Laplacian Powers for Graph-Based Semi-Supervised Learning

Esteban Bautista Ruiz

Numéro National de Thèse : 2019LYSEN081

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée par
l'Ecole Normale Supérieure de Lyon

Ecole Doctorale N° 512
École Doctorale en Informatique et Mathématiques de Lyon

Spécialité de doctorat : Informatique
Discipline : **Traitement du signal, apprentissage**

Soutenue publiquement le 27/11/2019, par :
Esteban BAUTISTA RUIZ

Laplacian Powers for Graph-Based Semi-Supervised Learning

Puissances du Laplacien pour L'apprentissage Semi-Supervisé sur Graphes

---

Devant le jury composé de :

COURTY, Nicolas	Professeur des universités	Université de Bretagne sud	Rapporteur
LITVAK, Nelly	Full Professor	University of Twente	Rapporteure
BONIFATI, Angela	Professeure des universités	Université de Lyon 1	Examinatrice
FORBES, Florence	Directrice de recherche	INRIA Grenoble Rhône-Alpes	Examinatrice
RICHARD, Cédric	Professeur des Universités	Université de Nice Sophia-Antipolis	Examinateur
ABRY, Patrice	Directeur de Recherche	CNRS-ENS de Lyon	Coencadrant de thèse
GONÇALVES, Paulo	Directeur de Recherche	INRIA-ENS de Lyon	Directeur de thèse

To my parents and my sister

Acknowledgements

This dissertation is the result of three years of work carried at the Dante team of the LIP laboratory from the ENS Lyon. During these three fruitful years, many people have directly or indirectly contributed to the realization of this dissertation, whom I would like to thank.

First and foremost, I would like to thank my thesis advisors: Paulo Gonçalves and Patrice Abry. To you both, thank you for welcoming me to your group. I can not be as grateful as with you for giving me the opportunity to live this wonderful experience. Thank you for helping me in an endless list of things that made my PhD possible. From getting an apartment, going through obtaining funding for me, up to supporting me in my desire of doing an international visit to EPFL, you have been as supportive as someone can be. I would also like to thank you for your great academic influence. Many thanks for showing me how to become a researcher, for all those endless meetings where you would push me to go beyond my boundaries, and for inviting me to take part in numerous important projects and collaborations.

I would like to thank all my collaborators. A big thanks to Sarah de Nigris and Konstantin Avrachenkov for all the discussions and joint work on Lévy flights. Many thanks to Romain Fontugne for the discussions and joint work on Internet routing. I am also grateful to Pierre Vandergheynst, Benjamin Ricaud and Andreas Loukas for hosting me at the LTS2 group from EPFL and for all the discussions on evolving graph data and updating methods.

I also thank all the members of my jury for the keen interest they have shown in reviewing this thesis work.

I thank the CONACyT and the Labex Milyon for the financial support.

I would also like to thank all the people from the Dante team. In particular, I thank Gaëtan, Samuel, Jacob, Marija, Mohammed, Hadrien, Sébastien, and Mikhail for all the passionate discussions and good memories.

Last but not least, I thank all my family and friends from the bottom of my heart. Especially, I would like to thank Nadine for the emotional support.

Abstract

Graph-Based Semi-Supervised Learning (G-SSL) exploits labeled data along with the structure of unlabelled data to build better classifiers. This classification paradigm has received considerable attention since modern applications allow to collect large amounts of unlabelled but structured data, naturally encoded by a graph, in a relatively easy and inexpensive manner, while tagged data is expensive to obtain. However, despite its great success, the performance of G-SSL can still be improved, particularly in cases of graph topologies with unclear clusters, or unbalanced data settings, that this dissertation aims to address.

The main contribution of this dissertation is a novel algorithm for G-SSL coined as the L^γ-PageRank method: a generalization of the PageRank-based G-SSL by using (non-necessarily integers) powers of the combinatorial Laplacian matrix L^γ ($\gamma > 0$). The theoretical analysis of L^γ-PageRank is divided in two regimes. In the regime $\gamma < 1$, we show that L^γ-PageRank extends the standard PageRank algorithm to adopt the dynamics of Lévy processes: where random walkers are now allowed to perform long-distant jumps in a single step. In the regime $\gamma > 1$, we show that L^γ-PageRank operates on signed graphs: where nodes belonging to one same class are more likely to share positive edges while nodes from different classes are more likely to be connected with negative edges. Our main theoretical contribution is to show that L^γ-PageRank is guaranteed to outperform the standard PageRank method if γ is properly chosen. By means of numerical experimentations we point the existence of an optimal γ value maximizing performance, for which a method for its automatic estimation is devised and assessed. The practical evaluation of L^γ-PageRank on synthetic and real-world datasets commonly used for classification shows that (i) in the regime $\gamma < 1$, L^γ-PageRank can leverage the Lévy flight random walkers to enhance the detection of classes with complex local structures, such as hubs or sub-clusters; and (ii) in regime $\gamma > 1$, due to the signed graphs enhancing the separability of the data, L^γ-PageRank can significantly improve classification performance and also override the issue of unbalanced labelled data.

To increase the value of L^γ-PageRank, we investigate fast and efficient implementations that avoid the costly matrix inversion step demanded by its closed form solution. Towards this goal, by leveraging results from the field of Graph Signal Processing, we derive extensions of Power Iteration and Gauss-Southwell, successful algorithms for efficient computation of the standard PageRank algorithm, to L^γ-PageRank. Moreover, the dynamic versions of Power Iteration and Gauss-Southwell, which can update the solution of standard PageRank in sub-linear time complexity when the graph evolves or new data arrive, are also extended to L^γ-PageRank.

The main goal of G-SSL is to help solve real world problems. Towards this aim, in the last part of this dissertation we use G-SSL to address current issues in the context of Internet routing. Firstly, we use G-SSL to provide the first characterization of the scope of BGP zombies: routers that maintain routes towards of IP prefixes that have already withdrawn the Internet. By measuring the state of routers in a small set of Autonomous Systems (AS), we show that standard PageRank can predict the state of routers in ASes over which measurements are not available with an accuracy of 97% for zombie ASes and 99% for normal ASes. Then, we use G-SSL inferences to characterize the scope of BGP zombies. Secondly, we use G-SSL to address the problem of identifying the AS of inter-AS links from a network of IP addresses and AS public registers. By building a graph from traceroute measurements collected from the Internet and by collecting various types of expertized data with varying degrees of confidence from AS public registers, we show that L^γ-PageRank can solve this inference task with no errors, even when the expert does not provide labelled examples of all classes.

Résumé

Les méthodes d'apprentissage semi-supervisé sur graphes (G-SSL) exploitent un nombre raisonnable de données étiquetées, conjointement à des informations structurelles sur l'ensemble de ces données, et ce afin de construire des classifieurs plus performants. Ce paradigme de classification a fait l'objet d'une attention considérable, d'autant que les applications actuelles génèrent des quantités de données structurées de plus en plus importantes, facilement accessibles et naturellement encodées par des structures de graphes, alors que l'équitetage de ces données reste un processus souvent coûteux qui limite l'accès à des données labelisées. En dépit des nombreux progrès réalisés dans ce domaine, les performances des G-SSL sont encore perfectibles, notamment lorsqu'il s'agit de traiter des graphes présentant une faible séparabilité entre classes, ou dans le cas de forts déséquilibres entre les données des différentes classes. Ce sont précisment ces situations difficiles auxquelles nous nous intéressons dans le cadre de ce travail de thèse.

La principale contribution de cette thèse est une extension des méthodes de G-SSL classiques, qui nous a conduit à une approche originale, appelée L^γ-PageRank. L'idée que nous avons développée consiste à élever la matrice combinatoire Laplacienne de graphe – qui est au centre de la méthode PageRank – à des puissances (non nécessairement entières) L^γ $(\gamma > 0)$.

L'analyse théorique de notre proposition nécessite alors de considérer deux régimes conceptuellement distincts. Pour le cas dit fractionnaire, où $0 < \gamma < 1$, nous montrons que L^γ-PageRank généralise le concept de marches aléatoires qui sous-tend l'algorithme de PageRank standard à des dynamiques de marches plus riches, tels que les vols de Lévy. Ces derniers permettent aux marcheurs aléatoires d'atteindre des nœuds du graphes, distants de leur position courante, en un seul saut, accélérant ainsi la diffusion des étiquettes à travers les nœuds d'une même classe.

L'autre régime que nous avons étudié correspond aux valeur de $\gamma > 1$. Contrairement au cas précédent, L^γ introduit ici des poids négatifs sur les liens, rendant la méthode ininterprétable en termes de matrices de probabilité de transition. Nous avons alors montré que L^γ-PageRank effectue une classification sur un nouveau graphe signé, où les nœuds appartenant à une même classe ont une plus grande probabilité d'être connectés via des liens positifs, alors que des nœuds de classes différentes sont plus susceptibles d'être connectés par des liens pondérés négativement. La principale contribution théorique de notre travail est de garantir que L^γ-PageRank atteint des performances de classification supérieures à celles de PageRank standard, dès lors que le paramètre γ est correctement sélectionné. Expérimentalement donc, nous vérifions l'existence d'une puissance γ optimale qui max-

imise les performances de L^γ-PageRank. Puis, nous proposons une routine empirique opérationnelle qui permet de déterminer à partir du graphe des données et des étiquettes disponibles, la valeur optimale de la puissance γ.

Pour illustrer les améliorations apportées par les méthodes proposées, nous avons testé L^γ-PageRank sur un grand nombre de jeux de données, couramment utilisées pour évaluer les performances des classifieurs semi-supervisés. Les résultats obtenus montrent que: *(i)* dans le rgime $0 < \gamma < 1$, les vols de Lévy permettent à L^γ-PageRank de mieux identifier les classes prsentant des structures locales complexes, telles que des *hubs* ou des sous-groupes; *(ii)* avec $\gamma > 1$, les graphes signés accentuent significativement la séparabilité des classes, ce qui permet en particulier aux L^γ-PageRank de répondre plus efficacement au problème du déséquilibre du nombre de données étiquetées par classes.

Les contributions de ce travail de thèse sont également d'ordre algorithmique.

Un autre avantage des méthodes d'apprentissage semi-supervisé sur graphe est de fournir une solution explicite au problème de classification. Cependant, les expressions analytiques de ces solutions impliquent toutes, quel que soit L^γ-PageRank choisi, une inversion matricielle coûteuse, mal adaptée aux grands jeux de données. Nous avons donc développé des implémentations efficaces de L^γ-PageRank, qui s'appuient sur des résultats obtenus en traitement du signal sur graphes. Comme cela avait déjà été fait pour PageRank standard, nous utilisons des méthodes d'approximation de type *Power Iteration* et *Gauss-Southwell*, pour obtenir des solutions numériques de L^γ-PageRank capables de passer à l'échelle. Enfin, nous nous intéressons au contexte de l'apprentissage évolutif sur graphes, où, soit la strucuture de graphe change au cours du temps, soit les données arrivent séquentiellement. Dans les deux cas, il faut pouvoir intégrer ces évolutions à la classification, sans avoir à re-calculer intégralement la solution à chaque pas de temps. Nous avons alors développé des algorithmes dynamiques incrémentaux en complexité sous-linéaire, permettant de calculer la solution de L^γ-PageRank au fil de l'eau.

Dans la dernière partie de cette thèse, nous traitons deux applications originales de G-SSL dans le contexte du routage Internet. Tout d'abord, nous utilisons PageRank standard pour fournir une caractérisation inédite de la portée de l'influence des *zombies Boarder Gate Protocol* (BGP). Ces derniers sont des routeurs qui ont conservé les chemins vers certains préfixes ayant déjà disparu du réseau Internet. En mesurant l'état d'un groupe restreint de systèmes autonomes (AS), nous montrons que PageRank permet de prédire l'état des routeurs dans d'autres AS, sur lesquels nous n'avons aucune mesure. La précision atteinte est alors de 97% pour les AS avec routeurs zombies et 99 % pour les AS n'ayant que des routeurs à jour. Les résultats fournis par cette classification G-SSL nous permettent ensuite de caractériser le domaine d'influence des zombies BGP. Dans une deuxime application, nous abordons le problème de l'identification des systèmes autonomes (AS) connectés par des liens inter-AS, et ce, uniquement à partir du réseau d'adresses IP et des registres publics d'AS. Des expériences à partir de mesures *traceroute* d'Internet montrent que seuls les L^γ-PageRank, avec $\gamma > 1$, permettent de résoudre cette tâche sans erreur, alors même lorsqu'on ne dispose pas d'exemples étiquetés par l'expert, pour la totalité des AS (i.e., des classes).

Contents

List of Figures

List of Tables

Symbols

$\mathcal{G}$	Weighted undirected graph
W	Graph adjacency matrix
D	Graph degree matrix
$vol(S)$	Volume of set S
Δ_{uv}	Geodesic distance between nodes u and v
P	Transition probability matrix
χ_t	t-th step distribution of a random walk
π	Stationary distribution of a random walk
L^γ	γ-th Laplacian power
f	G-SSL solution
y	vector of labelled points
$f(S)$	Sum of f in the set S
$\hat{f}$	Graph Fourier Transform of f
L	Combinatorial Laplacian matrix
$\mathcal{L}_n$	Normalized Laplacian matrix
$\mathcal{L}_{rw}$	Random walk Laplacian matrix
Λ	Laplacian spectrum
$h(\Lambda)$	Graph filter
$\mathcal{V}_L$	Set of labelled points
μ	regularization parameter of G-SSL
α	Restart probability
S_{gt}	Ground truth class
h_S	Cheeger ratio of S
q	Permutation vector
W_γ	L^γ-graph adjacency matrix
D_γ	L^γ-graph degree matrix
P_γ	L^γ-graph transition matrix
$h_S^{(\gamma)}$	Generalized Cheeger ratio
$\mathcal{A}_{in}^{(\gamma)}(S)$	Within cluster agreements of set S in L^γ-graph
$\mathcal{A}_{out}^{(\gamma)}(S)$	Between cluster agreements of set S in L^γ-graph
$\mathcal{D}_{in}^{(\gamma)}(S)$	Within cluster disagreements of set S in L^γ-graph
$\mathcal{D}_{out}^{(\gamma)}(S)$	Between cluster disagreements of set S in L^γ-graph
$vol_\gamma(S)$	Volume of set S in L^γ-graph
π_γ	Stationary distribution in L^γ-graph
$\widetilde{\mathcal{G}}$	Evolved graph
$\mathcal{R}$	Reference operator of G-SSL

$\mathrm{pr}_\alpha(y)$	Personalized Pagerank vector with parameter α and seed y
$\widetilde{\mathrm{pr}}_\alpha(y)$	Personalized Pagerank vector with parameter α and seed y in evolved graph
$\mathrm{arma}_{\rho,\psi}(y)$	ARMA filter with coefficients ρ, ψ and seed y
$\widetilde{\mathrm{arma}}_{\rho,\psi(y)}$	ARMA filter with coefficients ρ, ψ and seed y in evolved graph

My Publications

Journals

1. E. Bautista, P. Abry, P. Gonçalves, ***"$\mathbf{L}^{\gamma}$-PageRank for Semi-Supervised Learning"***, Applied Network Science, volume 4, 2019.

Conference Proceedings

1. R. Fontugne, E. Bautista, C. Petrie, Y. Nomura, P. Abry, P. Gonçalves, K. Fakuda, E. Aben ***"BGP Zombies: an Analysis of Beacons Stuck Routes"***, in PAM 2019 - 20th Passive and Active Measurements Conference, 2019, (**Best paper award**).

2. S. de Nigris, E. Bautista, P. Abry, K. Avrachenkov, P. Gonçalves, ***"Fractional Graph-based Semi-Supervised Learning"***, in Proceedings of the 25th European Signal Processing Conference, 2017.

3. E. Bautista, S. de Nigris, P. Abry, K. Avrachenkov, P. Gonçalves, ***"Lévy Flights for Graph-Based Semi-Supervised Classification"***, in Proceedings of the 26th colloquium GRETSI, 2017.

Abstracts

1. S. de Nigris, E. Bautista, P. Abry, K. Avrachenkov, P. Gonçalves, ***"Fractional Graph-Based Semi-Supervised Learning"***, in International Conference on Network Science, NetSci 2018, Paris, France.

2. E. Bautista, S. de Nigris, P. Abry, P. Gonçalves, ***"$\mathbf{L}^{2}$-based PageRank for Graph-Based Semi-Supervised Learning"***, in Graph Signal Processing Workshop, GSP 2018, Lausanne, Switzerland.

Introduction

Data have a fundamental role in society. It can be argued that one important reason for the progress of civilization is that we have leveraged data to take better decisions. For instance, the deep understanding of the data output by medical tests has allowed doctors to make more precise diagnoses and to better identify optimal treatments. It is thanks to weather records and CO2 measurements that we now better understand the impact that our carbon footprint has on climate change. Also, evolutionary theory by natural selection, which is one of the major breakthroughs in human history, was devised from data collected from birds.

In today's world, data are more important than ever. They are involved in almost any human related activity. From listening to music records, to trade in goods, passing through medical examinations, up to sharing photos on internet. All these activities generate massive amounts of data which, due to technological progress, we can now store and effortlessly access. If we have historically used few available data to better solve problems, then the monumental amount of data at our disposal nowadays is a gold mine to devise a better world.

Over the years, the *machine learning* and the *signal processing* communities have proposed numerous data classification techniques to better organize and understand the large mass of data arising from the *big data* trend. Classification refers to the task of grouping data instances according to some properties they have in common. For example, given a set of documents, classification can consist in separating them by topic. From a set of bank transactions, classification may aim to group those who correspond to a fraud. From a set of emails, it consists in identifying those that are spam. It can also be the categorisation of music files according to genre, instruments, or language. The list of possible data applications is endless and classifiers constitute one of the essential tools to capitalize on them.

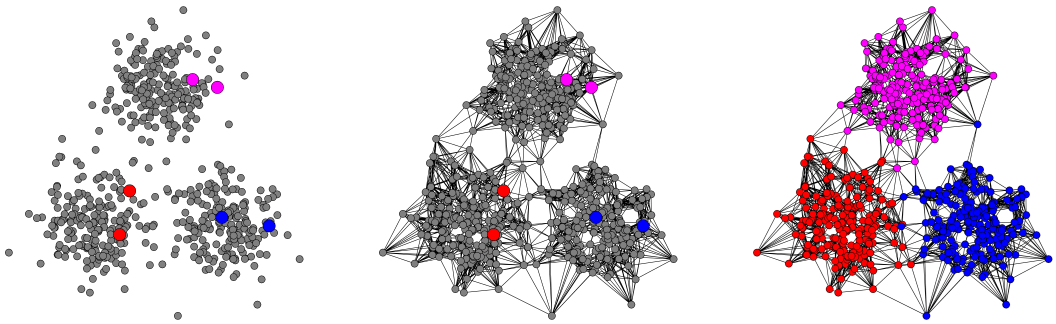
Yet, one fundamental question that arises is why state-of-the-art classifiers, which have been shown to categorize pet images with even better accuracy than humans [1], have had negligible impact in important areas such as medicine [2], despite an estimated 1.7 billion users of healthcare applications [3] and a market of roughly 600 million wearable medical devices [4] generating data? The answer is that classifiers need to *learn*: a process that involves discerning what are the determining features that make two points share a class (or not) after seeing large amounts of data instances annotated with their true label. The issue is that annotated data do not follow the *big data* trend, instead they remain scarce in numerous application domains, such as medicine, in which the process of hand-labelling

data usually requires from the intervention of human experts and the use of specialized devices. In this context, it is certainly unfeasible that medical experts can tag an amount of examples equivalent to the 1.28 million annotated images needed by the classifier of [1] to attain such reliable predictions. This lack of sufficient labelled data necessary to make *supervised* classifiers trustworthy has caused most of the data that we continue to collect to remain unanalyzed.

How to draw reliable inferences when the labelled data is insufficient? This fundamental question is the fuelling force of the modern classification paradigm known as *semi-supervised* learning on graphs (G-SSL). In G-SSL, classifiers not only learn from annotated examples but also from unlabelled data. This gives them the ability to take advantage of the *big data* trend and the power to deliver reliable inferences even from limited amounts of labelled data. The idea of learning from unlabelled data is not new to G-SSL, as previous approaches under the umbrella of *unsupervised* learning have noticed that raw data posses structural patterns informative of its class nature, yet *unsupervised* approaches have failed to have much impact since they require extensive intervention from human experts to judge if the patterns they find are beneficial in some manner. G-SSL revisits those ideas and uses the structure of unlabelled data to enrich the labelled data. To attain a synergy between these two sources of information, G-SSL represents the data by means of a similarity graph. Graphs are a powerful tool to represent and capture the structure of datasets because the usually high-dimensional datapoints become simple vertices and the similarity between datapoints is encoded by edges linking those vertices. In this context, G-SSL essentially amounts to transform the usual classification problem into one in which a graph has some of its nodes labelled and one aims to find the label for the remainders. Yet since the class nature of the data is implicit in the graph, this one allies with the labelled data to deliver reliable inferences.

To illustrate how G-SSL can draw reliable predictions by learning from both the structure of data and few annotated examples, let us give a simple illustrating example in Figure 1. Let us assume that Figure 1a represents a dataset with three classes where a few points have been hand-labelled (red, blue, magenta) and the rest are raw data that we aim to classify (grey). By themselves, the labelled points are simply too few to learn something from them, however the grey datapoints have a structure reminiscent of the class nature of the data. This extra information is key in G-SSL which builds a similarity graph to encode for it, as shown in Figure 1b. This is where the G-SSL inference problem is reached: given a graph with some labelled nodes, one aims to predict the class label for the remainder of nodes. To solve it, the graph structure and the annotated nodes operate together by successively propagating the labels to adjacent nodes in the graph until all nodes have inherited a class. As it can be seen in Figure 1c, despite having few annotated examples, G-SSL is able to enrich this information with the graph structure to deliver reliable inferences.

G-SSL procedures have already set the state-of-the-art in various applications domains [5, 6, 7, 8, 9]. Moreover, the recent proliferation of non-euclidean graph-structured datasets, such as the web-graph, social networks, protein networks, or citation networks, are tailor-made to be addressed by G-SSL, making it a tool of utmost importance to tackle some of the central problems today.



(a) raw data (b) inferred graph (c) G-SSL classification

Figure 1: Illustration of the G-SSL classification process

Nevertheless, G-SSL is still not perfect. Indeed, the state-of-the-art results indicate that when the labelled data are limited, G-SSL gives accurate classifications only under rather simple data conditions: such as when the structure of classes is well defined (separable) and without complex local structures like multiple sub-classes constituting one larger class. In this regard, if the data increases in complexity, then G-SSL may require significantly more tagged examples to remain reliable. For example, the authors of [5] employed the state-of-the-art G-SSL method [10] to classify $1,126,670$ internet users according to the topic of videos they downloaded. Their results show that if 500 labelled points per class are used, G-SSL attains a classification error of 10%, which may be sufficient to draw conclusions, but if one uses 5 labelled points per class, then the error grows to 37%, which is now too large to be beneficial. This lack of reliability in challenging data settings has caused G-SSL to continue to be mainly employed in applications where some miss-classifications can be tolerated, such as text categorization [6], or handwritten digit recognition [8]. Furthermore, it explains why it has not been seriously considered in other types of application domains such as medicine [2].

In addition, recent theoretical results [10] indicate that G-SSL methods produce biased outputs when the ratio of class size and number of annotated examples is not the same for all classes. This certainly harms the trustability of G-SSL classifiers because the size of classes cannot be known a priori and it is against the G-SSL philosophy that, in order to make a classifier reliable, one would need to discard tagged data that may have been expensive to collect. Indeed, it is unfortunate that for large datasets consisting of thousands of classes, G-SSL demands to collect labelled data in a balanced way for each and every class, or otherwise the classifier will either be biased or will missclassify an entire class whose label was never collected.

All these results point in the direction that the G-SSL paradigm is one of the most promising approaches to get the most out of data. Yet, its output tends to only be accurate under simple data settings with rather ideal conditions on the separability of data and the balancedness and availability of annotated data. In this context, the main question that I would like to address in this dissertation is the following:

How can we improve G-SSL to address the limitations listed above?

To address this question, the main contribution of this dissertation consists of a new method for G-SSL referred to as the L^γ-PageRank G-SSL. Elaborating on [10], L^γ-PageRank introduces a new degree of freedom into the G-SSL problem: γ, which, for a chosen value, changes the topology of the data and makes L^γ-PageRank solve the G-SSL problem in the new topology. The following summary of characteristics of L^γ-PageRank highlights the significance that our contribution brings to G-SSL:

1. L^γ-PageRank is theoretically guaranteed to deliver more reliable classifications than the state-of-the-art PageRank method [10] if γ is properly chosen.

2. L^γ-PageRank overrides the issue of unbalanced sets of labelled points

In addition:

- L^γ-PageRank provides an algorithm for the automatic estimation of the optimal γ to maximize performance

- L^γ-PageRank addresses better complicated data structures, such as graphs with unclear clusters or classes with sub-clusters structures or hubs

- L^γ-PageRank is tailor-made to classify data via sweep-cuts [11], implying that it does not necessarily need labelled points of all classes to operate: it can be run with labelled points of just one class and returns the nodes belonging to such class.

- L^γ-PageRank can be efficiently computed and fastly updated to classify new data or evolving graph structures

- L^γ-PageRank has been empirically assessed on extensive datasets commonly used for classification, showing significant improvements over the state-of-the-art PageRank method.

The thesis is organised as follows:

Chapter 1: In this chapter, we present fundamental background in graph theory, random walk theory and an introduction to the emerging field of Graph Signal Processing, fields upon which the thesis is built. The chapter also serves to introduce definitions and notations used throughout the thesis.

Chapter 2: This chapter starts with a thorough tour of the field of G-SSL. Then, we provide our fist contribution.

In the first part, we review the most influential and widely used G-SSL propositions [12, 13, 14, 10] and point that the PageRank-based method [10] arises as the state-of-the-art approach for G-SSL. We cover results showing that the classification assignment given by the PageRank method can be explained in terms of the theory of random walks.

Then, we point that such results imply that G-SSL methods suffer from biased outputs when facing unbalanced labelled sets. We introduce another line of works indicating that G-SSL is ill-posed when the data grows infinitely large. Proposed solutions are reviewed and special emphasis is given to the approach of [15] that proposes the use iterations of the Laplacian kernel to amend the issue.

The second part introduces G-SSL from the perspective of graph partitioning. We review partition problems and revisit results originally developed in the context of local clustering [11, 16, 17] showing that the PageRank algorithm, in conjunction with a technique called the sweep-cut, can be used to partition the graph into clusters of small Cheeger ratio. We contribute pointing that the results of [11, 16, 17], can be re-interpreted in the context of G-SSL. Moreover, we show that the idea of taking sweeps can be directly applied to G-SSL, embedding it with a larger degree of flexibility in which only the tagged points of a class are needed to run G-SSL and find the nodes belonging to such class. Lastly, we highlight that those results serve to explain better the success of the PageRank-based G-SSL method of [10].

Chapter 3: This chapter describes the core contribution of this thesis: the L^γ-PageRank method, a generalization of PageRank to (non-necessarily integers) γ-th powers of the *combinatorial* Laplacian matrix L^γ ($\gamma > 0$). For our developments, the chapter commences revisiting the Laplacian powers, already considered in [15], as a means to improve G-SSL. The key difference between our approach and the one in [15] is that [15] interprets the Laplacian powers as a Sobolev regulariser, while in our approach we show that the L^γ operator, for every fixed γ value, generates a new graph. These new graphs, which we refer to as the L^γ-graphs, reweight the links of the original structure and create edges between originally far-distant nodes. Thus, our generalized L^γ-PageRank formulation is an extension of PageRank to operate on the L^γ-graphs (for $\gamma = 1$ our algorithm then reduces to the standard PageRank algorithm). To analyse L^γ-PageRank, we show that two regimes arise: (i) $\gamma < 1$: leading to random walk transition matrices encoding for Lévy processes; and (ii) $\gamma > 1$: leading to signed graphs where edges can be positive or negative.

Regime $\gamma < 1$: we show that our L^γ-PageRank extends the regular PageRank algorithm to incorporate Lévy flight random walkers instead of the regular random walkers. The Lévy flight random walkers can jump between far-distant nodes in the graph in a single step, contrary to the regular random walkers that can only transition to adjacent neighbours. We show that the improved capacity of the Lévy walkers to explore the graph can improve the classifications of graphs with trapping regions, like strong hubs, or sub-cluster structures, that tend to harm the significance of the functions learned by the regular PageRank algorithm.

Regime $\gamma > 1$: we show that, albeit no longer modelled by random walkers because of the graph being signed, L^γ-PageRank remains a well behaved diffusion process. Then, we extend the definition of a cluster to L^γ-graphs: we say that a cluster in a signed graph is a group of nodes whose members strongly agree (positive edges) and that strongly disagree with members of other clusters (negative edges). We provide a generalization of the Cheeger ratio to assess clusters in the signed L^γ-graphs and show that, similar to the regular PageRank method that can partition graphs into clusters of small Cheeger ratio,

L^γ-PageRank is a tool to partition L^γ-graphs into clusters of small generalized Cheeger ratio. This result implies that if a L^γ-graph increases the separability of the data, then it is easier for L^γ-PageRank to classify the data. In other words, we theoretically show that if the ground truth class under search has a smaller Cheeger ratio in a L^γ-graph than in the initial graph ($\gamma = 1$), then we can more accurately identify it with L^γ-PageRank using the sweep-cut technique. By means of numerical investigations, we point the existence of an optimal γ value that maximizes performance. Therefore, we propose an algorithm that allows to estimate the optimal γ directly from the initial graph and the labeled points. Lastly, we demonstrate the classification improvements permitted by L^γ-PageRank on several real world datasets commonly used in classification, as well as the relevance of the estimation procedure for the optimal tuning. Particularly, our results demonstrate that L^γ-PageRank can: (i) significantly improve classification performance; and (ii) amend the issue of unbalanced labelled data.

Chapter 4: This chapter investigates fast and efficient implementations for our propositions in Chapter 3 and represents our third contribution. We start reviewing highly successful algorithms for efficient PageRank computation: power iteration [18, 19, 20] and Gauss-Southwell [11, 21], but that rely on the Markov chain structure of PageRank and cannot be directly used in our propositions. Then, we show that our L^γ-PageRank method can be framed in the context of graph filters, allowing us to use techniques from the field of Graph Signal Processing to efficiently implement graph filters: ARMA filters [22] and Chebyshev polynomials [23]. We then show that by using the ARMA filter structure, we can derive extensions of the power iteration and Gauss-Southwell algorithms to compute L^γ-PageRank. Part of the strong success of power iteration and the Gauss-Southwell methods for standard PageRank computation is that they possess dynamic versions [24, 25] that can update the PageRank solution in sub-linear time when the graph evolves. Therefore, we elaborate on the ARMA-based extensions of those algorithms derived in the first part of the chapter to obtain dynamic extensions of the algorithms of [24, 25] that permit to update L^γ-PageRank in sub-linear time.

Chapter 5: This chapter uses G-SSL to address issues in Internet routing. It represents our fourth contribution. The chapter starts using G-SSL to provide the first characterization of BGP zombies under BGP protocol. Then, it employs G-SSL to address the challenge of inferring topologies of autonomous systems from networks of IP addresses.

In the first part of the chapter, we use G-SSL to provide the first characterization of the scope of BGP zombies. For this characterization, we perform measurements of the Internet in a controlled environment during three periods that span across one year an a half. Our measurements track the state of a restricted set of routers, where we occasionally observe some of them maintaining a route towards a prefix that has withdrawn the Internet more than 1.5 hours ago. To assess if this anomaly occurs in isolation or at a large scale, every time it is detected in our measurements we use G-SSL to predict which autonomous systems, other than the ones over we have measurements, also have affected routers. We show that the standard PageRank algorithm detects affected autonomous systems with 97% accuracy and non-affected ones with 99% accuracy, according to a validation set constructed from tracerout measurements of the Internet. The G-SSL predictions are then used to characterize the scope of affected autonomous systems, with our results indicating

that, on average, 10% (IPv4) and 17% (IPv6) of the monitored autonomous systems are affected when the issue appears in our measurements.

In the second part of the chapter, we use G-SSL to solve the issue of inferring the topology of autonomous systems from the network of IP addresses. We perform tracerout measurements of the Internet and build a graph of IP addresses from them. Then, from publicly available ASNs registered for the IP addresses of the graph, we show that various labelled datasets with varying degrees of confidence can be constructed. We study the advantages/disadvantages of the various types of semi-supervision proposed, which offer a trade-off between amount of annotated examples and how much we can trust them. Our results show that, for the studied dataset, L^γ-PageRank with $\gamma = 2$ can solve this inference task with no errors, contrary to standard PageRank which always miss-classifies data.

Chapter 6: This chapter concludes the work and discusses future directions.

Chapter 1

Preliminaries

1.1 Graph theory

Graph theory is an important area of discrete mathematics with a long history dating back to the 18th century [26]. It focuses on the study of graphs: mathematical objects that represent pair-wise interactions between elements. Graphs were initially used to solve combinatorial problems [27], although the technological developments from the last century have given rise to numerous modern systems that can be effectively modelled by graphs. Examples of such systems range from the Internet [28], social networks [29], financial systems [30], healthcare [31], protein networks [32], smart grids [33], communication systems [34], or sensor networks [35], to name a few. Indeed, the list of applications, and the systems themselves, evolve every day, and, accordingly, the complexity of the questions they give rise to. To address such questions, it is crucial to better understand the data generated by these systems and graph theory provides the fundamental building block towards developing better data processing tools capable of taking into account the complex interconnected nature of these systems.

Graphs are made of two fundamental ingredients: a set of nodes (also referred to as vertices) and a set of edges linking these nodes. From here, variants can be devised: the edges may code for strength through a weight coefficient or may have a direction. In this thesis, we will only consider networks that are undirected, with positively weighted edges, and without self-loops (a link connecting a vertex with itself), unless otherwise stated. While this restriction leaves out various networks of major interest, the tradeoff is that it allows a much more amenable mathematical treatment while still covering the majority of modern applications.

A *graph* is denoted by the triplet $\mathcal{G}(\mathcal{V}, \mathcal{E}, w)$. By $\mathcal{V}$, we refer to the set of vertices, which we assume of carnality $|\mathcal{V}| = N$. By $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, we denote the set of edges, in which a connected pair $u, v \in \mathcal{V}$, denoted $u \sim v$, implies both $(u, v) \in \mathcal{E}$ and $(v, u) \in \mathcal{E}$. A graph is labeled as sparse if $|\mathcal{E}| = \mathcal{O}(N)$. Lastly, $w : \mathcal{E} \to \mathbb{R}^+$ is a function that assigns a real positive weight to edges.

The *adjacency matrix* of the graph is an important matrix condensing all the information from the triplet $\mathcal{G}(\mathcal{V}, \mathcal{E}, w)$. Without loss of generality, assume that each node in the graph is assigned an arbitrary and unique index from 1 to N. Then, is is defined as follows:

Definition 1. *The graph adjacency matrix is the matrix $W \in \mathbb{R}^{N\times N}$ with elements given by*

$$W_{uv} = \begin{cases} w(u,v) & u \sim v, \\ 0 & otherwise. \end{cases} \tag{1.1}$$

Note that since we assumed undirected edges the adjacency matrix is symmetric. Moreover, the no-self loops condition implies that the diagonal of W is full of zeros.

The *degree matrix* is another important matrix which encodes for the degrees of nodes. The latter are a measure of how strong are the connections towards a node. Both are defined as follows:

Definition 2. *Let $u \in \mathcal{V}$ be an arbitrary node. The degree of u, denoted d_u, is given as*

$$d_u = \sum_v W_{uv} \tag{1.2}$$

Definition 3. *The graph degree matrix is the diagonal matrix $D \in \mathbb{R}^{N\times N}$ with elements given by $D_{uu} = d_u$*

Graphs do not live in euclidean spaces, therefore we cannot compute distances between nodes using euclidean approaches. However, graphs possess an intrinsic metric usually referred to as the *shortest path distance* or the *geodesic distance* between nodes. Let the sequence $(v_1, v_2, \ldots, v_k) \in \mathcal{V} \times \mathcal{V} \times \cdots \times \mathcal{V}$, with $v_i \sim v_{i+1}$, $v_1 = u$ and $v_k = v$, denote a path of $k-1$ edges between u and v. The shortest path distance is defined as follows:

Definition 4. *The shortest path between $u, v \in \mathcal{V}$, denoted by Δ_{uv}, is the path between u and v with the minimum number of edges.*

In this work, we will extensively work with groups of nodes. We will refer to such groups via the set notation $S \subseteq \mathcal{V}$. Further, let $\mathbb{1}_S$ denote the indicator function of S, so that $(\mathbb{1}_S)_u = 1$ if $u \in S$ and $(\mathbb{1}_S)_u = 0$ otherwise. As illustrated in our next definition, the indicator function notation will allow us to write most operations involving sets in matrix form.

The *volume* of a set is a quantity that assesses how many connections reach the nodes in the set. It is defined as follows:

Definition 5. *Let $S \subseteq \mathcal{V}$. The volume of S is given by*

$$vol(S) = \sum_{u\in S} d_u = \mathbb{1}_S^T D \mathbb{1}_S \tag{1.3}$$

When we refer to the volume of the entire graph it will be denoted by $vol(\mathcal{G})$.

1.2 Graph data

1.2.1 Graph models

As discussed above, numerous graphs can naturally arise in several real world applications. However, one has hardly any control over these graphs. It is thus important to rely on graph generative models that, on the one hand, allow to vary properties under investigation by means of tunable parameters, and, on the other hand, have matrix representations with amenable theoretical properties. This subsection introduces graph models that will be employed throughout this work.

Cyclic graph

The ring graph is a 1-dimensional regular lattice with periodic boundary conditions. If we label the graph vertices from 1 to N, then the defining property of ring graphs is that the shortest path distance between nodes satisfies

$$\Delta_{uv} = \begin{cases} |u-v| & \text{if } |u-v| = 0, 1, \ldots, \lfloor N/2 \rfloor, \\ N - |u-v| & \text{if } |u-v| = \lfloor N/2 \rfloor + 1, \ldots, N, \end{cases} \tag{1.4}$$

where $\lfloor \cdot \rfloor$ is the floor function. As a result, the adjacency matrix of ring graphs has the following circulant form: $W_{uv} = 1$ if $v = u + 1$ or $v = u - 1$, and $W_{uv} = 0$ otherwise. It is well known that circulant matrices have amenable mathematical properties. Two of the most important are [36]: (i) they are diagonalized by the discrete Fourier transform; and (ii) they can be interpreted as a convolution operator on cyclic groups of N elements.

The planted partition model

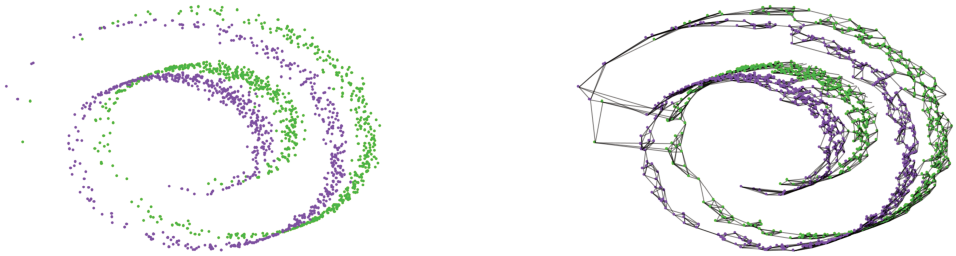
The planted partition model is an important generative model for random graphs. It is tailored to generate graphs having a community structure. A community (also referred to as a cluster in this work) denotes a group of nodes satisfying: (i) nodes in the group are strongly connected between them; and (ii) nodes in the group are poorly connected towards the rest of nodes in the graph. The planted partition model allows to control the degree of attachment between nodes within and between communities. Thus, it widely used for benchmarking tasks that involve communities.

To construct the planted partition, let the set of nodes be split into two disjoint subsets S_1 and S_2 as $\mathcal{V} = S_1 \cup S_2$, with $S_1 \cap S_2 = \varnothing$. Further, let two parameters p_{in} and p_{out} denote the probabilities of within-cluster connections and between-cluster connections. Then, for every possible pair, $u \in S_i$ and $v \in S_j$, $u \neq v$, an unweighted edge is drawn between u and v with probability p_{in} if $i = j$, or with probability p_{out} if $i \neq j$.

It is clear that if $|S_1| = |S_2|$ and $p_{in} > p_{out}$ are both satisfied, then S_1 and S_2 should form separate communities. Yet, due to the probabilistic approach, nodes from S_1 can still end up more connected to nodes from S_2 than to the rest of nodes in S_1 (depending on how much bigger is p_{in} than p_{out}). Thus, the clustering task in the planted partition is: for a given realization of the model, one must recover the true partitioning, i.e. which nodes belonged initially to S_1 and which ones to S_2.

The planted partition has been subject of extensive theoretical studies [37, 38, 39]. An important result due to Mossel et al. [40] demonstrates the existence of a detectability transition above which any unsupervised algorithm is unable to detect communities positively correlated with the true partition. To state their result, let $|S_1| = |S_2| = n$ and let $C_{out} = (p_{out})(n)$ and $C_{in} = (p_{in})(n-1)$ be the mean number of within cluster and without cluster connections of a node, respectively. The mean degree of a node is thus given by $C_{avg} = C_{in} + C_{out}$.

Theorem 1 ([40])**.** *Consider a planted partition model. Then, as $n \to \infty$, it is possible to recover a cluster that is positively correlated with the true partition, in an unsupervised manner, if $(C_{in} - C_{out})^2 > 2(C_{in} + C_{out})$, and impossible otherwise.*



(a) raw data (b) inferred graph

Figure 1.1: The Swiss roll dataset.

