report

[5] J. Paratte and L. Martin. Fast Eigenspace Approximation using Random Signals. *arXiv preprint arXiv:1611.00938*, 2016.

Toolbox

[6] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. GSPBOX: A toolbox for signal processing on graphs. *arXiv preprint arXiv:1408.5781*, 2014.

Bibliography

[1] E. Abbe, A. S. Bandeira, and G. Hall. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2016.

[2] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[3] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine learning*, 6(1):37–66, 1991.

[4] A. Ahmad and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, 63(2):503–527, 2007.

[5] K. S. Al-Sultan. A tabu search approach to the clustering problem. *Pattern recognition*, 28 (9):1443–1451, 1995.

[6] Z. Allen-Zhu and Y. Li. Faster principal component regression and stable matrix chebyshev approximation. In *International Conference on Machine Learning*, pages 107–115, 2017.

[7] D. Archambault, T. Munzner, and D. Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE transactions on visualization and computer graphics*, 14(4): 900–913, 2008.

[8] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics*, 9(1):17–29, 1951.

[9] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[10] Y. Bai. High performance parallel approximate eigensolver for real symmetric matrices. 2005.

[11] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the solution of algebraic eigenvalue problems: a practical guide*, volume 11. Siam, 2000.

[12] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[13] M. Belkin and P. Niyogi. Towards a theoretical foundation for laplacian-based manifold methods. *Journal of Computer and System Sciences*, 74(8):1289–1308, 2008.

[14] P. Berkhin et al. A survey of clustering data mining techniques. *Grouping multidimensional data*, 25:71, 2006.

[15] J. C. Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013.

[16] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008 (10):P10008, 2008.

[17] C. Böhm, S. Goebl, A. Oswald, C. Plant, M. Plavinski, and B. Wackersreuther. Integrative parameter-free clustering of data with mixed type attributes. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 38–47. Springer, 2010.

[18] J. A. Bondy, U. S. R. Murty, et al. *Graph theory with applications*, volume 290. Citeseer, 1976.

[19] C. Boutsidis, P. Drineas, and M. Magdon-Ismail. Near-optimal column-based matrix reconstruction. *SIAM Journal on Computing*, 43(2):687–717, 2014.

[20] C. Boutsidis, P. Kambadur, and A. Gittens. Spectral clustering via the power method-provably. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 40–48, 2015.

[21] U. Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[22] J.-P. Calbimonte, J. Eberle, and K. Aberer. Semantic data layers in air quality monitoring for smarter cities. In *Proc. of the 6th Workshop on Semantics for Smarter Cities S4SC 2015, at ISWC 2015*, number EPFL-CONF-212731, 2015.

[23] D. Calvetti, L. Reichel, and D. C. Sorensen. An implicitly restarted lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2 (1):21, 1994.

[24] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 554–560. ACM, 2006.

[25] G. Chandrashekar and F. Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[26] C.-H. Chang. Simulated annealing clustering of chinese words for contextual text recognition. *Pattern Recognition Letters*, 17(1):57–66, 1996.

[27] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6):1417–1440, 2004.

[28] N. V. Chawla and D. A. Davis. Bringing big data to personalized healthcare: a patient-centered framework. *Journal of general internal medicine*, 28(3):660–665, 2013.

[29] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovacevic. Signal denoising on graphs via graph filtering. In *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, pages 872–876. IEEE, 2014.

[30] Y. Chi. Compressive graph clustering from random sketches. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5466–5469. IEEE, 2015.

[31] Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng. Evolutionary spectral clustering by incorporating temporal smoothness. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 153–162. ACM, 2007.

[32] F. Chung and A. Tsiatas. Finding and visualizing graph clusters using pagerank optimization. *Internet Mathematics*, 8(1-2):46–72, 2012.

[33] F. R. Chung. *Spectral graph theory*, volume 92. AMS Bookstore, 1997.

[34] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[35] M. B. Cohen, S. Elder, C. Musco, C. Musco, and M. Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 163–172. ACM, 2015.

[36] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová. Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications. *Physical Review E*, 84 (6):066106, 2011.

[37] I. Derényi, G. Palla, and T. Vicsek. Clique percolation in random networks. *Physical review letters*, 94(16):160202, 2005.