Stochastic block model

The stochastic block model (SBM) is a generalization of the planted partition to an arbitrary number of communities. For the K-class SBM, let the set of vertices be split in K disjoint subsets $S_1, \cdots, S_K$ as $\mathcal{V} = \cup_{k=1}^{K} S_k$. Further, let $p_{ij} = p_{ji}$ denote the probability of connecting two vertices from sets S_i and S_j. Then, for every possible pair of nodes $u \in S_i$ and $v \in S_j$, $u \neq v$, an unweighted edge is drawn between u and v with probability p_{ij}.

It was conjectured in [41] that the detectability transition of the planted partition extends to K-class SBM, so that a partition correlated with the true partition can be found if $(C_{in} - C_{out})^2 > K(C_{in} + (K-1)C_{out})$, and impossible otherwise. However, the work of [42] shows that for $K \geq 4$, it is possible to detect communities information-theoretically beyond the conjectured threshold.

Lolipop graph

The two-headed lollipop graph or barbell graph is a toy graph model useful to represent hub regions, i.e. nodes or small regions with much larger degree with respect to their surroundings. The (m, n)-lollipop graph consists of a path of n vertices with cliques (groups of nodes fully interconnected) of m nodes at the extremes of such path. The links in the path are unweighted and the links in the cliques have tunable weights.

1.2.2 Graphs constructed from raw data

Thus far, we have discussed applications in which a graph implicitly arises. However, graphs can also be effective to analyse Euclidean structured datasets. In such cases, the graph is used as a way to encode for the structure of the data. Take for instance the case of data living in a low dimensional manifold embedded in a high dimensional euclidean space. One common example of this type of data is the *Swiss roll* dataset displayed in Fig 1.1a. The datapoints of the Swiss roll are embedded in $\mathbb{R}^3$, yet their natural dimension is $\mathbb{R}^2$. Clearly, processing the data in $\mathbb{R}^3$ can easily lead to conclude that points having different color can be closer than points of the same color. Now, consider a graph inferred from the data in Fig 1.1b. Here, datapoints represent nodes in the graph and edges are placed between a point and its closest neighbours. Observe how the graph connects, for the most part, nodes sharing the same color, thus effectively capturing the structure of the data. This subsection describes popular approaches for inferring graphs from data.

Before proceeding, it is important to define a metric to assess the similarity between datapoints. Various possibilities exist: correlations, inverse of the euclidean distance, kernel functions, etc. In this work, we will employ the so-called *radial basis function* (RBF). It is defined as follows: let $u, v \in \mathbb{R}^m$ be two data points in an m-dimensional vector space and let σ be a width parameter. The similarity between u and v is thus given as

$$\text{sim}(u, v) = \exp\left(-\frac{\|u - v\|_2^2}{\sigma^2}\right) \tag{1.5}$$

ϵ-neighborhood graph construction

In the ϵ-neighborhood graph construction approach, the adjacency matrix of the graph is constructed employing the following rule

$$W_{uv} = \begin{cases} \text{sim}(u, v) & \text{if } \text{sim}(u, v) \leq \epsilon, \\ 0 & \text{otherwise,} \end{cases} \tag{1.6}$$

This is the most natural approach to capture the structure of manifolds as they seem euclidean in local regions. However, the tuning of ϵ is critical and there are not clear insights on how to choose this parameter in practice. Indeed, a bad choice of ϵ can easily lead to disconnected vertices.

K-nearest neighbor graph construction

In the K-nearest neighbor graph construction approach, the adjacency matrix is computed as

$$W_{uv} = \begin{cases} \text{sim}(u, v) & \text{if } v \text{ is within the } K \text{ most similar points of } u, \\ 0 & \text{otherwise,} \end{cases} \tag{1.7}$$

Note that this approach solves the problem of disconnected nodes. However, in practice, the tuning of K is crucial and its selection remains an open question. Normally, choosing it around $K = 10$ tends to give good results. Finally, observe that u may be in the K closest neighbors of v, but the converse may not be true. Thus, each time we set $W_{uv} = \text{sim}(u, v)$, we must also set $W_{vu} = \text{sim}(u, v)$.

1.3 Random walks on graphs

Now that we have seen that numerous problems can be modelled by graphs, we focus on tools to analyse graphs. A simple, yet powerful approach to analyse graphs is to navigate the graph and to compute statistics about it. This is the underlying mechanism of a *random walk* on a graph. The random walk process operates as follows: a walker is located at a node u at a specific time t, then it selects one node from its neighbors with probability proportional to the strength of their connection, and moves to this node at time step $t+1$. While simple, the statistics of this process, in the limit of infinite number of realizations, yield a very useful model for diffusion on graphs, upon which several precise statements can be made. Indeed, this modeling capacity of random walks has been used to characterise: users surfing the web by clicking hyperlinks between sites [43]; a disease

propagating in a population [44]; or fake news being spread and influencing political preferences [45]. Further, walkers have also leveraged to solve graph problems such as graph clustering [46], graph coloring [47], graph critical point [48], or minimum spanning tree [49].

In math terms, a discrete time random walk on a graph is a discrete time Markov chain with state space given by the vertices of the graph. The transition probabilities of the chain are encoded by the so-called *transition probability matrix* of the walk defined as follows:

Definition 6. *The transition probability matrix of a random walk is the matrix* $P \in \mathbb{R}^{N\times N}$ *with elements given by*

$$P_{uv} = \frac{W_{uv}}{d_u}, \tag{1.8}$$

where P_{uv} *denotes the probability of a walker at node* u *moving to a node* v *in the following step.*

Observe that P can be computed in matrix form as $P = D^{-1}W$. Also, since the walker always moves to a neighbor, P is right stochastic, i.e. $P_{uv} \in [0,1]$ with $\sum_v P_{uv} = 1$, or, equivalently in matrix form, the all ones vector, denoted $\mathbb{1}$, is a right eigenvector of P with eigenvalue one: $P\mathbb{1} = \mathbb{1}$.

Note that, since an individual walker is not sufficient to derive statistics, one must work with the ensemble probabilities from an infinite number of walk realizations instead. Let χ_t denote the probability vector whose u-th entry, $(\chi_t)_u$, encodes the probability of finding the walker at node u at time t. Then, the probability of finding the walker at node v at time $t+1$ is given by

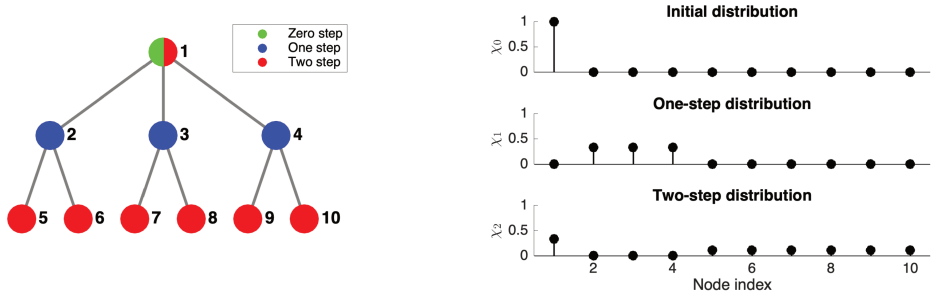
$$(\chi_{t+1})_v = \sum_u (\chi_t)_u P_{uv}. \tag{1.9}$$

In matrix form, this can be computed for all nodes as $\chi_{t+1}^T = \chi_t^T P$. Clearly, this recursive relation can be iterated to compute the distribution of the random walks at any time t by only knowing the distribution of its starting point. Thus, we have that the t-step distribution of the random walk, with starting distribution χ_0, is given as

$$\chi_t^T = \chi_0^T P^t. \tag{1.10}$$

It is important to highlight that Eq. (1.10) admits an interpretation as a diffusion process on the graph. This is, consider the entries of χ_0 to represent some amount of 'heat' or 'liquid' placed on the nodes of the graph. Then, the vector χ_t represents the state of having diffused this quantity, through the graph vertices, for t-steps via the random walkers. Let us clarify on what we intend by means of the simple example in Figure 1.2. In it, a unit mass of 'heat' is placed at node 1 at $t = 0$ (green node / top plot). Then, random walk steps are applied and, at each time step, the heat diffuses to its adjacent vertices (first step: blue/middle; second step: red/bottom). Observe that, due to $\sum_u (\chi_t)_u = 1$ for all t, the process is *mass preserving*, implying that there is no creation or dissipation of 'heat'. Thus, effectively just propagating the mass at each time step. Due to this duality, when we refer to the probability of finding a walker at a node or to the mass diffused by walkers to a node, both expressions will be indistinct.

In the limit of infinite time steps, irrespective of the initial condition χ_0, if the graph



(a) nodes visited

(b) amount of 'heat' at a node

Figure 1.2: Random walks as a diffusion process.

is connected and not bipartite (graphs that can be divided in two sets with all edges ending in different sets), then the distribution of the random walk converges to a unique *stationary distribution*. Let π denote the stationary distribution of the random walk. The defining property of this stationary state is that $\pi = \chi_t = \chi_{t+1}$ for some t, implying that the following relationship holds

$$\pi^T = \pi^T P. \tag{1.11}$$

It can easily be shown that the stationary probability at a node is always proportional to the degree of the node. This is, let $\pi_u = d_u/vol(\mathcal{G})$ and observe that it satisfies Eq. (1.11):

$$\pi^T P = \frac{1}{vol(G)} (D\mathbb{1})^T D^{-1} W = \frac{1}{vol(G)} \mathbb{1}^T W = \frac{1}{vol(G)} (D\mathbb{1})^T = \pi^T. \tag{1.12}$$

An important property of a random walk is its *mixing time*: the number of steps that the walk needs to do in order to converge to its stationary state. The mixing time is tightly related to the eigenvalues and eigenvectors of P. Indeed, this relationship appears from Eq. (1.11), where it can be seen that π is a left eigenvector of P with eigenvalue one. To elaborate, note that the right stochasticity of P implies the following two things: (i) P always has an eigenvalue equal to one; and (ii) the spectral radius of P is upper bounded by one due to Gershgoring circle theorem. Now, P has a unique eigenvalue equal to one (in absolute value) as long as the graph is not bi-partite [50, 51]. Thus, the random walk converges to a stationary state because when the matrix P is iterated in Eq. (1.10), all the eigenvalues within the unit circle decay towards zero. Clearly, since the last eigenvalue to vanish is the second largest eigenvalue (in absolute value) of P, then it controls the mixing time, meaning that if such eigenvalue is close to zero the walk mixes in only a few steps, while if it close to one the walk needs much more steps to mix.

1.4 Graph signal processing

In addition to the graph topology, in numerous applications we can also have data associated to the graph vertices. Take for example the case in which the graph represents interacting people and nodes have an age, civil status, or healthy status associated to them. It can also be that the graph represents the Internet and the nodes are websites with traffic associated to them. Clearly, such extra data living in the graph vertices can be a valuable source of information to better understand the network. To process these

data, standard data processing tools must be able to incorporate the irregular nature of the domain in which the data lives. For this reason, the field of *graph signal processing* (GSP) has emerged with the objective of extending traditional data processing tools (and to develop new approaches) to treat and analyse data supported on irregular domains. This section gives an overview of the field of GSP.

1.4.1 Graph signals

If the data supported on the graph vertices is represented by numerical values, then the concept of *signal* from classical signal processing can be generalised to arbitrary graphs.

Definition 7. *A graph signal f is defined as the mapping $f : \mathcal{V} \to \mathbb{R}$.*

Graph signals are represented by column vectors, where the u-th element of f, denoted f_u, represents the signal value at node u.

Therefore, the focal point of GSP is to process graph signals in a similar manner as we process time series or images in classical signal processing. Towards this aim, numerous efforts have been done to extend operations that are the cornerstone of classical signal processing to the graph setting. These include: filtering [52, 53, 54], prediction [55, 56], inpainting [57, 52], subsampling [58, 59], multi-resolution anaysis [60, 61], compression [62, 63], or classification [64, 65, 5].

1.4.2 Spectral theory

In classical signal processing, one of the fundamental operations to analyse a signal is to decompose it into its fundamental frequencies. As natural signals tend to be generated by multiple phenomena of diverse oscillating nature, the frequency analysis of a signal allows to isolate and enhance, extract, or suppress any source of interest. Therefore, it is of interest to extend such operation to graph signals. For it, the field of GSP has shown that the framework of *spectral graph theory*, which studies how the eigenvalues and eigenvectors of the matrix representations of graphs relate to graph properties, can be leveraged to develop a frequency analysis of graph signals.

The most fundamental decomposition of this form in classical signal processing is given by the Fourier transform. The Fourier transform of the continuous time signal $f(t)$ is given by

$$\hat{f}(\xi) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i t \xi} dt. \tag{1.13}$$

The inverse Fourier transform of $\hat{f}(\xi)$ is given by

$$f(t) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i t \xi} d\xi. \tag{1.14}$$

Thus, the inverse Fourier transform corresponds to an expansion of a signal in terms of complex exponentials, where the argument $(2\pi\xi)$ determines the frequency of oscillation of such functions. Observe how this basis also corresponds to the eigenfunctions of the 1-D Laplace operator

$$-\frac{\partial^2}{\partial t^2} e^{2\pi i t \xi} = (2\pi\xi)^2 e^{2\pi i t \xi}, \tag{1.15}$$

in which the eigenvalues $(2\pi\xi)^2$ are simply the square of the frequencies of their corresponding eigenfunction.

Building upon this observation, a graph Fourier transform can thus also be defined as expanding a graph signal in terms of the eigenfunctions of a Laplace operator defined on graphs. Thus, calling for a definition of the Laplace operator on graphs. Now, since the Laplace operator is a differential operator, various possible definitions of graph signal differentiation and graph Laplacians arise.

Laplacian definitions on graphs

Option 1: The combinatorial graph Laplacian. In this Laplacian definition, one departs from defining a differentiation operator on a graph signal f as $(df)(u,v) = \sqrt{W_{uv}}(f_u - f_u)$. Observe how the resulting derivative lives on the graph edges. Thus, to revert to functions supported on nodes, the adjoint of d is needed. The adjoint operator of d, denoted d^*, acts on a function H defined on the graph edges as $(d^*H)_u = \sum_{v\sim u}\sqrt{W_{uv}}(H(v,u) - H(u,v))$. In [66], these definitions of differentiation were shown to be consistent with the continuous definitions of the derivative of a function. Thus, they can be employed to define the Laplace operator on graphs. As the Laplacian is the adjoint of the difference operator, we obtain the following important definition:

Definition 8. *The combinatorial Laplacian operator of a graph, denoted L, acting on the graph signal f and evaluated in a node u is given by*

$$(Lf)_u = d^*(df(u,v)) = \sum_{v\sim u} W_{uv}(f_u - f_v). \tag{1.16}$$

Observe how this Laplacian definition effectively permits to incorporate information about the domain of f into the analysis of f. This is, if u and v share a strong link, then any small difference between f_u and f_v will drastically impact $(Lf)_u$, while if their connection is weak, then differences are less important.

The combinatorial Laplacian operator can be expressed in matrix form as follows:

Definition 9. *The combinatorial Laplacian matrix is the matrix $L \in \mathbb{R}^{N\times N}$ given by*

$$L = D - W \tag{1.17}$$

One important asset of this matrix representation is that the quadratic form of L, also called the *Dirichlet energy form*, gives a measure of the global regularity, or smoothness, of a graph signal with respect to its supporting domain. It is given by

$$f^T L f = \sum_{(u,v)\in\mathcal{E}} W_{uv}\left(f_u - f_v\right)^2. \tag{1.18}$$

Thus, if f is the constant function, i.e. it is completely smooth, the Dirichlet form return a value equal to zero, and, as f becomes more oscillating, the value of the form increases accordingly.

Option 2: The random walk Laplacian. A second approach to define differentiation of graph signals is by comparing the signal value of a node with respect the mean value in its vicinity.

Definition 10. *The random walk Laplacian of a graph, denoted $\mathcal{L}_{rw}$, acting on the graph signal f and evaluated in a node u is given by*

$$(\mathcal{L}_{rw}f)_u = f_u - \sum_{v\sim u} P_{uv}f_v. \tag{1.19}$$

The random walk Laplacian operator can be represented in matrix form as follows:

Definition 11. *The random walk Laplacian matrix is the matrix $\mathcal{L}_{rw} \in \mathbb{R}^{N\times N}$ given by*

$$\mathcal{L}_{rw} = \mathbb{I} - P = D^{-1}L \tag{1.20}$$

Thus, the random walk Laplacian matrix is indeed a degree normalised version of the combinatorial Laplacian matrix. However, this normalization makes this operator non-symmetric, what makes it hard to derive an equivalent of Eq. (1.18) for $\mathcal{L}_{rw}$.

Option 3: The symmetric normalized Laplacian. The last approach to define differentiation of graph signals is by comparing the signal value of a node with respect to a degree-normalized mean value of its vicinity.

Definition 12. *The normalized Laplacian of a graph, denoted $\mathcal{L}_n$, acting on the graph signal f and evaluated in a node u is given by*

$$(\mathcal{L}_n f)_u = f_u - \sum_{v\sim u} \frac{\sqrt{d_u}}{\sqrt{d_v}} P_{uv}f_v. \tag{1.21}$$

Observe that, in matrix form, this operator is a *similar* transformation of $\mathcal{L}_{rw}$ and is a symmetric normalization of L.

Definition 13. *The normalized Laplacian matrix is the matrix $\mathcal{L}_n \in \mathbb{R}^{N\times N}$ given by*

$$\mathcal{L}_n = \mathbb{I} - D^{\frac{1}{2}}PD^{-\frac{1}{2}} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}} \tag{1.22}$$

Thus, the relationship in Eq. (1.22) implies that a quadratic form with $\mathcal{L}_n$ leads to the following graph signal smoothness metric

$$f^T\mathcal{L}_n f = \sum_{(u,v)\in\mathcal{E}} W_{uv}\left(\frac{f_u}{\sqrt{d_u}} - \frac{f_v}{\sqrt{d_v}}\right)^2. \tag{1.23}$$

It is important to observe that, under this metric, the constant function f is no longer considered the smoothest function on the graph as compared to L. Thus, the choice of the Laplacian kernel plays a key role on the way regularity of graph signals is assessed.

The graph Fourier transform

Now that the Laplacian operator on graphs has been introduced, the graph Fourier transform can be defined. Let $\mathcal{L}$ denote any of the Laplace operators defined above. Then, $\mathcal{L}$ admits an spectral decomposition of the form

$$\mathcal{L} = Q\Lambda Q^{-1}, \tag{1.24}$$

where the columns of $Q \in \mathbb{R}^{N\times N}$ conform the right eigenvectors, the rows of $Q^{-1} \in \mathbb{R}^{N\times N}$ are the left eigenvectors, and $\Lambda \in \mathbb{R}^{N\times N}$ refers to a diagonal matrix of eigenvalues of $\mathcal{L}$. The graph Fourier transform (GFT) is defined as the projection of the graph signal onto the eigenfunctions of the Laplacian as follows:

Definition 14. *The graph Fourier transform of a graph signal f, denoted $\hat{f}$, is given by*

$$\hat{f} = Q^{-1} f. \tag{1.25}$$

Further, the transformation is invertible:

Definition 15. *The inverse graph Fourier transform of $\hat{f}$ is given by*

$$f = Q\hat{f}. \tag{1.26}$$

It is important to stress that in the cases of $\mathcal{L} = L$ and $\mathcal{L} = \mathcal{L}_n$, the symmetry of the operator implies a complete set of orthonormal vectors, thus $Q^{-1} = Q^T$.

We now give a discussion on how relevant it is to decompose a graph signal into the bases given by these Laplacian operators. For it, without loss of generality, assume that the eigenvalues of $\mathcal{L}$ are labeled and sorted according to their value, such that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$, and q_k denotes the eigenvector associated to λ_k.

The case of L. Let the eigenvalue problem for L be stated in terms of the Rayleigh quotient. This is, the eigenvalue λ_k and the eigenvector q_k of L are defined as the pair satisfying

$$\lambda_k = \inf_{q_k} \frac{q_k^T L q_k}{q_k^T q_k} \quad \text{s.t.} \quad q_k \perp \text{span}\{q_1, \ldots, q_{k-1}\} \tag{1.27}$$

Eq. (1.27) shows that the magnitude of the eigenvalues of L are proportional to the smoothness of their corresponding eigenvectors when measured via the Dirichlet energy form of L (see Eq. (1.18)). Thus, q_k can be interpreted as the smoothest possible function that lives in the orthogonal of $span\{q_1, \ldots, q_{k-1}\}$. This implies that the eigenvectors of L are in effect a sensible basis to decompose graph signals since eigenfunctions associated to small eigenvalues will capture the non-oscillating phenomena and the ones associated to larger eigenvalues will capture variations in the signal.

As a last note, we remark that Eq. (1.27) also shows that the eigenvalues L are always real non-negative satisfying $\lambda_1 = 0$, and if the graph has c disconnected components, then the multiplicity of the zero eigenvalue is equal to c.

The cases of $\mathcal{L}_n$ and $\mathcal{L}_{rw}$. These operators are similar, thus they share the same set of eigenvalues. Further, their eigenvectors are equal up to a rotation, i.e. if q_k denotes an eigenvector of $\mathcal{L}_n$, then $q'_k = D^{-\frac{1}{2}} q_k$ and $q''_k = D^{\frac{1}{2}} q_k$ are the right a left eigenvectors of $\mathcal{L}_{rw}$, respectively. As with L, the eigenvalue problem for these operators can also be expressed via the Rayleigh quotient. Doing it highlights that the eigenfunctions of these operators also capture oscillatory phenomena, though the basis given by $\mathcal{L}_n$ motivates smoothness through the quadratic form in Eq. (1.23), while the one of $\mathcal{L}_{rw}$ does it via the Dirichlet form of Eq. (1.18). In precise terms, let $q'_k = D^{-\frac{1}{2}} q_k$ (we show for the right eigenvectors of

$\mathcal{L}_{rw}$), then we have

$$\begin{aligned}
\lambda_k &= \inf_{q_k} \frac{q_k^T \mathcal{L}_n q_k}{q_k^T q_k} && \text{s.t.} \quad q_k \perp \text{span}\{q_1, \dots, q_{k-1}\}, \\
&= \inf_{q_k} \frac{q_k^T D^{-\frac{1}{2}} L D^{-\frac{1}{2}} q_k}{q_k^T q_k} && \text{s.t.} \quad q_k \perp \text{span}\{q_1, \dots, q_{k-1}\} \\
&= \inf_{q'_k} \frac{{q'_k}^T L q'_k}{{q'_k}^T D q'_k} && \text{s.t.} \quad q'_k \perp \text{span}\{Dq'_1, \dots, Dq'_{k-1}\}.
\end{aligned} \tag{1.28}$$

Lastly, these equations imply that both $\mathcal{L}_n$ and $\mathcal{L}_{rw}$, have non-negative real eigenvalues, with $\lambda_1 = 0$, and if the graph has c disconnected components, then the multiplicity of the zero eigenvalue is equal to c.

1.4.3 Graph filters

In classical signal processing, filtering is the process of suppressing the contribution of specific frequencies in the signal expansion. The filtering process can be done either by multiplying the transfer function of the filter with the frequency representation of the signal or by performing a convolution between the filter and the signal in the time domain. In the case of graphs, filters can also be defined, though only through the spectral domain as the convolution operation on graphs is not well defined (it is not clear what it means to shift a graph signal). In [67], the following definition of a graph filter is given:

Definition 16. *A graph filter $\mathcal{H}$ is an operator determined by the mapping $\lambda_i \to h(\lambda_i)$. In the spectral domain it is represented by the matrix $\hat{H}$ and acts on the GFT of a graph signal as*

$$\widehat{\mathcal{H}(f)} = \begin{bmatrix} h(\lambda_1)\hat{f}_1 \\ h(\lambda_2)\hat{f}_2 \\ \dots \\ h(\lambda_N)\hat{f}_N \end{bmatrix} = \begin{bmatrix} h(\lambda_1) & 0 & 0 & \dots & 0 \\ 0 & h(\lambda_2) & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & h(\lambda_N) \end{bmatrix} \begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \\ \dots \\ \hat{f}_N \end{bmatrix} = \hat{H}\hat{f}. \tag{1.29}$$

Applying the definitions of the GFT and the inverse GFT shows that the filter can be expressed in the vertex domain as

$$\mathcal{H}(f) = Q\widehat{\mathcal{H}(f)} = Qh(\lambda)Q^{-1}f = h(\mathcal{L})f. \tag{1.30}$$

Since any function of a matrix is a function on its eigenvalues, this last expression shows that any graph filter is a function on the graph Laplacian. As it will be detailed in Chapter 4, this turns out key to derive efficient implementations of graph filters.

1.4.4 The heat equation

The combinatorial graph Laplacian offers an alternative approach to random walks for diffusion on graphs. This is achieved by means of the heat equation with L as operator. It reads as follows:

$$\frac{\partial f}{\partial t} = -Lf. \tag{1.31}$$

Let f_0 denote an initial heat distribution. Then, it is easy to see that the solution to Eq. (1.31) is given as

$$f = e^{-tL} f_0 \tag{1.32}$$

By means of its Taylor expansion, it can be shown that Eq. (1.32) corresponds to a mass preserving process. This is,

$$\begin{aligned} \mathbb{1}^T f &= \mathbb{1}^T f_0 - t\mathbb{1}^T L f_0 + \frac{t^2}{2!}\mathbb{1}^T L^2 f_0 - \frac{t^3}{3!}\mathbb{1}^T L^3 f_0 + \dots \\ &= \mathbb{1}^T f_0. \end{aligned} \tag{1.33}$$

Thus, Eq. (1.32) effectively determines the state of the system after having diffused the initial mass f_0 during a time t. Indeed, we clarify that the restriction to L is because it is the only Laplacian where mass preservation can be shown for all t.

Similar to the random walk case, irrespective of the initial condition, this diffusion process converges to a predictable stationary state at $t \to \infty$. However, while the walk converges to the distribution of nodes' degrees, this process converges to a constant function. This can be seen by reverting to the spectral domain, where we have that $f = Qe^{-t\Lambda}Q^T f_0$, where $e^{-t\Lambda}$ is a diagonal matrix with entries $[e^{-t\Lambda}]_{uu} = e^{-t\lambda_u}$. Clearly, as $t \to \infty$, all the eigenmodes vanish but the one associated to $\lambda_1 = 0$, leading to $f = q_1 q_1^T f_0$, which is constant on all the nodes in the graph.

Lastly, it can be shown that the rate of convergence of this diffusion process is controlled by λ_2. This is because $e^{-t\lambda_2}$ is the last eigenmode to vanish and it vanishes faster the larger it is λ_2, thus showing again the importance of the first non-trivial eigenvalue in how fast a diffusion process is.

Chapter 2

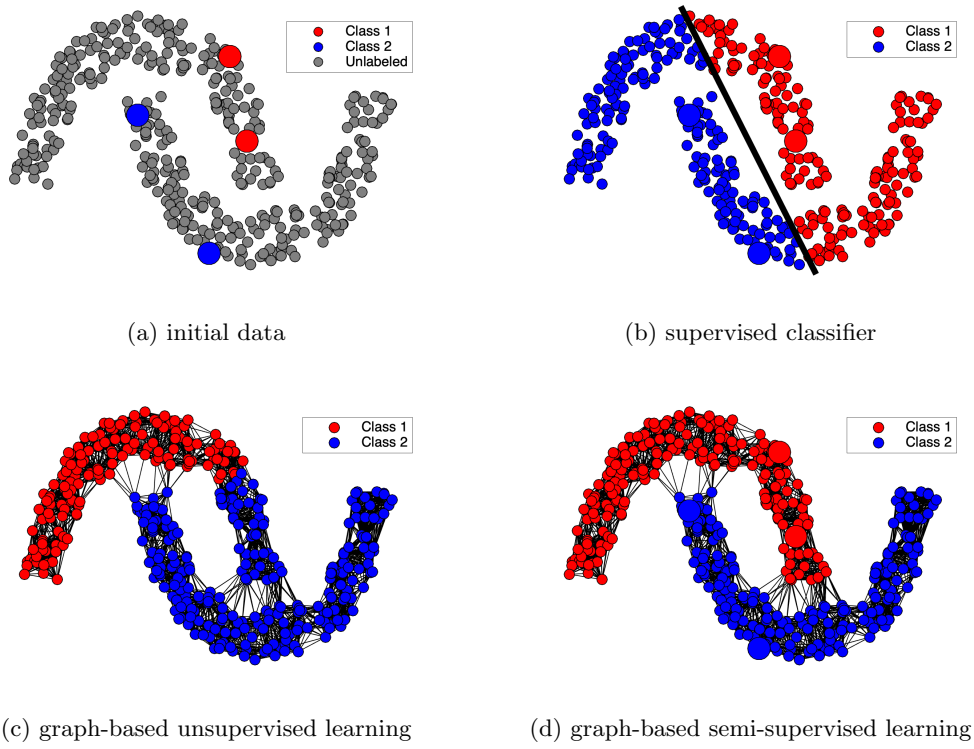
Graph-Based Semi-Supervised Learning

2.1 Introduction

In the last few years, numerous modern systems have become capable to generate massive amounts of data at a very small cost. Thus, substantially increasing the amount of data that can be readily accessed. However, despite all these data being a rich source of information, classical machine learning approaches can still learn little about it. On the one hand, *supervised learning* approaches require extensive amounts of labeled data to learn. This is problematic as the process of labelling data requires from both human intervention and specialised devices, thus it is susceptible to errors and prohibitively expensive to acquire at a large scale. On the other hand, *unsupervised learning* procedures are capable to leverage the structure of the data to learn. Nevertheless, when the data increases in complexity, it becomes too penalizing for them to overlook the valuable source of information given by the limited yet available labeled examples. As a synergy of these two approaches, *semi-supervised learning* procedures have recently emerged.

The goal of semi-supervised learning is to learn from both the structure of the data and the labelled examples. In early works, the structure of the data was incorporated to the learning problem via probabilistic approaches [68]. However, more recent works have shown that it is more effective to encode for the data structure using graphs [69]. As a result, the paradigm of graph-based semi-supervised learning (G-SSL) has attracted considerable attention. Indeed, while G-SSL was motivated for classification of structured euclidean data, the large amount of datasets found today as graphs (e.g. the Internet) motivate G-SSL on their own. G-SSL has been applied in numerous contexts, setting the state of the art in tasks such as classification of BitTorrent contents and users [5], text categorization [6], medical diagnosis [70], visual tracking [7], handwritten recognition [8], or classification of hyperspectral images [9], to name a few.

G-SSL operates under the assumption that similar points should be of the same class, so that the network structure and the labelled points can be exploited as follows: the category of the labelled nodes is propagated to their neighbors, then continues propagating to nodes further away until all nodes have inherited a class. Notice how, in this way, G-SSL is able to take full advantage of even very limited amounts of labelled points. To



(a) initial data

(b) supervised classifier

(c) graph-based unsupervised learning

(d) graph-based semi-supervised learning

Figure 2.1: The need for graph-based semi-supervised learning.

better highlight the potential of this learning philosophy, let us consider the example from Fig. 2.1. In Fig. 2.1a, we display a realization of the *two moons* data set, a classic toy example of structured data. In it, the learning task consists in detecting if a point belongs to the upper or the lower moon. To solve the task, we only dispose of a couple of labelled points per moon and the unlabelled data. Fig. 2.1b shows the result obtained by using a popular *supervised* classifier [71]. Clearly, it can be seen that the amount of labelled data is insufficient for it to learn. Now, as a way to capture the structure of the data, we build a graph from it. In Fig. 2.1c, we display the result of applying an *unsupervised* graph clustering method [72]. Notably, the incorporation of the graph structure can significantly improve the learning accuracy, albeit using only the graph structure is not sufficient to deliver a perfect result. Lastly, Fig. 2.1d shows the result of applying a G-SSL method [12]. As it can be seen, G-SSL is able to leverage the key information contained in the labelled and unlabelled points to deliver a reliable result.

In this Chapter, we give a thorough tour of the field of G-SSL, which will be the main topic of this dissertation. Traditionally, G-SSL is posed as an optimization problem under regularization constraint. As a result, most of the literature on GSSL, from books [69, 73] to recent PhD works [10, 74], have focussed on leveraging the optimization problem to propose novel and more performing objective functions. In this chapter, we will also approach G-SSL from the perspective of graph partitioning. Consequently, we devote two sections to introduce G-SSL: one covering the classical optimization perspective and the other deriving G-SSL from the point of view of graph partitioning. This alternative

perspective proves instrumental to: (i) embed G-SSL with a larger degree of flexibility in which only the tagged points of a class are needed to run G-SSL and find the nodes belonging to such class; and (ii) understand better the success of the PageRank-based G-SSL algorithm proposed in [10]. More precisely, Section 2.2 covers G-SSL from the perspective of Tikhonov regularization. In it, we introduce popular G-SSL approaches [12, 10, 13, 14] and the generalized optimization framework from [75] that encompasses most of these methods in a unified optimization framework. Special emphasis is given to the interpretation given by [75] in terms of competing random walks driving the classification process. Lastly, we cover recent methods [76, 15] addressing an issue highlighted in [77] indicating that traditional G-SSL approaches fail to operate in the limit of infinite unlabelled data. In Section 2.3, we introduce G-SSL from the point of view of graph partitioning. We show that G-SSL can be cast as a binary clustering problem. Therefore, we introduce cut problems on graphs tailored for clustering. Since these cut problems are unfeasible to solve exactly, we cover results showing that efficient approximations can be obtained, in an *unsupervised* manner, via spectral clustering [78], or, in a *semi-supervised* manner, via random walks [79, 80] and PageRank [11, 16, 17]. Section 2.4 discusses issues in G-SSL methods.

2.2 From Tikhonov regularization to G-SSL

Consider a weighted undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, w)$ and assume that each element of $\mathcal{V}$ belongs to one of K possible classes. Further, assume that the ground truth class is known for a fraction of vertices $\mathcal{V}_L \subset \mathcal{V}$. Thus, the G-SSL task is to classify the points in the complement of $\mathcal{V}_L$.

To solve the problem, G-SSL methods employ the so-called *smoothness assumption*: strongly connected nodes in the graph should be of the same class. This assumption motivates from the observation that, in the euclidean case, the decision boundary of classifiers usually lies in low density regions, indicating that it is much more likely that two data points in a dense region belong to the same class. As a result, when data points are mapped into a graph, those large density regions translate into strongly connected groups of nodes, thus motivating the smoothness assumption.

Algorithmically, G-SSL methods incorporate the smoothness assumption by casting the problem as one in which one searches for smooth functions on the graph that are consistent with the labelled data. In the literature, there are are deluge of G-SSL propositions following this rationale [81, 82, 83, 84, 85, 86, 87, 88]. However, four formulations stand out as the most performing, influential and widely used: [12, 13, 14, 10]. We detail these four propositions next.

Before proceeding, we define two important matrices. First, let $Y \in \mathbb{R}^{N\times K}$ denote a matrix of labelled points. Note that $\mathcal{V}_L$ can be written as $\mathcal{V}_L = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_K$, where $\mathcal{V}_k$ denotes the set of nodes tagged in class k. Thus, given that $|\mathcal{V}_k| \neq 0$ for all k, the entries of Y are given as

$$Y_{uk} = \begin{cases} 1 & u \in \mathcal{V}_k, \\ 0 & otherwise. \end{cases} \tag{2.1}$$

And second, let $F \in \mathbb{R}^{N\times K}$ denote a classification matrix. The k-th column of F, denoted F_{*k}, is referred to as the classification function of class k. The goal of G-SSL algorithms is first to estimate the scores of the classification matrix F given the matrix of labelled points Y, and then, interpret entry F_{uk} as how likely it is that node u belongs to class k, upon which a final classification decision is made.

2.2.1 The unnormalized Laplacian G-SSL

One of the simplest G-SSL propositions was introduced in [12]. Despite its simplicity, this method is the building block for the more involved and performing propositions, thus it constitutes a reference method in the G-SSL literature. The method proposes to find the classification functions by solving an optimization problem with the following objective function:

$$\arg\min_{F_{*k}} \left\{ F_{*k}^T L F_{*k} + \mu \left(F_{*k} - Y_{*k}\right)^T \left(F_{*k} - Y_{*k}\right) \right\}. \tag{2.2}$$

The interpretation of problem (2.2) is straightforward: the left term searches for smooth classification functions, while the right term constraints the search space by penalizing solutions that deviate from the initial labelling, and the regularization parameter $\mu > 0$ offers a trade-off between these two terms.

Since the objective function in (2.2) is a mixture of a positive semi-definite quadratic form and an ℓ_2 norm, then the problem is convex. As a result, the classification functions can be found in closed form as

$$F_{*k} = \mu \left(L + \mu \mathbb{I}\right)^{-1} Y_{*k}. \tag{2.3}$$

Finally, once the classification functions have been computed, node u is assigned to the class k satisfying:

$$F_{uk} \geq F_{uk'} \quad \forall\ k' \neq k \tag{2.4}$$

Thus, this method basically operates assigning large function values to nodes strongly connected to the labelled points and then let the function smoothly decay toward zero as we move farther away from the labels. Then, each node has K function values assigned to it and inherits the class that assigns to it the larger value. It is clear that special care must be given to the selection of μ: $\mu \to 0$ implies only considering the smoothness term in (2.2) whose minimizer is the constant function and from whom Eq. (2.4) would not be able to do any decision.

2.2.2 The normalized Laplacian-based G-SSL

In [13], the authors propose the normalized Laplacian-based G-SSL as a variation of the unnormalized Laplacian G-SSL. We observe that problem (2.2) measures smoothness via the Dirichlet form of L, thus the most natural change is to use the Dirichlet form of $\mathcal{L}_n$ instead. The work of [13] adopts this change, defining the classification functions as the solution to:

$$\arg\min_{F_{*k}} \left\{ F_{*k}^T \mathcal{L}_n F_{*k} + \mu \left(F_{*k} - Y_{*k}\right)^T \left(F_{*k} - Y_{*k}\right) \right\} \tag{2.5}$$

Following similar arguments as for the unnormalized Laplacian approach, problem (2.5) is convex with closed form solution given by

$$F_{*k} = \mu \left(\mathcal{L}_n + \mu \mathbb{I}\right)^{-1} Y_{*k}. \tag{2.6}$$

Lastly, node u is assigned to the class k according to the same rule of Eq. (2.4).

2.2.3 The standard Laplacian-based G-SSL

In [14], the so-called standard Laplacian G-SSL was proposed. This work introduces the idea that labelled points should have different importance depending on which node they are. More precisely, it is assumed in [14] that a labelled point in a node of large degree (e.g. a hub) is more informative than a labelled point in a node of small degree (e.g. an outlier), thus the objective function should be more penalizing if the classification function does not fit well labelled points in large degree nodes. As a result, [14] proposes the following optimization problem:

$$\arg\min_{F_{*k}} \left\{ F_{*k}^T L F_{*k} + \mu \left(F_{*k} - Y_{*k}\right)^T D \left(F_{*k} - Y_{*k}\right) \right\} \tag{2.7}$$

As we can see, this change is implemented by simply tweaking the norm in the fidelity term from $\langle \cdot, \cdot \rangle$ to $\langle \cdot, \cdot \rangle_D$. Moreover, the name of standard Laplacian comes from the fact that this formulation reverts to the the standard smoothness metric using L.

Since $\langle \cdot, \cdot \rangle_D$ defines a proper inner product, problem (2.7) remains convex. Therefore, its solution can be expressed in closed form as

$$F_{*k} = \mu \left(\mathcal{L}_{rw} + \mu \mathbb{I}\right)^{-1} Y_{*k}. \tag{2.8}$$

Notably, by comparing the solutions of the unnormalized Laplacian approach (Eq. (2.3)), the normalized Laplacian (Eq. (2.6)), and the standard Laplacian (Eq. (2.8)), it can be seen that the three Laplacian propositions covered in Chapter 1 appear as the fuelling force for each of these methods. Indeed, the their only difference is the election of the Laplacian kernel under consideration.

Lastly, we recall that a node u is assigned to the class k that satisfies Eq. (2.4).

2.2.4 The PageRank-based G-SSL

Building upon similar ideas, [10] proposes the PageRank-based G-SSL. This work proposes to tweak both the fitting term and the smoothness term. Concerning the fitting term, the converse to the standard Laplacian is assumed: namely, labelled points in nodes of small degree are more important than labelled points in nodes of large degrees. On the other hand, the smoothness term is normalized to impose a stronger regularity constraint between nodes having small degree. Precisely, the PageRank-based G-SSL is defined as the solution to [10]:

$$\arg\min_{F_{*k}} \left\{ F_{*k}^T D^{-1} L D^{-1} F_{*k} + \mu \left(F_{*k} - Y_{*k}\right)^T D^{-1} \left(F_{*k} - Y_{*k}\right) \right\}. \tag{2.9}$$

Observe that problem (2.9) defines a new smoothness metric:

$$F_{*k}^T D^{-1} L D^{-1} F_{*k} = \sum_{(u,v)\in\mathcal{E}} W_{uv} \left(\frac{F_{uk}}{d_u} - \frac{F_{vk}}{d_v} \right)^2. \tag{2.10}$$

Clearly, this metric and data fidelity term combine to favour nodes of small degree by both fitting better their labelled points and enforcing more regular functions on them.

Now, since D is a non-negative matrix, both terms in the objective function remain positive for all F_{*k} and all Y_{*k}. Thus, implying that problem (2.9) is convex with closed form solution given as

$$F_{*k} = \mu \left(\mathcal{L}_{rw}^T + \mu \mathbb{I} \right)^{-1} Y_{*k}. \tag{2.11}$$

The name of the method stems from the fact that, by doing a change of variable, then Eq. (2.11) can be shown to be equivalent to the personalized PageRank vector from [89]. This will be made precise below.