[38] C. Dhanjal, R. Gaudel, and S. Clémençon. Efficient eigen-updating for spectral graph clustering. *Neurocomputing*, 131:440–452, 2014.

[39] E. Di Napoli, E. Polizzi, and Y. Saad. Efficient estimation of eigenvalue counts in an interval. *Numerical Linear Algebra with Applications*, 23(4):674–692, 2016.

[40] J. Duch and A. Arenas. Community detection in complex networks using extremal optimization. *Physical review E*, 72(2):027104, 2005.

[41] J. C. Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. 1973.

[42] D. Engel, L. Hüttenberger, and B. Hamann. A survey of dimension reduction methods for high-dimensional data analysis and visualization. In *OASIcs-OpenAccess Series in Informatics*, volume 27. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2012.

[43] P. Erdos and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[44] L. Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii academiae scientiarum Petropolitanae*, 8:128–140, 1741.

[45] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[46] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *IEEE transactions on pattern analysis and machine intelligence*, 26(2):214–225, 2004.

[47] G. Gan, C. Ma, and J. Wu. *Data clustering: theory, algorithms, and applications*. SIAM, 2007.

[48] L. Gauvin, A. Panisson, and C. Cattuto. Detecting the community structure and activity patterns of temporal networks: a non-negative tensor factorization approach. *PloS one*, 9 (1):e86028, 2014.

[49] C. Giraud-Carrier and T. Martinez. An efficient metric for heterogeneous inductive learning applications in the attribute-value language. *Intelligent Systems*, pages 341–350, 1995.

[50] B. Girault, P. Gonçalves, and É. Fleury. *Translation and Stationarity for Graph Signals*. PhD thesis, École Normale Supérieure de Lyon; Inria Rhône-Alpes, 2015.

[51] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[52] A. Gittens, P. Kambadur, and C. Boutsidis. Approximate spectral clustering via randomized sketching. *Ebay/IBM Research Technical Report*, 2013.

[53] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[54] D. E. Goldberg and J. H. Holland. Genetic algorithms and machine learning. *Machine learning*, 3(2):95–99, 1988.

[55] R. Görke, P. Maillard, A. Schumm, C. Staudt, and D. Wagner. Dynamic graph clustering combining modularity and smoothness. *Journal of Experimental Algorithmics (JEA)*, 18: 1–5, 2013.

[56] R. Guimera, M. Sales-Pardo, and L. A. N. Amaral. Modularity from fluctuations in random graphs and complex networks. *Physical Review E*, 70(2):025101, 2004.

[57] A. Haghani, M. Hamedi, K. Sadabadi, S. Young, and P. Tarnoff. Data collection of freeway travel time ground truth with bluetooth sensors. *Transportation Research Record: Journal of the Transportation Research Board*, (2160):60–68, 2010.

[58] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[59] G. E. Hinton and S. T. Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 833–840, 2002.

[60] P. W. Holland and S. Leinhardt. Transitivity in structural models of small groups. *Comparative Group Studies*, 2(2):107–124, 1971.

[61] Z. Huang. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD)*, pages 21–34. Singapore, 1997.

[62] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

[63] B. Hunter and T. Strohmer. Performance analysis of spectral clustering on compressed, incomplete and inaccurate measurements. *arXiv preprint arXiv:1011.0997*, 2010.

[64] M. Ichino and H. Yaguchi. Generalized minkowski metrics for mixed feature-type data analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):698–708, 1994.

[65] E. Isufi, A. Loukas, A. Simonetto, and G. Leus. Autoregressive moving average graph filtering. *IEEE Transactions on Signal Processing*, 65(2):274–288, 2017.

[66] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[67] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[68] N. Jardine and C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information storage and retrieval*, 7(5):217–240, 1971.

[69] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

[70] I. Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

[71] L. Kaufman and P. Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.

[72] B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 49(2):291–307, 1970.

[73] B. King. Step-wise clustering procedures. *Journal of the American Statistical Association*, 62(317):86–101, 1967.

[74] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, et al. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

[75] R. W. Klein and R. C. Dubes. Experiments in projection and clustering by simulated annealing. *Pattern Recognition*, 22(2):213–220, 1989.