Lastly, [10] also proposes to assign node u to the class k satisfying to Eq. (2.4).

2.2.5 The generalized optimization framework for G-SSL

In the series of works: [75, 90, 10], a unified optimization framework for G-SSL is proposed. In these works, the following optimization formula is given:

$$\arg\min_{F_{*k}} \left\{ F_{*k}^T D^{\sigma-1} L D^{\sigma-1} F_{*k} + \mu \left(F_{*k} - Y_{*k} \right)^T D^{2\sigma-1} \left(F_{*k} - Y_{*k} \right) \right\} \tag{2.12}$$

As it can be seen, these works tweak the optimization problem by incorporating a new parameter σ. The remarkable feature is that, by properly choosing this parameter, then one can recover as particular cases some of the methods introduced above. More precisely, by setting $\sigma = 1$, objective (2.12) reduces to the standard Laplacian approach; $\sigma = 1/2$ recovers the Normalized Laplacian; and $\sigma = 0$ leads to the PageRank optimization problem.

Notably, [75] shows that the solution of this optimization problem can be cast in a unique random walk framework that helps to highlight some differences between G-SSL methods. Precisely, since the real powers of the degree matrix remain a positive semi-definite matrix, the objective above is convex with closed form solution given by

$$F_{*k} = \mu \left(D^{-\sigma} L D^{\sigma-1} + \mu \mathbb{I} \right)^{-1} Y_{*k}. \tag{2.13}$$

Then, by making the change of variable $\alpha = 1/(1+\mu)$, Eq. (2.13) can be rewritten as

$$F_{*k}^T = (1-\alpha) Y_{*k}^T \sum_{k=0}^{\infty} \alpha^k \left(D^{\sigma} P D^{-\sigma} \right)^k \tag{2.14}$$

Eq. (2.14) allows to interpret G-SSL methods as a random walk process. This is, for the PageRank case ($\sigma = 0$), F_{uk} is proportional to the expected number of visits made by random walkers to node u when they start from the labelled points of class k and, at each step, they diffuse to a neighbor with probability α or restart to the starting point with probability $1 - \alpha$. On the other hand, the standard Laplacian ($\sigma = 1$) implies that F_{uk} is proportional to the number of visits made by walkers to the labelled points of class k when they start at node u and, at each step, they diffuse to a neighbor with probability α or restart to the starting point with probability $1 - \alpha$.

This interpretation of G-SSL as a random walk process was exploited in [10] to derive the following theorem explaining the classification:

Theorem 2. *[10] Let pr_{vu} denote the probability that a random walk reaches node u before restarting to node v. Also, let unlabelled node u be assigned to the class k according to the rule: $argmax_k F_{uk}$. Then, u is assigned to the class k that satisfies the inequality*

$$\sum_{v \in \mathcal{V}_k} d_v^{\sigma} pr_{vu} \geq \sum_{w \in \mathcal{V}_{k'}} d_w^{\sigma} pr_{wu}, \quad \forall k' \neq k \tag{2.15}$$

This theorem was used in [10] to demonstrate that PageRank is the only method that provides stable classifications when $\alpha \to 1$ ($\mu \to 0$). This is, when $\alpha \to 1$ we have that $pr \to 1$, hence the inequality is controlled by the degree of the labelled points. Thus, if the classes have unbalanced densities, standard Laplacian and Normalized Laplacian easily assign all points into a single class, while PageRank can still give a meaningful classification. Moreover, the inequality implies that if the labelled points all have equal degrees, then all classification methods perform equality.

Furthermore, in [10], all the G-SSL propositions stemming from the generalized optimization framework have been extensively compared on synthetic and real datasets (les miserables, planted partition, Wikipedia, P2P traffic), and their results indicate that the PageRank-based method is the superior approach in terms of scalability, stability, robustness to classes of different densities, and performance.

2.2.6 Fitting on the labels vs fitting on the graph

In the G-SSL literature, there are two philosophies to fit the labelled data. In the first, which is the one we have considered thus far and considered in [13, 75, 90, 10], the fitting term acts on all the vertices of the graph. In the second approach, considered in [12, 14, 73], the fitting term only acts on the labelled vertices. In this subsection, we discuss their differences as we have not seen a discussion in the literature (only [73] briefly points this difference).

To simplify notations, let us denote $f = F_{*k}$ and $y = Y_{*k}$. Also, for simplicity, we elaborate on the unnormalized Laplacian approach (see Sec. 2.2.1), yet keep in mind that our discussion applies to all the G-SSL methods presented above.

Penalty 1: In this case, one fits over all the nodes in the graph as follows

$$\mathscr{L}_1 = \mu (f - y)^T (f - y) = \mu \sum_{u \in \mathcal{V}} (f_u - y_u)^2 \tag{2.16}$$

Penality 2: In this case, one only fits over the labelled data as follows

$$\mathscr{L}_2 = \mu (f - y)^T \mathbb{I}_{\mathcal{V}_L} (f - y) = \mu \sum_{u \in \mathcal{V}_L} (f_u - y_u)^2 \tag{2.17}$$

Despite looking minor, this change has important implications. We list some of them:

$\mathscr{L}_1$ enforces a stronger regularization. Note that $\mathscr{L}_1 = \mathscr{L}_2 + \sum_{u \in \mathcal{V}_L^c} f_u^2$. Thus, $\mathscr{L}_1$ enforces a stronger regularization by trying to keep the norm of f as small as possible

$\mathscr{L}_1$ leads to a graph filter and $\mathscr{L}_2$ to a non-convolutive matrix. Let us cast the G-SSL solution given by both objective functions in a unique expression as $f = \mu(L + \mu\mathcal{I})^{-1}\mathcal{I}y$,

where $\mathcal{I} = \mathbb{I}$ or $\mathcal{I} = \mathbb{I}_{\mathcal{V}_L}$ depending on whether $\mathscr{L}_1$ or $\mathscr{L}_2$ is chosen, respectively. In the former, we have that $(L + \mu\mathbb{I})^{-1} = Q(\Lambda + \mu\mathbb{I})^{-1}Q^T$, which clearly corresponds to a graph filter with response $h(\Lambda) = 1/(\Lambda + \mu)$. For the latter, the matrix $\mathbb{I}_{\mathcal{V}_L}$ is not diagonalizable under Q, thus $(L + \mu\mathbb{I}_{\mathcal{V}_L})^{-1}$ is a non-convolutive filter in the sense of [91].

$\mathscr{L}_1$ defines an invertible kernel while $\mathscr{L}_2$ requires extra regularization. From the graph filter interpretation above, we observe that the role of $\mu\mathbb{I}$ is to shift the spectrum of L, which makes the matrix $(L + \mu\mathbb{I})^{-1}$ always invertible. For the case of $\mathscr{L}_2$, it cannot be guaranteed that $\mu\mathbb{I}_{\mathcal{V}_L}$ shifts the zero eigenvalue of L, thus $(L + \mu\mathbb{I}_{\mathcal{V}_L})^{-1}$ may not be invertible. In [15], the authors propose to solve the problem in the orthogonal to the null space of L (thus working with the pseudo-inverse). Most other works address the issue by adding an extra regularization term as $\mathscr{L}_2' = \mathscr{L}_2 + \epsilon \sum_{u \in \mathcal{V}_L^c} f_u^2$. Thus, they make it resemble $\mathscr{L}_1$. As a result, the G-SSL kernel given as $(L + \mu\mathbb{I}_{\mathcal{V}_L} + \epsilon\mathbb{I})^{-1}$, which is always invertible. We are not aware of a comparison between these two approaches, though empirical experience suggest that the latter is more stable. However, such solution introduces a new parameter and there are no insights on how to tune it.

$\mathscr{L}_2$ allows to assign infinite confidence to the labelled data ($\mu \to \infty$). Consider the case of letting $\mu \to \infty$. For $\mathscr{L}_2$, this could be interpreted as forcing first $f_u = y_u$ for all $u \in \mathcal{V}_L$ and then minimize $f^T L f$. Let $f = [f_l; f_u]$, $y = [y_l; y_u]$ and $L = [L_{ll}, L_{lu}; L_{ul}, L_{uu}]$ be a split of f, y and L into their labelled and unlabelled parts. By forcing $f = [y_l, f_u]$, then minimization of $f^T L f$ leads to $f_u = -L_{uu}^{-1} L_{ul} y_l$, which is doable upon invertibility of L_{uu}. On the other hand, if we set $\mu \to \infty$ in $\mathscr{L}_1$, then we pull down towards zero the norm of f (see item 1 above) and hence no learning occurs.

$\mathscr{L}_1$ allows a random walk interpretation for some methods. As shown by theorem 2 above, when using $\mathscr{L}_1$ the classification result can be interpreted in terms of competing random walks. These type of conclusions cannot be attained under $\mathscr{L}_2$ as important properties, such as mass preservation, are not satisfied.

Due to the close connection with graph signal processing, the improved stability, and the diffusion properties, in this work we will only consider G-SSL under penalty terms of the type $\mathscr{L}_1$.

2.2.7 The limit of infinite unlabelled data

Let N data points $(x_u, u \in 1, \ldots, N)$ be drawn from a smooth probability distribution $p(x)$ on a compact manifold $\Omega \subset \mathbb{R}^N$ of intrinsic dimension d. Now, assume that a similarity graph is built by means of the Guassian kernel $W_{uv} = exp(-\frac{\|x_u - x_v\|}{\sigma})$. Further, assume that a constant amount of labelled data is given $|\mathcal{V}_L| = c$. We are interested in the behavior of G-SSL when $N \to \infty$.

Intuition says that the more data we collect, the better our knowledge of $p(x)$ will be, and G-SSL methods will profit more from it. However, it was found in [77] that when $d \geq 2$, $\sigma \to 0$ and $N \to \infty$, G-SSL methods are plagued by the so-called curse of dimensionality. More precisely, [77] elaborates on the unnormalized Laplacian (see Sec. 2.2.1) to show that, independently of μ, problem (2.2) can be minimized by 'spiky' functions that per-

fectly fit the labelled data and are constant everywhere else in the graph. In other words, it is possible to minimize the G-SSL objective with discontinuous functions that do not generalize to unlabelled data. The authors coin this problem to the first order gradient not being restrictive enough in large dimensions, causing the solution space to be too large. Namely, the regularizer converges to $f^T L f \to \int_\Omega \|\nabla f(x)\| p^2(x) dx$ when $N \to \infty$. Thus, since dx has a rather small volume in large dimensions, the integral can still be close to zero even when $\|\nabla f(x)\| \to \infty$.

The iterated Laplacian G-SSL

As a solution to the aforementioned problem, the authors of [15] proposed a novel regularization approach based on Laplacian iterations. The underlying idea of this work is to restrict the solution space of G-SSL to just continuous functions. Towards this aim, [15] proposes to use, as smoothness penalty, the m-th order Sobolev semi-norm of f instead of the standard Dirichlet form. To induce such property, the authors iterate the Laplacian operator m times in the quadratic form as follows:

$$\arg\min_f \left\{ f^T L^m f + \mu (f - y)^T (f - y) \right\}. \tag{2.18}$$

The effectiveness of this regularizer follows from the fact that, when $N \to \infty$, the regularizer satisfies:

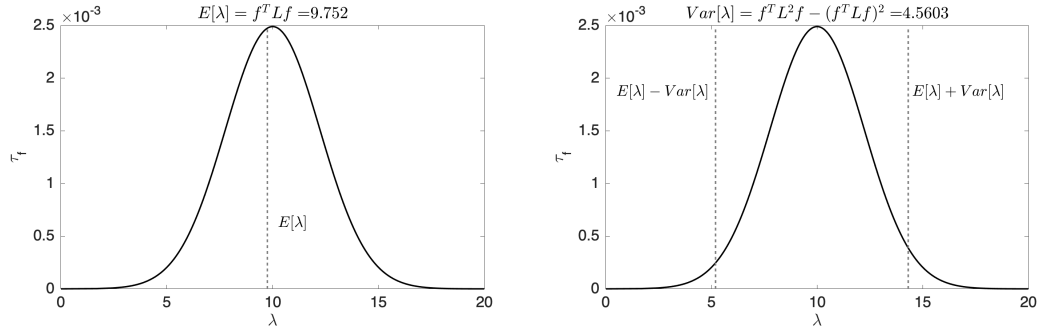
$$f^T L^m f = \int_\Omega f(x) \Delta^m f(x) dx = \sum_u \lambda_u^m |\hat{f}_u|^2. \tag{2.19}$$

Thus, pulling down $f^T L^m f$ implies searching in the space of m-times continuously differentiable functions (the Sobolev space of order m). Moreover, the authors show that such space corresponds to a reproducing kernel Hilbert space iff $2m > d$. Thus, showing that, under such condition, problem (2.18) can provide an effective solution to the curse of dimensionality.

It is important to stress that regularizer $f^T L^m f$ is not only interesting from the perspective of infinite data. Indeed, in the finite case, the spectral representation of the regularizer allows a nice interpretation in terms of the high order statistical moments of the energy spectral density of f. More precisely, since L is a self-adjoint operator, Riez-Markov representation theorem implies that

$$f^T L^m f = \sum_u \lambda_u^m |\hat{f}_u|^2 = \int \lambda^m d\tau_f \tag{2.20}$$

where $\tau_f(\lambda) = \sum_{u=1}^N |\hat{f}_u|^2 \delta(\lambda - \lambda_u)$ is a unique positive Borel measure assigned to each frequency. Clearly, expression (2.20) corresponds to the m-th moment of the set λ with respect to the measure τ_f. In other words, analogous to probabilities assigned to a random variable, the coefficients $\tau_f(\lambda_u) = |\hat{f}_u|^2$ assign a measure to each frequency λ_u and hence regularizer $f^T L^m f$ computes the m-th moment of the shape τ_f with support λ. To give a concrete example, consider a band pass graph signal f whose energy spectral density τ_f is displayed in Figure 2.2. Then, we compute the quadratic form $f^T L f$ and observe in Fig 2.2a that, indeed, its value corresponds to the first moment of the energy spectral density f (vertical line). Now, we compute $f^T L^2 f$ and show, in Figure 2.2b, that it effectively delivers the second raw moment of the energy spectral density f, from which



(a) Mean (b) Variance

Figure 2.2: Interpretation of $f^T L^m f$ as the m-th raw moment of the energy spectral density of f.

we extract and display the variance by the vertical lines. This observation allows to give a new interpretation to optimization problem (2.18). Namely, while it still searches for functions that fit the labels, for high order m values the quadratic term no longer aims for solutions with minimum mean frequency, but also discriminates parameters such as the spread the frequency content. This interpretation also explains why regularizer $f^T L^2 f$ has empirically shown superior performance to $f^T Lf$ [12, 92, 93]: minimization of the second raw moment always implies minimization of the first, but also of the variance.

It is trivial to show that the solution to problem (2.18) can be given in closed form as

$$f = \mu \left(L^m + \mu \mathbb{I}\right)^{-1} y \tag{2.21}$$

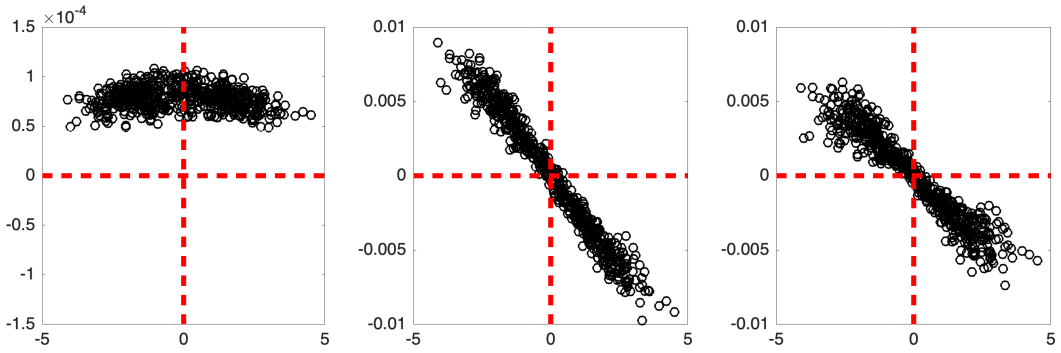
From Eq. (2.21) one can easily see that this G-SSL method corresponds to high-order low pass graph filter. Namely, by taking the spectral decomposition of L, we have that $\hat{f} = \frac{\mu}{\lambda^m + \mu} \hat{y}$. Thus, the effect of m is indeed to further penalize frequencies ($\lambda > 1$) and to soften the effect in frequencies ($\lambda < 1$).

As all the G-SSL propositions suffer from the curse of dimensionality problem, [15] proposes to replace L with any of the other Laplacian kernels, i.e $\mathcal{L}_n$, $\mathcal{L}_{rw}$, or $\mathcal{L}_{rw}^T$, so as to generalize this framework towards the other more performing G-SSL propositions.

Lastly, we recall that one of the main drawbacks of this approach is that there are no insights on how to choose m in practice.

The re-centered kernel G-SSL

A related approach has been proposed in a series of works [94, 76, 74]. These works study the more ambitious regime in which both $N \to \infty$ and $d \to \infty$ at the same rate. As part of their results, the authors demonstrate that, in this regime, $W_{uv} \to \kappa$ for all u and v. Thus, the data structure gets hidden as small 'perturbations' in the adjacency matrix. By relying on advanced techniques from random matrix theory, the authors show, on a gaussian mixture model, that by taking the Taylor series around κ, then a re-centering tweak of the G-SSL kernel is sufficient to amend the aforementioned problem and recover



(a) Normalized Laplacian (b) Iterated Normalized Laplacian (c) Re-centered kernel

Figure 2.3: Illustration of the curse of dimensionality issue in G-SSL and proposed solutions. The horizontal axis represents the spacial dimension of the data and the vertical axis represents the function value learned by G-SSL

the structure contained in the 'perturbation' of W. Precisely, the following optimization problem is proposed:

$$\arg\min_f \left\{-f^T K f + \mu(f-y)^T(f-y)\right\} \tag{2.22}$$

where $K = PWP$ and $P = \mathbb{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^T$. It is important to note that $PWP\mathbb{1} = 0$, hence this new matrix K can be interpreted as a new adjacency matrix of a new graph, where the degree of nodes is zero (hence the name of re-centering). Clearly, doing so implies creation of positive and negatives entries in K which can make the problem above non-convex, though this does not pose an issue as the fitting term implicitly constraints the norm of f. It is also important to stress that despite K being the adjacency matrix of a new graph, such graph codes for the same pair-wise similarities as the involved transformation only implies a translation of the data in the feature space. [74] gives a rigorous theoretical justification of why such translation step solves the curse of dimensionality. Those results are beyond the scope of this work, nonetheless, an intuitive explanation can be given as follows: since K has a null degree matrix, then its associated Dirichlet form satisfies $-f^T K f = \sum_{(u,v)\epsilon\mathcal{E}} K_{u,v}(f_u - f_v)^2 \leq 0$, which, when minimized, the non-informative 'flat' function caused by the curse of dimensionality now becomes the least favoured solution as it achieves the upper bound.

Illustrating example

To illustrate this phenomena, we replicate the experiment of [15] and show how these two methods solve the issue. We generate datapoints from a mixture of two gaussians in $\mathbb{R}^{20}$, where $\sigma_1 = \sigma_2 = 1$, and $\mu_1 = -1.5$, $\mu_2 = 1.5$ on the first dimension, and $\mu_1 = \mu_2 = 0$ in the remaining dimensions. A complete graph is constructed with similarities computed according to the RBF (see Eq. (1.5)) with $\sigma^2 = 20$. One labelled point per class is chosen, and, since we are in a binary setting, we set them to +1 and −1 in a single vector of labels y. We draw 300 points for each gaussian and the task is to recover the two gaussians (left is class +1, right is -1). In Fig. 2.3a, we display the results of applying the normalized Laplacian method on this data set. The points correspond to the classification function

of the unlabelled data projected into the first dimension (the one where we shifted the means). As it can be seen, the normalized Laplacian method suffers from the curse of dimensionality, retrieving a flat solution that is completely offset towards one side of the decision boundary. In Fig. 2.3b, we display the result obtained with the Iterated Laplacian approach. Here, we simply iterate the normalized Laplacian from the experiment of Fig. 2.3a for 32 times. Clearly, the Laplacian iterations correct the offset and learn a smooth function f that is consistent with the two gaussians. Lastly, in Fig. 2.3c, we display the result of applying the re-centerd kernel method. It can be seen that it is also able to alleviate the curse of dimensionality, though it displays a higher variance with respect to the iterated Laplacian. As a last remark, we highlight that the tuning μ in the re-centered kernel approach is (i) critical: a bad selection leads the method to not perform at all; (ii) narrow: the range of values where the method performs can be too restricted; and (iii) unstable: this region could significantly vary between different realizations of the gaussians or by increasing the amount of data.

2.3 From graph partitioning to G-SSL

In this section, we introduce G-SSL from the perspective of graph partitioning. In graphs, a ground truth class is represented by a group of vertices in the graph. Let us denote such group by $S_{gt} \subset \mathcal{V}$. Thus, in graphs, the classification challenge corresponds to finding the binary partition of the graph nodes as:

$$\mathcal{V} = S_{gt} \cup S_{gt}^{c}. \tag{2.23}$$

According to the smoothness assumption (see page 25), if the data is structured, then S_{gt} should form a cluster. Thus, the classification problem can be posed as a clustering one. As a result, we can use techniques to identify clusters on graphs to find a proxy $\mathcal{V} = \hat{S}_{gt} \cup \hat{S}_{gt}^{c}$.

In the graph literature, numerous techniques have been proposed to find clusters. Since the standard definition of a cluster, i.e. a group of nodes strongly connected internally and weakly connected externally, is not very precise about when a group of nodes can be considered as a cluster and when it cannot, a deluge of clustering methods have been proposed, such as clique searching [95], minimum common neighbours [96], maximum modularity [97], or minimum cuts [98], to name a few. In this work, we concentrate on the minimum cut approaches, that we detail in the next section.

2.3.1 Cut problems on graphs

Min-cut

Since clusters must be poorly connected, one of the simplest approaches to find clusters is to find two sets that have the least possible amount of connections between them. This is the rationale of the min-cut problem [99], which retrieves as a cluster, the partition minimizing the following objective:

$$\arg\min_{S} \left\{ \sum_{u \in S} \sum_{v \in S^c} W_{uv} \right\} \tag{2.24}$$

Despite this opimization problem being solvable in polynomial time [98], only under mild conditions it leads to satisfactory partitions. Its main issue is that it is extremely prone

to outliers: take an isolated node connected by a single edge with the rest of the graph, then it minimizes objective (2.24).

Ratio-cut

To amend to the min-cut outlier problem, the ratio-cut was proposed in [100]. The ratio-cut adds a constraint to penalize small partitions as follows:

$$\arg\min_{S} \left\{ \frac{\sum_{u \in S} \sum_{v \in S^c} W_{uv}}{|S| \, |S^c|} \right\} \tag{2.25}$$

Clearly, this objective should retrieve much consistent clusters, as these should not only be poorly connected between them but also should have similar sizes. However, this objective function still has some limitations. The most notorious is that it overlooks the internal connectivity of sets. Therefore, as long as two sets have similar between-cluster connections and are of similar sizes, it is indifferent for this objective function that one of them may be more strongly connected internally than the other, indicating that it delineates a better cluster.

Normalized cut

The normalized cut was proposed in [101] as an improved alternative to the ratio-cut. To capture better whether a group of nodes is a cluster, the normalized cut proposes a new metric which counts the ratio of external and internal connections of this group of nodes. This metric is usually refereed to as the Cheeger ratio of a set, or the conductance of a set. It is formally defined as follows:

Definition 17. *Let $S \subseteq \mathcal{V}$ be an arbitrary set of nodes. The Cheeger ratio, or conductance, of S is defined as:*

$$h_S = \frac{\sum_{u \in S} \sum_{v \in S^c} W_{uv}}{\min\{vol(S), vol(S^c)\}}. \tag{2.26}$$

By definition, the Cheeger ratio satisfies $h_S \in [0, 1]$ and we have that $h_S = h_{S^c}$. Furthermore, it is clear that the more a group of nodes forms a clear cluster, the smaller its Cheeger ratio. Thus, if we aim to find clusters we must search for sets of small Cheeger ratio. This is precisely the rationale of the normalized cut problem, which aims to find the partition with smallest Cheeger ratio:

$$\arg\min_{S} \{h_S\}. \tag{2.27}$$

The normalized cut problem has been shown to be NP-complete [101]. However, one can efficiently approximate its solution, in an *unsupervised* manner, by leveraging the spectral properties of graphs, or in a *semi-supervised* manner, by running a diffusion process on the graph. These relaxations are the subject of our next subsections.

2.3.2 Partitioning via spectral clustering

Let us start by introducing the relaxation via spectral clustering. This approach is unsupervised in nature as it fully relies on the spectral properties of graphs. The basis of this relaxation is the Cheeger inequality, a mathematical result that relates the minimum

Algorithm 1 Sweep-cut procedure for partitioning a graph from a vector

Input: a real valued vector f
Output: a binary partition $\mathcal{V} = \hat{S}_{gt} \cup \hat{S}_{gt}^c$
1) Let $v_1, \dots, v_N$ be a rearrangement of the vertices in descending order, so that the permutation vector q satisfies $q_{v_i} = f_{v_i}/d_{v_i} \geq q_{v_{i+1}} = f_{v_{i+1}}/d_{v_{i+1}}$
2) Let $S_j = \{v_1, \dots, v_j\}$ be the set of vertices indexed by the first j largest elements of q
3) Let $\phi(f) = \min_j h_{S_j}$
4) Retrieve $\hat{S}_{gt} = S_j$ for the set S_j achieving $\phi(f)$

value attained by the normalized cut objective and the eigenvalues of the graph.

We start defining the Cheeger constant of the graph:

$$h_G = \min_S h_S = \min_S \frac{\mathbb{1}_S^T L \mathbb{1}_S}{\mathbb{1}_S^T D \mathbb{1}_S}, \tag{2.28}$$

where the last term follows from $h_S = h_{S^c}$. The hard constraint of searching on the space of indicator functions can be relaxed to searching on the space of real-valued functions ($\mathbb{1}_S \approx g \in \mathbb{R}^N$). By implementing this change, we obtain that $h_G \approx \min_g (g^T Lg)/(g^T Dg)$. Notably, this expression corresponds to eigenvalue problem of $\mathcal{L}_{rw}$ (see page 20), thus relaxing the problem implies converting it into an eigenvalue one. However, as it is, the relaxation is useless for clustering purposes since it assumes that $h_G \approx \lambda_1 = 0$, and therefore $g = q_1 = \mathbb{1}$. A better supposition is that

$$h_G \approx \lambda_2 = \min_{g \perp D\mathbb{1}} \frac{g^T Lg}{g^T Dg}, \tag{2.29}$$

whose solution is given by the second smallest eigenvector of $\mathcal{L}_{rw}$. This eigenvector is commonly referred to as the Fiedler vector [95, 102]. Then, the Cheeger inequality essentially amounts to define a way to partition the graph from the Fiedler vector, and, if $\phi(g)$ denotes the Cheeger ratio of such partition, then the inequality bounds how far are λ_2, h_G and $\phi(g)$. In precise terms, to retrieve a partition from the Fiedler vector, we introduce the sweep-cut technique in Algorithm 1. Then, the Cheeger inequality reads as follows:

Theorem 3. *[78] Let $g' = Dg$ where D refers to the degree matrix, and let $\phi(g')$ be the Cheeger ratio of the partition obtained by applying a sweep-cut on g'. Then, h_G, λ_2, and $\phi(g')$ are related as follows:*

$$2h_G \geq \lambda_2 \geq \frac{\phi(g')^2}{2} \geq \frac{h_G^2}{2}. \tag{2.30}$$

The Sweep-cut procedure reduces the exponential search to a linear search. Moreover, theorem 3 implies that the partition retrieved by the sweep is granted to have a Cheeger ratio that is within a quadratic factor of the optimum. While this bound is not very tight, if h_G is very small then the partition retrieved by the method is likely to be a good cluster as it correponds to a small Cheeger ratio. Moreover, we recall that this lower bound guarantee is still an open problem for most other graph partitioning methods relying on the Fiedler vector, such as the popular approaches based on k-means [72].

2.3.3 Partitioning via random walks for G-SSL

Diffusion processes can also be used to find, in a semi-supervised manner, partitions with small Cheeger ratio. The rationale is the following: assume that S is a set of small Cheeger ratio (may not necessarily be the smallest), thus if a diffusion process is started within S, then it should be hard for such diffusion process to escape S as it is poorly connected with the rest of the graph. As a result, by looking at the nodes in which the diffusion process spent most of the time we should be able to identify a good approximation of S. There are two important things to remark: (i) the methodology assumes starting a diffusion process inside the set under search, thus semi-supervision plays a key role; and (ii) for the set S to display a confinement of diffusion processes, it is not necessary that it possesses the smallest Cheeger ratio in the entire graph. This highlights that diffusion-based techniques bring more flexibility allowing to find partitions with small Cheeger ratio but not necessarily the smallest (as spectral clustering does), yet we recall that if the diffusion is started within the set of smallest Cheeger ratio, then the diffusion processes will also display the confinement phenomenon and can be used to approximate the normalized cut.

In this subsection, we introduce results showing that random walks are one of the basic diffusion processes that can be used to implement these ideas. This is, if a random walk is started within a set of small Cheeger ratio, then it should get trapped for a long time within the set, meaning that if one picks the nodes with largest random walk probability as a partition, then this partition should have small Cheeger ratio.

We introduce the following result from [80], which formalizes the fact that random walks cannot not escape clusters easily.

Proposition 1. *[80] Let $S \subset \mathcal{V}$ be an arbitrary set of nodes. Let $u \in S$ be a labelled point selected with probability proportional to its degree in S, i.e. $d_u/vol(S)$. Further, let $\chi_t^{(in)}(S)$ denote the probability that a lazy random walk starting from u stays entirely in S after t steps. Then, for all $t \geq 0$, we have*

$$\mathbb{E}\left[\chi_t^{(in)}(S)\right] \geq 1 - t\frac{h_S}{2} \tag{2.31}$$

This proposition shows that smallest the Cheeger ratio of a set, the higher the probability that a walker that started inside the set remains within the set. Then, if we look at the random walk probability vector, we should identify large scores in the nodes indexing S and small scores in the nodes of S^c.

The next result we introduce states that if such mass concentration phenomenon is present on the t-step random walk probability vector χ_t, then applying the sweep-cut procedure on all the probability vectors up to t steps is guaranteed to retrieve a partition with small Cheeger ratio. Let us recall that the notation $\chi_t(S) = \sum_{u \in S}(\chi_t)_u$ means sum of entries in the nodes indexing S. Thus, we have [79, 103]:

Lemma 1. *[79, 103] Let $S \subset \mathcal{V}$ be an arbitrary set of $vol(S) \leq vol(G)/2$. Let χ_t denote the t-step lazy random walk distribution vector with seed u and π be the stationary distribution of the random walk. Let $\phi(\chi_t)$ be the Cheeger ratio of the partition obtained by applying a*

sweep-cut on χ_t. Then, the following inequality holds

$$\chi_t(S) - \pi(S) \leq \sqrt{\frac{vol(S)}{d_u}} \left(1 - \frac{\beta_t^2}{8}\right)^t \tag{2.32}$$

where $\beta_t = \inf_{t'} \phi(\chi_{t'})$ for all $t' \leq t$ and $u \in \mathcal{V}$.

This lemma says that if the probability of finding a walker in a set S after t steps is much larger than the probability of finding the walker in the set in the stationary state, then applying a sweep-cut on all the probability vectors $\chi_0 \dots \chi_t$ up to step t implies finding a partition with small Cheeger ratio. Clearly, Proposition 1 and Lemma 1 complement each other to demonstrate that if a set S has small Cheeger ratio, then diffusing random walks and applying sweeps will find a good proxy of S.

It is important to stress that the selection of t is critical and can significantly vary between graphs. For small t, walkers may not visit enough all the members of the set under search, which harms the sweep. Thus, it is better to let t grow so that the walk closely mixes. However, doing so can greatly increase the complexity of the method, as it is necessary to perform too many sweeps.

2.3.4 Partitioning via PageRank for G-SSL

PageRank is another diffusion-based algorithm that can be used to identify clusters with small Cheeger ratio. Its numerous theoretical studies [11, 16, 17, 104, 105], applications [106, 107, 108, 109] and implementations [110, 11, 111, 112] have made of PageRank a state-of-the-art clustering algorithm. In Sec 2.2.4, we introduced PageRank as a solution to a Thikhonov regularization problem and briefly discussed the interpretation of PageRank as random walk process. In this subsection, we deepen into this diffusion interpretation and introduce results which show that this diffusion process can more simply identify clusters in the graph.

Let us recall that, given a distribution, y, the PagerRank vector can be expressed as $f = \mu \left(\mathcal{L}_{rw}^T + \mu \mathbb{I}\right)^{-1} y$. Further, by doing a small change of variable $\alpha = 1/(1+\mu)$, then the PageRank vector can be cast as the solution to the fixed point equation: $f^T = (1-\alpha) y^T + \alpha f^T P$. Thus, PageRank can be interpreted as the equilibrium state of a process that, at each step, with probability α does a random walk step, or with probability $(1-\alpha)$ revisits the starting distribution y. Clearly, given proper normalization of y, the PageRank score at a particular node is equal to the probability of finding a walker, at equilibrium, at this node. Notably, this equilibrium state is the result of a well-behaved diffusion process on the graph as stated by the following Lemma [105]:

Lemma 2. *[105] Let f denote the personalized PageRank vector with personalization vector y. Then f satisfies the following properties*

1. **mass preservation**: $\sum_{u \in \mathcal{V}} f_u = \sum_{u \in \mathcal{V}} y_u$
2. **stationarity**: $f = \pi$ *if* $y = \pi$
3. **limit behavior**: $f \to \pi$ *as* $\mu \to 0$ *and* $f \to y$ *as* $\mu \to \infty$

Lemma 2 is important as it shows that PageRank is well behaved diffusion process propagating the initial mass y, through the graph, with a diffusion rate controlled by the parameter μ.

In [16], it is shown that the behavior of this diffusion process is tightly related to the cluster structure of graphs. This connection between PageRank and clustering is quantified in the following result.

Lemma 3. *[16] Let $S \subset \mathcal{V}$ be an arbitrary set with $vol(S) \leq vol(G)/2$. For a labeled point placed at a node $u \in S$ selected with probability proportional to its degree in S, i.e. $d_u/vol(S)$, the PageRank satisfies*

$$\mathbb{E}[f(S^c)] \leq \frac{h_S}{\mu}. \tag{2.33}$$

This lemma implies that if we apply PageRank diffusion from the labelled points of a set S and it has a small h_S, then the probability of finding a walker outside S is small and the nodes with largest PageRank value should index S.

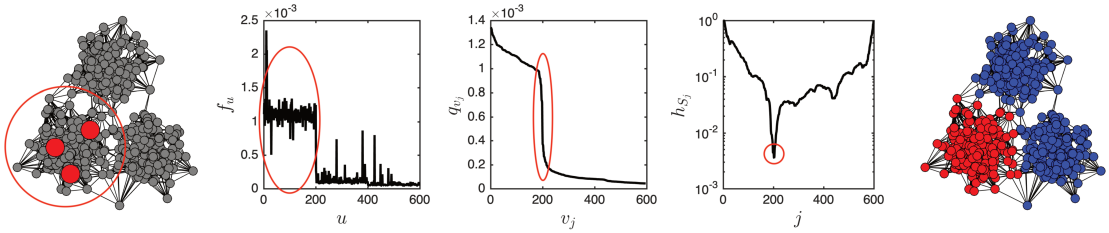
The works of [16] and [17] formalize the notion that a high concentration of PageRank mass implies a good cut. The former shows that a set with small Cheeger ratio can be found by looking for regions of high concentration of PageRank mass. The latter improves that result, showing that such set can be found more easily by looking for a sharp drop in the sorted PageRank scores. More precisely, the result of [17] shows that, when doing a sweep of the PageRank vector, if there is a sharp drop in rank at the set S_j, then the set S_j has small Cheeger ratio.

Lemma 4. *[17] Let $h \in (0,1)$, j be any index in $[1,N]$ and $\alpha \in (0,1]$ denote the PageRank restarting probability. Let $C(S_j, S_j^c) = \sum_{u \in S_j} \sum_{v \in S_j^c} W_{uv}$ be the numerator of the Cheeger ratio. Then, S_j satisfies one of the following: (a) $C(S_j, S_j^c) < 2hvol(S_j)$; or (b) there is some index $k > j$ such that $vol(S_k) \geq vol(S_j)(1+h)$ and $q_k \geq q_j - \alpha/hvol(S_j)$*

In other words, this lemma implies that either S_j has a small Cheeger ratio, or there is no sharp drop at q_j (recall that q is the permutation vector in the sweep-cut).

Illustrating example

To have a better grasp on these results, we give an illustrating example. In Fig. 2.4a, we display a simple synthetic dataset generated from a mixture of three gaussians in $\mathbb{R}^2$ and a graph build from the data. In this case S_{gt} corresponds to one of the gaussians and y is given by the red nodes. We index the vertices such that S_{gt} consists of the first 200 nodes. In Fig. 2.4b, we display the PageRank vector with initial condition y. In accordance to Lemma 3, we can observe that the PageRank vector concentrates a large amount of mass in the nodes indexing S_{gt}, thus effectively revealing it. Now, in Fig. 2.4c, we display the degree-normalized scores sorted in decreasing order, where we note that a sharp drop appears. According to Lemma 4, if the sorted scores display a sharp drop in rank, then the set associated to this rank is granted to have small Cheeger ratio. As it can be seen in Fig 2.4d, the set with smallest Cheeger ratio from the sweep coincides with the sharp drop. Finally, we output this set as our partition $\hat{S}_{gt}$. Since $\hat{S}_{gt}$ has a small Cheeger ratio, then, by definition, it is a good cluster and hence a good approximation of S_{gt} as confirmed in Fig. 2.4e.



(a) Data (b) PageRank vector (c) Sharp drop (d) Sweep-cut (e) Partition

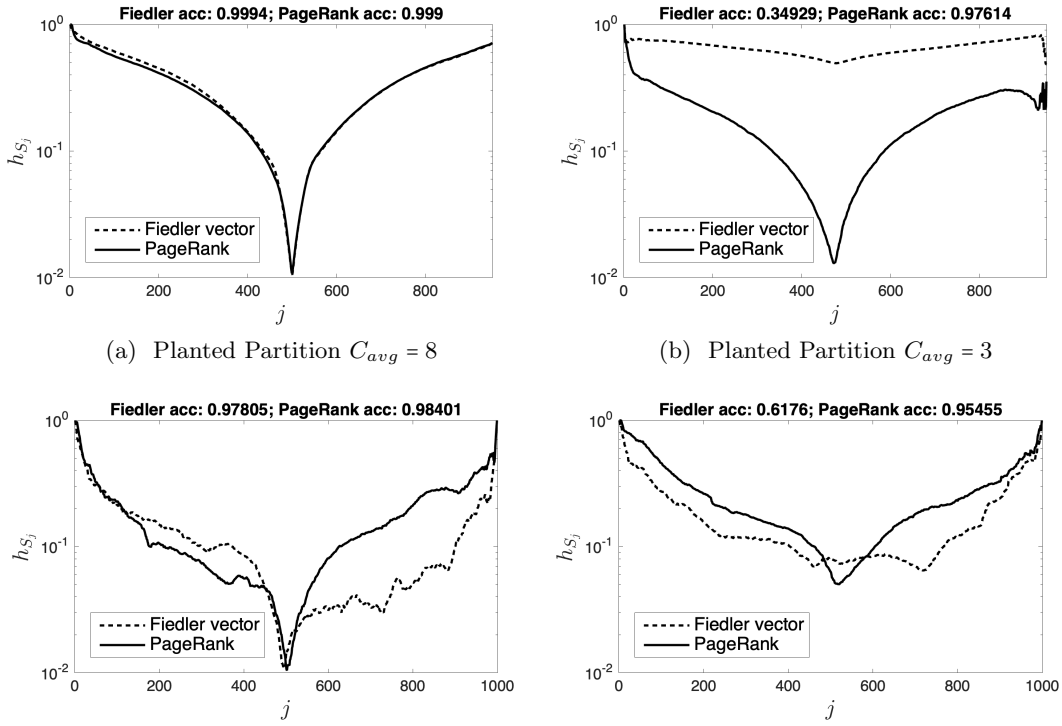
Figure 2.4: Illustration of semi-supervised graph partitioning using PageRank vectors

2.3.5 Semi-supervised vs unsupervised

We now compare the performance of PageRank and spectral clustering as relaxations to the normalized cut. We do not include the random walk-based method as PageRank is clearly easier to tune and it is much less computational demanding. Clearly, PageRank should do better than spectral clustering as it adds more information to the learning task, however this is not clear from the theoretical results as both methods have good guarantees. Thus, we present two simple numerical experiments illustrating how both methods can perform remarkably well in simple scenarios, yet, in more difficult situations, semi-supervision becomes critical.

Let us starting by considering the planted partition model. We draw 20 realizations of the model with $C_{out}/C_{in} = 0.01$, $C_{avg} = 8$, $n = 500$ and 1% of randomly labelled points (we remove disconnected nodes). Therefore, these graphs are far from the detectability transition (see theorem 1) and define clusters that, on average, have $h_S = 0.009$. Then, we apply both methods and display in Fig. 2.5a the mean shape of their sweeps and the mean accuracy of the partitions they found. Accuracy is assessed in terms of the Matthews correlation coefficient, so that a value of 1 corresponds to perfect agreement with the true partition, and a value of 0 indicates a random decision. As it can be seen, both methods perform remarkably well and, on average, retrieve partitions with Cheeger ratios close to the optimal. Now, we set $C_{avg} = 3$ while fixing the other parameters and we draw another 20 realizations of the model (we remove disconnected nodes). Here we have two important things to remark: (i) since we keep C_{out}/C_{in} fixed, the mean Cheeger ratio of these realisations remains $h_S = 0.009$; and (ii) this simple change in the mean degree drives the model much closer towards the detectability threshold, thus it is now more challenging to detect. Then, we apply both methods and display the sweep and the accuracy in Fig. 2.5b. Notably, the spectral clustering approach is no longer able to retrieve a good partition, while the PageRank method barely decreases its performance.

Our second experiment considers a point cloud formed by the images of digits 3 and 8 of the MNIST dataset. We build a graph with 500 images per digit and choose 1% of randomly labelled points. Then, we apply both methods display the sweep and the cluster accuracies in Fig. 2.5c. As it can be seen, both methods perform similarly, retrieving partitions with very high accuracies. Now, we add gaussian noise with $\mu = 0$ and $\sigma = 10$ to the point cloud and build a graph, keeping the same conditions, from this noisy point cloud. The noise causes the structure of the data to be less clear. We apply the methods



(a) Planted Partition $C_{avg} = 8$

(b) Planted Partition $C_{avg} = 3$

(c) MNIST 3vs8

(d) MNIST 3vs8 with noise

Figure 2.5: Comparing sweeps on the Fiedler vector (unsupervised) and the PageRank vector (semi-supervised) on both the Planted Partition and digits 3vs8 from the MNIST dataset.

again and show the results in Fig. 2.5d. Clearly, perturbing the data completely disrupts the eigenvectors of the Laplacian, causing the spectral clustering approach to retrieve poor partitions. On the other hand, the PageRank method displays robustness by still retrieving a very accurate partition.

2.4 Open problems

In this section, we highlight some open problems in G-SSL.

The problem of extending sweeps beyond PageRank. Section 2.2 introduced G-SSL methods that rely on comparing classification functions to classify data. One of such methods relied on comparing PageRank vectors. In Section 2.3, we boosted G-SSL with more flexibility by showing that, by doing a sweep, we only needed a single PageRank vector to find a binary partition and classify a targeted class. Therefore, we can readily extrapolate this idea to the other G-SSL propositions and boost them with extra flexibility by doing sweeps on their classification functions. However, this is not straightforward as the relationship between these other G-SSL proposition and the graph topology remains an open problem. Indeed, it is not even clear if there is any.[1] Thus, it cannot be given

[1]In turn, this lack of topological insights for most G-SSL propositions, in conjunction with the results indicating that the PageRank vectors have the natural tendency to reveal clusters, may help understand

guarantees that doing a sweep on such approaches will lead to a meaningful classification.

The problem of graphs with unclear (fuzzy) clusters. By revisiting Lemma 3, it can be seen that if S_{gt} has a small $h_{S_{gt}}$, then the PageRank method is bound to accurately detect S_{gt}. In other words, if the ground truth class under search designates a strongly disconnected cluster, then it is a set that PageRank can eventually easily detect. Clearly, the Lemma also indicates the cases in which PageRank cannot operate. Namely, as $h_{S_{gt}}$ increases, then the Lemma implies that the random walkers escape S_{gt} more easily and, as a result, the PageRank vector will not display the confinement of the information necessary for the sweep to deliver a reliable result. Thus, not very well delineated clusters pose a big challenge for PageRank.

The problem of unbalanced number of labelled points: Theorem 2 highlights an important issue affecting the G-SSL methods that arise from the generalized optimization framework. Precisely, by looking at the inequality, it can be seen that the summations depend on the cardinality of labelled points. Thus, cases of unbalanced number of labeled points can potentially bias the classification, causing unlabelled nodes prone to be assigned to the class with more labelled data. It is important to stress that preprocessing the data to even the labelled points is not a solution. Take for instance the case of two classes, one 10 times larger than the other and accordingly with 10 more labeled points. Despite starting with unbalanced labels, the problem and the inequality in Theorem 2 are balanced because, even if we have ten more units of mass to diffuse, we have ten times more nodes to reach. Thus, normalization of labelled data is only valid if classes are assumed of the same size. Notably, for the PageRank case, classifying the data using sweeps provides a solution to this problem as the behavior of the sweep is independent of the number of labelled points. Thus, we can do a multi-class classification by doing a sweep on the PageRank vector for each class and combining results. However, while this solves the unbalanced labelled data issue (and embeds topological guarantees on the multi-class output), the price to pay is that some nodes may be assigned into more than one class or to none.

The problem of hubs: The classification functions for various G-SSL can be interpreted in terms of random walks exploring the graph. Thus, the effectiveness of such approaches is based on walkers being able to visit enough the nodes under search. However, graphs with hubs or regions more densely connected than their surroundings may cause walkers to be trapped in such regions for a large time, harming the exploration capabilities of walkers and degrading the qualities of the classification functions. Therefore, graphs with a hub-like structure pose challenges to G-SSL.

better why, in general, the PageRank method performed extremely well and outperformed the other G-SSL propositions in the comparisons done in [10].

Chapter 3

L^γ-PageRank for Semi-Supervised Learning

3.1 Introduction

In Chapter 2, numerous G-SSL proposition were introduced. It was shown that in terms of flexibility, stability and theoretical understanding, the PageRank method arises as the state-of-the-art approach for G-SSL. However, various issues can still degrade the performance of PageRank, that we aim to address in this chapter.

In the face of such challenging settings, it is natural to try to enhance the performance of PageRank by embedding it with stronger regularity properties. As proposed by [15], such effect can be attained by iterating the *random walk* Laplacian in the PageRank solution. However, an issue with such approach is that it causes most of the PageRank properties to be lost. Namely, there is no known optimization problem having such expression as a solution and it is unclear if it can be given diffusion properties that can be related to the graph topology, implying that one cannot derive guarantees that a sweep still leads to a meaningful partition. Thus, due to the lack of insights on the properties and qualities of the partitions retrieved by this approach, it is hard to build upon and to address the issues listed in Chapter 2.

In this chapter, we revisit the Laplacian powers as a means to improve G-SSL and to address the aforementioned problems. We propose a novel generalization of PageRank by using (non necessarily integers) powers of the *combinatorial* Laplacian matrix L^γ ($\gamma > 0$). We coin this generalization as the L^γ-PageRank method and it constitutes the main contribution of this dissertation. In contradistinction to an iterated PageRank [15], our L^γ-PageRank: (i) enables us to have an explicit closed form expression of the underlying optimization problem; (ii) permits a diffusion and a topological interpretation; (iii) allows us to use the regime of fractional γ; and (iv) gives us insights on how to optimally tune γ to maximize performance.

In our formulation, the key ingredient is a reinterpretation of the L^γ operator. Precisely, while [15] interprets the Laplacian powers as a Sobolev norm regularizer, in our approach we show that, for every fixed γ value, a new graph is generated. These emerging graphs, which we refer to as the L^γ-graphs, reweight the links of the original structure and create

edges between originally far-distant nodes. Thus, our generalized L^γ-PageRank procedure is an extension of PageRank to operate on the L^γ-graphs. To analyse our algorithm, we show that two regimes of L^γ-graphs arise: (i) the regime of $\gamma < 1$: in this regime, the L^γ-graphs make emerge the so-called Lévy flight random walk, i.e. walkers that, with small probability, can jump between far-distant nodes in a single step; and (ii) the regime $\gamma > 1$: here, the L^γ-graphs give rise to signed graphs, i.e. graphs with positive and negative edges. Notably, both regimes carry the potential to improve G-SSL. On the one hand, the capacity of the Lévy walkers to jump far away can be convenient in settings in which the significance of the learned functions degrades due to normal random walkers getting stuck for too long in undesired graph regions, like strong hubs. On the other hand, the emergence of positive and negative edges bear the potential to enhance clustering as the signed edges introduce what can be seen as agreements (positive edges) or disagreements (negative edges) between nodes, allowing us to revamp clusters as groups of nodes agreeing between them and disagreeing with the rest of the graph. Thus, throughout the chapter we investigate the potential of these L^γ-graphs to better delineate targeted ground truth class S_{gt}. Our results are the following:

Regime $\gamma < 1$: our analysis shows that, in this regime, our L^γ-PageRank procedure corresponds to an extension of the regular PageRank algorithm that is now driven by Lévy processes. Since such extension consists in applying the regular PageRank algorithm to a L^γ-topology, then, by proving that the latter is always undirected with positively weighted edges, we guarantee that the full frame of theoretical results given in Chapter 2 directly applies to our setting. Through numerical experimentations, we demonstrate that the non-local nature of the Lévy random walkers can be useful to overcome skewed graphs with trapping regions and to enhance the detection of classes with sub-cluster structures.

Regime $\gamma > 1$: we theoretically show that, while not necessarily modelled by random walkers, our L^γ-PageRank method remains a well behaved diffusion process propagating labelled data on the signed L^γ-graphs (still, preserving the PageRank properties). Thus, we extend the Cheeger ratio definition to L^γ-graphs and prove that if there is a L^γ-graph in which S_{gt} has a smaller Cheeger ratio (w.r.t the standard case of $\gamma = 1$), then we can more accurately identify it with our generalized L^γ-PageRank procedure using the sweep-cut technique. By means of numerical investigations, we point the existence of an optimal γ value that maximizes performance. Thus, we propose an algorithm that allows to estimate the optimal γ directly from the initial graph and the labeled points. Lastly, we demonstrate the classification improvements permitted by L^γ-PageRank on several real world datasets commonly used in classification, as well as the relevance of the estimation procedure for the optimal tuning. Such results demonstrate that our L^γ-PageRank can significantly increase classification performance and also amend the issue of unbalanced labelled data.

The chapter is organized as follows. Section 3.2 introduces L^γ-graphs. Section 3.3 defines L^γ-PageRank and derives its solution. Section 3.4 analyses our algorithm in the regime $\gamma < 1$: Section 3.4.1 shows that L^γ-PageRank is an extension of standard PageRank to Lévy processes; Section 3.4.2 evaluates the classification benefits brought by the incorporation of the Lévy processes. Section 3.5 analyses our algorithm in the regime $\gamma > 1$: Section 3.5.1 analyses its clustering capabilities; Section 3.5.2 discusses the existence of

an optimal γ and its estimation; Section 3.5.3 evaluates L^γ-PageRank and the algorithm for the optimal γ estimation in practice. Section 3.6 discusses differences between L^γ-PageRank and iterated Laplacian [15] and compares them in practice. Section 3.7 extends the generalized G-SSL framework of [75] to L^γ-graphs and numerically demonstrates that the new topologies can also enhance the other G-SSL propositions.

3.2 The L^γ-graphs

In this contribution, we propose to change the graph topology in which the problem is solved as a means to improve classification. We evoke such change by considering powers of the Laplacian matrix, noting that the L^γ operator, for $\gamma > 0$, generates a new graph for every fixed γ value. More precisely, by exploiting the Laplacian definition, we define the L^γ-graphs as follows:

Definition 18. *Given a $\gamma > 0$, the L^γ-graph with adjacency matrix W_γ and degree matrix D_γ is given as*

$$L^\gamma = Q\Lambda^\gamma Q^T = D_\gamma - W_\gamma, \tag{3.1}$$

where $[D_\gamma]_{uu} = [L^\gamma]_{uu}$, and $[W_\gamma]_{uv} = -[L^\gamma]_{uv}$, with $u \neq v$.

It can easily be shown that the L^γ-graphs satisfy the Laplacian property:

Lemma 5. *For all $\gamma > 0$, the L^γ-graphs satisfy the Laplacian property:*

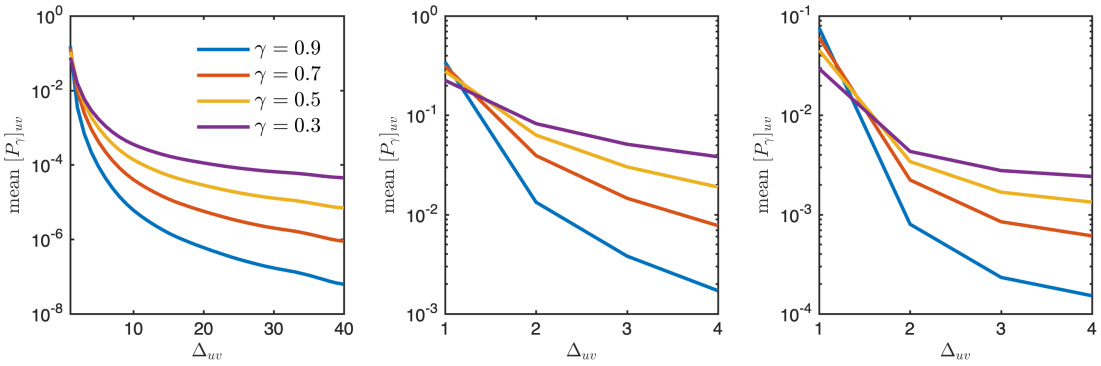
$$[D_\gamma]_{uu} = \sum_v [W_\gamma]_{uv}\,. \tag{3.2}$$

Proof. Follows trivially from the fact that, for all γ, we have that $L^\gamma \mathbb{1} = 0$. ∎

We now highlight that the L^γ-graphs possess two operational regimes: one in which the emanating graphs are undirected positively weighted graphs ($\gamma \leq 1$); and another in which the emerging topologies are undirected signed graphs ($\gamma > 1$). Precisely, let us expand the L^γ operator in its binomial series as follows:

$$\begin{aligned}
L^\gamma &= (D - W)^\gamma \\
&= \left(D^{1/2}D^{1/2} - D^{1/2}D^{-1/2}WD^{-1/2}D^{1/2}\right)^\gamma \\
&= D^{\gamma/2}\left(I - D^{-1/2}WD^{-1/2}\right)^\gamma D^{\gamma/2} \\
&= D^{\gamma/2}\Big[I - \gamma D^{-1/2}WD^{-1/2} \\
&\quad + \frac{\gamma(\gamma-1)}{2}(-D^{-1/2}WD^{-1/2})^2 \\
&\quad - \frac{\gamma(\gamma-1)(\gamma-2)}{6}(D^{-1/2}WD^{-1/2})^3 + \cdots\Big] D^{\gamma/2}.
\end{aligned} \tag{3.3}$$

Then, we can see that: (i) for $0 < \gamma \leq 1$, the infinite sum within the squared brackets entails non-positive off-diagonal terms in L^γ, thus implying that W_γ codes for an undirected graph with positive edges; and (ii) for $\gamma > 1$, the off-diagonal terms can entail positive entries, meaning that W_γ codes for an undirected signed graph. Now, while graphs with only positive edges can be considered as particular cases of signed graphs, numerous useful



(a) Swiss roll (b) Lollipop (c) Planted partition

Figure 3.1: Mean transition probabilities arising in the matrix $P_\gamma = D_\gamma^{-1} W_\gamma$ as a function of the initial geodesic distance between nodes. We recall that $[P_1]_{uv} = 0$ if $\Delta_{uv} > 1$.

results developed for the former have not yet been extended to the latter, which remains widely unexplored. Thus, it is better to treat each regime independently.

The remarkable aspect about the L^γ-graphs is that the topologies emerging from both regimes carry the potential to improve classification.

3.2.1 Regime of $\gamma < 1$

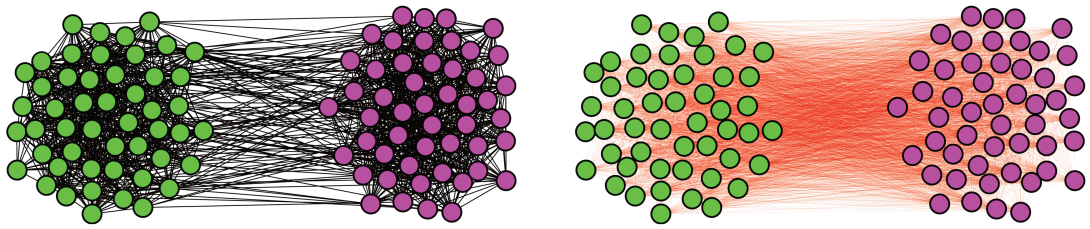
As described above, in this regime, the L^γ-graphs make emerge positively weighted graphs. Thus, we can compute a stochastic transition matrix from these graphs as $P_\gamma = D_\gamma^{-1} W_\gamma$. In [113], the following result is given concerning P_γ:

Theorem 4 ([113]). *Consider an infinitely large 1-D regular lattice with periodic boundary conditions. Then, as $\gamma \to 0$ and $\gamma \to 1$, we have*

$$[P_\gamma]_{uv} \sim \Delta_{uv}^{-(2\gamma+1)} \tag{3.4}$$

This theorem says that, on very large rings, the associated L^γ-graphs make emerge transition matrices that allow random walkers to jump between any pair of nodes in the graph with a probability that is a power law of their distance. Thus, while in the initial ring a walker can only transition to its adjacent neighbors, in the L^γ-topologies the walkers can also jump farther away with small probability. In other words, the L^γ-graphs make emerge the so-called Lévy flight random walk [114]. To the best of our understanding, this is the only analytic result developed for the topologies emanating from fractional Laplacian matrices, thus it remains to be theoretically proven that such result extends to arbitrary networks. Nonetheless, extensive numerical evidence suggest that this same behavior extends to arbitrary networks [113]. In Figure 3.1, we illustrate how the Lévy flights arises on a K-NN graph build from the swiss-roll dataset (Figure 3.1a), the lollipop graph (Figure 3.1b), and the planted partition (Figure 3.1c).

Clearly, the new dynamics of these more volatile random walkers entail an improved capacity to explore the graph. Particularly, their ability to jump far away can allow to more



(a) Positive edges (b) Negative edges

Figure 3.2: Exemplification of the topology emerging from L^2 on a realization of the Planted Partition with 100 nodes and parameters $p_{in} = 0.5$ and $p_{out} = 0.05$. The positive edges coincide with the original structure but are reweighted. The negative ones appear between nodes that are initially at a 2-hop distance. It can be seen that a considerable amount of negative edges appear between clusters, bringing the potential to boost their detection.

frequently reach nodes that, otherwise, may be rarely visited by normal walkers. This can be the case in which a graph presents a trapping region, like a strong hub. Thus, since the learning process of PageRank mainly depends on walkers visiting enough the target set of nodes, extending PageRank to incorporate such long-range transitions bear the potential to enhance the classification outcome.

3.2.2 Regime of $\gamma > 1$

In this regime, the topologies emanating from the L^γ operator are signed graphs. In this case, the edges of the original structure are reweighted (still, remanining positive) and the new edges appearing between far distant nodes can be positive or negative. We note that, for $\gamma \in \mathbb{Z}$, such edge proliferation process can be related to paths of different lengths. To have a better grasp on this, let us take the topology from $\gamma = 2$ as an example: for $L^2 = (D-W)^2 = D^2+W^2-(DW+WD)$, the elements of the emanating graph are given as $[D_2]_{uu} = [L^2]_{uu} = D_{uu}^2 + \sum_v W_{uv}^2$ and $[W_2]_{uv} = -[L^2]_{uv} = (D_{uu}+D_{vv})W_{uv} - \sum_{l\neq u,v} W_{ul}W_{lv}$, showing that, in W_2, nodes originally connected get their link reweighted (still, remaining positive) while those at a 2-hop distance become linked by a negatively weighted edge.

We stress that this change in the topology has the potential to improve classification, as the emergence of positive and negative edges opens the door for an interpretation in terms of an agreement (positive edge) or a disagreement (negative edge) between datapoints. Hence, clustering can be revamped to assume that nodes agreeing should belong to the same cluster and nodes disagreeing should belong to different ones. From this perspective, a revisit to the case of $\gamma = 2$ shows that this is indeed a potentially good topology since, for numerous graphs, it is more likely that vertices having a 2-hop distance lie in different clusters than in the same one, thus creating a considerable amount of disagreements between clusters, that may enhance their separability. This idea is illustrated in Figure 3.2, where for a realization of the planted partition model we show that with $\gamma = 2$ a big amount of negative edges appear between clusters.

3.3 The L^γ-PageRank method

In this section, we introduce our main contribution: the L^γ-PageRank G-SSL, a generalization of PageRank that operates on the L^γ-graphs.

Departing from the optimization problem in Eq. (2.9) (see page 27), we revamp PageRank to operate on the L^γ-topology as follows:

Definition 19. *The L^γ-PageRank G-SSL is defined as the solution to the optimization problem:*

$$\arg\min_f \left\{ f^T D_\gamma^{-1} L^\gamma D_\gamma^{-1} f + \mu (f-y)^T D_\gamma^{-1} (f-y) \right\}. \tag{3.5}$$

Since this proposition is designed to operate on both unsigned and signed graphs, we must take care that it is a well-behaved object. In the following Lemma, we show that, for all $\gamma > 0$, problem (3.5) is convex and its solution can be found in closed form.

Lemma 6. *Let $\gamma > 0$. Then, problem (3.5) is convex with closed form solution given as:*

$$f = \mu \left(L^\gamma D_\gamma^{-1} + \mu \mathbb{I} \right)^{-1} y. \tag{3.6}$$

The proof of this Lemma is deferred to Appendix 3.A.

Now that we have shown that, for all γ, our formulation is a well-posed problem, we proceed to study its theoretical properties and clustering capabilities.

3.4 Analysis of $\gamma < 1$: Lévy flights for classification

3.4.1 Lévy flight driven PageRank

Let us note that our L^γ-PageRank solution in Eq. (3.6) corresponds to the regular PageRank expression but in which the initial graph has been replaced by one of the L^γ-topologies. Thus, since we showed in Section 3.2 that these topologies, for $\gamma \leq 1$, are positively weighted undirected graphs, then it is clear that analysis given for regular PageRank in Chapter 2 directly applies to our setting. As a result, by using the change of variable $\alpha = 1/(1+\mu)$, we can recast our L^γ-PageRank solution as:

$$f^T = (1-\alpha) \sum_{k=0}^{\infty} \alpha^k y^T P_\gamma^k. \tag{3.7}$$

Hence, our contribution is that Eq. (3.7) corresponds to a generalization of the PageRank algorithm to the Lévy process encoded by P_γ. This leads our L^γ-PageRank formulation to admit the following interpretation: the L^γ-PageRank vector at node u is proportional to the expected number of visits made by Lévy random walkers to node u, when, at each step, they decide to either continue the walk with probability α, or revisit the starting point with probability $(1-\alpha)$.

Clearly, having the ability to tune the dynamics of the walkers through the γ parameter can be useful to enhance, in some cases, our learned functions f.

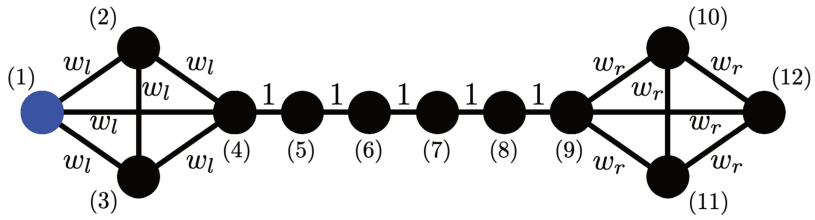


Figure 3.3: Lollipop graph.

3.4.2 Numerical experiments

Correct partitioning of a skewed graph

Experimental setup and goals. Our goal is to show that the enhanced exploration capabilities of the walkers can help to improve the performance of PageRank. Towards this goal, let us consider the lollipop graph displayed in Figure 3.3. The graph consists of a line graph with two cliques at the tails. We can tune the strength of links in each clique through the parameters w_l (left clique) and w_r (right clique). By priming symmetry, we assume that the best partition should cut the graph between nodes (6) and (7). Indeed, such partition implies a very good cut as there is only a single edge to remove. To solve the task, we dispose of one labelled point illustrated by the blue node. The challenge of the task is that, when the topology is skewed (i.e. $w_l \gg w_r$), then the random walkers can easily get trapped in the hub created by the clique, harming the significance of the PageRank vector.

Results and discussion. Let us start illustrating that, in a balanced setting, the standard PageRank algorithm always finds the right partition and that, in an unbalanced setting, it always fails. In Figure 3.4, we show the classification assignment given to each node by the standard PageRank algorithm ($\gamma = 1$) in a balanced setting $w_r = w_l = 1$. Please recall that PageRank possesses one degree of freedom: the $\alpha = 1/(1+\mu)$ parameter. Thus, it can be seen that, irrespective of α, PageRank always retrieves the right partition. Now, let us skew the balance of the graph by setting one of the cliques with larger strength than the other: $w_l = 10$ and $w_r = 1$. Notably, by doing this change it is now impossible for PageRank to solve the task. Precisely, Figure 3.5 displays the new classification assignment given to each node by the standard PageRank algorithm. As it can be seen, there is no longer a value of α that allows PageRank to assign nodes (5) and (6) to the correct class. Clearly, this behaviour is due to the hub created by the first four nodes that hinders the walkers capacity to explore the graph and visit nodes (5) and (6) often enough. This is, when the walker, which starts at the labelled point, is at node (4) and with probability α decides continue the walk, it will only choose to transition to node (5) with probability 1/31. Moreover, even if the walker manages to escape the hub, this latter will be revisited at the next step with probability $(1-\alpha)$. Thus, highlighting how regions that trap random walkers can pose serious challenges for PageRank.

To enhance classification performance, we now revert to our L^γ-PageRank formulation. Let us recall that our formulation is driven by a transition matrix that follows $[P_\gamma]_{uv} \propto \Delta_{uv}^{-\gamma}$. Thus, we embed the long-range transitions by setting $\gamma = 0.1$ and we display the classification retrieved by our L^γ-PageRank algorithm in Figure 3.6. As it can be seen, the added

degree of freedom allows to now, for all α, get node (5) assigned to the correct class. However, the walkers are still not diffusive enough to get node (6) correctly classified. Thus, we increase the diffusion capacity of the walkers even more by further reducing $\gamma = 0.01$. As it can be seen in Figure 3.7, we can now obtain, irrespective of α, the desired partition. This example illustrates how the added degree of freedom allows us to override an skewed graph setting which standard PageRank is not able to cope with.

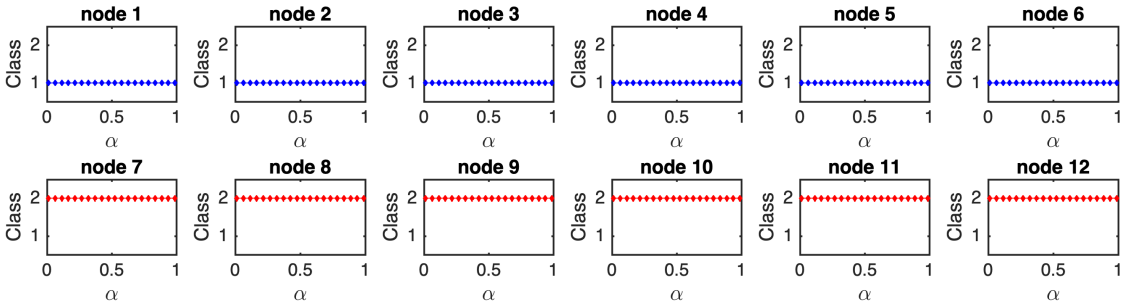


Figure 3.4: Classification with $\gamma = 1$ in balanced setting ($w_l = w_r = 1$). For all α, standard PageRank correctly partitions the graph between nodes (6) and (7).

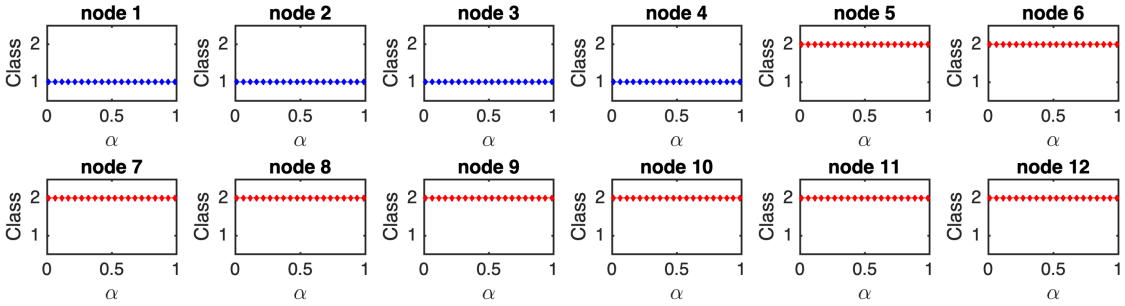


Figure 3.5: Classification with $\gamma = 1$ in a skewed setting ($w_l = 10$, $w_r = 1$). Due to walkers getting trapped in the hub, nodes (5) and (6) are now always incorrectly assigned.

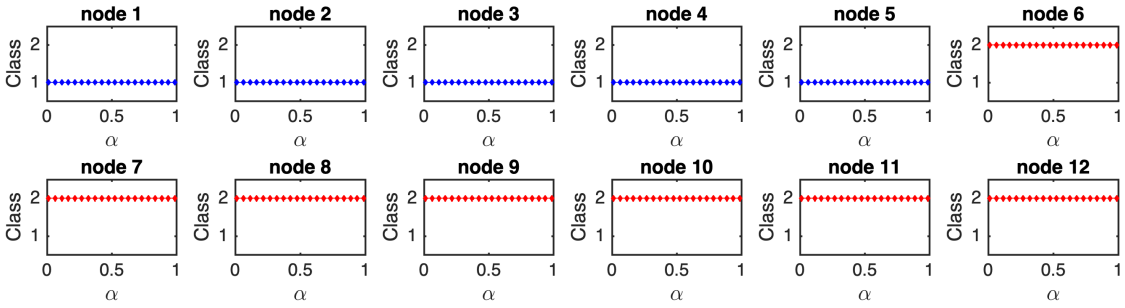


Figure 3.6: Classification with $\gamma = 0.1$ in a skewed setting ($w_l = 10$, $w_r = 1$). The Lévy walkers escape the hub more often, correctly assigning node (5). Yet, node (6) is still not enough visited, getting incorrectly assigned.

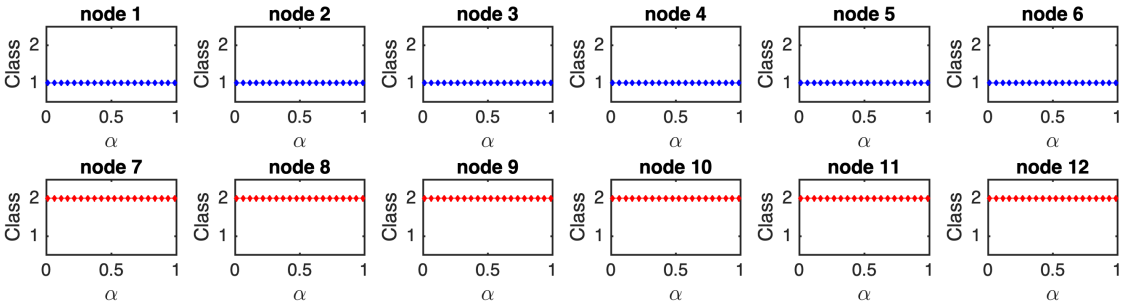


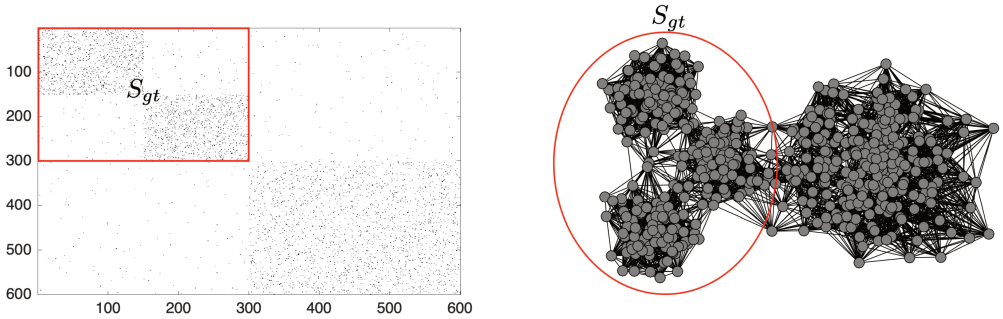
Figure 3.7: Classification with $\gamma = 0.01$ in a skewed setting ($w_l = 10$, $w_r = 1$). By properly tuning γ, now all the nodes get correctly classified.

Enhanced detection of graphs with sub-community structures

Experimental setup and goals. In this experiment, our goal is to illustrate that our L^γ-PageRank can also help to more easily indentify classes with sub-cluster structures.

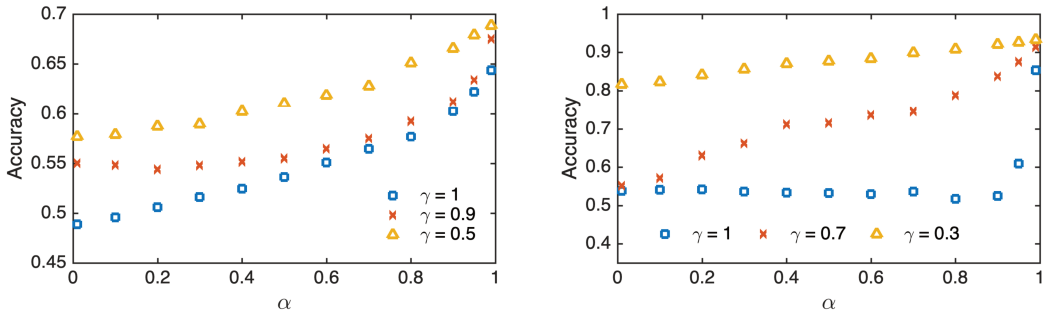
Our experimental setup is as follows. We consider two graphs in which the target set S_{gt} has internal sub clusters (see Figure 3.8). On the one hand, we generate data from a SBM with three clusters ($\mathcal{V} = S_1 \cup S_2 \cup S_3$) and define $S_{gt} = S_1 \cup S_2$, so that the partition to find is $\mathcal{V} = S_{gt} \cup S_3$. We set $|S_1| = |S_2| = 0.5 \times |S_3| = 150$. The mean degrees are set as $C_{33} = 7.6$, $C_{23} = C_{13} = 0.4$, $C_{11} = C_{22} = 7.3$, and $C_{12} = C_{21} = 0.3$ (leading to $p_{12} = p_{21} = 0.002$ which is bigger than $p_{13} = p_{23} = 0.0013$). This way, the mean degree of nodes in S_{gt} are equal to the mean degrees of nodes in S_3. Moreover, we grant that, despite various clusters arising, $\mathcal{V} = S_{gt} \cup S_3$ is the optimal partition (the one with smallest Cheeger ratio). An adjacency matrix drawn from this model is shown in Figure 3.8a. On the other hand, we consider a K-NN graph drawn from a mixture of gaussians. We first generate 300 data points from a mixture of three Gaussians in $\mathbb{R}^2$, where $\mu_1 = [0, 0]$, $\mu_2 = [-2, 2]$, $\mu_3 = [-2, -2]$ and $\sigma_1 = \sigma_2 = \sigma_3 = 0.3$. Secondly, we generate a single more spreaded gaussian of 300 points with $\mu_4 = [4, 0]$ and $\sigma_4 = [1.5]$. Then, we build a graph from this data using $K = 30$ and $\sigma = 2$. Due to the closeness in space and the balance of the cut, we consider the three smaller gaussians as S_{gt}. The resulting graph from one realization of data is displayed in Figure 3.8b. Then, to classify the data, one labelled point is sampled at random from S_{gt} and the L^γ-PageRank method (via sweep) is applied for different values of the α parameter lying on a discrete grid covering the range $\alpha \in [0.01, 0.99]$. The procedure is repeated for 50 data realizations on both datasets. Performance is assessed in terms of the Matthews Correlation Coefficient (MCC), which we recall that a value of 1 implies perfect agreement with the true partition and 0 a random decision.

Results and discussion. Figure 3.9 displays the performance of L^γ-PageRank on the classification of the two graphs from Figure 3.8. Observe how, in both cases, the standard PageRank algorithm is not able to accurately recover S_{gt} despite it having a small Cheeger ratio. This is because, in both graphs, our diffusion starts inside another set of small Cheeger ratio, thus, as implied by the results from Chapter 2, it is hard for the diffusion process to escape such cluster and cover all S_{gt}. Now, let us note how the introduction of γ helps to improve the detection of S_{gt}. Clearly, this is because the long-jump



(a) Graph 1: Planted Partition with sub-cluster (b) Graph 2: Mixture of Gaussians

Figure 3.8: Graphs with sub-cluster structures.



(a) Classification of graph 1 (b) Classification of graph 2

Figure 3.9: Improved detection of graphs with sub-cluster structures.

nature of the Lévy walkers can override, to some extend, the cluster in which the diffusion process starts. The impact of the Lévy walkers is particularly prominent on the mixture of gaussians graph, where our L^γ-PageRank can significantly enhance performance. The planted partition graph also benefits from our formulation, albeit the impact is less drastic. This is natural, as the planted partition is already a small world network, meaning that every pair of nodes is already at a short distance. Thus, in it, there is not too much room to jump 'far away'. Indeed, in it, one needs to take more care on the tuning of γ, as for not very long jumps one can already fall in S_{gt}^c which, of course, is undesired.

3.5 Analysis of $\gamma > 1$: Signed graphs for classification

3.5.1 Clustering with L^γ-PageRank

In this regime, the L^γ-graphs give rise to signed graphs: i.e. graphs with positive and negative edges. Thus, in this regime, we cannot longer interpret the closed form solution of our algorithm (see Lemma 6) as the equilibrium state of a random walk diffusion process. Clearly, this is because P_γ now becomes a matrix in which 'negative transitions' appear, meaning that it is no longer stochastic and, consequently, it no longer codes for a random walk process. In this section, we show that, while not necessarily modelled by random walkers, our L^γ-PageRank algorithm, remains a well-behaved diffusion process that can

be used to find clusters in the L^γ-graphs.

For our analysis, it is useful to first extend some of the graph topological definitions to the L^γ-graphs. Namely, let $vol_\gamma(S) = \sum_{u\in S} [D_\gamma]_{uu}$ denote the generalized volume of S. Let π_γ denote a generalized stationary distribution with entries given by $(\pi_\gamma)_u = [D_\gamma]_{uu}/vol_\gamma(G)$. It is important to stress that $[D_\gamma]_{uu} = \sum_k \lambda_k^\gamma Q_{uk}^2 \geq 0$. Thus, for all $\gamma > 0$, the generalized volume and the generalized stationary distribution are non-negative quantities.

Another important topological definition introduced in Chapter 2 was the Cheeger ratio metric. Clearly, it cannot longer be employed to assess the presence of clusters in the L^γ-graphs as it lacks the ability to account for the sign of edges. Thus, we generalize the Cheeger ratio definition to the new graphs as follows.

Definition 20. *For a set of nodes $S \subseteq V$, the generalized Cheeger ratio, or generalized conductance, of S is defined as:*

$$h_S^{(\gamma)} = \frac{\sum_{u\in S}\sum_{v\in S^c}[W_\gamma]_{uv}}{\min\left(vol_\gamma(S), vol_\gamma(S^c)\right)}. \tag{3.8}$$

Please note that this generalization of the Cheeger ratio is mathematically sound: (i) first, it is a non-negative quantity since $\sum_{u\in S}\sum_{v\in S^c}[W_\gamma]_{uv} = \mathbb{1}_S^T L^\gamma \mathbb{1}_S \geq 0$; and (ii) the set S attaining the minimum value coincides with a sensible clustering. To show the latter, let us split the edges in W_γ according to their sign as $W_\gamma = W_\gamma{}^+ + W_\gamma{}^-$. Now, let us consider the following definitions:

- Let $\mathcal{A}_{in}^{(\gamma)}(S) = \sum_{u\in S}\sum_{w\in S}|\left[W_\gamma^+\right]_{uw}|$ be the sum of agreements within S
- Let $\mathcal{A}_{out}^{(\gamma)}(S) = \sum_{u\in S}\sum_{v\in S^c}|\left[W_\gamma^+\right]_{uv}|$ be the agreements between S and S^c
- Let $\mathcal{D}_{in}^{(\gamma)}(S) = \sum_{u\in S}\sum_{w\in S}|\left[W_\gamma^-\right]_{uw}|$ be the disagreements within S
- Let $\mathcal{D}_{out}^{(\gamma)}(S) = \sum_{u\in S}\sum_{v\in S^c}|\left[W_\gamma^-\right]_{uv}|$ the disagreements between S and S^c

Then, we state the following lemma.

Lemma 7. *Let $S^* = \arg\min_S h_S^{(\gamma)} \quad \forall S$ s.t. $vol_\gamma(S) \leq vol_\gamma(G)/2$. Then, S^* is the set that attains the best balance of:*

- *Maximal $\mathcal{D}_{out}^{(\gamma)}(S^*)$ and $\mathcal{A}_{in}^{(\gamma)}(S^*)$;*
- *Minimal $\mathcal{D}_{in}^{(\gamma)}(S^*)$ and $\mathcal{A}_{out}^{(\gamma)}(S^*)$.*

The proof of this Lemma is deferred to Appendix 3.B.

Lemma 7 shows that, for a given L^γ-graph, sets of small generalized Cheeger ratio are bound to have strong between-cluster disagreements and strong within-cluster agreements as well as small between-cluster agreements and small within-cluster disagreements, thus coinciding with our definition of clusters in signed graphs. In other words, to find clusters in the L^γ-graphs, we should look for sets of small generalized Cheeger ratio.

Now, we show that, while not necessarily modelled by random walkers, our L^γ-PageRank remains a well-behaved diffusion process having π_γ as stationary state and diffusion rate controlled by the μ parameter.