[76] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

[77] B. Laurent and P. Massart. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pages 1302–1338, 2000.

[78] L. Le Magoarou and R. Gribonval. Are there approximate fast fourier transforms on graphs? In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4811–4815. IEEE, 2016.

[79] B. Leibe, K. Mikolajczyk, and B. Schiele. Efficient clustering and matching for object class recognition. In *BMVC*, pages 789–798, 2006.

[80] M. Li, X.-C. Lian, J. T. Kwok, and B.-L. Lu. Time and space efficient spectral clustering via column sampling. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2297–2304. IEEE, 2011.

[81] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28 (2):129–137, 1982.

[82] A. Loukas, A. Simonetto, and G. Leus. Distributed autoregressive moving average graph filters. *IEEE Signal Processing Letters*, 22(11):1931–1935, 2015.

[83] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[84] M. W. Mahoney and L. Orecchia. Implementing regularization implicitly via approximate eigenvector computation. *arXiv preprint arXiv:1010.0703*, 2010.

[85] J. Mao and A. K. Jain. A self-organizing network for hyperellipsoidal clustering (hec). *Ieee transactions on neural networks*, 7(1):16–29, 1996.

[86] V. A. Marchenko and L. A. Pastur. Distribution of eigenvalues for some sets of random matrices. *Matematicheskii Sbornik*, 114(4):507–536, 1967.

[87] D. S. Markovic, D. Zivkovic, I. Branovic, R. Popovic, and D. Cvetkovic. Smart power grid and cloud computing. *Renewable and Sustainable Energy Reviews*, 24:566–577, 2013.

[88] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro. Stationary graph processes and spectral estimation. *IEEE transactions on signal processing*, 65(22):5911–5926, 2017.

[89] P.-G. Martinsson, V. Rokhlin, and M. Tygert. A randomized algorithm for the decomposition of matrices. *Applied and Computational Harmonic Analysis*, 30(1):47–68, 2011.

[90] C. P. Massen and J. P. Doye. Identifying communities within energy landscapes. *Physical Review E*, 71(4):046101, 2005.

[91] J. McQueen. Some methods for classification and analysis of multivariate observations. In *Proc. Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967*, pages 281–297, 1967.

[92] A. Medus, G. Acuña, and C. Dorso. Detection of community structures in networks via global optimization. *Physica A: Statistical Mechanics and its Applications*, 358(2): 593–604, 2005.

[93] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014.

[94] M. E. Newman. Scientific collaboration networks. i. network construction and fundamental results. *Physical review E*, 64(1):016131, 2001.

[95] M. E. Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

[96] M. E. Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[97] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[98] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[99] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang. Incremental spectral clustering with application to monitoring of evolving blog communities. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, pages 261–272. SIAM, 2007.

[100] R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 165–176. IEEE, 2006.

[101] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *arXiv preprint physics/0506133*, 2005.

[102] J. Paratte. *Graph-based Methods for Visualization and Clustering.* PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2017.

[103] J. Paratte, N. Perraudin, and P. Vandergheynst. Compressive embedding and visualization using graphs. *arXiv preprint arXiv:1702.05815*, 2017.

[104] L. A. Park. Fast approximate text document clustering using compressive sampling. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 565–580. Springer, 2011.

[105] N. Perraudin. *Graph-based structures in data science: fundamental limits and applications to machine learning*. PhD thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE, 2017.

[106] N. Perraudin and P. Vandergheynst. Stationary signal processing on graphs. *IEEE Transactions on Signal Processing*, 65(13):3462–3477, 2017.

[107] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond. Gspbox: A toolbox for signal processing on graphs. *arXiv preprint arXiv:1408.5781*, 2014.

[108] N. Perraudin, B. Ricaud, D. Shuman, and P. Vandergheynst. Global and local uncertainty principles for signals on graphs. *arXiv preprint arXiv:1603.03030*, 2016.

[109] P. T. Peter Tang and E. Polizzi. Feast as a subspace iteration eigensolver accelerated by approximate spectral projection. *SIAM Journal on Matrix Analysis and Applications*, 35 (2):354–390, 2014.