Lemma 8. *Let $\gamma > 0$. The L^γ-PageRank solution in Eq. (3.6) satisfies the following properties:*

1. **mass preservation***: $\sum_{u\in\mathcal{V}} f_u = \sum_{u\in\mathcal{V}} y_u$*
2. **stationarity***: $f = \pi_\gamma$ if $y = \pi_\gamma$*
3. **limit behavior***: $f \to \pi_\gamma$ as $\mu \to 0$ and $f \to y$ as $\mu \to \infty$*

The proof of this Lemma is deferred to Appendix 3.C.

Our next results shows that it is hard for such diffusion process to escape clusters in the L^γ-graphs.

Lemma 9. *Let $\gamma > 0$ and let $S \subset \mathcal{V}$ be an arbitrary set with $vol_\gamma(S) \leq vol_\gamma(G)/2$. For a labeled point placed at node $u \in S$ with probability proportional to its generalized degree in S, i.e. $\frac{[D_\gamma]_{uu}}{vol_\gamma(S)}$, L^γ-PageRank satisfies:*

$$\mathbb{E}\left[f(S^c)\right] \leq \frac{h_S^{(\gamma)}}{\mu}. \tag{3.9}$$

The proof of this Lemma is deferred to Appendix 3.D

Lemma 9 admits a similar interpretation as Lemma 3 (see page 39). Namely, if L^γ-PageRank is applied to the labeled points of some set S with small $h_S^{(\gamma)}$, then diffusion is confined to S and the score values outside of S are expected to be small. Thus, by looking at the nodes with largest score values we should be able to retrieve a good estimation of S.

Now, let us note that if such score concentration phenomenon takes place, then a sharp drop must appear after sorting the L^γ-PageRank scores in descending order. We will use the following lemma to show that if a sharp drop is present, then the sweep cut procedure applied on the L^γ-PageRank vector retrieves a partition $\hat{S}$ that has small $h_{\hat{S}}^{(\gamma)}$.

Lemma 10. *Let q denote the permutation vector and S_j denote the set associated to q_j obtained by applying the sweep-cut procedure on the L^γ-PageRank vector. Then, the partition $\mathcal{V} = S_j \cup S_j^c$ satisfies the inequality:*

$$\begin{aligned}\mathcal{A}_{out}^{(\gamma)}(S_j)\left(2 - \frac{(q_j - q_{j+1})}{(q_1 - q_N)}\right) - \mathcal{D}_{out}^{(\gamma)}(S_j)\left(2\frac{(q_j - q_{j+1})}{(q_1 - q_N)} - 1\right) \geq \frac{\mu\left(y(S_j) - f(S_j)\right)}{(q_1 - q_n)} \\ \geq \mathcal{A}_{out}^{(\gamma)}(S_j)\left(2\frac{(q_j - q_{j+1})}{(q_1 - q_N)} - 1\right) - \mathcal{D}_{out}^{(\gamma)}(S_j)\left(2 - \frac{(q_j - q_{j+1})}{(q_1 - q_N)}\right).\end{aligned} \tag{3.10}$$

The proof of this Lemma is deferred to Appendix 3.E.

To show that this Lemma implies a good partition if a sharp drop is present, let us depart noticing that

$$\sum_{u\in S_j}\sum_{v\in S_j^c} [W_\gamma]_{uv} = \mathcal{A}_{out}^{(\gamma)}(S_j) - \mathcal{D}_{out}^{(\gamma)}(S_j) \geq 0. \tag{3.11}$$

Thus, the generalized Cheeger ratio of S_j is small if $\mathcal{A}_{out}^{(\gamma)}(S_j)$ is not much larger than $\mathcal{D}_{out}^{(\gamma)}(S_j)$. In the inequality (3.10), we have two cases in which $(q_j - q_{j+1})/(q_1 - q_N) \approx 1$: (i) q is approximately constant; and (ii) q has a drop that satisfies $q_j \approx q_1$ and $q_{j+1} \approx q_N$. The former can only occur if $f \to \pi_\gamma$ and clearly no cluster can be retrieved from that vector, as confirmed by the inequality growing unbounded. The latter case is what we coin as having a sharp drop between q_j and q_{j+1}. In such case, the inequality is controlled by the difference $y(S_j) - f(S_j)$ which, due to the mass preserving property and the assumption that $q_{j+1} \approx q_N$, should be small. Thus, granting that $\mathcal{A}_{out}^{(\gamma)}(S_j)$ is not much larger than $\mathcal{D}_{out}^{(\gamma)}(S_j)$ and S_j has a small $h_{S_j}^{(\gamma)}$.

Discussion. The previous results show that L^γ-PageRank is a sensible tool to find clusters in the L^γ-graphs, i.e. groups of nodes with small generalized Cheeger ratio. Thus, revisiting the classification case in which we target group of nodes S_{gt}, we have that the smaller the value of $h_{S_{gt}}^{(\gamma)}$, the better the L^γ-PageRank method can recover it. This observation, in addition to noting that standard PageRank emerges as the particular case of $\gamma = 1$, indicate that we should be able to enhance the performance of G-SSL in the detection of S_{gt} by finding the graph, i.e. the γ value, in which $h_{S_{gt}}^{(\gamma)} < h_{S_{gt}}^{(1)}$.

3.5.2 The selection of γ

Case of $\gamma = 2$: analytic study

In Section 3.2.2, it was argued that the topology emerging from L^2 places a negatively weighted link between nodes at a 2-hop distance, thus carrying the potential to place a big amount of disagreements between clusters that may enhance their separability. Our next result formalizes this claim, demonstrating that, on assortative graphs from the Planted Partition model, it is expected that the L^2-graph improves the generalized Cheeger ratio.

Theorem 5. *Consider a Planted Partition model of parameters (p_{in}, p_{out}) and cluster sizes $|S_{gt}| = |S_{gt}^c| = n$. Then, as $n \to \infty$ we have that*

$$\mathbb{E}\left[h_{S_{gt}}^{(2)}\right] = 2\mathbb{E}\left[h_{S_{gt}}^{(1)}\right]^2, \tag{3.12}$$

where $\mathbb{E}\left[h_{S_{gt}}^{(1)}\right] = p_{out}/(p_{in} + p_{out})$.

The proof of this Theorem is deferred to Appendix 3.F.

Corollary 1. *If $p_{in} \geq p_{out}$, then $\mathbb{E}\left[h_{S_{gt}}^{(2)}\right] \leq \mathbb{E}\left[h_{S_{gt}}^{(1)}\right]$, with equality occurring in the case $p_{in} = p_{out}$.*

The proof of this Corollary is deferred to the Appendix 3.G

Theorem 5 and Corollary 1 open the door to investigate, on arbitrary graphs, in which cases the L^2-graph improves the generalized Cheeger ratio of a set. In the next Proposition, we provide a sufficient condition in which the L^2-graph improves the generalized Cheeger ratio a set.

Proposition 2. *Let $\langle D_{S_{gt}} \rangle$ denote the mean degree of S_{gt}. A sufficient condition on S_{gt} so that $h^{(2)}_{S_{gt}} \leq h^{(1)}_{S_{gt}}$ is that*

$$\langle D_{S_{gt}} \rangle \geq \max_{u \in S_{gt}} \sum_{v \in S^c_{gt}} W_{uv} + \max_{w \in S^c_{gt}} \sum_{\ell \in S_{gt}} W_{w\ell}. \tag{3.13}$$

The proof of this Proposition is deferred to Appendix 3.H

Proposition 2 points in the same direction as Theorem 5, saying that graphs having a cluster structure are bound to benefit from L^2. Concretely, the first term on the right hand side of the inequality searches, among all the nodes of S_{gt}, the one that has the maximum number of connections towards S^c_{gt}. The second term does the reverse for the nodes of S^c_{gt}. Hence, asking for the nodes of S_{gt} to have, on average, more connections than the maximum possible boundary implies that S_{gt} should have a cluster structure.

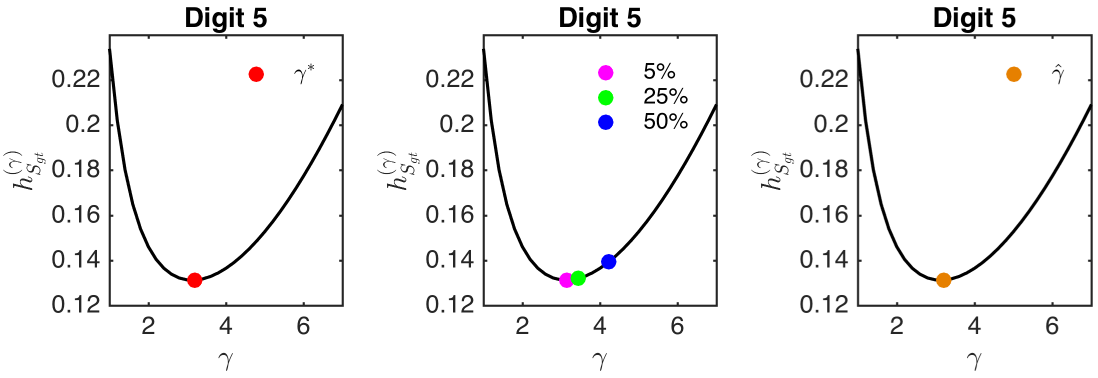
An algorithm for the optimal selection of γ

Numerical experiments show that increasing γ can further decrease the generalized Cheeger ratio up to a point where it starts increasing. We show an example of this phenomenon in Figure 3.10a, displaying the evolution of $h^{(\gamma)}_{S_{gt}}$ as a function of γ when S_{gt} corresponds to a digit of the MNIST dataset. From Figure 3.10a, it is evident that an optimal value appears:

$$\gamma^* = \arg\min_{\gamma} h^{(\gamma)}_{S_{gt}}. \tag{3.14}$$

Hence, this raises the question of how to find such value. Clearly, since the behavior of $h^{(\gamma)}_{S_{gt}}$ depends on S_{gt}, in practice, taking the derivative or performing a greedy search to find γ^* cannot be employed since S_{gt} is unknown. A second question that arises is whether such optimal value changes drastically or smoothly with changes in S_{gt}. To address it, we perform the following test: for a given S_{gt} (same MNIST digit), we remove some percentage of the nodes in S_{gt} and record the optimal value on subsets of S_{gt}. More precisely, recall that $h^{(\gamma)}_{S_{gt}} = \mathbb{1}^T_{S_{gt}} L^\gamma \mathbb{1}_{S_{gt}} / \mathbb{1}^T_{S_{gt}} D_\gamma \mathbb{1}_{S_{gt}}$, hence we randomly select some percentage of the entries indexing S_{gt} in $\mathbb{1}_{S_{gt}}$, set them to zero and obtain a new indicator function indexing a subset of S_{gt}. Mean results are evaluated in the original curve and displayed in Figure 3.10b. As it can be seen, the figure suggest that it is not necessary to know S_{gt} to find a proxy $\hat{\gamma}$ of γ^*, it suffices to know a subset of S_{gt}.

Based on the insights from our last experiment, we propose Algorithm 2 for the estimation of γ^*. The rationale of the algorithm is to exploit the labeled points and the initial graph to find a proxy $\hat{S}$ of S_{gt} on which we can compute the estimate. The procedure consists in letting walkers started from the label points, run for a number of steps that is determined by the maximum geodesic distance between the labels. This allows walkers to explore S_{gt} without having the chance to go far from it. Then, after running the walk, we list the nodes in descending order according to the probability of finding a walk at a node. We take the first element on the list (the one where it is more likely to find a walker), add it to $\hat{S}$ and remove it from the list, so that the former second element becomes the first in the listing. We repeat the procedure until the probability of finding a walker in the nodes conforming $\hat{S}$ is 0.7.



(a) $\gamma^* = \arg\min_\gamma h^{(\gamma)}_{S_{gt}}$ (b) γ^* on subsets of S_{gt} (c) $\hat{\gamma}$ by algorithm

Figure 3.10: Generalized Cheeger ratio of S_{gt} as a function of γ. Percentages in (b) represent a fraction of nodes randomly chosen from S_{gt} and removed from S_{gt}. For the plot, S_{gt} is a digit of the MNIST dataset

Algorithm 2 Estimation of γ^*

Input: $\mathcal{G}, \mathcal{V}_{S_{gt}}$ and a grid of γ values.
Output: $\hat{\gamma}$
Compute $\Delta_{uv} \ \forall \ u, v \in \mathcal{V}_{S_{gt}}$.
Set $k = \max_{u,v} \{\Delta_{uv}\}$
Set $\chi = y/\|y\|_1$
Run a k-step walk with seed χ: $x^T = \chi^T P^k$
Reorder the vertices as $v_1, \cdots, v_N$, so that $x_{v_i} \geq x_{v_{i+1}}$
for $i = 1 : N$ **do**
  if $\sum_{j=1}^{i} x_{v_j} < 0.7$ **then**
    Set $(\mathbb{1}_{\hat{S}})_{v_i} = 1$
  else
    Set $(\mathbb{1}_{\hat{S}})_{v_i} = 0$
  end if
end for
Compute $h^{(\gamma)}_{\hat{S}} = \frac{\mathbb{1}^T_{\hat{S}} L^\gamma \mathbb{1}_{\hat{S}}}{\mathbb{1}^T_{\hat{S}} D_\gamma \mathbb{1}_{\hat{S}}} \ \forall \ \gamma$
Return $\hat{\gamma} = \arg\min_\gamma h^{(\gamma)}_{\hat{S}}$.

In Table 3.1, we evaluate the performance of Algorithm 2 on the estimation of γ^* for all the digits of the MNIST. The graph construction guidelines are given in Section 3.5.3 (Real world datasets). For the test, 500 realizations of labeled points and a grid of γ ranging from 1 to 7 with a resolution of 0.2 were used. The first row displays, as γ^*, the value of γ (from the input range) attaining the minimum generalized Cheeger ratio. The second row displays the performance of the algorithm when estimating such value. The last three rows show the value of the generalized Cheeger ratio evaluated at γ^*, $\hat{\gamma}$ and $\gamma = 1$, respectively. As it can be seen, our estimator finds values of $\hat{\gamma}$ whose Cheeger ratios are: (i) significantly smaller than those of $\gamma = 1$; and (ii) close to the optimal.

Digit	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
γ^*	7.0	3.0	7.0	3.2	3.2	7.0	7.0	3.2	4.2
$\hat{\gamma}$	5.45 (0.15)	3.10 (0.14)	6.41 (0.11)	4.92 (0.16)	3.20 (0.14)	6.04 (0.15)	4.98 (0.17)	4.40 (0.18)	5.08 (0.15)
$h_{S_{gt}}^{(\gamma^*)}$	0.065	0.166	0.035	0.141	0.131	0.011	0.052	0.116	0.135
$h_{S_{gt}}^{(\hat{\gamma})}$	0.073 (9e-4)	0.174 (8e-4)	0.041 (1e-3)	0.185 (4e-3)	0.148 (2e-3)	0.017 (1e-3)	0.074 (2e-3)	0.142 (2e-3)	0.149 (9e-4)
$h_{S_{gt}}^{(1)}$	0.175	0.248	0.216	0.258	0.233	0.107	0.203	0.215	0.285

Table 3.1: Evaluation of Algorithm 2 on the MNIST Dataset: each cell reports MCC, 95% confidenceinterval (parenthesis)

3.5.3 Numerical experiments

Planted Partition

Experimental setup and goals. In the following experiment, we show that L^γ-PageRank can increase the performance of G-SSL as the graph approaches the Planted Partition detectability transition. In Chapter 1, we covered results indicating that the Planted Partition possesses a detectability threshold above which unsupervised methods are unable to retrieve a meaningful clustering (see Theorem 1 in page 11). As for G-SSL, the work in [115] shows that such threshold can be overcome when a fraction of labeled points is introduced to the task. Nonetheless, the performance of G-SSL drastically degrades when approaching the detectability transition.

The experimental setup is the following: for a given C_{out}/C_{in}, a realization of the Planted Partition is drawn with $n = 500$ and $C_{avg} = 3$. Then, 1% of labeled points are sampled at random and the L^γ-PageRank method is applied for different values of μ lying on a discrete grid. The clusters are determined via a sweep-cut procedure, and the best performance is retained. The whole procedure is repeated for 10 different realizations of the labeled points. Finally, all the preceding steps are repeated for 100 graph realizations. Performance is assessed in terms of the Matthews Correlation Coefficient (MCC).

Results and discussion. Figure 3.11 displays the performance of L^γ-PageRank at recovering the Planted Partition as a function of the ratio C_{out}/C_{in}. As it can be seen, standard PageRank ($\gamma = 1$) performs poorly when the configuration approaches the phase transition (referred by the vertical line) since $h_S^{(1)}$ becomes large. Then, we observe that the introduction of γ allows to decrease $h_{S_{gt}}^{(\gamma)}$, which, accordingly, enhances the clustering performance. Furthermore, the figure verifies that the smaller the value of $h_{S_{gt}}^{(\gamma)}$ (right plot), the better the L^γ-PageRank recovers the true partition (left plot). It is important to remark that, for this experiment, while $\gamma = 2$ shows good improvements, larger values of γ keep improving $h_S^{(\gamma)}$, until it reaches a saturation plateau, designating a region of optimal γ values ($\gamma \geq 6$).

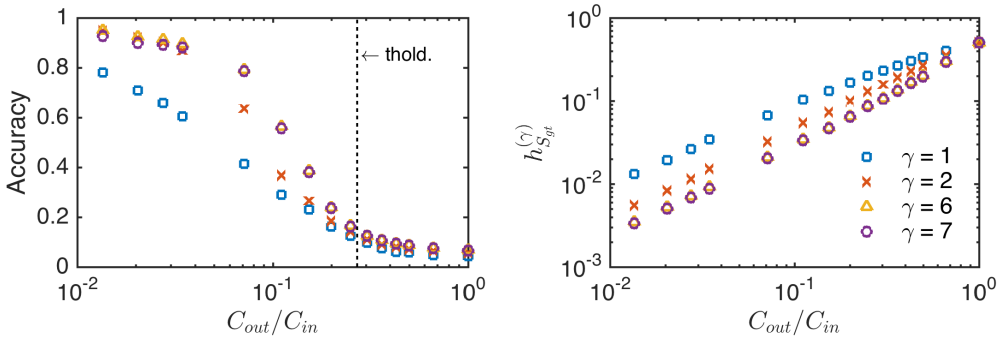


Figure 3.11: Improved detection of the Planted Partition.

Real world datasets

Experimental setup and goals. In our following experiment, we assesses the performance of L^γ-PageRank and Algorithm 2 on real world datasets.

The experimental setup is as follows: graphs are built connecting the K-Nearest Neighbors (KNN) with distances computed via the Gaussian kernel, so that the weight between points $\mathbf{x}_u$ and $\mathbf{x}_v$ is given by $W_{uv} = \exp\{-\|\mathbf{x}_u - \mathbf{x}_v\|_2^2/\sigma^2\}$. For each class, 2% of labeled points are randomly selected, L^γ-PageRank is applied for a grid of μ values, partitions are retrieved via the sweep-cut, and the best performance, assessed in terms of MCC, is retained. Such procedure is repeated for 100 realization of labeled points, except for the MNIST on which 30 realizations only are employed. In all cases, classes are balanced in size and the graph construction parameters are selected to provide a good distribution of weights as follows: (a) MNIST [116]: Images of handwritten digits (1 to 9). From the entire dataset, 200 images of each digit are selected and used to build the graph with KNN = 10 and $\sigma = 10^4$; (b) Gender Images [117]: Images of male and female subjects for gender recognition. From the entire dataset, 200 images of each gender are selected and used to build the graph with KNN = 60 and $\sigma = 10^4$. The large value of KNN is to avoid disconnected components; (c) BBC articles [118]: Word frequency attributes from news media articles. From the entire dataset, 200 business and 200 entertainment articles are used to build the graph with KNN = 5 and $\sigma = 50$; and (d) Phoneme [119]: Five attributes to discern nasal sounds from oral sounds. From the entire dataset, 200 oral and 200 nasal sounds are used to build the graph with KNN = 10 and $\sigma = 2$.

Results and discussion. Table 3.2 shows the performance of L^γ-PageRank on the classification of the real world datasets. Clearly, the introduction of γ can significantly improve performance and, in general, the estimation $\hat{\gamma}$ performs close to the optimal value γ^*. It can be seen that some datasets are more sensitive to γ than others. For instance, in the BBC articles we observe that a small change in γ, going from $\gamma = 1$ to $\gamma^* = 1.1$, increases performance, and going further to $\hat{\gamma} = 1.3$ and $\gamma = 2$ significantly worsens the classification. On the other hand, the MNIST dataset is less sensitive to γ, obtaining similar performances with larger variations in γ.

	S_{gt}	$\gamma = 1$	$\gamma = 2$	$\gamma = \hat{\gamma}$	$\gamma = \gamma^\star$
MNIST	Digit 1	0.67 (0.075)	0.78 (0.032)	0.78 (0.034) [5.4]	0.80 (0.027) [7.0]
	Digit 2	0.38 (0.042)	0.60 (0.064)	0.64 (0.059) [3.3]	0.64 (0.059) [3.0]
	Digit 3	0.47 (0.040)	0.61 (0.032)	0.61 (0.028) [6.0]	0.61 (0.028) [7.0]
	Digit 4	0.39 (0.022)	0.48 (0.036)	0.53 (0.044) [4.7]	0.53 (0.037) [3.2]
	Digit 5	0.44 (0.036)	0.56 (0.046)	0.61 (0.036) [3.3]	0.64 (0.035) [3.2]
	Digit 6	0.90 (0.039)	0.94 (0.003)	0.94 (0.002) [6.0]	0.94 (0.002) [7.0]
	Digit 7	0.43 (0.027)	0.66 (0.043)	0.71 (0.042) [4.8]	0.75 (0.032) [7.0]
	Digit 8	0.47 (0.062)	0.65 (0.057)	0.74 (0.038) [4.8]	0.72 (0.050) [3.2]
	Digit 9	0.43 (0.020)	0.52 (0.026)	0.53 (0.023) [4.9]	0.56 (0.026) [4.2]
Gender images	Female	0.51 (0.039)	0.57 (0.028)	0.57 (0.020) [3.0]	0.57 (0.028) [2.0]
	Male	0.55 (0.028)	0.61 (0.021)	0.60 (0.022) [3.3]	0.61 (0.021) [2.4]
BBC articles	Business	0.80 (0.020)	0.53 (0.038)	0.72 (0.040) [1.3]	0.81 (0.021) [1.1]
	Entmt.	0.84 (0.027)	0.57 (0.040)	0.76 (0.047) [1.5]	0.86 (0.025) [1.3]
Phoneme	Nasal	0.37 (0.030)	0.41 (0.028)	0.43 (0.025) [2.9]	0.43 (0.025) [3.0]
	Oral	0.41 (0.025)	0.44 (0.022)	0.46 (0.019) [2.8]	0.46 (0.019) [3.0]

Table 3.2: Performance on real world datasets: each cell reports MCC, 95% confidence interval (parenthesis) and the value of γ [squared brackets].

It is important to stress that, thus far, we have assumed possession of the proper tuning of the diffusion rate (μ) that attains the best results. However, when working with real data, clusters may have intricate local structures, e.g. sub-clusters, that play an important role in the way information diffuses, and that can make more difficult the finding of the optimal diffusion rate μ. As a result, two clusters may have equal Cheeger ratios but one of them being harder to find if its local structure is complex (this will be made more clear in our last experiment from this section). Digit 8 poses an example of this phenomenon, where the mean performance for $\hat{\gamma}$ is slightly better than that of $\gamma^\star$. This anomaly can be explained as an aftereffect of using a finite grid on μ: for some realization of labeled points, the best performance for $\gamma^\star$ falls in a region not covered by the grid.

Unbalanced labelled data

Experimental setup and goals. In Chapter 2, it was shown that an implication of Theorem 2 (see page 29) is that PageRank, in the multi-class setting (where multiple classification functions are compared instead of doing sweeps), suffers from biased outputs when operating on unbalanced labelled datasets. Thus, in our next experiment, we aim to show that L^γ-PageRank, adapted to the multi-class setting, can improve the performance of G-SSL in the presence of unbalanced labelled data.

The experimental setup is as follows: graphs with two balanced classes (in size) are built using the datasets from the preceding experiments. The parameters of the graphs' construction follow the guidelines detailed in our experiment above. For the Planted Partition, the configuration is $n = 200$, $C_{avg} = 3$, $C_{out} = 0.1$. Then, unbalanced labelled points are

	Planted Partition	MNIST 4vs9	MNIST 3vs8	BBC articles	Gender images	Phoneme
$\gamma = 1$	0.81 (1.1e-2)	0.51 (1.5e-2)	0.70 (1.4e-2)	0.66 (1.8e-2)	0.63 (2.1e-2)	0.44 (2.3e-2)
$\gamma = 2$	0.87 (8.7e-3)	0.56 (1.5e-2)	0.76 (1.2e-2)	0.92 (5.0e-3)	0.73 (1.6e-2)	0.48 (1.4e-2)
γ = Best	0.90 (7.0e-3) [6]	0.57 (1.5e-2) [3]	0.78 (1.2e-2) [4]	0.93 (1.5e-3) [3]	0.75 (1.7e-2) [3]	0.48 (1.4e-2) [1.9]

Table 3.3: Performance on unbalanced labelled data: each cell reports MCC, 95% confidence interval (parenthesis) and the value of γ [squared brackets].

drawn at random: 2% from one class and 6% from the other. Lastly, L^γ-PageRank, in the multi-class setting, is applied for a grid of μ values and the best performance, assessed by MCC, is recorded. For the planted partition, the procedure is repeated over 15 realizations of the labeled points and for 100 graph realizations. For the other datasets, 100 realizations of labeled points are employed.

Results and discussion. Table 3.3 displays the performance L^γ-PageRank in the presence of unbalanced labeled data. It is important to stress that, in this framework, a unique value of γ is used to retrieve all the clusters at the same time, precluding the notion of an optimal γ as defined in Section 3.5.2. However, one value of γ seems to perform better, we denote it as γ = Best. As it can be seen, the introduction of γ helps to significantly improve the classification performance in the presence of the unbalanced labeled data.

The generalized Cheeger ratio: necessary but not sufficient

Experimental setup and goals. Our theoretical results indicate that, for a target S_{gt}, the smaller the value of $h^{(\gamma)}_{S_{gt}}$, then the easier it is for our L^γ-PageRank algorithm to detect S_{gt}. In our next experiment, we show that having a small $h^{(\gamma)}_{S_{gt}}$ is a necessary condition but not sufficient. This is because the Cheeger ratio only counts a ratio of external and internal connections of a group of nodes, but it does not consider how such connections are distributed. Thus, while a set S_{gt} may have a small Cheeger ratio, the distribution of its connections may not be favorable for information to properly diffuse.

The experimental setup is as follows: we create a graph from the SBM with three clusters: $\mathcal{V} = S_1 \cup S_2 \cup S_3$. We set $|S_1| = |S_2| = 100$, and $|S_3| = 200$. We fix $C_{13} = C_{23} = 2$, $C_{33} = 18$. Then, we let $S_{gt} = S_1 \cup S_2$ and we make vary the internal structure of S_{gt}. Precisely, by keeping $C_{avg} = 20$ fixed, we make vary $C_{12}/C_{11} = C_{21}/C_{22}$ from poorly connected until S_1 and S_2 merge into a single uniformly connected cluster ($C_{12}/C_{11} = C_{21}/C_{22} = 1$). Then, 1 labeled point is sampled at random from S_{gt} and the L^γ-PageRank method is applied for different values of μ lying on a discrete grid. The clusters are determined via a sweep-cut procedure, and the best performance is retained. The whole procedure is repeated for 10 different realizations of the labeled points. Finally, all the preceding steps are repeated for 100 graph realizations.

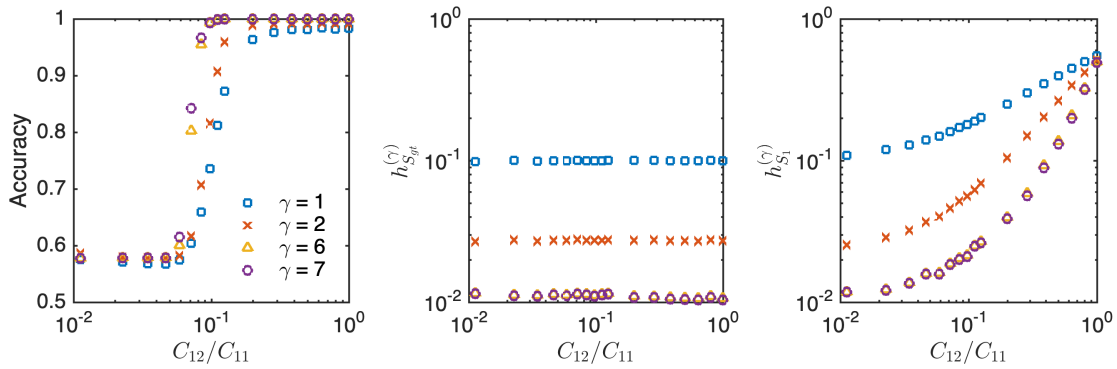


Figure 3.12: Classification performance as we vary the internal structure of S_{gt} through the ratio C_{12}/C_{11} while keeping constant the Cheeger ratio of S_{gt}.

Results and discussion. Figure 3.12 displays the performance of L^γ-PageRank at recovering S_{gt} as its internal connectivity varies through the ratio C_{12}/C_{11}. As it can be seen from Figure 3.12b, despite C_{12}/C_{11} varying, the Cheeger ratio of S_{gt} remains constant. Indeed, we can observe that the L^γ-topologies can help to further reduce $h^{(\gamma)}_{S_{gt}}$. However, as shown in Figure 3.12a, the Cheeger ratio of S_{gt} is not the only factor involved in its detection since for small values of C_{12}/C_{11} we cannot detect S_{gt} accurately while for larger values the performance significantly improves. Clearly, this is because the internal sub-cluster structure of S_{gt}, in the regime of small C_{12}/C_{11}, makes it hard to correctly diffuse information. Thus, as displayed by Fig 3.12c, we need to increase the internal connectivity fo S_{gt} to enhance the recovery of S_{gt}.[1]

3.6 Differences with Iterated Laplacian

In this section, we highlight some differences between our L^γ-PageRank and the generalization obtained by iterating the *random walk* Laplacian in the PageRank solution as proposed in [15].

Let us start emphasising the differences between the closed form solutions

- L^γ-PageRank:
$$f = \mu \left(L^\gamma D_\gamma^{-1} + \mu \mathbb{I} \right)^{-1} y. \tag{3.15}$$
- Iterated PageRank:
$$f = \mu \left(\left[L D^{-1} \right]^m + \mu \mathbb{I} \right)^{-1} y. \tag{3.16}$$

Thus, we can see that the contrast lies on the operators $L^\gamma D_\gamma^{-1}$ and $\left[L D^{-1} \right]^m$: the former corresponds to the *random walk* Laplacian of a L^γ-graph, while latter is high order version of the *random walk* Laplacian of the initial graph. By reverting to the spectral domain, we can more clearly highlight the differences between both operators. Let $L^\gamma D_\gamma^{-1} = Q_\gamma \Lambda_\gamma Q_\gamma^{-1}$,

[1] This phenomenon was pointed in our experiments from Section 3.4.2, where it was shown that the Lévy flight can override, to some extend, such sub-cluster structures.

then we have that $\left[LD^{-1}\right]^m = Q_1\Lambda_1^m Q_1^{-1}$. Thus, when comparing both methods with respect to the standard PageRank, we have that our L^γ-PageRank formulation changes both the spectrum and eigenvectors, while, in the Iterated Laplacian case, the eigenvectors are left unchanged. Clearly, relating Q_γ to Q_1 and Λ_γ to Λ_1^γ is not a trivial task. Numerical evidence suggests that $\Lambda_\gamma \approx \Lambda_1^\gamma$ only for small γ but that Q_γ and Q_1 can significantly differ. Thus, we leave the formal link between the output of both methods as an open problem. In practice, these spectral difference translate into the following differences between both methods:

1. There is no known optimization problem having the Iterated PageRank as a solution.
2. It is unclear if Iterated PageRank can be related to the equilibrium state of a diffusion process
3. It is unclear if Iterated PageRank can be related to the graph topology
4. There are not guarantees that doing a sweep on the Iterated PageRank still leads to a meaningful clustering
5. Iterated PageRank is not defined in the regime of fractional powers.
6. There are no insights on how to optimally tune m for the Iterated PageRank
7. L^γ-PageRank cannot be related to a Sobolev norm regularization

3.6.1 Numerical comparison

Performance on real world data

Experimental setup and goals. In this experiment, we aim to compare the performance between both L^γ-PageRank and Iterated PageRank when classifying the MNIST.

The experimental setup is as follows: a graph from the MNIST dataset is build using the guidelines given in Section 3.5.3 (Real world datasets). Then, one labelled point per class is sampled at random and both methods are applied, in the multi-class setting, for a grid of μ values and Laplacian powers in the range $\gamma = m = 1, ..., 5$. Accuracy, assessed by MCC, is computed for each class and then the scores of all classe are combined to obtain a global one. The configuration that leads to the best global score is retained. Lastly, the whole procedure is repeated for 30 different realizations of labelled points.

Results and discussion. Table 3.4 displays the performance of both L^γ-PageRank and Iterated PageRank in the classification of the digits from the MNIST dataset. We observe that, in both cases, the extra degree of freedom allows to enhance the classification performance but that L^γ-PageRank achieves the best overall performance. Indeed, for both methods the sweet spot is attained at $\gamma = m = 3$, but the accuracy differences are due to L^γ-PageRank being capable to double the improvements given by the iterated Laplacian in digits 1 and 2, while the iterated Laplacian does better in digits 7 and 9 but only by one unit. Now, for other γ and m, L^γ-PageRank performs better for $\gamma = m = 2$ and worse for $\gamma = m = 5$. The drop at $\gamma = 5$ is due to the L^5-graph permitting to much better recover digit 2 but at the price of degrading the detection of the other digits, particularly digit 6.

Digit		1	2	3	4	5	6	7	8	9	Total
It. PR	$m = 1$	0.57 (0.041)	0.57 (0.072)	0.52 (0.073)	0.36 (0.078)	0.53 (0.068)	0.82 (0.042)	0.65 (0.057)	0.56 (0.104)	0.38 (0.062)	0.55 (0.106)
	$m = 2$	0.59 (0.042)	0.58 (0.077)	0.53 (0.074)	0.36 (0.082)	0.54 (0.072)	0.84 (0.038)	0.67 (0.054)	0.59 (0.109)	0.39 (0.063)	0.56 (0.108)
	$m = 3$	0.59 (0.044)	0.60 (0.078)	0.54 (0.074)	0.36 (0.083)	0.54 (0.072)	0.84 (0.036)	0.69 (0.054)	0.60 (0.110)	0.40 (0.062)	0.57 (0.110)
	$m = 4$	0.58 (0.043)	0.59 (0.082)	0.55 (0.076)	0.35 (0.083)	0.54 (0.070)	0.83 (0.041)	0.69 (0.051)	0.61 (0.104)	0.39 (0.062)	0.57 (0.111)
	$m = 5$	0.55 (0.044)	0.60 (0.079)	0.55 (0.078)	0.35 (0.072)	0.54 (0.066)	0.81 (0.045)	0.66 (0.058)	0.60 (0.103)	0.39 (0.059)	0.56 (0.105)
L^γ-PR	$\gamma = 1$	0.57 (0.041)	0.57 (0.072)	0.52 (0.073)	0.36 (0.078)	0.53 (0.068)	0.82 (0.042)	0.65 (0.057)	0.56 (0.104)	0.38 (0.062)	0.55 (0.106)
	$\gamma = 2$	0.60 (0.042)	0.60 (0.077)	0.53 (0.074)	0.36 (0.079)	0.54 (0.070)	0.84 (0.039)	0.67 (0.055)	0.59 (0.108)	0.39 (0.062)	0.57 (0.110)
	$\gamma = 3$	0.61 (0.045)	0.63 (0.074)	0.54 (0.074)	0.36 (0.080)	0.54 (0.068)	0.84 (0.037)	0.68 (0.053)	0.61 (0.106)	0.38 (0.059)	**0.58 (0.113)**
	$\gamma = 4$	0.60 (0.044)	0.65 (0.076)	0.54 (0.074)	0.35 (0.079)	0.54 (0.067)	0.83 (0.042)	0.68 (0.054)	0.61 (0.104)	0.37 (0.058)	0.57 (0.116)
	$\gamma = 5$	0.54 (0.045)	0.66 (0.065)	0.53 (0.079)	0.35 (0.065)	0.55 (0.062)	0.72 (0.056)	0.63 (0.057)	0.58 (0.097)	0.37 (0.059)	0.55 (0.093)

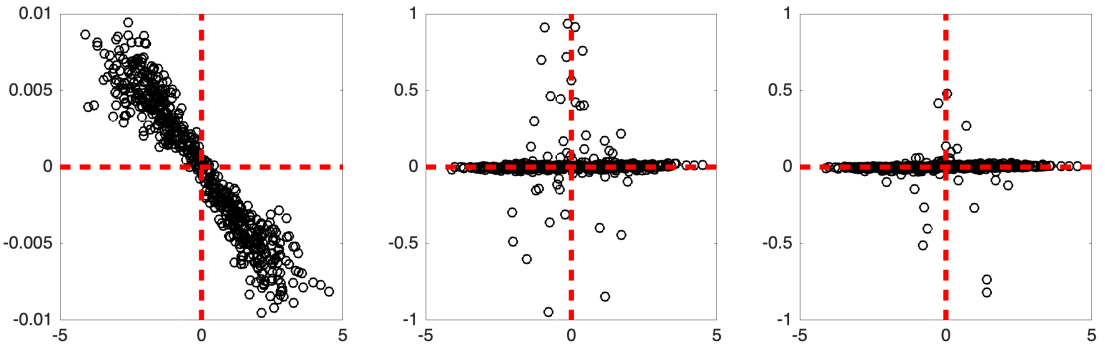
Table 3.4: Performance of both L^γ-PageRank and Iterated PageRank on the MNIST dataset (multi-class setting): each cell reports MCC and the 95% confidence interval (parenthesis).

Solving the curse of flatness

Experimental setup and goals. In Chapter 2, it was pointed that, in the limit of infinite unlabelled data, G-SSL methods suffer from the so-called curse of flatness (see Section 2.2.7, page 30). Indeed, it was shown there that the Iterated Laplacian G-SSL was proposed to amend such issue. In our following experiment, we show that our L^γ-PageRank is not suited to address the curse of flatness.

Our experimental setup is as follows: we replicate the experiment given in Section 2.2.7 (see page 33).

Results and discussion. Figure 3.13 displays the classification functions of L^γ-PageRank and the Iterated PageRank when applied on data that suffers from the curse of flatness. Since Iterated PageRank is designed to embed Sobolev regularity onto the PageRank vector, then it is able to overcome the curse of flatness and provide meaningful classification functions as shown by Figure 3.13a. Now, in Figures 3.13b and 3.13c, we display the L^γ-PageRank vector with $\gamma = 32$ and $\gamma = 62$, respectively. Clearly, our formulation is not able to cope with the curse of dimensionality issue. We presume that this is because Λ_γ and Λ_1^γ significantly differ for large γ, meaning that the L^γ-PageRank vector does not carry Sobolev regularites.



(a) It. PageRank ($m = 32$) (b) L^γ-PageRank ($\gamma = 32$) (c) L^γ-PageRank ($\gamma = 64$)

Figure 3.13: Comparison of L^γ-PageRank and Iterated PageRank for the curse of flatness. The horizontal axis represents the spacial dimension of the data and the verticalaxis represents the function value learned by G-SSL

3.7 Extending the generalized optimization framework to L^γ-graphs

In this section, we show that the generalized optimization framework for G-SSL [75] (see page 28) can also be readily extended to the L^γ-graphs and incorporate their benefits.

Let us depart from optimization problem (2.12) and extend it to the L^γ-graphs as follows:

$$\arg\min_f \left\{ f^T D_\gamma^{\sigma-1} L^\gamma D_\gamma^{\sigma-1} f + \mu (f-y)^T D_\gamma^{2\sigma-1} (f-y) \right\} \tag{3.17}$$

Thus, we observe that by properly choosing σ, we can recover the L^γ extension of the three popular methods. Namely, by choosing $\sigma = 1$ we obtain the L^γ-Standard Laplacian method; $\sigma = 1/2$ leads to the L^γ-Normalized Laplacian G-SSL; and $\sigma = 0$ reduces to our L^γ-PageRank.

Now, since the σ-th power of a positive diagonal matrix remains positive, then the same argumentation used in the proof of Lemma 6 can be applied to problem (3.17) to show that, for all γ, it is convex and its solution can be given in closed form as

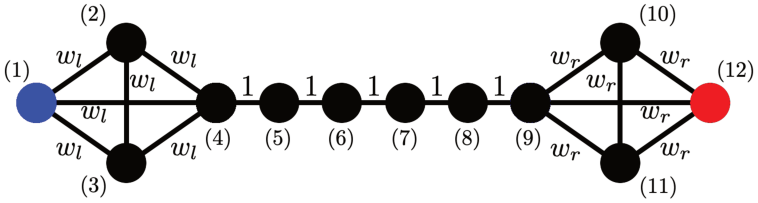
$$f = \mu \left(D_\gamma^{-\sigma} L^\gamma D_\gamma^{\sigma-1} + \mu \mathbb{I} \right)^{-1} y. \tag{3.18}$$

Then, by making the change of variable $\alpha = 1/(1+\mu)$, we can recast Eq. (3.18) as

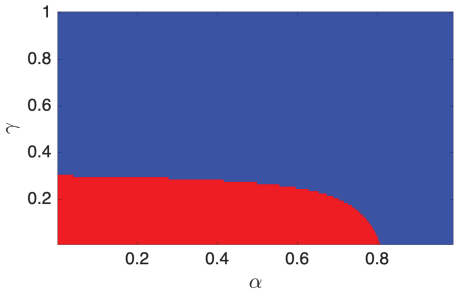
$$f^T = (1-\alpha) y^T \sum_{k=0}^{\infty} \alpha^k \left(D_\gamma^\sigma P_\gamma D_\gamma^{-\sigma} \right)^k \tag{3.19}$$

Thus, in the regime $\gamma < 1$, Eq. (3.19) corresponds to an extension of the generalized optimization framework to the Lévy processes encoded by P_γ.