[110] I. Pohl. A method for finding hamilton paths and knight's tours. *Communications of the ACM*, 10(7):446–449, 1967.

[111] M. A. Porter, J.-P. Onnela, and P. J. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.

[112] G. Puy, N. Tremblay, R. Gribonval, and P. Vandergheynst. Random sampling of bandlimited signals on graphs. *Applied and Computational Harmonic Analysis*, 2016.

[113] M. S. Pydi and A. Dukkipati. Spectral clustering via graph filtering: Consistency on the high-dimensional stochastic block model. *arXiv preprint arXiv:1702.03522*, 2017.

[114] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[115] D. Ramasamy and U. Madhow. Compressive spectral embedding: sidestepping the svd. In *Advances in Neural Information Processing Systems*, pages 550–558, 2015.

[116] A. Ruta and F. Porikli. Compressive clustering of high-dimensional data. In *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, volume 1, pages 380–385. IEEE, 2012.

[117] Y. Saeys, I. Inza, and P. Larrañaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[118] T. Sakai and A. Imiya. Fast spectral clustering with random projection and sampling. In *MLDM*, pages 372–384. Springer, 2009.

[119] A. Sandryhaila and J. M. Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656, 2013.

[120] A. Sandryhaila and J. M. Moura. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. *IEEE Signal Processing Magazine*, 31(5):80–90, 2014.

[121] P. Schuetz and A. Caflisch. Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E*, 77(4):046112, 2008.

[122] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[123] D. I. Shuman, P. Vandergheynst, and P. Frossard. Chebyshev polynomial approximation for distributed signal processing. In *Distributed Computing in Sensor Systems and Workshops (DCOSS), 2011 International Conference on*, pages 1–8. IEEE, 2011.

[124] D. I. Shuman, P. Vandergheynst, and P. Frossard. Distributed signal processing via chebyshev polynomial approximation. *arXiv preprint arXiv:1111.5239*, 2011.

[125] 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. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[126] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[127] P. H. Sneath, R. R. Sokal, et al. *Numerical taxonomy. The principles and practice of numerical classification.* 1973.

[128] D. C. Sorensen. Implicit application of polynomial filters in ak-step arnoldi method. *Siam journal on matrix analysis and applications*, 13(1):357–385, 1992.

[129] A. Susnjara, N. Perraudin, D. Kressner, and P. Vandergheynst. Accelerated filtering on graphs using lanczos method. *arXiv preprint arXiv:1509.04537*, 2015.

[130] J. Tang, J. Liu, M. Zhang, and Q. Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

[131] P. A. Traganitis, K. Slavakis, and G. B. Giannakis. Big data clustering via random sketching and validation. In *Signals, Systems and Computers, 2014 48th Asilomar Conference on*, pages 1046–1050. IEEE, 2014.

[132] N. Tremblay, G. Puy, R. Gribonval, and P. Vandergheynst. Compressive spectral clustering. In *33rd International Conference on Machine Learning*, 2016.

[133] K. Tu, B. Ribeiro, A. Swami, and D. Towsley. Detecting cluster with temporal information in sparse dynamic graph. *arXiv preprint arXiv:1605.08074*, 2016.

[134] L. Van Der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10:66–71, 2009.

[135] R. Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[136] K. Wakita and T. Tsurumi. Finding community structure in mega-scale social networks. In *Proceedings of the 16th international conference on World Wide Web*, pages 1275–1276. ACM, 2007.

[137] D. J. Watts and S. H. Strogatz. Collective dynamics of'small-world'networks. *nature*, 393 (6684):440, 1998.

[138] S. H. Weintraub. Jordan canonical form: theory and practice. *Synthesis Lectures on Mathematics and Statistics*, 2(1):1–108, 2009.

[139] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Journal of artificial intelligence research*, 1997.

[140] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[141] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420, 1997.

[142] O. Yürüten, J. Zhang, and P. H. Pu. Predictors of life satisfaction based on daily activities from mobile sensor data. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 497–500. ACM, 2014.

[143] A. Zhang. *Protein interaction networks: computational analysis*. Cambridge University Press, 2009.