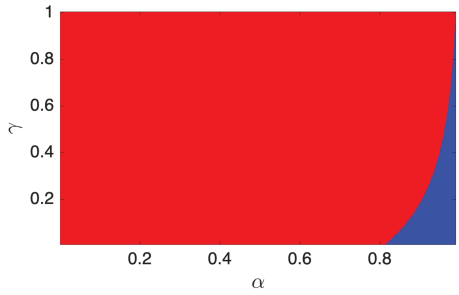
Now, as discussed in Chapter 2, PageRank is the only method with a known relationship to the graph topology. Thus, it is unclear in which manner Eq. (3.18) benefits from



(a) Lollipop with 2 classes



(b) Assigment of node (6)



(c) Assigment of node (7)

Figure 3.14: New competing classes with L^γ-Normalized Laplacian ($\gamma \leq 1$)

the signed L^γ-topologies in the regime $\gamma > 1$. We leave such theoretical analysis as an open problem. Nonetheless, below, we present numerical experiments pointing that the extension of these other G-SSL propositions to L^γ-graphs also imply a significant boost in their performance.

3.7.1 Numerical experiments

Regime $\gamma < 1$

Experimental setup and goals. Our goal in this experiment is to show that the new degree of freedom add extra flexibility that allows to tweak the classification in a desired manner. Towards this aim let us revisit the lollipop graph in Figure 3.14. For the test, we set the weights of the cliques as $w_l = 2$ and $w_r = 1$. As G-SSL method, we consider the L^γ-Normalized Laplacian ($\sigma = 1/2$) in the multi-class setting and we track the class assignment given to nodes (6) and (7) for a fine grid of α and γ.

Results and discussion. In Figure 3.14b and Figure 3.14c we display the class assignment of nodes (6) and (7) as a function of α and γ, respectively. As it can be seen, the standard Normalized Laplacian method ($\gamma = 1$) is very rigid assigning, for all α, nodes (6) and (7) always to the same classes. Thus, by introducing the γ parameter, we can see that non-local nature of P_γ allows to get regimes in the parameter space (γ, α) where the method is more flexible, allowing us to tweak the classification output if desired.

Regime $\gamma > 1$

Experimental setup and goals. Our goal in this experiment is to show that the L^γ-graphs, in the regime $\gamma > 1$, also enhance the performance of the other methods arising from the generalized optimization framework.

	Planted Partition	MNIST 4vs9	MNIST 3vs8	BBC articles	Gender images	Phoneme
			Balanced labels			
$\gamma = 1$	0.89 (8.3e-3)	0.50 (2.4e-2)	0.83 (1.0e-2)	0.85 (1.6e-2)	0.58 (3.3e-2)	0.44 (2.2e-2)
γ = Best	0.93 (5.4e-3) [5]	0.51 (2.3e-2) [2.5]	0.84 (1.1e-2) [1.5]	0.90 (1.2e-2) [4]	0.63 (2.3e-2) [1.5]	0.45 (1.9e-2) [2.5]
			Unbalanced labels			
$\gamma = 1$	0.79 (9.6e-2)	0.49 (1.6e-2)	0.71 (1.5e-2)	0.66 (2.2e-2)	0.63 (2.2e-2)	0.43 (2.5e-2)
γ = Best	0.88 (8.1e-3) [7]	0.55 (1.5e-2) [4]	0.79 (1.5e-2) [4]	0.90 (1.7e-3) [4]	0.71 (1.8e-2) [3.5]	0.46 (1.7e-2) [1.9]

Table 3.5: Performance of L^γ-Normalized Laplacian: each cell reports MCC, 95% confidence interval (parenthesis) and the value of γ [squared brackets].

Our experimental setup is as follows: two cases are considered (i) balanced labelled points; and (ii) unbalanced labelled points. For the experiments, we follow the same graphs and guidelines used for the experiments in Section 3.5.3. The only variation we use is that, for the experiment with balanced labels, we consider 2% of labelled points in both classes. As classification method we employ the L^γ-Normalized Laplacian ($\sigma = 1/2$).

Results and discussion. Table 3.5 displays the classification performance of L^γ-Normalized Laplacian in the classification of various real world datasets in the presence of both balanced labelled points and unbalanced labelled points. As it can be seen, the introduction of γ helps to (i) get moderate increments in performance in the case of balanced labels; and (ii) significantly improve the performance when facing unbalanced labelled datasets.

Appendix: technical proofs

3.A Proof of Lemma 6

Proof. It suffices to show the positive semi-definiteness of the functional and to apply the first order optimality condition. Let $\hat{f} = Q^T D_\gamma^{-1} f$. Then, the left term satisfies $\sum_j \lambda_j^\gamma \hat{f}_j^2 \geq 0$. It can be shown that $[D_\gamma]_{uu} = \sum_j Q_{uj}^2 \lambda_j^\gamma \geq 0$ granting the right term satisfies $\sum_u (f_u - y_u)^2 / [D_\gamma]_{uu} \geq 0$. Now, computing the derivative of the functional with respect to f and equaling to 0 leads to: $L^\gamma D_\gamma^{-1} f + \mu(f - y) = 0$. The lemma is proved after isolating f. ■

3.B Proof of Lemma 7

Proof. For an arbitrary and fixed $\gamma > 0$, let S denote an arbitrary set of $vol_\gamma(S) \leq vol_\gamma(G)/2$. Let $r = (\mathcal{A}_{out}^{(\gamma)}(S) - \mathcal{D}_{out}^{(\gamma)}(S))/(\mathcal{A}_{in}^{(\gamma)}(S) - \mathcal{D}_{in}^{(\gamma)}(S))$. It is easy to show that $h_S^{(\gamma)} = r/(r+1)$, which is monotonically increasing with r. Thus, the set S that minimizes r also minimizes $h_S^{(\gamma)}$. ■

3.C Proof of Lemma 8

Proof. From the demonstration of Lemma (6) we have that $L^\gamma D_\gamma^{-1} f + \mu(f - y) = 0$. Then, $\mathbb{1}^T L^\gamma D_\gamma^{-1} + \mu \mathbb{1}^T f = \mu \mathbb{1}^T y$. Since $\mathbb{1}^T L^\gamma = 0$ we have that $\mathbb{1}^T f = \mathbb{1}^T y$, proving (i). We prove property (iii) using the same expression. We only develop the case $\mu \to 0$ since the case $\mu \to \infty$ follows the same steps: taking $\lim_{\mu \to 0} \left\{ L^\gamma D_\gamma^{-1} f + \mu(f - y) = 0 \right\}$ leads to $L^\gamma D_\gamma^{-1} f = 0$, whose solution is proportional to $\pi_\gamma = D_\gamma \mathbb{1} / vol_\gamma(G)$. Lastly, we prove (ii) by noting that the operator $L^\gamma D_\gamma^{-1}$ has a positive real spectrum as it is similar to $D_\gamma^{-1/2} L^\gamma D_\gamma^{-1/2}$ which is positive semi-definite. Thus, we can use the inverse Laplace transform of the resolvent $(L^\gamma D_\gamma^{-1} + \mu \mathbb{I})^{-1} = \int_0^\infty e^{-t} e^{-t L^\gamma D_\gamma^{-1}/\mu} dt$, which, after using its Taylor expansion, allows to rewrite the L^γ-PageRank solution as $f = \sum_{k=0}^\infty \frac{(-1)^k}{\mu^k} \left(L^\gamma D_\gamma^{-1} \right)^k y$. If $y = \pi_\gamma$, the previous equation is only non-zero for $k = 0$, proving (ii). ■

3.D Proof of Lemma 9

Proof. Let $y = D_\gamma \mathbb{1}_S / vol_\gamma(S)$. Using Eq. (3.6) we can see that

$$\mathbb{1}_{S^c}^T f = \sum_{u \in S} \frac{[D_\gamma]_{uu}}{vol_\gamma(S)} \mathbb{1}_{S^c}^T \left[\mu \left(L^\gamma D_\gamma^{-1} + \mu \mathbb{I} \right)^{-1} \delta_u \right], \tag{3.20}$$

showing that $\mathbb{1}_{S^c}^T f$ can be interpreted as $\mathbb{E}[f(S^c)]$ when labels are selected with probability proportional to their generalized degree in S. Using the fact that

$$\left(L^\gamma D_\gamma^{-1} + \mu\mathbb{I}\right)^{-1}\left(L^\gamma D_\gamma^{-1} + \mu\mathbb{I}\right) = \mathbb{I}, \tag{3.21}$$

we express

$$f = \left(\mathbb{I} - \frac{1}{\mu}L^\gamma D_\gamma^{-1} + \frac{1}{\mu}L^\gamma D_\gamma^{-1}\left(L^\gamma D_\gamma^{-1} + \mu\mathbb{I}\right)^{-1} L^\gamma D_\gamma^{-1}\right) y. \tag{3.22}$$

The upper bound is thus obtained by substituting y and summing over S.

$$\begin{aligned}
\mathbb{1}_S^T f &= \frac{\mathbb{1}_S^T D_\gamma \mathbb{1}_S}{vol_\gamma(S)} - \frac{\mathbb{1}_S^T L^\gamma \mathbb{1}_S}{\mu\ vol_\gamma(S)} + \frac{\mathbb{1}_S^T L^\gamma \left(L^\gamma + \mu D_\gamma\right)^{-1} L^\gamma \mathbb{1}_S}{\mu\ vol_\gamma(S)} \\
&\geq \frac{\mathbb{1}_S^T D_\gamma \mathbb{1}_S}{vol_\gamma(S)} - \frac{\mathbb{1}_S^T L^\gamma \mathbb{1}_S}{\mu\ vol_\gamma(S)} \\
&= 1 - \frac{h_S^{(\gamma)}}{\mu}.
\end{aligned} \tag{3.23}$$

Employing property (i) from Lemma 8 finishes the proof. ∎

3.E Proof of Lemma 10

Proof. We only show the proof of the lower bound as the upper bound follows a similar derivation. We recast Eq. (6) as $L^\gamma D_\gamma^{-1} f = \mu(y - f)$. Thus, the set S_j satisfies:

$$\begin{aligned}
\mu\left((y(S_j) - f(S_j)\right) &= \mathbb{1}_{S_j}^T L^\gamma D_\gamma^{-1} f \\
&= \mathbb{1}_{S_j}^T L^\gamma q \\
&= \sum_{u\in S_j, v\in S_j^c} [W_\gamma]_{uv}(q_u - q_v) \\
&= \sum_{u\in S_j, v\in S_j^c} |\left[W_\gamma^+\right]_{uv}|(q_u - q_v) - \sum_{u\in S_j, v\in S_j^c} |\left[W_\gamma^-\right]_{uv}|(q_u - q_v) \\
&\quad + \sum_{u\in S_j, v\in S_j^c} |\left[W_\gamma^+\right]_{uv}|(q_j - q_{j+1}) - \sum_{u\in S_j, v\in S_j^c} |\left[W_\gamma^+\right]_{uv}|(q_j - q_{j+1}) \\
&\quad + \sum_{u\in S_j, v\in S_j^c} |\left[W_\gamma^-\right]_{uv}|(q_j - q_{j+1}) - \sum_{u\in S_j, v\in S_j^c} |\left[W_\gamma^-\right]_{uv}|(q_j - q_{j+1}) \\
&\geq (q_j - q_{j+1})\left(2\mathcal{A}_{out}^{(\gamma)}(S_j) + \mathcal{D}_{out}^{(\gamma)}(S_j)\right) \\
&\quad - (q_1 - q_N)\left(2\mathcal{D}_{out}^{(\gamma)}(S_j) + \mathcal{A}_{out}^{(\gamma)}(S_j)\right).
\end{aligned} \tag{3.24}$$

Re-ordering terms finishes the proof. ∎

3.F Proof of Theorem 5

Proof. Let $n = |S|$. For $u, v \in S$ and $w \in S^c$ the Planted Partition satisfies $\sum_v W_{uv} \sim B(n-1, p_{in})$ and $\sum_w W_{uw} \sim B(n, p_{out})$. The key step in the proof is to show that, in the

limit $n \to \infty$, $\mathbb{E}\left[h_S^{(1)}\right] = \mathbb{E}\left[\frac{\mathbb{1}_S^T L \mathbb{1}_S}{vol(S)}\right] = \frac{\mathbb{E}[\mathbb{1}_S^T L \mathbb{1}_S]}{\mathbb{E}[vol(S)]}$, and the same for $h_S^{(2)}$. By application of the Chebyshev inequality we have that

$$Pr\left(d_u - \mathbb{E}\left[d_u\right] \geq \mathbb{E}\left[d_u\right]\right) \leq \frac{var(d_u)}{var(d_u) + \mathbb{E}\left[d_u\right]^2} = \mathcal{O}(n^{-1}). \tag{3.25}$$

Thus, in the limit of $n \to \infty$ we can establish the inequality $d_u < 2\mathbb{E}\left[d_u\right]$ and further that $vol(S) < 2\mathbb{E}\left[vol(S)\right]$. This latter allows to express $\mathbb{E}\left[h_S^{(1)}\right]$ as follows [120]:

$$\begin{aligned}
\mathbb{E}\left[h_S^{(1)}\right] &= \mathbb{E}\left[\frac{\mathbb{1}_S^T L \mathbb{1}_S}{vol(S)}\right] \\
&= \frac{\mathbb{E}\left[\mathbb{1}_S^T L \mathbb{1}_S\right]}{\mathbb{E}\left[vol(S)\right]} + \sum_{i=1}^{\infty}(-1)^i \frac{\mathbb{E}[\mathbb{1}_S^T L \mathbb{1}_S]\langle\!\langle {}^i vol(S)\rangle\!\rangle + \langle\!\langle \mathbb{1}_S^T L \mathbb{1}_S, {}^i\, vol(S)\rangle\!\rangle}{\mathbb{E}\left[vol(S)\right]^{i+1}} \\
&= \frac{\mathbb{E}\left[\mathbb{1}_S^T L \mathbb{1}_S\right]}{\mathbb{E}\left[vol(S)\right]} + \sum_{i=1}^{\infty}(-1)^i \frac{\mathbb{E}\left[\mathbb{1}_S^T L \mathbb{1}_S (vol(S) - \mathbb{E}[vol(S)])^i\right]}{\mathbb{E}[vol(S)]^{i+1}} \\
&= \frac{\mathbb{E}\left[\mathbb{1}_S^T L \mathbb{1}_S\right]}{\mathbb{E}\left[vol(S)\right]} + \sum_{i=1}^{\infty}(-1)^i \mathbb{E}\left[\frac{\mathbb{1}_S^T L \mathbb{1}_S}{\mathbb{E}[vol(S)]}\left(\frac{vol(S)}{\mathbb{E}[vol(S)]} - 1\right)^i\right] \\
&= \frac{\mathbb{E}\left[\mathbb{1}_S^T L \mathbb{1}_S\right]}{\mathbb{E}\left[vol(S)\right]} + \sum_{i=1}^{\infty}(-1)^i c_i,
\end{aligned} \tag{3.26}$$

where $\langle\!\langle a, {}^i\, b\rangle\!\rangle = \mathbb{E}\left[(a - \mathbb{E}[a])(b - \mathbb{E}[b])^i\right]$. The fact that $vol(S) < 2\mathbb{E}\left[vol(S)\right]$ and the monotonicity of the expected value imply that the sequence $\sum_i |c_i|$ decreases monotonically. Also, it can be shown that its dominant term: $c_1 = \mathcal{O}(n^{-2})$. Replacing the expectations and evaluating the limit leads to:

$$\lim_{n\to\infty} \mathbb{E}\left[h_S^{(1)}\right] = \frac{p_{out}}{p_{in} + p_{out}}. \tag{3.27}$$

The case of $\mathbb{E}[h_S^{(2)}]$ follows a similar derivation. Since $[D_2]_{uu} = d_u^2 + d_u$, the Jensen inequality implies that $[D_2]_{uu} < 2\mathbb{E}\left[[D_2]_{uu}\right]$ and consequently that $vol_2(S) < 2\mathbb{E}\left[vol_2(S)\right]$. Thus, we cast:

$$\begin{aligned}
\mathbb{E}\left[h_S^{(2)}\right] &= \mathbb{E}\left[\frac{\mathbb{1}_S^T L^2 \mathbb{1}_S}{vol_2(S)}\right] \\
&= \frac{\mathbb{E}\left[\mathbb{1}_S^T L^2 \mathbb{1}_S\right]}{\mathbb{E}\left[vol_2(S)\right]} + \sum_{i=1}^{\infty}(-1)^i \mathbb{E}\left[\frac{\mathbb{1}_S^T L^2 \mathbb{1}_S}{\mathbb{E}[vol_2(S)]}\left(\frac{vol_2(S)}{\mathbb{E}[vol_2(S)]} - 1\right)^i\right] \\
&= \frac{\mathbb{E}\left[\mathbb{1}_S^T L^2 \mathbb{1}_S\right]}{\mathbb{E}\left[vol_2(S)\right]} + \sum_{i=1}^{\infty}(-1)^i c_i^{(2)}.
\end{aligned} \tag{3.28}$$

Let the random variable $O_u = \sum_{w \in S^c} W_{uw}$. Then we have that $\mathbb{1}_S^T L^2 \mathbb{1}_S = 2\sum_{u\in S}(O_u)^2$. This fact, in addition to $vol_2(S) = \sum_{u\in S} d_u^2 + d_u$, allow to show that the sequence $\sum_i |c_i^{(2)}|$ is monotonically decreasing with $c_1^{(2)} = \mathcal{O}(n^{-1})$. Replacing the expectations and evaluating the limit leads to:

$$\lim_{n\to\infty} \mathbb{E}\left[h_S^{(2)}\right] = 2\left(\frac{p_{out}}{p_{in} + p_{out}}\right)^2. \tag{3.29}$$

■

3.G Proof of Corollary 1

Proof. Let $p_{in} = p_{out} + \epsilon$ and assume that $h_S^{(1)} \geq h_S^{(2)}$. Thus $p_{out}/(p_{in} + p_{out}) \geq 2(p_{out}/(p_{in} + p_{out}))^2$, which can be further simplified to $1 \geq 2p_{out}/(2p_{out} + \epsilon)$. We observe that such expression holds for $\epsilon \geq 0$ and equality occurs when $\epsilon = 0$. ■

3.H Proof of Proposition 2

Proof. We search a condition on S that permits $\frac{\mathbb{1}_S^T L \mathbb{1}_S}{\mathbb{1}_S^T D \mathbb{1}_S} \geq \frac{\mathbb{1}_S^T L^2 \mathbb{1}_S}{\mathbb{1}_S^T D_2 \mathbb{1}_S}$, or equivalently, that satisfies the inequality $\frac{\mathbb{1}_S^T D_2 \mathbb{1}_S}{\mathbb{1}_S^T D \mathbb{1}_S} - \frac{\mathbb{1}_S^T L^2 \mathbb{1}_S}{\mathbb{1}_S^T L \mathbb{1}_S} \geq 0$. We have

$$\begin{aligned}
\frac{\mathbb{1}_S^T D_2 \mathbb{1}_S}{\mathbb{1}_S^T D \mathbb{1}_S} - \frac{\mathbb{1}_S^T L^2 \mathbb{1}_S}{\mathbb{1}_S^T L \mathbb{1}_S} &\geq \frac{\mathbb{1}_S^T D^2 \mathbb{1}_S}{\mathbb{1}_S^T D \mathbb{1}_S} - \frac{\mathbb{1}_S^T L^2 \mathbb{1}_S}{\mathbb{1}_S^T L \mathbb{1}_S} \\
&\geq \frac{\mathbb{1}_S^T D^2 \mathbb{1}_S}{\mathbb{1}_S^T D \mathbb{1}_S} - \left(\max_{u \in S} \sum_{w \in S^c} W_{uw} + \max_{\ell \in S^c} \sum_{v \in S} W_{\ell v} \right) \\
&\geq \frac{\mathbb{1}_S^T D \mathbb{1}_S}{\mathbb{1}_S^T \mathbb{1}_S} - \left(\max_{u \in S} \sum_{w \in S^c} W_{uw} + \max_{\ell \in S^c} \sum_{v \in S} W_{\ell v} \right) \\
&= \frac{vol(S)}{|S|} - \left(\max_{u \in S} \sum_{w \in S^c} W_{uw} + \max_{\ell \in S^c} \sum_{v \in S} W_{\ell v} \right), \qquad (3.30)
\end{aligned}$$

where we have used Lehmers and Holders inequalities and that $\mathbb{1}_S^T L^2 \mathbb{1}_S = \sum_{u \in S} \left(\sum_{w \in S^c} W_{uw} \right)^2 + \sum_{\ell \in S^c} \left(\sum_{v \in S} W_{\ell v} \right)^2$. Thus, it is sufficient that S satisfies:

$$\frac{vol(S)}{|S|} - \left(\max_{u \in S} \sum_{w \in S^c} W_{uw} + \max_{\ell \in S^c} \sum_{v \in S} W_{\ell v} \right) \geq 0. \qquad (3.31)$$

■

Chapter 4

Fast and efficient implementations

4.1 Introduction

In Chapter 3, we extended PageRank and other popular G-SSL propositions to powers of Laplacian matrices. We showed that our extensions permit to significantly enhance the performance and flexibility of G-SSL. Moreover, it was proven that our methods enjoy from a well-behaved solution expressed in closed form. However, a strong limitation of our closed form expressions is that they require to perform matrix inversion, meaning that they can be prohibitively expensive (or even unfeasible) to calculate in large scale contexts. Clearly, this arises as a serious drawback of G-SSL since numerous modern systems foster applications involving thousands or millions of dimensions. Let us take the web graph as an example: it has 60×10^{12} web-sites and it evolves at a rate of 600×10^{3} new pages created every second [121]. Thus, despite their remarkable performance, the value of our G-SSL propositions remains limited if they cannot be used to address such important applications. In order to bring us a step closer towards this goal, in this chapter we investigate computationally efficient implementations that can better cope with such large scale rapidly evolving scenarios.

In terms of efficient implementations, the standard PageRank algorithm constitutes the state of the art approach. This is because, as we discussed in Chapter 2, standard PageRank possesses a random walk interpretation that numerous works have exploited to derive efficient ways to compute PageRank vectors [122, 123, 124, 111, 125]. Among these, three highly successful algorithms, practically employed in networks involving millions of vertices and billions of edges, arise: (a) Power iteration [18, 19, 20]; (b) Monte-Carlo simulation [126]; and (c) Gauss-Southwell method [11, 21]. *Power iteration* is the fundamental approach for efficient PageRank computing. It consists on iterating the PageRank fixed point equation until convergence. Due to its recursive nature, it allows to efficiently compute PageRank vectors via matrix-vector products, thus allowing a distributed implementation with a cost proportional to the number of edges in the graph. *Monte-Carlo simulation* runs individual random walkers and then estimates the entries of the PageRank vector according to the frequency of visits made to nodes by the walkers. It has the advantage of being easily parallelizable and it has been reported to give good estimates from only a few iterations. The *Gauss-Southwell method* is a fast approach to compute approximate PageRank vectors. It employs two vectors: the approximate PageRank vector and a residual vector coding for the error in the approximation. Then, it pushes mass from the

residual vector into the approximate PageRank vector iteratively until the residual vector diminishes below certain threshold. Albeit the algorithm is centralized, it is inexpensive to compute with a running time that is independent of the size of the graph.

The success of these algorithms also lies in that they have been effectively adapted to more efficiently cope with evolving graph structures. Let us stress that, in real world applications, it is common that sensors constantly collect new data, users in social networks continuously make new followers/friends or join/leave the network, or articles in citation networks become more or less influential as time passes by. Thus, to classify all these new data, or to adapt the class of a given node to a new state of the network, one needs to constantly re-compute the output of G-SSL. Evidently, doing this at a fast pace can simply be too computationally demanding to be practical. Notably, the works of [24, 127, 25] have shown that, when the graph changes smoothly, it is wasteful to discard an existing PageRank vector in order to compute a new one. Instead, such works show that the algorithms introduced above can be adapted to update an already existing PageRank vector in sublinear time. Their key observation is that changes in the graph tend to only affect the entries of the PageRank vector that lie close to where the change happened. Thus, implying that one only needs to update the entries of the PageRank vector in the local vicinity of a change. To have an idea on the practical gains given by these works, [25] is able to update PageRank vectors in a web graph of 105 million vertices and 3.7 billion edges in only 3 microseconds, which is more than 10000 faster than doing full re-computation via the power method. Clearly, we aim to attain such efficient implementations to the remainder of G-SSL methods, particularly the ones proposed in this work. However, it is not straightforward to apply these algorithms beyond the PageRank case as they heavily rely on its Markov chain structure, which, we recall, our L^γ-based extensions do not possess.

In this chapter, we leverage results from the field of Graph Signal Processing (GSP) to derive extensions of the aforementioned algorithms to the general family of G-SSL methods. Towards this goal, we exploit the fact that, despite not necessarily having a Markov chain structure, the G-SSL methods considered in this work do have an interpretation in terms of graph filters. Concerning the latter, the field of GSP has developed a solid toolbox that allows to efficiently compute the output of graph filters. In particular, there are two main approaches: the *Chebyshev polynomials* [23] and the *autoregressive moving average* (ARMA) filters [22]. In the former, one approximates the graph filter response by means of a truncated polynomial of the graph operator. In the latter, the transfer function of the filter is implemented by means of a recursive formula that converges as long as the filter poles lie within a stability region. Both approaches are very appealing as they permit distributed implementations with a cost proportional to the number of edges. The ARMA recursions are particularly relevant four us, as our main contribution from this chapter is to show that they permit an extension of the algorithms from [19, 11, 24, 25] to the general G-SSL setting. In principle, if one directly uses the definition of ARMA filters to implement the G-SSL frequency response, then the ARMA filters only serve to implement G-SSL for a narrow regime of values of the μ parameter. Thus, by mirroring [23], we do a shifting of the operator's domain and show that by doing this, then the region of stability can be extended to all $\mu > 0$, leading us to obtain a recursive formula that extends the power method to the general G-SSL setting. By elaborating on this ARMA recursive formula, we then extend the Gauss-Southwell method of [11] to general

G-SSL procedures. Lastly, in the second part of the chapter, we use these results to extend the updating algorithms of [24, 25] to the general family of G-SSL propositions. In addition to extending these important algorithms, in this chapter we study other efficient approaches. Concerning the polynomial representations of G-SSL covered in the first part of the chapter, we propose and assess the effectiveness of a novel representation based on Greens functions that also leads to an efficient and distributed algorithm. With respect to the update algorithms, we propose a novel promising approach based on neural networks to more effectively solve the update problem.

4.2 State-of-the-art approaches for PageRank computation

In this section, we recall the state of the art approaches of [19, 11, 24, 25] to efficiently compute PageRank vectors, that we aim to extend to the more general G-SSL propositions. To simplify notations, let us denote by $\mathrm{pr}_\alpha(y)$ the personalized PageRank vector of restarting probability α and initial condition y.

4.2.1 PageRank on static networks

Power iteration

In Section 2.3.4 (see page 38), it was shown that PageRank can be cast as the solution to the fixed point:

$$\mathrm{pr}_\alpha(y) = (1-\alpha)y + \alpha P^T \mathrm{pr}_\alpha(y) \tag{4.1}$$

Note that Eq. (4.1) corresponds to an eigenvector problem. This can be seen by rewriting it as $\mathrm{pr}_\alpha(y) = \left[(1-\alpha)y\mathbb{1}^T + \alpha P^T\right]\mathrm{pr}_\alpha(y)$. The matrix in squared brackets is normally referred to as the Google matrix, thus it can be seen that PageRank corresponds to the eigenvector with eigenvalue 1 of the Google matrix. If P is irreducible, then such matrix codes for a Markov chain with a stationary state, meaning that it only posses one eigenvalue lying in its spectral circle and hence the PageRank vector can be found via the power method. This latter is defined through the recursive formula [18, 19, 20]:

$$p^{(t)} = (1-\alpha)y + \alpha P^T p^{(t-1)}, \tag{4.2}$$

where $p^{(t)}$ denotes the PageRank estimation at iteration t. This recursion is very efficient to implement as to obtain the new iteration one simply needs to: (i) perform a (usually sparse) matrix-vector product; (ii) re-scale the resulting vector; and (iii) add the (re-scaled) labelled points. Thus, the cost of the algorithm is determined by the matrix-vector step and the number of iterations until convergence. Concerning the latter, it is not hard to show that the following relationship holds:

$$\|\mathrm{pr}_\alpha(y) - p^{(t)}\|_\infty \leq \alpha\|\mathrm{pr}_\alpha(y) - p^{(t-1)}\|_\infty \leq \alpha^t\|\mathrm{pr}_\alpha(y) - p^{(0)}\|_\infty \tag{4.3}$$

This inequality says that the power method converges in exact arithmetic and it has an asymptotic convergence rate determined by $\alpha^t \to 0$. As a result, if the algorithm requires from K iterations to converge, then its running time is $\mathcal{O}(K|\mathcal{E}|)$. Importantly, this procedure allows a distributed implementation given in Algorithm 3.

Algorithm 3 [18, 19, 20] Distributed computation of PageRank via the Power method

Input at node u: y_u, α, P_{vu} $\forall$ $v \sim u$, K, and $p^{(0)}$.
Output at node u: $p^{(K)}$
for $t = 1 : t_{max}$ **do**
  Transmit $p_u^{(t-1)}$ to all neighbors $v \sim u$
  Receive $p_v^{(t-1)}$ from all neighbors $v \sim u$
  $p_u^{(k)} = (1-\alpha)y_u + \alpha \sum_{v \sim u} p_v^{(t-1)} P_{vu}$
end for
Return $p^{(K)}$

The Gauss-Southwell method

The Gauss-Southwell algorithm [11, 21] is an iterative method that efficiently computes approximate PageRank vectors. In this algorithm two vectors are used: (i) p: an approximate PageRank vector; and (ii) r: a residual vector coding for the difference between $\mathrm{pr}_\alpha(y)$ and p. The method initiates with an initial guess $p^{(0)}$ and then, at iteration t, the algorithm transfers mass from $r^{(t-1)}$ into $p^{(t)}$ such that the following invariant is preserved:

$$\mathrm{pr}_\alpha(y) = p^{(t)} + \frac{1}{1-\alpha}\mathrm{pr}_\alpha(r^{(t)}) \tag{4.4}$$

Hence, the goal of the algorithm is to minimize the residual as that implies that p converges to the PageRank vector. Let us assume that, at iteration t, the largest entry of $r^{(t)}$ corresponds to vertex u. Then, the state of the algorithm at iteration $t+1$ is determined by the following set of update equations:

$$p^{(t+1)} = p^{(t)} + r_u^{(t)}\delta_u \tag{4.5}$$

$$r^{(t+1)} = r^{(t)} - r_u^{(t)}\delta_u + \alpha P^T r_u^{(t)}\delta_u \tag{4.6}$$

This procedure is repeated until all entries from the residual vector diminish below some threshold value ϵ. The whole procedure is detailed in Algorithm 4.

Algorithm 4 is extremely efficient to implement as the update equations only imply addition and re-scaling of vectors. To see it, recall that $\alpha P^T r_u^{(t-1)}\delta_u$ corresponds to a re-scaling the u-th column of P^T by the scalar $\alpha r_u^{(t-1)}$. Hence, cost of the algorithm is determined by the number of iterations required until convergence. Naturally, such number of iterations depends on the desired accuracy, as stated by the following result:

Lemma 11 ([11]). *When Algorithm 4 terminates, its output p satisfies $\|pr_\alpha(y) - p\|_\infty \leq \frac{\epsilon}{1-\alpha}$ and it does it within $\|r^{(0)}\|_1/((1-\alpha)\epsilon)$ iterations.*

This Lemma implies that cost of the Gauss-Southwell algorithm is $\mathcal{O}(\|r^{(0)}\|_1/((1-\alpha)\epsilon)$, which is independent of the number of edges in the graph.

4.2.2 Updating PageRank on dynamic networks

In the introduction of this Chapter, we pointed that graphs arising in real world applications are not only massive but also dynamic. Intuition says that if the graph does not

Algorithm 4 [11] Approximate PageRank vectors via Gauss-Southwell method

Input: P, $p^{(0)}$, and ϵ
Output: Approximate PageRank vector p and its residual r
$r^{(0)} = (1-\alpha)y - \left(\mathbb{I} - \alpha P^T\right)p^{(0)}$
t = 0
while $\|r^{(t)}\|_\infty > \epsilon$ **do**
  $u = \arg\max_u (r^{(t)})_u$
  $p^{(t+1)} = p^{(t)} + (r^{(t)})_u \delta_u$
  $r^{(t+1)} = r^{(t)} - (r^{(t)})_u \delta_u + \alpha P^T (r^{(t)})_u \delta_u$
  $t++$
end while
Return $p = p^{(t)}$ and $r = r^{(t)}$

drastically change, then the G-SSL solution should be relatively stable, implying that there may be better alternatives than re-computing from scratch the PageRank vector every time a change occurs. In this section we present adaptations of the algorithms introduced above that allow to perform efficient local updates to the PageRank vector.

To proceed, consider two graphs: $\mathcal{G}(\mathcal{V}, \mathcal{E}, W)$ and an evolved version of such graph given as $\widetilde{\mathcal{G}}(\widetilde{\mathcal{V}}, \widetilde{\mathcal{E}}, \widetilde{W})$. Moreover, let $P = D^{-1}W$ and $\widetilde{P} = \widetilde{D}^{-1}\widetilde{W}$ denote the transition matrices of $\mathcal{G}$ and $\widetilde{\mathcal{G}}$, respectively. To have consistent sized matrices, let nodes joining/leaving the network be modelled as isolated nodes that get connected/disconnected. By definition, we set these isolated nodes with zero rows and columns in P and $\widetilde{P}$. Please note that by adding rows full of zeros to P we do not change the convergence of the chain as this only adds adds eigenvalues equal to zero and preserves the unique eigenvalue in the spectral circle.

Now, we present the following result from [128], saying that if a small change in the graph occurs, then the PageRank vector should not drastically change.

Lemma 12 ([128])**.** *Let $pr_\alpha(y)$ and $\widetilde{pr}_\alpha(y)$ denote the Pagerank vectors on $\mathcal{G}$ and $\widetilde{\mathcal{G}}$, respectively. Then, we have that*

$$\|\widetilde{pr}_\alpha(y) - pr_\alpha(y)\|_\infty \leq \frac{\alpha}{1-\alpha}\|\widetilde{P} - P\|_\infty \tag{4.7}$$

In addition to this result, it is important to stress that changes only reflect locally in the perturbation. To have a grasp on this, let us illustrate such phenomenon in Figure 4.1. In the figure, we display the Minnesota graph (a well known toy graph) and we show the effect that adding a new node has on the PageRank vector. More precisely, in the left figure, we show (in red) the new node joining the graph. Then, in the right plot, we show the difference between the PageRank vectors from the initial graph and the evolved graph. As it can be seen, the differences only appear in the region surrounding the arriving node. The implication of this observation is that one can keep the same scores in the PageRank vector for nodes that are far from the perturbation, hence avoiding the need to spend computational resources to re-estimate the score of such vertices, and then one only needs to compute the new scores for the nodes encircling the change.

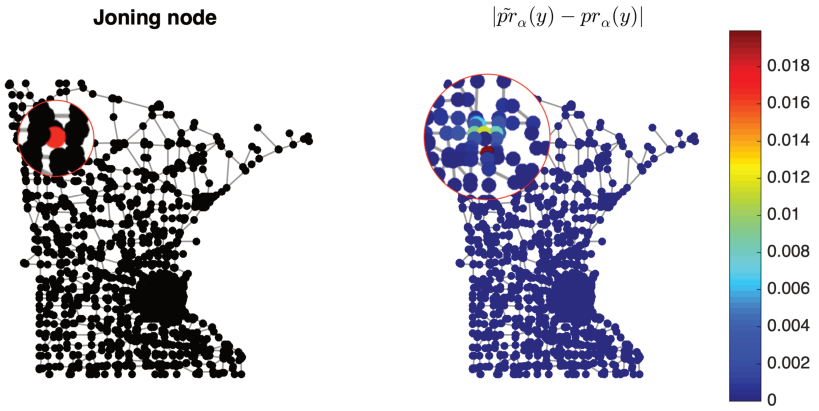


Figure 4.1: Effect that adding one node to the Minnesota graph has on the PageRank vector.

Local updating via power iteration

By examining Eq. (4.3), it can be seen that the the initial guess, $p^{(0)}$, has a key role in he convergence time of the power method. Thus, since it is assumed that $\mathrm{pr}_\alpha(y)$ and $\widetilde{\mathrm{pr}}_\alpha(y)$ are close, Eq. (4.3) implies that by setting $\widetilde{p}^{(0)} = \mathrm{pr}_\alpha(y)$ (the new initial guess when using the power method for $\widetilde{\mathcal{G}}$) can significantly reduce the number of iterations required for power the method to compute $\widetilde{\mathrm{pr}}_\alpha(y)$. This approach of using the previous PageRank vector as the new initial guess for a new run of the power method is normally known as the power method with warm restart. In Figure 4.2, we compare the approximation error obtained by computing $\widetilde{\mathrm{pr}}_\alpha(y)$ either from scratch or by using the warm restarts on the same Minnesota graph example used above. From the figure, it can be seen that the warm restarts can significantly reduce the number of iterations needed to achieve a target error. However, the problem with this approach is that the entire graph still needs to be seen or loaded into memory to compute the updated PageRank vector, as one still needs to perform the matrix-vector products with $\widetilde{p}^{(0)} = \mathrm{pr}_\alpha(y)$ which is usually a dense (global) vector.

In [24], the authors amend this issue by showing that the power method with warm restart can be cast as a local update problem as long as α and y do not change between graph realizations. More precisely, [24] shows that by using $\widetilde{p}^{(0)} = \mathrm{pr}_\alpha(y)$ in the power method, then the following holds:

$$\widetilde{\mathrm{pr}}_\alpha(y) = \mathrm{pr}_\alpha(y) + \frac{1}{(1-\alpha)}\widetilde{\mathrm{pr}}_\alpha(r), \tag{4.8}$$

where

$$r = \alpha[\widetilde{P}^T - P^T]\mathrm{pr}_\alpha(y) \tag{4.9}$$

Notably, Eq. (4.8) says that to update an existing PageRank vector, then one needs to compute another PageRank vector. However, this novel PageRank vector is much more efficient to compute. This is because $[\widetilde{P}^T - P^T]$ is only non-zero on the rows corresponding to nodes that changed, implying that r is completely localized (only non-zero) in the

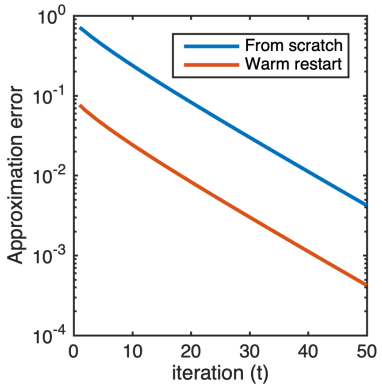


Figure 4.2: Normalized approximation in the power method when doing computation from scratch or by using the warm restarts. For the test, we use the same Minessota graph example from Figure 4.1.

Algorithm 5 [24] Local PageRank updating via power iteration

Input: α, $\mathrm{pr}_\alpha(y)$, P, $\widetilde{P}$.
Output: $\widetilde{\mathrm{pr}}_\alpha(y)$.

$r = \alpha[\widetilde{P}^T - P^T]\mathrm{pr}_\alpha(y)$
Compute $\widetilde{\mathrm{pr}}_\alpha(r)$ via the power method
Return $\widetilde{\mathrm{pr}}_\alpha(y) = \mathrm{pr}_\alpha(y) + \widetilde{\mathrm{pr}}_\alpha(r)/(1-\alpha)$

1-hop vicinity of the perturbed nodes. Hence, since $\widetilde{\mathrm{pr}}_\alpha(r)$ entails starting a diffusion process from the initial condition r, then one can interpret Eq. (4.8) as r containing the information for an update and $\widetilde{\mathrm{pr}}_\alpha(r)$ as a way to diffuse this update in $\widetilde{\mathcal{G}}$, starting from the perturbed nodes and then propagating it to nodes farther away. Since $\widetilde{\mathrm{pr}}_\alpha(r)$ can be computed by means of the power method, then one can control the scope of propagation for an update depending on the number of iterations used in the algorithm. This is, if one uses K iterations to estimate $\widetilde{\mathrm{pr}}_\alpha(r)$, then the update is propagated to nodes in a K-hop vicinity from the perturbations. This provides significant computational benefits as one only needs to explore the graph within this K-hop region to get the new PageRank vector, implying that (i) the algorithm has sublinear complexity; and (ii) if, at most, there are M edges are contained in the K-hop vicinity of a node that changed, and c changes occur, then the cost of the update is bounded by $\mathcal{O}(cKM)$. If one uses large values of K, then one gets better approximations, yet one may end up exploring most of the graph while getting negligible practical benefits. Thus, the goal is find a good tradeoff between accuracy and complexity. The procedure proposed in [24] is summarized in Algorithm 5.