Lionel Martin

Birth date: 10.08.1990
Nationality: French
Marital Status: Single
Residence permit: C

Rue des Fossés 6
1110 – Morges
+41 (0)76 / 822 34 42
lionel.lmartin@gmail.com

Education

PhD in Computer and Communication Sciences — 2012 - 2017
Laboratoire de Traitement du Signal 2 (LTS2) — *EPFL, Lausanne (CH)*

Thesis title: "Robust and Efficient Data Clustering with Signal Processing on Graphs"
Thesis director: Prof. Dr. Pierre Vandergheynst

Master degree in Communication Systems — 2010 - 2012
Specialization in Information Security — *EPFL, Lausanne (CH)*

Thesis title: "Intelligent Filtering for Graph Visualization"
Thesis supervisors: Dr. Pearl Pu, Dr. Geraldine Bous

Bachelor degree in Communication Systems — 2007 - 2010
EPFL, Lausanne (CH)

Working Experience

Co-founder — 2016 - present
IT services company. Company executive management and project management. — *Arcanite Solutions Sàrl, Renens (CH)*

Part-time developer — 2017 (Feb - June)
CTI project in collaboration with RAPT Touch SA for the development of signal processing techniques improving the signal to noise ratio for touch detection. — *EPFL, Lausanne (CH)*

Team member in NewbornCare project — 2015 - 2017
Nano-Tera research project in collaboration with CSEM and USZ on neonate's vital signs monitoring using non-invasive sensors. Development of camera based techniques for tracking infants and extracting heart rate signal. — *EPFL, Lausanne (CH)*

Teaching assistant — 2012 - 2017
Supervising students during projects, managing exercise sessions for various courses (Signal Processing, Computer Science, Physics, Calculus), attending and/or supervising the examination preparation and corrections. — *EPFL, Lausanne (CH)*

Co-founder — 2014 (June - Nov)
Startup project based on my PhD research (Big Data emotion detection) done in HCI under the direction of Dr. Pearl Pu. — *LivelyPlanet SAS, Versailles (FR)*

Business Intelligence Research Intern — 2012 (Feb - Aug)
Realization of the EPFL master thesis entitled "Intelligent Filtering for Graph Visualization". — *SAP Labs France, Mougins (FR)*

Extra-Curriculum Activities

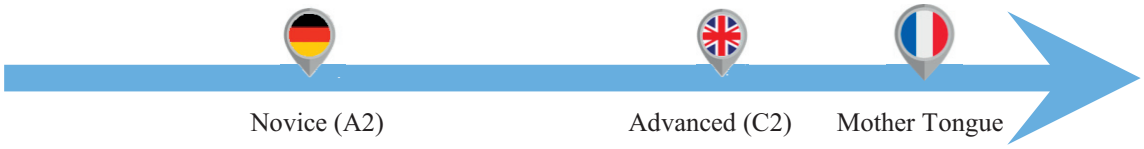
Student association "AGEPoly"
Student representation at EPFL, organization of events all along the year for EPFL students.
Member (2010/2012), Treasurer (2011/2012), Honorary Member (2012), Auditor (2013/2015)

Student association "PolyJapan"
Promotion of the Japanese culture on campus, organization of the largest cultural event in Switzerland.
Member (2008/2013), President (2009/2010), Vice-president (2011/2012)

Student association "PolyLAN"
Organization of LAN-parties on campus, Switzerland's largest event of the kind.
Member (2012/2015), Vice-president (2013/2015), Treasurer (2013/2015), Honorary Member (2015)

Badminton Club Lausanne (CSCL)
Member (2014/present), Account auditor (2016/2020)

Language Skills



IT Skills

Programming Matlab (fluent), C++ (average), Java (average), C (notions), Scala (notions)

Scripting Python (fluent), Javascript (average), Shell (average), SQL (average), Perl (notions), PHP (notions)

Markup CSS (average), HTML (average), Latex (average)

Prizes

Winner of the first phase of the World Innovation Challenge — **June 2014**
LivelyPlanet was recognized as a high potential startup in the domain of Big Data by the French government.

Laureate of the Outstanding Teaching Assistant Award — **December 2013**
Outstanding TA prize rewards students for their teaching involvement.

Laureate of the EPFL Youth Prize — **October 2012**
The Youth prize rewards the youngest of the graduates who has brilliantly completed his studies.