Local updating via Gauss Southwell

In [25], the authors extended the Gauss Southwell method to perform efficient updates to approximate PageRank vectors. The underlying idea of their approach is to also use warm restarts for the Gauss-Southwell method. In precise terms, if the restart parameter and initial condition do not change, and the Gauss-Southwell method, for graph $\mathcal{G}$, outputs p and r as an approximate PageRank vector and its residual, respectively, then [25] proposes

Algorithm 6 [25] Local PageRank updating via Gauss-Southwell

Input: α, P, $\widetilde{P}$, p and r.
Output: Updated PageRank vector $\widetilde{p}$ and updated residual $\widetilde{r}$.

$\widetilde{p}^{(0)} = p$
$\widetilde{r}^{(0)} = r + \alpha[\widetilde{P}^T - P^T]p$
Apply Gauss-Southwell on $\widetilde{p}^{(0)}$ and $\widetilde{r}^{(0)}$, store result in $\widetilde{p}$ and $\widetilde{r}$
Return $\widetilde{p}$ and $\widetilde{r}$

to use, for graph $\widetilde{\mathcal{G}}$, the following initialization (still, preserving Eq. (4.4)):

$$\widetilde{p}^{(0)} = p \tag{4.10}$$

$$\widetilde{r}^{(0)} = r + \alpha[\widetilde{P}^T - P^T]p \tag{4.11}$$

The method proposed in [25] is shown in Algorithm 6. By definition, $\|r\|_\infty < \epsilon$ and, as discussed above, the term $\alpha[\widetilde{P}^T - P^T]p$ is non-zero everywhere but in the 1-hop vicinity of the perturbed nodes. As a result, only few entries in $\widetilde{r}^{(0)}$ can surpass the tolerance threshold, having the implication that with only a few push operations the algorithm will drive these entries below the threshold again. Thus, the algorithm is extremely efficient in practice. However, since $\widetilde{r}^{(0)}$ may now have positive or negative entries, deriving theoretical bounds on the running time of Algorithm 6 significantly complexifies. The authors of [25] give two results in this regard: one using amortized analysis and another in expected value sense. For the former, [25] shows that Algorithm 6 is amortized by $\mathcal{O}(1/\epsilon)$. With respect to the latter, [25] shows that if m edges are randomly and sequentially inserted, then, in expected value, it takes $\mathcal{O}(\log(m)/\epsilon)$ iterations for Algorithm 6 to converge.

4.3 Fast and efficient implementations of G-SSL on static graphs

In this section, we derive extensions of the algorithms from Section 4.2.1 to the general family of G-SSL propositions. Moreover, we explore other distributed approaches for efficient G-SSL computation. Let us commence clarifying that we cannot replace P by P_γ in the algorithms above and use them to implement our L^γ-PageRank method. This is because P_γ, for $\gamma > 1$, is no longer a stochastic matrix, implying that its spectral radius is larger than 1 and the recursive Eq. (4.1) diverges. Thus, other alternatives must be sought.

A better alternative consist in noticing that all the G-SSL methods presented in this PhD work can be expressed in a general linear system form:

$$\mathcal{R}f + \mu f = \mu y, \tag{4.12}$$

where $\mathcal{R} \in \mathbb{R}^{N\times N}$ denotes a generalized reference operator. Thus, the choice of $\mathcal{R}$ determines the G-SSL method under consideration. Table 4.1 lists possible values of $\mathcal{R}$.

Consider the spectral representation of $\mathcal{R}$ and the definition of the Graph Fourier Transform (GFT) given in Chapter 1. Then, we have that the solution to the linear system above can be expressed as

$$\hat{f}_u = \frac{\mu}{\lambda_u + \mu}\hat{y}_u. \tag{4.13}$$

Method	L^γ-PageRank	L^γ-Norm. Lap.	L^γ-Std. Lap.	Iter. PageRank	Recenterd kernel
$\mathcal{R}$	$L^\gamma D_\gamma^{-1}$	$D_\gamma^{-\frac{1}{2}} L^\gamma D_\gamma^{-\frac{1}{2}}$	$D_\gamma^{-1} L^\gamma$	$(LD^{-1})^m$	$-PWP$ with $P = \mathbb{I} - \frac{1}{N}\mathbb{1}\mathbb{1}^T$

Table 4.1: Possible choices for reference operators in G-SSL.

This equation shows that the u-th frequency component of f corresponds to a re-scaling of the u-th frequency component of y according to the transfer function $h(\Lambda) = \frac{\mu}{\mu+\Lambda}$. In other words, Eq. (4.13) shows that G-SSL methods correspond to a low-pass graph filter in which the labelled points constitute the signal to be filtered and the regularization parameter μ determines the cut-off frequency of the filter.

Classical results from matrix theory say that any function on the spectrum of a matrix is a function of the matrix itself. Therefore, implying that if $h(\Lambda)$ is approximated through a polynomial function: $h(\Lambda) \approx \sum_{t=0}^{K} c_t \Lambda^t$, then such approximation can be expressed in the vertex domain as $h(\Lambda) = h(\mathcal{R}) \approx \sum_{t=0}^{K} c_t \mathcal{R}^t$. The assets of this rewriting is that the filter output is then expressed as

$$f = \sum_{t=0}^{K} c_t \mathcal{R}^t y, \tag{4.14}$$

which can be computed recursively, in a distributed manner, and with cost $\mathcal{O}(K|\mathcal{E}|)$.

Thus, the question that arises is how to find the best set of coefficients for such polynomial. In the GSP literature, there are two state-of-the-art approaches to approximate filter functions via polynomials: (i) the Chebyshev polynomials; and (ii) the ARMA recursions. In this section, we leverage them to efficiently implement G-SSL. In particular, we show that the ARMA recursions imply a direct extension of the algorithms from Section 4.2. In addition, we derive a novel polynomial representation based on the Greens function and explore its feasibility.

4.3.1 Generalized implementation via Chebyshev polynomials

In [23], the Chebyshev polynomials for graph filter approximation were proposed. Such work proposes to approximate the filter function $h(\Lambda)$ by the following truncated series, which has been shown to be optimal in the ∞-norm sense [23, 129]:

$$h(\Lambda) \approx \frac{1}{2} c_0 + \sum_{t=1}^{K} c_t \bar{T}_t(\Lambda), \quad \text{for} \quad \Lambda \in [0, \lambda_{max}], \tag{4.15}$$

where

$$\bar{T}_t(\Lambda) = \begin{cases} 1, & t = 0 \\ \frac{\Lambda - \phi}{\phi}, & t = 1 \\ 2\left(\frac{\Lambda-\phi}{\phi}\right)\bar{T}_{t-1} - \bar{T}_{t-2}, & t \geq 2 \end{cases}. \tag{4.16}$$

and $\phi = \lambda_{max}/2$, $c_t = \frac{2}{\pi}\int_0^\pi cos(t\theta) h(\phi(cos(\theta)+1)) d\theta$.

Algorithm 7 [23] Distributed computation of G-SSL via Chebyshev polynomials

Input at node u: λ_{max}, y_u, μ, $\mathcal{R}_{uv}$ $\forall$ $v \sim u$, K, and $\{c_t : t = 0, \dots, K\}$.
Output at node u: $f_u^{(K)}$
$(\bar{T}_0(\mathcal{R})y)_u = y_u$
Transmit y_u to all neighbours $v \sim u$
Transmit y_v from all neighbours $v \sim u$
$(\bar{T}_1(\mathcal{R})y)_u = \sum_{v=\{v\sim u\}\cup u} \frac{1}{\phi}\mathcal{R}_{uv}y_v - y_u$
for $t = 2 : K$ **do**
  Transmit $(\bar{T}_{t-1}(\mathcal{R})y)_u$ to all neighbours $v \sim u$
  Receive $(\bar{T}_{t-1}(\mathcal{R})y)_v$ from all neighbors $v \sim u$
  $(\bar{T}_t(\mathcal{R})y)_u = \sum_{v=\{v\sim u\}\cup u} \frac{2}{\phi}(\bar{T}_{t-1}(\mathcal{R})y)_v - 2(\bar{T}_{t-1}(\mathcal{R})y)_u - (\bar{T}_{t-2}(\mathcal{R})y)_v$
end for
Return $f_u^{(K)} = \frac{1}{2}c_0 y_u + \sum_{t=1}^{K} c_t(\bar{T}_t(\mathcal{R})y)_u$

Then, in the vertex domain, we obtain the following K-term approximation of G-SSL

$$f^{(K)} = \frac{1}{2}c_0 y + \sum_{t=1}^{K} c_t \bar{T}_t(\mathcal{R})y \tag{4.17}$$

It can be seen from Eqs. (4.16) and (4.17) that to implement this approximation one needs to know λ_{max} and K. We recall λ_{max} can be efficiently estimated, for instance via the Lanczos method [130]. With respect to the choice of K there are not clear insights on how to choose it to attain a target approximation error. This is because it is hard to derive the asymptotic convergence rate of the series from its coefficient formula. Thus, in practice, the selection of K remains a design problem. We summarize the distributed implementation of G-SSL via Chebyshev polynomials in Algorithm 7.

4.3.2 Generalized implementation via Greens functions

In this subsection, we contribute with a novel polynomial expansion of G-SSL based on the Greens function that leads to an efficient and distributed algorithm.

Let us commence by identifying Eq. (4.12) as a discrete Helmholtz equation in which μ acts as the diffusion rate and the labels, y, take the role of an external forcing as an inhomogeneous term. Then, we retrieve this PDE Green's function by means of the inverse Laplace transform of the resolvent $(\mathcal{R} + \mu\mathbb{I})^{-1}$. Doing so allows us to express G-SSL in an integral form as

$$f = \int_0^\infty \mu e^{-\mu\tau} e^{-\tau\mathcal{R}}\ y\ d\tau. \tag{4.18}$$

For the family of G-SSL methods under consideration the spectrum of $\mathcal{R}$ is positive real granting convergence of the transformation. The only exception is the Re-centered Kernel method which may present negative eigenvalues and $\mu > -\lambda_0$ must be considered for convergence. Eq. (4.18) represents a novel approach to represent the G-SSL label propagation process: the integrand tracks the state of the system after the initial mass distribution y

has diffused during a (continuous) time τ via an exponentially distributed weighted heat kernel. Then, we can use this expression to obtain a polynomial representation of G-SSL by using the Taylor expansion as follows:

$$f = \int_0^\infty e^{-\tau} \left[\sum_{t=0}^{\infty} \frac{(-1)^t \tau^t \mathcal{R}^t}{\mu^t t!} \right] y \; d\tau \tag{4.19}$$

$$= \sum_{t=0}^{\infty} \frac{(-1)^t}{\mu^t t!} \left[\int_0^\infty \tau^t e^{-\tau} \; d\tau \right] \mathcal{R}^t y \tag{4.20}$$

$$= \sum_{t=0}^{\infty} \frac{(-1)^t}{\mu^t t!} \Gamma(t+1) \mathcal{R}^t y \tag{4.21}$$

$$= \sum_{t=0}^{\infty} \frac{(-1)^t}{\mu^t} \mathcal{R}^t y \tag{4.22}$$

We recognize in Eq. (4.19) an alternating sequence that monotonically converges to zero in the case of $\mu > \lambda_{\max}$. Hence, if $f^{(K)}$ denotes the approximation obtained by truncating the series to K terms, we have that, for $\mu > \lambda_{max}$, this representation is guaranteed to satisfy

$$\|f - f^{(K)}\| \leq \frac{1}{\mu^{K+1}} \|\mathcal{R}^{K+1}\| \|y\|. \tag{4.23}$$

In the case of $\mu \leq \lambda_{max}$, a truncation via Eq. (4.19) is not longer practical as it corresponds to a divergent sequence. However, we can still obtain a polynomial representation in such regime, although the procedure is more intricate. To clarify, let us give a step back to Eq. (4.18). In it, when $\tau \to \infty$, we have that $e^{-\tau \mathcal{R}/\mu} \to 0$. Thus, when we try to use a truncated Taylor expansion as an approximating function, the error grows unbounded as $\tau \to \infty$ since any alternating and non-monotonically convergent polynomial of fixed degree never goes to 0 at $t \to \infty$, instead they go to either $+\infty$ or $-\infty$. Nonetheless, Taylor's theorem does guarantee that a polynomial of K-terms will be able to approximate a function e^{-z} as long as this is restricted to disc of radius z_f centred at $z = 0$. Thus, this observation says that, on the regime $\mu < \lambda_{max}$, with a finite number of terms we can still approximate our label propagation process as long as it is up to a maximal time τ_f. Then, for this regime, we consider the following approximation of G-SSL

$$f^{(K)} = \int_0^{\tau_f} e^{-\tau} \left[\sum_{t=0}^{K} \frac{(-1)^t \tau^t \mathcal{R}^t}{\mu t!} \right] y \; d\tau \tag{4.24}$$

$$= \sum_{t=0}^{K} \frac{(-1)^t}{\mu^t t!} \left[\int_0^{\tau_f} \tau^t e^{-\tau} \; d\tau \right] \mathcal{R}^t y \tag{4.25}$$

$$= \sum_{t=0}^{K} \frac{(-1)^t}{\mu^t t!} \gamma(t+1, \tau_f) \mathcal{R}^t y, \tag{4.26}$$

where $\gamma(\cdot, \cdot)$ refers to the incomplete gamma function. Clearly, the more time we run our diffusion process, the better the accuracy of our approximation, but at the cost of more terms needed in the polynomial. To address the question of given a number of terms K, what is maximum disc radius z_f so that we can approximate e^{-z_f} with an error at most ϵ, let $T_K(z) = \sum_{t=0}^{K} (-1)^k z^k / k!$ and $R_K(z) = e^{-z} - T_K(z)$. The approximation error is bounded by $|R_K(z)| \leq z^{K+1}/(K+1)!$, thus we can estimate z_f by considering the equation, $z^{K+1}/(K+1)! = \epsilon$, and solving it for z. The solution can be derived as follows:

Algorithm 8 Distributed computation of G-SSL via Greens functions

Input at node u: $y_u, \mu, \lambda_{max}, K, \epsilon$, and $R_{uv}\ \forall\ \{v \sim u\} \cup u$.
Output node u: $f_u^{(K)}$
$x_u^{(0)} = y_u$
if $\mu > \lambda_{max}$ **then**
  $f_u^{(0)} = x_u^{(0)}$
  for $t = 1 : K$ **do**
    transmit $x_u^{(t-1)}$ to all neighbors $v \sim u$
    receive $x_v^{(t-1)}$ from all neighbors $v \sim u$
    $x_u^{(t)} = -\frac{1}{\mu} \sum_{v \in \{v \sim u\} \cup u} R_{uv} x_v^{(t-1)}$
    $f_u^{(t)} = f_u^{(t-1)} + x_u^{(t)}$
  end for
else
  compute τ_f via Eq. (4.27)
  $f_u^{(0)} = \gamma(1, \tau_f) x_u^{(0)}$
  for $t = 1 : K$ **do**
    transmit $x_u^{(t-1)}$ to all neighbors $v \sim u$
    receive $x_v^{(t-1)}$ from all neighbors $v \sim u$
    $x_u^{(t)} = -\frac{1}{\mu t} \sum_{v \in \{v \sim u\} \cup u} R_{uv} x_v^{(t-1)}$
    $f_u^{(t)} = f_u^{(t-1)} + \gamma(t+1, \tau_f) x_u^{(t)}$
  end for
end if
Return $f_u^{(t)}$

$\log(\epsilon) = (K+1)\log(K) - \log((K+1)!) \approx (K+1)\log(z) - (K+1)\left(\log(K+1) - 1\right)$ using Stirling's approximation. To estimate τ_f from z_f, we note that $z_f = \tau_f \lambda_{max}/\mu$, hence leading to the result that for K terms we can run our diffusion process until

$$\tau_f = \frac{\mu}{\lambda_{max}} \exp\left\{\frac{(K+1)(\log(K+1) - 1) + \log(\epsilon)}{K+1}\right\}. \tag{4.27}$$

The distributed implementation of this algorithm is summarized in Algorithm 8.

4.3.3 Generalized implementation via ARMA recursions

In this subsection, we contribute with an extension of the Algorithms from Section 4.2) by using the ARMA filters. The ARMA graph filters, introduced in [22], are a type of filter characterized for being able to implement filter functions with rational frequency response. They are defined by the following recursive formula:

$$p^{(t)} = \rho y + \psi \mathcal{R} p^{(t-1)}. \tag{4.28}$$

In [22], it is shown that when this recursive formula converges, then it defines a filter in the spectral domain of $\mathcal{R}$ with input y:

Theorem 6 ([22]). *Let $\mathcal{R}$ be a symmetric operator and λ_{max} denote its spectral radius bound. Further, let $r = -\frac{\rho}{\psi}$ be a residue and $p = \frac{1}{\psi}$ denote a pole of a rational transfer*

function. Then, the ARMA recursion in Eq. (4.28) converges linearly to the following frequency response on the spectrum of $\mathcal{R}$*:*

$$h(\Lambda) = \frac{r}{\Lambda - p}; \quad \textit{subject to} \quad |p| > \lambda_{max} \tag{4.29}$$

Let us start clarifying that the symmetry assumption on $\mathcal{R}$ used in Theorem 6 is to ensure that $\mathcal{R}$ has real spectrum. Thus, Theorem 6 still holds for our non-symmetric operators $\mathcal{R}$ as long as they possess real spectrum. Having made this clarification, we now observe that, by direct analogy with Eq. (4.13), Theorem 6 can be readily used to implement the frequency response of G-SSL by choosing $r = \mu$ and $p = -\mu$ and, accordingly, $\psi = -\frac{1}{\mu}$ and $\rho = 1$. Hence, leading to a recursive implementation of G-SSL given as

$$p^{(t)} = y + \frac{1}{\mu}\mathcal{R}p^{(t-1)}. \tag{4.30}$$

However, an issue with Eq. (4.30) is that its stability region, determined by the condition $|p| > \lambda_{max}$, implies that the recursion only converges if $|\mu| > \lambda_{max}$, which is too penalizing for G-SSL. For this reason, we now revert to a shifting of the domain of $\mathcal{R}$ (as done by the Chebyshev method) and show that by doing this transformation, then the stability region of G-SSL can be extended to all $\mu > 0$.

More precisely, let us consider the mapping $\Lambda \rightarrow \frac{\lambda_{max}}{2}(s+1)$, where $s \in [-1, 1]$, that shifts the spectrum of the operator $\mathcal{R}$. Then, we shift the filter to this new domain by applying this transformation to the G-SSL transfer function:

$$h(\Lambda) = \frac{\mu}{\Lambda + \mu} = \frac{\mu}{(\lambda_{max}/2)(s+1) + \mu} = \frac{2\mu/\lambda_{max}}{s + 1 + (2\mu/\lambda_{max})} = \frac{r}{s - p} = h(s) \tag{4.31}$$

where $r = 2\mu/\lambda_{max}$ is the new residue, $p = -[(2\mu/\lambda_{max}) + 1]$ is a new pole, and s refers to the spectrum of the shifted operator $\mathcal{S} = (2/\lambda_{max})\mathcal{R} - \mathbb{I}$. Now, for these values of p and r we obtain the following ARMA coefficients: $\psi = -(\lambda_{max})/(2\mu + \lambda_{max})$ and $\rho = (2\mu)/(2\mu + \lambda_{max})$. Thus, leading to the following ARMA implementation of G-SSL:

$$p^{(t)} = \left(\frac{2\mu}{2\mu + \lambda_{max}}\right) y - \left(\frac{\lambda_{max}}{2\mu + \lambda_{max}}\right) \mathcal{S}p^{(t-1)}. \tag{4.32}$$

To see that this recursion now converges for all $\mu > 0$, we have that the spectrum of $\mathcal{S}$ is bounded between [-1,1], leading to the stability constraint: $|\lambda_{max}/(2\mu + \lambda_{max})| < 1$, which holds for all $\mu > 0$. As a result, Eq. (4.32) corresponds to an extension of the power method to the general family of G-SSL algorithms. We detail its distributed implementation in Algorithm 9.

Let us now determine the asymptotic convergence rate of Algorithm 9 and show that, remarkably, using it to compute the standard PageRank algorithm implies faster convergence than the classical power method (see Algorithm 3 from Section 4.2).

Lemma 13. *Let* $arma_{\rho,\psi}(y)$ *denote the steady state of Eq. (4.32). Then, the output of Algorithm 9 satisfies*

$$\left\| arma_{\rho,\psi}(y) - p^{(K)} \right\| \leq \left| \frac{\lambda_{max}}{2\mu + \lambda_{max}} \right|^K \|\mathcal{S}^K\| \, \| arma_{\rho,\psi}(y) - p^{(0)} \| \tag{4.33}$$

Algorithm 9 Distributed computation of G-SSL via the Power method

Input at node u: y_u, μ, $\mathcal{R}_{uv}$ $\forall$ $\{v \sim u\} \cup u$, K, $p^{(0)}$ and λ_{max}.
Output at node u: $p_u^{(K)}$
$\rho = (2\mu)/(2\mu + \lambda_{max})$
$\psi = -(\lambda_{max})/(2\mu + \lambda_{max})$
$\phi = 2/\lambda_{max}$
for $t = 1 : K$ **do**
  Transmit $p_u^{(t-1)}$ to all neighbors $v \sim u$
  Receive $p_v^{(t-1)}$ from all neighbors $v \sim u$
  $p_u^{(t)} = \rho y_u - \psi p_u^{(t-1)} + \psi\phi \sum_{v=\{v\sim u\}\cup u} \mathcal{R}_{uv} p_v^{(t-1)}$
end for
Return $p^{(K)}$

The proof of this Lemma is deferred to Appendix 4.A

We have that the spectrum of $\mathcal{S}$ lies within the unit circle, implying that when K grows, $\mathcal{S}^K$ converges to a rank one matrix. As a result, $\|S^K\|$ converges to a constant value that has a negligible role on the asymptotic convergence rate, making this determined by (i) $\left|\frac{\lambda_{max}}{2\mu+\lambda_{max}}\right|^K \to 0$; and (ii) how accurate is the initial guess. Now, we compare Algorithm 9 with Algorithm 3 for computation of standard PageRank. We have that the latter has a convergence rate determined by $\alpha^K = \left(\frac{1}{\mu+1}\right)^K \to 0$, while the one of the former can be recast as $\left(\frac{\lambda_{max}/2}{\mu+\lambda_{max}/2}\right)^K \to 0$. This implies that Algorithm 9 converges faster if $\lambda_{max}/2 < 1$, which is always true as standard PageRank is driven by the *random walk* Laplacian $\mathcal{R} = LD^{-1}$, which, as mentioned in Chapter 1, it always has a λ_{max} smaller than 2 (unless the graph is bipartite).

4.3.4 Generalized implementation via Gauss-Southwell method

In this subsection, we leverage the ARMA recursion from Eq. (4.32) to contribute with an extension of the Gauss-Southwell algorithm to the general family of G-SSL methods. In precise terms, by analogy with standard Gauss-Southwell method (see Algorithm 4 in Section 4.2), we propose to have two vectors: an approximate G-SSL vector p and its residual vector r, so that, at each iteration t, the algorithm satisfies the following invariant:

$$\text{arma}_{\rho,\psi}(y) = p^{(t)} + \frac{1}{\rho}\text{arma}_{\rho,\psi}(r^{(t)}) \tag{4.34}$$

Hence, the main goal of our proposition remains to drive r towards zero as that implies that p converges to the G-SSL solution. In the following, we extrapolate the update equations from the standard Gauss-Southwell to the ARMA case and show that they preserve the invariant (4.34).

Lemma 14. *Assume that, at iteration t, the largest entry of $|r^{(t)}|$ corresponds to vertex u.*

Algorithm 10 Approximate G-SSL computation via Gauss-Southwell method

Input: μ, $\mathcal{R}$, $p^{(0)}$, ϵ, and λ_{max}
Output: Approximate G-SSL solution p and its residual r
$\rho = (2\mu)/(2\mu + \lambda_{max})$
$\psi = -(\lambda_{max})/(2\mu + \lambda_{max})$
$\phi = 2/\lambda_{max}$
$r^{(0)} = \rho y - (\mathbb{I} - \psi\mathcal{S})\, p^{(0)}$
t = 0
while $\|r^{(t)}\|_\infty > \epsilon$ **do**
  $u = \arg\max_u |(r^{(t)})_u|$
  $p^{(t+1)} = p^{(t)} + (r^{(t)})_u \delta_u$
  $r^{(t+1)} = r^{(t)} - (1+\psi)(r^{(t)})_u \delta_u + \psi\phi\mathcal{R}(r^{(t)})_u \delta_u$
  $t++$
end while
Return $p = p^{(t)}$ and $r = r^{(t)}$

Then, the following set of update equations satisfy, for all t, the invariant of Eq. (4.34):

$$p^{(t+1)} = p^{(t)} + r_u^{(t)} \delta_u \tag{4.35}$$

$$r^{(t+1)} = r^{(t)} - r_u^{(t)} \delta_u + \psi\mathcal{S} r_u^{(t)} \delta_u \tag{4.36}$$

where $r^{(0)} = \rho y - (\mathbb{I} - \psi\mathcal{S}) p^{(0)}$

The proof of this Lemma is deferred to Appendix 4.B.

Algorithm 10 summarizes our extension of the Gauss-Southwell algorithm to general family of G-SSL methods. Now, we stress that the general nature of $\mathcal{R}$ and $\text{arma}_{\rho,\psi}(y)$, which can code for a signed graph and a vector of negative entries, respectively, make notoriously hard to theoretically bound the running time of Algorithm 10. Indeed, it is not even straightforward to guarantee that $\|r^{(t+1)}\|_\infty \leq \|r^{(t)}\|_\infty$ as, in theory, $r_u^{(t)}$ may be negative and the vector $\psi\mathcal{R} r_u^{(t)} \delta_u$ may have positive and negative entries. We stress that despite this theoretical challenge, in practice, we have experienced that the update equations satisfy $\|r^{(t+1)}\|_\infty \leq \|r^{(t)}\|_\infty$ for all t, implying that the algorithm always terminates. We leave the formal analysis on the time complexity of Algorithm 10 as an open problem.

4.3.5 Numerical assessment

In this subsection, we assess the approximation algorithms presented above. We present two experiments: in the first, we compare our ARMA-based methods (Algorithm 9 and 10) versus their random walk-based counterparts (Algorithms 3 and 4) by noting that standard PageRank arises as a particular case of our generalized approaches; in our second experiment, by using our L^γ-PageRank as base method, we compare the approximation quality of Algorithms 7 - 10 and assess the sensitivity of these algorithms to variations in μ and γ. To assess the quality of the approximations, we employ the following error metrics:

- **ℓ_2-norm error**: standard metric to compute the relative distance between two vectors.

- **Ranking error**: metric that counts the fraction of nodes that do not rank in the same spot in the true G-SSL solution and in the approximation vector. The metric is computed by first sorting the entries from both the true G-SSL solution and the approximation vector, so as to obtain the relative ranking of nodes in the graph, and then by counting the fraction of nodes that rank differently in both vectors. This metric motivates from the observation that G-SSL essentially amounts to compare how one node ranks with respect to the rest, meaning that the output of an approximation algorithm may be far from the ground truth in the ℓ_2-norm sense, while still preserving the same relative ranking of nodes, thus constituting a reliable approximation.

- **Sweep error**: metric that assesses to what extend a partition obtained by doing a sweep on the ground truth G-SSL solution differs from the one obtained by doing a sweep on the approximation vector. For the metric, the similarity between partitions is assessed by MCC.

- **Multi-class error**: metric that assesses to what extend a classification obtained on a set of ground truth G-SSL solutions (multi-class approach) differs from the classification obtained deciding on approximation vectors output by the algorithms above. For the metric, the similarity between classifications is assessed by MCC.

- **Number of iterations (Gauss-Southwell only)**: strictly speaking, this is not an error metric but it complements the error metrics when the Gauss-Southwell method is employed. This is because, for the Gauss-Southwell, a parameter ϵ is set so that the algorithm stops when $\|r\|_\infty \leq \epsilon$. Therefore, it is important to: (i) assess the influence that μ, ϵ and γ have on the number of iterations taken by the algorithm to finish; and (ii) verify, to what extend, Algorithm 4 and Algorithm 10 vary in their running times as they involve different update equations.

ARMA vs random walks for standard PageRank computation

Experimental setup and goals. In this experiment, we assess the benefits the ARMA-based extensions of the Power iteration and Gauss Southwell (Algorithms 9 and 10, respectively) over their random walk-based counterparts (Algorithm 3 and 4, respectively). As implied by our Lemma 13, for standard PageRank computation, the ARMA-based version of the power method should imply an improvement in convergence rate over the random walk-based approach. Therefore, we aim to quantify if such improvements are significant.

The experimental setup is as follows: we generate data from the planted partition model with parameters $C_{avg} = 20$, $C_{in} = 18$, $C_{out} = 2$, $N = 1000$. Then, 1% of labelled points is randomly sampled. The ground truth classification function is computed through MATLAB backslash operator and it consists of the PageRank with parameter $\mu = 0.1$ ($\alpha \approx 0.9$). Then, we apply Algorithm 9 (ARMA-based power iteration) and Algorithm 3 (Walk-based power iteration) under the same complexity (K) and report the error in their approximations using the metrics defined above. The whole procedure is repeated for 20 data realizations. In addition, this same procedure is employed to compare Algorithm 10 (ARMA-based Gauss-Southwell) and Algorithm 4 (Walk-based Gauss-Southwell). For these algorithms, the approximation errors are compared under a common input error tolerance ϵ.

Results and discussion.
Figure 4.3: comparison of Algorithm 3 and Algorithm 9. As implied by Lemma 13, this figure displays that Algorithm 9 has a faster convergence rate than Algorithm 3. After $K = 100$ iterations, Algorithm 9 shows improvements of roughly 2 orders of magnitude over Algorithm 3, which translates into an improved ranking of nodes (second plot from left to right). Surprisingly, this faster convergence rate and improved node ranking do not reflect as improvements in the sweep error or the multi-class error. Indeed, the last two figures on the right show that, for small K, a sweep on the random walk-based approximation implies a slightly smaller error over doing a sweep on the ARMA-based approximation. Despite these minor differences, it is important to stress the G-SSL decision rules display robustness to inaccuracies on the classification function as, already from small K values, the sweep error and the multi-class error are small.

Figure 4.4: comparison of Algorithm 4 and Algorithm 10. In this figure, we observe that the ARMA-based Gauss-Southwell method also displays an improvement over the random walk-based version of the algorithm. In this case, the benefits are more evident after $\epsilon < 10^{-3}$, as, for this regime, Algorithm 10 terminates in less iterations than Algorithm 4 while attaining less ℓ_2-norm error and more accurate rankings. Remarkably, the sweep error is already very small from rather large values of ϵ. This may be due to the simple graph model that we are considering, where there is only one clear partition of small Cheeger ratio, hence it is should be doable for the sweep to find this true partition despite the inaccurate functions. However, the multi-class error displays high values for large values of ϵ and only after $\epsilon < 10^{-3}$ the true partition is always obtained.

Comparison of methods for L^γ-PageRank and influence of parameters

Experimental setup and goals. In our next experiment, by using L^γ-PageRank as the base method, our goal is to compare the approximation accuracy of Algorithms 7 - 10 and to assess the impact that changes in μ and γ have on these algorithms, so as to determine the best approach to implement our L^γ-PageRank.

The experimental setup is as follows: we replicate the setup of the previous experiment. To define our ground truth classification functions, we consider L^γ-PageRank for the following parameter combinations: (i) $\gamma = 1, \mu = 5$; (ii) $\gamma = 1, \mu = 0.1$; (iii) $\gamma = 5, \mu = 5$; (iv) $\gamma = 5, \mu = 0.1$. We stress that these values have been carefully chosen: for setting (i) we have that $\mu = 5 > \lambda_{max}$, thus implying that the Greens function method (Algorithm 8) operates in the region where it is guaranteed to converge; setting (ii) allows us to compare with the previous experiment; setting (iii) allows us to see the impact of increasing γ (w.r.t. (i)), moreover, $\gamma = 5$ is a common value where the planted partition starts to display a saturation plateau on its Cheeger ratio, hence it is related to the optimal γ. Lastly, setting (iv) allows to see the impact of either decreasing μ (w.r.t (iii)), or to increase γ (w.r.t (ii)).

Results and discussion.
Figure 4.5: comparison of distributed algorithms: (i) Chebyshev (Algorithm 7); (ii) Greens function (Algorithm 8); and (iii) ARMA-recursions (Algorithm 9). In the figure array, rows correspond to error metrics and columns correspond to a combination of (γ, μ). It is clear from the figure that the Chebyshev approach outperforms the other methods by several orders of magnitude. Indeed, in every test, the Chebyshev approach is able min-

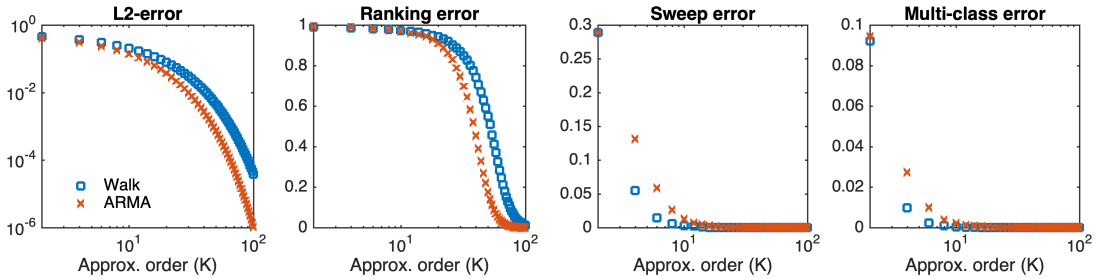


Figure 4.3: Comparison of Algorithm 3 (Random walk-based power iteration) vs Algorithm 9 (ARMA-based power iteration).

imize the ranking error, sweep error and multi-class error in a maximum of roughly 30 iterations. This remarkable performance seems to be a consequence of its much lower sensitivity to changes in γ and μ than the other approximation methods. Precisely, let us consider $\gamma = 1, \mu = 5$ as a baseline setting, since all methods tend to converge to the true G-SSL solution in only a few iterations for this setting. Then, by increasing γ (third column), we see that the Greens function method severely decreases performance. Even though the ARMA recursion remains relatively insensitive to changes in γ, when the μ parameter decreases it drastically worsens performance (second column). On the contrary, the Chebyshev approach cases remains performing remarkably well despite the parameter variations.

Figure 4.6: sensitivity of Algorithm 10 to μ and γ. In this figure, we observe that the Gauss-Southwell, as its based on the ARMA recursion, is also sensitive to variations in γ, μ. Namely, as we increase γ, we increase the number of iterations and the ℓ_2 error. Then, if μ is further decreased these quantities increase even more. Interestingly, the figures from the sweep error and multi-class error suggest that the introduction of γ implies to more quickly converge towards the true partitions.

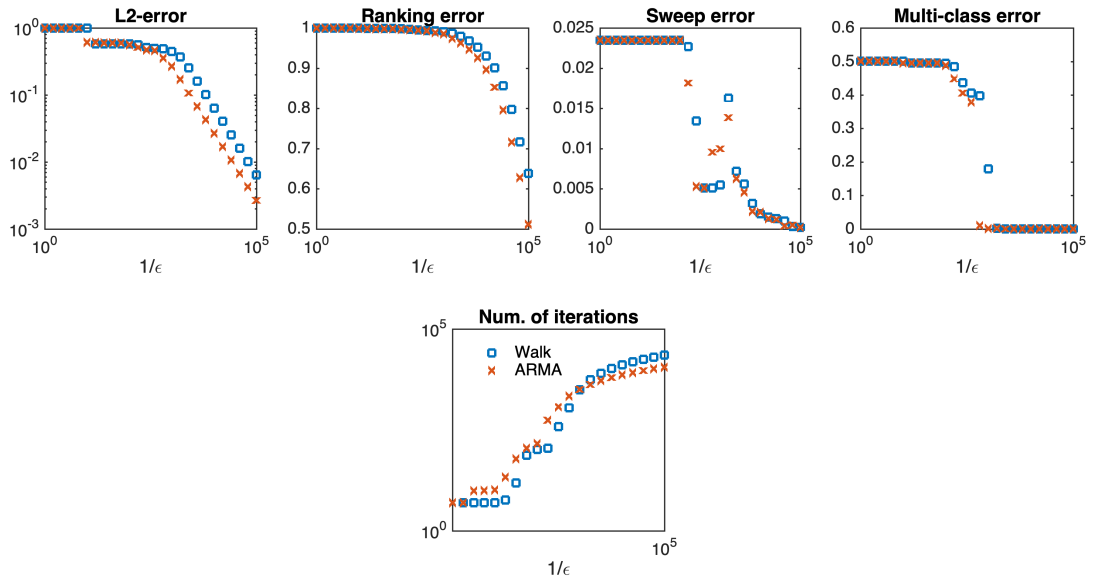
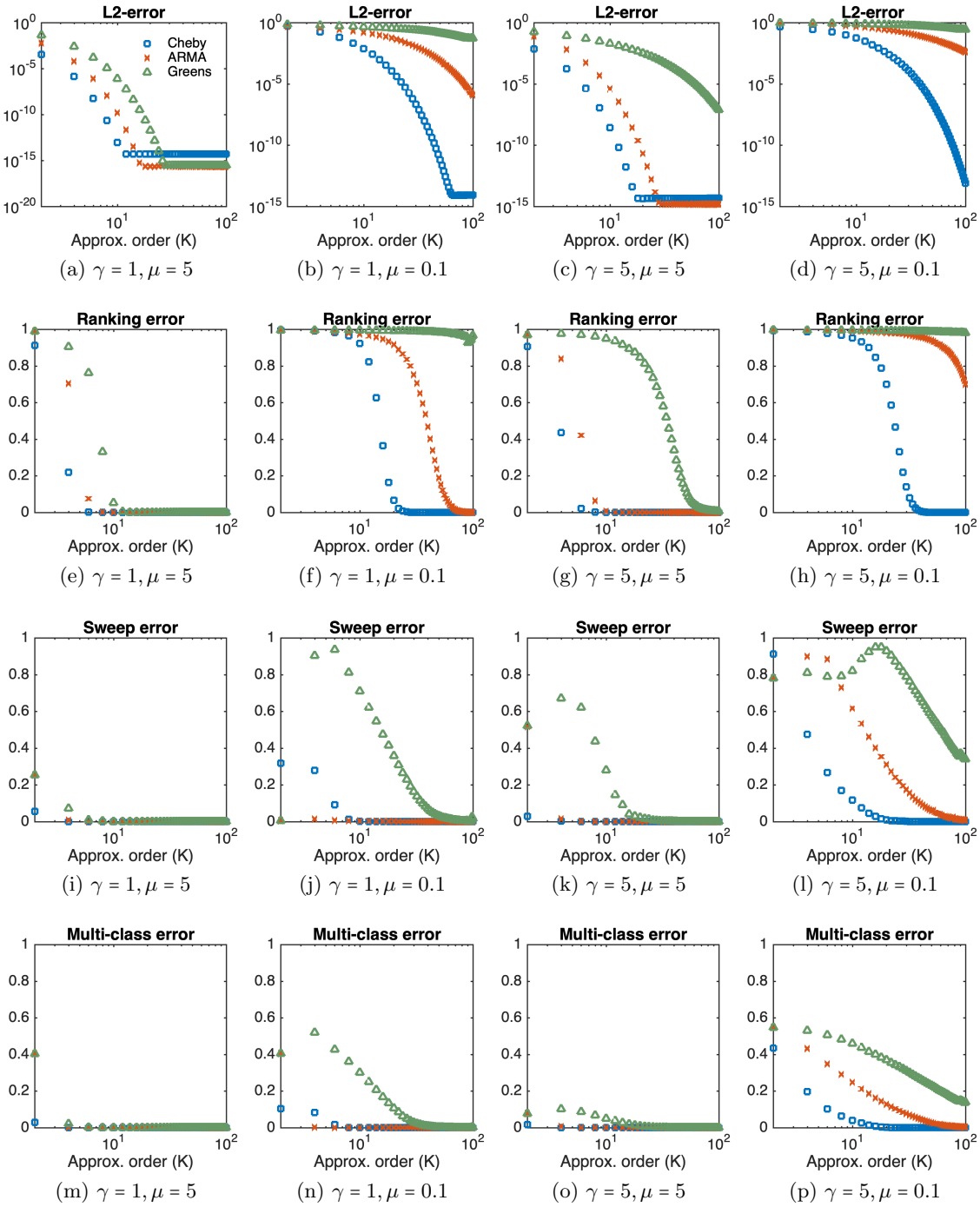


Figure 4.4: Comparison of Algorithm 4 (Random walk-based Gauss-Southwell) vs Algorithm 10 (ARMA-based Gauss-Southwell)



(a) $\gamma = 1, \mu = 5$ (b) $\gamma = 1, \mu = 0.1$ (c) $\gamma = 5, \mu = 5$ (d) $\gamma = 5, \mu = 0.1$

(e) $\gamma = 1, \mu = 5$ (f) $\gamma = 1, \mu = 0.1$ (g) $\gamma = 5, \mu = 5$ (h) $\gamma = 5, \mu = 0.1$

(i) $\gamma = 1, \mu = 5$ (j) $\gamma = 1, \mu = 0.1$ (k) $\gamma = 5, \mu = 5$ (l) $\gamma = 5, \mu = 0.1$

(m) $\gamma = 1, \mu = 5$ (n) $\gamma = 1, \mu = 0.1$ (o) $\gamma = 5, \mu = 5$ (p) $\gamma = 5, \mu = 0.1$

Figure 4.5: Comparison of distributed algorithms: (i) Chebyshev (Algorithm 7); (ii) Greens function (Algorithm 8); and (iii) ARMA-recursions (Algorithm 9). In the figure array, rows correspond to error metrics and columns correspond to a combination of (γ, μ).

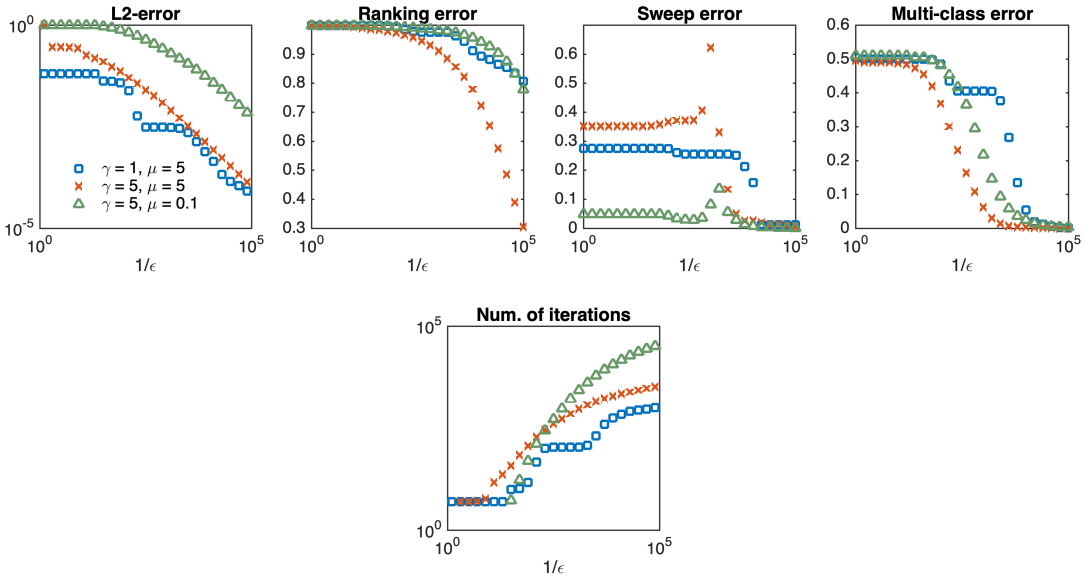


Figure 4.6: Sensitivity of the ARMA-based Gauss-Southwell method (Algorithm 10) to μ and γ

4.4 Fast updating of G-SSL on evolving networks

In this section, we address the updating problem for the general G-SSL case. From the experiments of the last section, it may seem that the value of our ARMA-based extensions is limited when they are compared to the Chebyshev approach. However, one critical feature that the ARMA-based algorithms possess that is missing on the Chebyshev method is the possibility to employ warm restarts. Let us show in Figure 4.7 the effect of using a warm restart on the ARMA recursion versus using the Chebyshev method (from scratch) when one node joins the graph. For the plot, we employ one of the planted partition graphs from the experiments in Section 4.3.5 and use L^2-PageRank with $\mu = 0.01$ as the base G-SSL method. As it can be seen, due to the very accurate initial guess, the ARMA recursion now outperforms the Chebyshev method by several orders of magnitude. Moreover, as we will show in this section, the ARMA-based methods allows to cast the update problem in a local manner, thus permitting sub-linear time algorithms, contrary to the Chebyshev method which, even though it converges extremely fast, remains a global method that needs to see all the graph to operate.

4.4.1 Local G-SSL updating via the power method

In this subsection, we use the ARMA recursion from Eq. (4.32) to extend the local update method of Algorithm 5 to the general family of G-SSL methods. To proceed, let us denote the steady state of Eq. (4.32) for the initial graph $\mathcal{G}$ by $\text{arma}_{\rho,\psi}(y)$ and for the evolved graph $\widetilde{\mathcal{G}}$ by $\widetilde{\text{arma}}_{\rho,\psi}(y)$. We assume that the initial condition y and the set of ARMA coefficients ρ, ψ are the same for both $\mathcal{G}$ and $\widetilde{\mathcal{G}}$. Thus, we leverage the warm restarts $\widetilde{p}^{(0)} = \text{arma}_{\rho,\psi}(y)$ to show the following:

Lemma 15. *If $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ share the same set of ARMA coefficients (ρ, ψ) and initial*

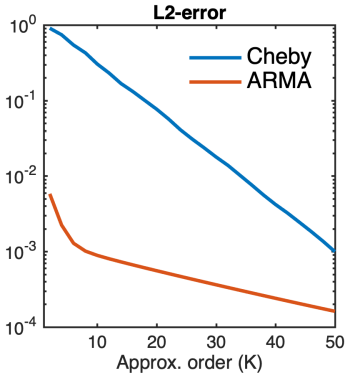


Figure 4.7: Effect of using a warm restart on the ARMA recursion versus computing from scratch via the Chebyshev approach when one node joins the graph.

condition y, then we have that

$$\widetilde{arma}_{\rho,\psi}(y) = arma_{\rho,\psi}(y) + \frac{1}{\rho}\widetilde{arma}_{\rho,\psi}(r) \tag{4.37}$$

where

$$r = \psi\left[\widetilde{\mathcal{S}} - \mathcal{S}\right] arma_{\rho,\psi}(y) \tag{4.38}$$

The proof of this Lemma is deferred to Appendix 4.C.

This Lemma generalizes the local update rule of Eq. (4.8) to the general G-SSL setting. Namely, it says that to update the G-SSL solution, then one needs to solve another G-SSL problem, but that this new one is much more efficient to compute since the initial distribution r is local in the perturbation. Hence, by computing $\widetilde{\text{arma}}_{\rho,\psi}(r)$ via the power method, we can control the scope of the update according to the number of terms used in the approximation. Namely, if one uses K-terms, then one propagates the update in a K-hop vicinity, leading to a method with cost bounded by $\mathcal{O}(cKM)$ if, at most, there are M edges are contained in the K-hop vicinity of a node that changed, and c changes occur. We detail our proposed local update procedure in Algorithm 11.

We now show that the convergence rate of Algorithm 11 is the same of Algorithm 9 and that the cost of the algorithm is proportional to the update needed.

Lemma 16. *Let K-terms be used in the estimation of $\widetilde{arma}_{\rho,\psi}(r)$ in Algorithm 11 and $\widetilde{p}^{(K)}$ denote the output of Algorithm 11 under such truncation. Then, the output of Algorithm 11 satisfies*

$$\|\widetilde{arma}_{\rho,\psi}(y) - \widetilde{p}^{(K)}\| \leq |\psi^{K+1}|\|\widetilde{\mathcal{S}}^{K+1}\|\|\frac{1}{\rho}\widetilde{arma}_{\rho,\psi}(r)\| \tag{4.39}$$

The proof of this Lemma is deferred to Appendix 4.D.

The implications of this lemma are twofold. On the one hand, it points in the same direction as Lemma 12, saying that the norm of the update is bounded by the norm of the perturbation (encoded by $\|\frac{1}{\rho}\widetilde{\text{arma}}_{\rho,\psi}(r)\|$). On the other hand, since $\|\frac{1}{\rho}\widetilde{\text{arma}}_{\rho,\psi}(r)\|$

Algorithm 11 Local G-SSL updating via power iteration

Input: μ, $\text{arma}_{\rho,\psi}(y)$, $\mathcal{R}$, $\widetilde{\mathcal{R}}$ and the same ρ, ψ, ϕ used to compute $\text{arma}_{\rho,\psi}(y)$.
Output: $\widetilde{\text{pr}}_\alpha(y)$.

$r = \psi\phi[\widetilde{\mathcal{R}} - \mathcal{R}]\text{arma}_{\rho,\psi}(y)$
Compute $\widetilde{\text{arma}}_{\rho,\psi}(r)$ via Algorithm 9
Return $\widetilde{\text{arma}}_{\rho,\psi}(y) = \text{arma}_{\rho,\psi}(y) + \widetilde{\text{arma}}_{\rho,\psi}(r)/\rho$

Algorithm 12 Local G-SSL updating via Gauss-Southwell

Input: p, r, $\mathcal{R}$, $\widetilde{\mathcal{R}}$ and the same ρ, ψ, ϕ used to compute $\text{arma}_{\rho,\psi}(y)$.
Output: Updated G-SSL solution $\widetilde{p}$ and updated residual $\widetilde{r}$.

$\widetilde{p}^{(0)} = p$
$\widetilde{r}^{(0)} = r + \psi[\widetilde{\mathcal{S}} - \mathcal{S}]p$
Apply Gauss-Southwell on $\widetilde{p}^{(0)}$ and $\widetilde{r}^{(0)}$, store result in $\widetilde{p}$ and $\widetilde{r}$
Return $\widetilde{p}$ and $\widetilde{r}$

denotes the update, the Lemma implies that the convergence rate of the algorithm is determined by (i) the size of the update; and (ii) the rate at which $\psi^{K+1} = |\frac{\lambda_{max}}{2\mu+\lambda_{max}}|^{K+1} \to 0$, which is the same of Algorithm 11.

4.4.2 Local G-SSL updating via Gauss-Southwell

We now finalize our extension of algorithms by revamping the Gauss-Southwell method to perform local updates to the G-SSL solution. In analogy with Algorithm 6, we embed the local update property by means of the warm restarts. Namely, let p and r denote the approximate G-SSL solution and residual vector, respectively, output by the Gauss-Southwell algorithm for graph $\mathcal{G}$. Then, under the assumption that the initial condition y and the set of ARMA coefficients ρ, ψ do not change, we use p as the initial guess $\widetilde{p}^{(0)}$ when running the Gauss-Southwell for $\widetilde{\mathcal{G}}$. In math terms, we use the following initial state of the algorithm

$$\widetilde{p}^{(0)} = p \tag{4.40}$$

and

$$\begin{aligned}
\widetilde{r}^{(0)} &= \rho y - (\mathbb{I} - \psi\widetilde{\mathcal{S}})\widetilde{p}^{(0)} && (4.41)\\
&= \rho y - (\mathbb{I} - \psi\widetilde{\mathcal{S}})p && (4.42)\\
&= (\mathbb{I} - \psi\mathcal{S})p + r - (\mathbb{I} - \psi\widetilde{S})p && (4.43)\\
&= r + \psi(\widetilde{\mathcal{S}} - \mathcal{S})p. && (4.44)
\end{aligned}$$

We identify this residual to be local on the perturbations, thus implying that only a few entries of r can be triggered beyond the threshold. As a result, with only a few iterations the Gauss-Southwell will drive these entries below the threshold again. We summarize the procedure for local G-SSL updating via the Gauss-Southwell method in Algorithm 12.

4.4.3 Updating via neural networks

In this subsection, we present a novel and promising approach to solve the update problem which consists on using neural networks. The idea we propose to investigate is the following: if we pay the price of computing the true G-SSL solution for some sequence of graphs, can we feed into a neural network the previous G-SSL solution, the perturbation, and the evolved G-SSL solution (from our training set), so that the network can learn the necessary mapping to update the G-SSL solution for future graphs not seen during training?

The Multi-Layer Perceptron

Neural networks are computational models that draw inspiration from biological brain networks. In a neural network, individual processing units, called neurons, receive information from other neurons, process that information, and output information that is used by other neurons to repeat this task. There are several variants of neural networks, but one of the most popular is the multilayer perceptron (MLP). In this model, neurons are organized in layers, and there are always, at least, three of them. The input layer, the hidden layer(s), and the output layer. Neurons from the input layer send information to the ones in the hidden layer, these to the successive hidden layers, until the information reaches the output layer. Each neuron from a layer is connected to each neuron in the next layer and each connection possess a real-valued coefficient reflecting the importance if this connection. Under certain assumptions, this type of models are known to be universal approximators, meaning that they are bound to realize any mapping between vector spaces. Thus, if a mapping from $\mathbb{R}^l$ to $\mathbb{R}^m$ is desired, then the input layer must have l neurons and the output layer m neurons. The hidden layers can have distinct number of neurons and their number is a design choice.

If we let the information output from each of the neurons in layer k be encoded by the vector $h^{(k)}$, the value of the MLP at layer k is described by the equation

$$h^{(k)} = \sigma(W^{(k)}h^{(k-1)} + \theta^{(k)}), \tag{4.45}$$

where $\sigma(\cdot)$ is a non-linear activation function, $W^{(k)} \in \mathbb{R}^{j_{k-1},j_k}$, with j_k being the number of neurons in layer k, is a weight matrix encoding the strength of connections between layers $k-1$ and k, and $\theta^{(k)}$ is an offset parameter. Popular choices of $\sigma(\cdot)$ are: (i) sigmoid, (ii) softmax, (iii) tanh, or (iv) ReLU. Each one of them has strengths and drawbacks and their selection is a design choice. For a more thorough discussion on activation functions we refer the reader to [131].

Neural networks are trained to learn. In math terms, training means adjusting the values of the weight matrices within layers so that we fit the transformation for a subset of datapoints for which we know their value in $\mathbb{R}^l$ and corresponding image in $\mathbb{R}^m$. Thus, for the pair $(x \in \mathbb{R}^l, z \in \mathbb{R}^m)$, x is fed into the network which outputs h_{out} (the output layer) as an estimate of z. Then, a loss function is employed to assess how much the network fits the training data and consequently adjust the weights of the network. The most common approach to adjust the weights is by means of the back-propagation algorithm. In it, the

weight between neurons i and j is updated according to the rule

$$w_{ij}^{(new)} = w_{ij}^{(old)} - \zeta \left(\frac{\partial Loss}{\partial w_{ij}} \right), \tag{4.46}$$

where ζ is the learning rate. By using the chain rule, Eq. (4.46) can be expressed in terms of the layers between the output layer and w_{ij} and the update 'back-propagated', hence the name.

As mentioned above, MLPs are used to learn mapping between euclidean vector spaces. However, numerous research efforts have been recently made to leverage the power of neural networks to solve graph problems [132, 133, 134, 135]. Yet, due to its irregular nature, graph data poses a challenge for neural networks. The traditional approach to adapt neural networks to graph data consists on creating a latent space for the vertices by assigning them a feature vector, thus 'embedding' the graph vertices into an euclidean representation. Moreover, a popular approach consists on using graph convolutional layers, which basically consist on using graph filters in between regular layers of the MLP (see [132] for details) so as to use the graph structure as a guide to perform linear combinations of the euclidean representation of the data.

The MLP for the updating problem

In this work, we consider MLP's for the following problem: let $\{\mathcal{G}_t : t = 1, 2, \ldots, t_{max}\}$ denote a sequence of graphs. Let $f_t = h(\mathcal{G}_t, y_t)$ be a transformation applied on the graph signal $y = y_1 = y_2 = \cdots = y_{t_{max}}$ in graph $\mathcal{G}_t$. The transformation is the same for all the graphs in the sequence. Now, if the changes between $\mathcal{G}_t$ and $\mathcal{G}_{t+1}$ are small, then the functions f_t and f_{t+1} should be similar. Thus, the question we address is the following: by training the network with the mapping (updates) $f_t \to f_{t+1}$ for all t up to t_{train}, then, if the changes in the graph follow a pattern, can the network learn the distribution of this pattern so that it can perform the updates for future graph changes $f_t \to f_{t+1}$ with $t > t_{train}$?

In contradistinction with the Graph Convolutional Network (GCN) architecture from [132], or its simplification [133], which rely on the use of convolutional layers to incorporate graph information, in this work we propose to use the pure and simpler MLP (no convolutional layers) to solve this task. The reasons for this choice are: (i) the GCN is more challenging to train than the simpler MLP; (ii) in the GCN, one needs to see the entire graph in between its layers (thus constantly having to load it into memory); and (iii) it appears to be more intricate to encode the information about graph changes in a GCN that in our simpler, yet motivated by our theoretical results from this chapter, model. More precisely, in this chapter, we have seen that when one uses the warm restarts, the ARMA recursions allow to encode the necessary update to be performed by means of projecting a previous G-SSL solution onto the matrix $[\widetilde{\mathcal{R}} - \mathcal{R}]$, which is then diffused. Based on this observation, we claim that if we diffuse f_t in both $\mathcal{G}_{t+1}$ and $\mathcal{G}_t$ for up to a given number of steps k, then the resulting diffusion vector in $\mathcal{G}_{t+1}$ will differ from the diffusion vector on $\mathcal{G}_t$ proportional to the change between $\mathcal{G}_{t+1}$ and $\mathcal{G}_t$ in a k-hop vicinity from where the diffusion starts. Hence, is a means to encode for the perturbation and the G-SSL solution. In precise terms, we propose to use the following embedding of the graph

vertices into a latent euclidean space (feature vector of $2k+1$ entries):

$$\text{features of node } u = \left[(f_t)_u, \underbrace{(\mathcal{R}f_t)_u, (\mathcal{R}^2 f_t)_u, \cdots, (\mathcal{R}^k f_t)_u}_{\text{Diffusion on } \mathcal{G}_t}, \underbrace{(\widetilde{\mathcal{R}}f_t)_u, (\widetilde{\mathcal{R}}^2 f_t)_u, \cdots, (\widetilde{\mathcal{R}}^k f_t)_u}_{\text{Diffusion on } \mathcal{G}_{t+1}}\right]$$

Then, we propose to feed this vector into a multi-layer perceptron with $2k+1$ neurons in the input layer (feature vector), 1 neuron in the output layer (updated G-SSL value) and 50 neurons in a hidden layer with ReLU activations. As it can be seen, the feature vector is straightforward to compute in a recursive manner. Moreover, once the features are computed, our model acts individually on nodes, thus it can easily scale to massive graphs as it is highly paralellizable and bound to be trained on any graphic card. Indeed, our model calls for a local (sublinear) extension in which one only computes the feature entries for nodes surrounding the perturbations and then the training is performed on this much smaller subset of data. How to extend the feature vector to effectively code for changes in the graph locally is one of our immediate future research directions.

4.4.4 Numerical experiments

A Python toolbox for evolutionary stochastic block model

To alleviate the challenge of collecting graph evolving data, we contribute with a Python 3 class that allows to easily generate a temporal sequence of graphs from the Stochastic Block Model (SBM). The toolbox can be found in the Github repository: [136]

For this toolbox, we developed a class that initially generates a standard SBM realization and then creates the temporal sequence by recursively perturbing this initial graph according to the following user tunable parameters: (a) expected number of nodes joining; (b) expected number of nodes leaving; (c) probability of a link appearing inside classes; (d) probability of a link appearing between classes; (e) probability of a link disappearing inside classes; (f) probability of a link disappearing between classes.

In addition, our class facilitates G-SSL as any object drawn from the class possess the following attributes: (i) the sequence of graphs; (ii) what is the time index for each graph in the sequence; (i) the corresponding graph matrices (adjacency matrix, degree matrix, transition matrix, shifted operator, etc) for each graph in the sequence; (iii) the exact Pagerank or L^γ-PageRank vector for each graph in the sequence (for an initial user defined distribution y and parameter μ); and (iv) quick access to the cluster membership of each node.

The expected number of nodes joining and leaving are drawn according to a Poisson trial. Moreover, to allow to treat G-SSL in the dynamic setting (which assumes common initial distribution), labelled points cannot leave the graph. Further, when a node adheres to the graph it links to members of its cluster following the SBM class membership probabilities. Concerning the arising edges, these can appear between any pair of nodes just respecting the probability of within class or between class. In addition, new edges can superpose with existing ones, just increasing their weight by one. With respect to G-SSL operators, to satisfy equal ARMA coefficients for the entire graph sequence, we assume an upper bound on λ_{max} of 10 for the generalized operator $\mathcal{R}$ of L^γ-PageRank. Yet, these

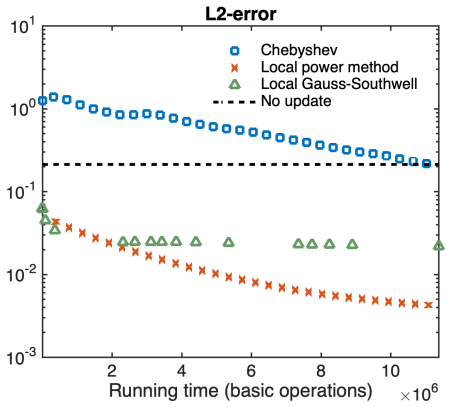


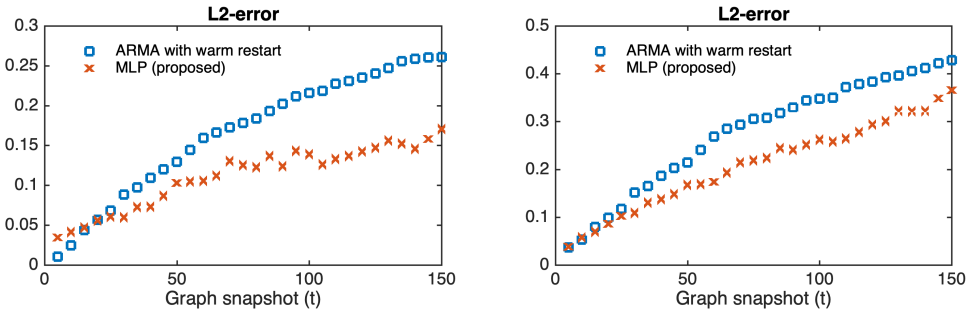
Figure 4.8: Approximation accuracy of the local algorithms (Algorithm 11 and Algorithm 12) and the Chebyshev method (Algorithm 7) as the number of fundamental operations they have performed in the task of updating an L^2-PageRank vector

parameters are easy to adapt for specific settings. Lastly, we stress that our toolbox only uses standard Python libraries (networkx and numpy) so that it can easily be combined with other libraries.

Impact of local updating algorithms

Experimental setup and goals. In this experiment, we aim to assess the benefits given by the local updating algorithms derived in this section.

The experimental setup is as follows: we take the same graph model from the experiments of Section 4.3.5, i.e., we use $C_{avg} = 20$, $C_{in} = 18$, $C_{out} = 2$, $N = 1000$. Then, 1% of labelled points is sampled at random and our dynamic SBM Python 3 class is used to make evolve the graph for 20 time steps. We use the following evolutionary parameters: (a) expected number of nodes joining: 1; (b) expected number of nodes leaving: 1; (c) probability of a link appearing inside classes: $C_{in}/(25N)$; (d) probability of a link appearing between classes: $C_{out}/(25N)$; (e) probability of a link disappearing inside classes: $C_{in}/(25N)$; (f) probability of a link disappearing between classes: $C_{out}/(25N)$. Despite these low rates, almost 40% of the nodes in the graph are affected between snapshots due to the small-world nature of the planted partition. Moreover, we stress that by making this selection of parameters, the class structure of the SBM is respected. We consider the L^2-PageRank vector with $\mu = 0.01$ as the base G-SSL method. We highlight that since we have a closed form expression for the elements of degree matrix D_2 of the L^2-graph, then taking matrix vector products with the L^2-PageRank operator $\mathcal{R} = L^2 D_2^{-1}$ can be done by re-scaling the input vector by $[D_2]_{uu} = [L^2]_{uu} = D_{uu}^2 + \sum_v W_{uv}^2$, and then apply the Laplacian matrix twice to such vector. Thus, we compute the exact L^2-PageRank for each graph realization and use the update algorithms introduced above (Algorithm 11 and Algorithm 12) as a means to update such vectors and predict the L^2-PageRank vector for the following graph realization. In addition, we apply the Chebyshev approach from scratch in each graph. To assess the benefits of the local update algorithms, we track the approximation accuracy given by these algorithms as a function of the number of basic operations that they have performed and report the mean value over the 20 graph



(a) Std. PageRank (b) L^2-PageRank

Figure 4.9: Updating via neural network (proposed) versus the analytic updates via the power method with warm restart (analytic)

realizations.

Results and discussion. Figure 4.8 displays the approximation accuracy of the local update algorithms (Algorithm 11 and Algorithm 12) and the Chebyshev method (Algorithm 7) as a function of the number of basic operations they have performed in the task of updating an L^2-PageRank vector. As it can be seen, when addressing changes in the graph, the local updating approaches developed in this chapter offer a much better solution than the Chebyshev approach. Indeed, due to the small world nature of the SBM, for this graph model the local property is less influential as compared to other graphs. Nonetheless, since the changes are small and the local methods are just a rewriting of the warm restarts, we can still clearly see that these approaches provide significant savings in computational power. For instance, in our experiment, it takes around 10 million operations for the Chebyshev approach to attain a similar amount of error as not performing any update, while the local power method after 10 million operations is close to two orders of magnitude below the no-update error. Moreover, if the graph increases in size, then these differences between the local and the Chebyshev method become more dramatic as the latter remains global and always starts from scratch. Lastly, we note that the Power method tends to perform better than the Gauss-Southwell. This observation is consistent with the observations made in [25], where the power method also leads to better accuracy, although, in such work it was not acknowledge that the power method could also be recasted locally.

Updating a sequence of graphs

Experimental setup and goals In this experiment, we aim to assess the feasibility of using the neural network to update a sequence of graphs.

The experimental setup is as follows: we generate a sequence of 170 graphs with our dynamic SBM python class. From these, the first 20 graphs are used as our training set and the remainder 150 snapshots are used for the test set. The graphs follow simple parameters: p_{in} = 0.5, p_{out} = 0.05, N = 200, the expected number of nodes joining/leaving is 1, the probabilities of edges appearing/disappearing within a class is $p_{in}/20$ and the probability of nodes appearing/disappearing between classes is $p_{out}/20$. Then, we train

the network to learn both standard PageRank updates and L^2-PageRank updates with $\mu = 0.1$. Then, we train the network to learn both PageRank updates and L^2-PageRank updates. For standard PageRank we use $\mathcal{R} = P$ as the operator used to compute the features. For L^2-PageRank the natural choice is $\mathcal{R} = L^2 D_2^{-1}$. Yet, interestingly, we have experienced that, although not being Markovian, using $\mathcal{R} = P_2 = D_2^{-1} W_2$ tends to give better results in practice, thus we use this choice for our experiment. In both cases we use features up to $k = 3$ hops. To train the network, we use the Pytorch model of the MLP via the ADAM optimizer with learning rate equal to 1e-6 for PageRank and 5e-5 for L^2-PageRank. Since our goal is to overfit the distribution of the updates, then we let the network overfit its training set until an early stop condition of 10 consecutive loss evaluations increasing is met. To stabilize the training, we use a weight decay of 5e-9. For the evaluation set, we split the scores from the 20 graphs constituting the training set into 20 epochs and each epoch is randomly splitted for training and evaluating. Then, as baseline, we compare with the power iteration with warm restarts under the same complexity: i.e., if to compute the features we use k-hops, leading to a vector of $2k$-matrix-vector products, then we use power iteration with $2k$ steps. Lastly, we stress that in both algorithms we use their predictions at time t as the new input for the predictions at time $t + 1$, meaning that we aim for the the method that accumulates less error.

Results and discussion. Figure 4.9 shows the approximation error attained by the neural network (proposed) versus using analytic updates via the power method with warm restart (analytic) in the problem of updating the G-SSL solution for a sequence of 150 graphs when both approaches are employed under the same complexity. As it can be seen, the network is able to cope better with changes by keeping a smaller error for both PageRank and L^γ-based PageRank as the sequence of graphs progresses. While this example is simple, these results are encouraging and point in the direction that our proposition for feature vector is pertinent. Moreover, they open the door to further investigate our model with more challenging data and network architectures and also cast it in sub-linear complexity.

Appendix: technical proofs

4.A Proof of Lemma 13

Proof. From (4.32), we have that

$$p^{(K)} = \left(\frac{2\mu}{2\mu + \lambda_{max}}\right) y - \left(\frac{\lambda_{max}}{2\mu + \lambda_{max}}\right) \mathcal{S} p^{(K-1)} \tag{4.47}$$

$$= \text{arma}_{\rho,\psi}(y) + \left(\frac{\lambda_{max}}{2\mu + \lambda_{max}}\right) \mathcal{S} \left(\text{arma}_{\rho,\psi}(y) - p^{(K-1)}\right). \tag{4.48}$$

Thus, by re-arranging terms

$$\text{arma}_{\rho,\psi}(y) - p^{(K)} = -\left(\frac{\lambda_{max}}{2\mu + \lambda_{max}}\right) \mathcal{S} \left(\text{arma}_{\rho,\psi}(y) - p^{(K-1)}\right) \tag{4.49}$$

$$= (-1)^K \left(\frac{\lambda_{max}}{2\mu + \lambda_{max}}\right)^K \mathcal{S}^K \left(\text{arma}_{\rho,\psi}(y) - p^{(0)}\right) \tag{4.50}$$

Taking the norm finishes the proof. ∎

4.B Proof of Lemma 14

Proof. By induction, for $t = 0$ we have:

$$r^{(0)} = \rho y - (\mathbb{I} - \psi\mathcal{S}) p^{(0)} = (\mathbb{I} - \psi\mathcal{S}) \left(\text{arma}_{\rho,\psi}(y) - p^{(0)}\right) \tag{4.51}$$

Then, by isolating $\text{arma}_{\rho,\psi}(y)$, we obtain:

$$\text{arma}_{\rho,\psi}(y) = p^{(0)} + (\mathbb{I} - \psi\mathcal{S})^{-1} r^{(0)} = p^{(0)} + \frac{1}{\rho} \text{arma}_{\rho,\psi}(r^{(0)}) \tag{4.52}$$

Now, suppose that $\text{arma}_{\rho,\psi}(y) = p^{(t)} + \frac{1}{\rho}\text{arma}_{\rho,\psi}(r^{(t)})$ holds for some t. Then, for $t+1$ we have:

$$r^{(t+1)} = r^{(t)} - (\mathbb{I} - \psi\mathcal{S})\, r_u^{(t)} \delta_u = r^{(t)} - (\mathbb{I} - \psi\mathcal{S}) \left(p^{(t+1)} - p^{(t)}\right) \tag{4.53}$$

The proof finishes by re-arranging terms

$$\frac{1}{\rho}\text{arma}_{\rho,\psi}(r^{(t+1)}) + p^{(t+1)} = \frac{1}{\rho}\text{arma}_{\rho,\psi}(r^{(t)}) + p^{(t)} = \text{arma}_{\rho,\psi}(y). \tag{4.54}$$

∎

4.C Proof of Lemma 15

Proof. Let us depart with a warm restart on the ARMA recursion as follows:

$$\widetilde{p}^{(1)} = \rho y + \psi\widetilde{\mathcal{S}}\widetilde{p}^{(0)} \tag{4.55}$$
$$= \rho y + \psi\widetilde{\mathcal{S}}\text{arma}_{\rho,\psi}(y) \tag{4.56}$$
$$= \text{arma}_{\rho,\psi}(y) - \psi\mathcal{S}\text{arma}_{\rho,\psi}(y) + \psi\widetilde{\mathcal{S}}\text{arma}_{\rho,\psi}(y) \tag{4.57}$$
$$= \text{arma}_{\rho,\psi}(y) + \psi\left[\widetilde{\mathcal{S}} - \mathcal{S}\right]\text{arma}_{\rho,\psi}(y) \tag{4.58}$$
$$= \text{arma}_{\rho,\psi}(y) + r \tag{4.59}$$

Then, for the second iteration we have:

$$\widetilde{p}^{(2)} = \rho y + \psi\widetilde{\mathcal{S}}\widetilde{p}^{(1)} \tag{4.60}$$
$$= \rho y + \psi\widetilde{\mathcal{S}}\left(\text{arma}_{\rho,\psi}(y) + r\right) \tag{4.61}$$
$$= \text{arma}_{\rho,\psi}(y) - \psi\mathcal{S}\text{arma}_{\rho,\psi}(y) + \psi\widetilde{\mathcal{S}}\left(\text{arma}_{\rho,\psi}(y) + r\right) \tag{4.62}$$
$$= \text{arma}_{\rho,\psi}(y) + \psi\left[\widetilde{\mathcal{S}} - \mathcal{S}\right]\text{arma}_{\rho,\psi}(y) + \psi\widetilde{\mathcal{S}}r \tag{4.63}$$
$$= \rho y + \psi\widetilde{\mathcal{S}}\text{arma}_{\rho,\psi}(y) + \psi\widetilde{\mathcal{S}}r \tag{4.64}$$
$$= \text{arma}_{\rho,\psi}(y) + \psi r + \psi\widetilde{\mathcal{S}}r \tag{4.65}$$

By successive applications of this procedure, we have that

$$\widetilde{p}^{(\infty)} = \text{arma}_{\rho,\psi}(y) + \sum_{t=0}^{\infty}\psi^t\widetilde{\mathcal{S}}^t r \tag{4.66}$$
$$= \text{arma}_{\rho,\psi}(y) + \frac{1}{\rho}\widetilde{\text{arma}}_{\rho,\psi}(r) \tag{4.67}$$

■

4.D Proof of Lemma 16

Proof. From Eq. 4, by truncating computation of $\widetilde{\text{arma}}_{\rho,\psi}(r)$ to K terms, we have that

$$\|\widetilde{p}^{(\infty)} - \widetilde{p}^{(K)}\| = \|\sum_{t=0}^{\infty}\psi^t\widetilde{\mathcal{S}}^t r - \sum_{t=0}^{K}\psi^t\widetilde{\mathcal{S}}^t r\| \tag{4.68}$$
$$= \|\sum_{t=K+1}^{\infty}\psi^t\widetilde{\mathcal{S}}^t r\| \tag{4.69}$$
$$= \|\sum_{i=0}^{\infty}\psi^{i+K+1}\widetilde{\mathcal{S}}^{i+K+1} r\| \tag{4.70}$$
$$= \|\psi^{K+1}\widetilde{\mathcal{S}}^{K+1}\sum_{i=0}^{\infty}\psi^i\widetilde{\mathcal{S}}^i r\| \tag{4.71}$$
$$= \|\psi^{K+1}\widetilde{\mathcal{S}}^{K+1}\frac{1}{\rho}\widetilde{\text{arma}}_{\rho,\psi}(r)\| \tag{4.72}$$
$$\leq |\psi^{K+1}|\|\widetilde{\mathcal{S}}^{K+1}\|\|\frac{1}{\rho}\widetilde{\text{arma}}_{\rho,\psi}(r)\| \tag{4.73}$$

■

Chapter 5

G-SSL for Internet routing

5.1 Introduction

The main goal of G-SSL is to help solve real world problems. Towards this aim, this Chapter uses G-SSL to address current issues in the context of Internet routing under *Border Gateway Protocol* (BGP).

BGP is the routing protocol that drives the Internet. It makes the exchange of information between entities connected to the Internet to be as efficiently as possible. For example, when a user introduces the address of a site in a web browser, a domain name system (DNS) server retrieves the IP address of the website, and then BGP determines what is the optimal route that the data packets need to go through in order to fetch the website's information from the host IP.

In more precise terms, the Internet is a network of *autonomous systems* (ASes), where an AS refers to a collection of IP prefixes (groups of IP addresses) that is administrated by a single organization (usually an Internet service provider). In this context, the main goal of BGP is to exchange information between ASes so that they become aware of what sequence of ASes needs to be traversed to reach a target IP prefix. To achieve this, when an AS has a new prefix, the AS advertises the new prefix to its adjacent ASes by means of an update message that is communicated through BGP speakers. This update message is received by the adjacent ASes which append to it their Autonomous System Number (ASN) before retransmitting the message to further adjacent ASes. This process is successively repeated until all ASes have received the message, giving them the information about the sequence of ASes that needs to be followed to reach the aforementioned prefix. Since there may be various paths to reach the prefix, the key feature of BGP is that it discards routes containing loops and only preserves the most efficient path according to some policies agreed between ASes [137].

BGP has established as a protocol of utmost importance, bringing order to the way in which information travels in the Internet. However, despite its great success, issues affecting it and the Internet traffic controlled by it arise in practice, which, due to the large scale and rapidly evolving nature of the web, are difficult to measure and characterize. Next, we detail two of such issues:

Issue of BGP zombies: when a prefix withdraws from the Internet, the origin AS stops announcing it, expecting that the remainder of ASes become aware that the prefix is unreachable anymore so that they remove it from the their routers' tables. Normally, a large sequence of exchanges between BGP speakers needs to occur before an IP prefix is fully removed. Such long process is because an AS ignores if a route has ceased from being announced due to some malfunction occurring in the path towards the prefix, or because the prefix has been withdrawn at the origin AS. Therefore, in an attempt to maintain connections alive, an AS searches in its routing tables for the best alternative path towards the prefix and communicates this new path to the other ASes. At a next round, the AS realizes that such path is also invalid and searches for another alternative. Only when the AS runs out of alternative paths is when it removes the prefix and transmits the withdrawal of it. Theoretically, this withdrawal process always ends with the prefix being removed from the routing tables of all BGP routers. Yet, in practice, network operators occasionally report issues where routers get stuck in the withdrawal process and maintain routes towards IP prefixes that have been withdrawn by their origin network. This failure, normally refereed to as *stuck routes* or *zombie routes*, is relatively unknown outside network operator circles and is not well understood despite it being a source of confusion and a burden to troubleshoot and clean by network operators.

Issue of IP to AS mapping: internet traffic may display, in practice, congestion due to a mismatch between installed capacity and demand [138], vulnerable nodes to DDoS attacks [139], or abnormal traffic changes due to facilities outages [140]. These issues are normally visible, and characterizable, from traceroute measurements of the Internet: paths of IP addresses accompanied by transit delay measurements made on data packets that try to reach a host IP. Yet, a limitation for the understanding of these problems and, consequently, for taking action on them, is that such issues normally affect routers lying at the boundaries between ASes (in inter-AS links), implying that the tracerout measurements, which lack information about ASes, do not provide the entire picture to address these issues. Therefore, the design of techniques that allow to infer the AS topology from traceroute measurements is of prime interest. The main difficulty is that, while one can map an IP address to its corresponding IP prefix and then retrieve the publicly registered ASN for that prefix, such procedure fails for inter-AS links. The reason is that a link from the internet always needs to have two IP addresses from the same prefix, which enforces ASes to agree on which IP they will use for their inter-AS links, resulting in one of the ASes having to use an IP from the other AS. Therefore, if we blindly map IP prefixes to ASNs, then routers at the ASes boundaries, which are the ones of interest, can get incorrectly mapped. In the literature, two main techniques exist to address the IP to ASN mapping challenge: [141] and [142]. The recent proposition of [143] merges both techniques to enhance accuracy. However, we stress that [143] is notoriously difficult to use in practice: first, it requires Alias resolution and AS relationships, which are extremely hard to compute and error-prone; second, it assumes that the user has control over multiple remote vantage points spread-out over the Internet. To put into context the cost of such method, the Center for Applied Internet Data Analysis (CAIDA), which currently maintains such project, is restrained to retrieve results from it only every six months due to its high cost. As a result, better solutions must be sought.

In this chapter, we propose to use G-SSL to address the problems listed above. The

chapter is divided in two parts: first, Section 5.2 characterizes the scope of zombie outbreaks in BGP using G-SSL, then Section 5.3 shows that G-SSL is an effective tool to identify the AS topology from the network of IP addresses. More precisely, Section 5.2 provides the first characterization of BGP zombies. Our study employs a controlled setting using RIS routing beacons and RIS peers, which permit to track routing tables at numerous ASes for prefixes announced and withdrawn at predetermined intervals. From the RIS measurements, we construct an expertized dataset that we fed into G-SSL to infer the state of ASes not measured by RIS peers. By constructing a validation set from tracerout measurements, we show that the standard PageRank algorithm possesses the sufficient predictive power to solve this inference task in almost exact accuracy, therefore there is no need to revert to L^γ-PageRank. Lastly, we use the inferences given by G-SSL to characterize the scope of zombie outbreaks, with our results showing that these are likely to be of considerable size if they appear in transit networks with large AS hegemony. Section 5.3 shows that L^γ-PageRank is an effective tool to solve the IP to AS mapping challenge. To show it, we perform tracerout measurements and build a graph from them. Then, from publicly available ASNs, we construct labelled datasets with varying degrees of confidence: strict, loose, weighted. We study the advantages/disadvantages of these three types of semi-supervision, which offer a trade-off between amount of annotated examples and how much we can trust them. Our results show that, for the studied dataset, L^γ-PageRank with $\gamma = 2$ can solve this inference task with no errors, contrary to standard PageRank which always miss-classifies data.

5.2 G-SSL to characterize the scope of BGP zombies

5.2.1 Experimental setup

Our goal is to characterize the scope of zombie propagation in a controlled setting. To setup terminology, we refer to a *BGP zombie* as an active Routing Information Base (RIB) entry for a prefix that has been withdrawn by its origin network, meaning that it is not reachable anymore. By *zombie ASes* and *zombie peers* we refer to ASes and BGP peers, respectively, whose routers have BGP zombies. We refer to all zombies that correspond to a same prefix and appear during the same two-hour time window as a *zombie outbreak*. The *outbreak size* is the number of zombie ASes in an outbreak.

To observe BGP zombies one must withdraw an IP prefix from its origin AS and assess whether or not it has been withdrawn from the routing tables of other ASes. To characterize zombie emergence, we use a controlled environment via RIPE's Routing Information Service (RIS) BGP beacons [144] and BGP data repository [145]. RIS BGP beacons are specifically designed to study Internet inter-domain routing. They consist of a set of IP prefixes (IPv4 and IPv6) that are announced and withdrawn at predetermined time intervals. More precisely, they are announced every day at 00:00, 04:00, 08:00, 12:00, 16:00, and 20:00 UTC and withdrawn two-hours after their announcement at 02:00, 06:00, 10:00, 14:00, 18:00, and 22:00 UTC, respectively. For this study, we employ 27 beacon prefixes (13 IPv4 and 14 IPv6) announced from Europe, U.S.A, Russia, Japan, and Brazil. In addition to beacons, RIS provides data collectors which archive the RIBs and BGP update messages from numerous ASes peering with Internet eXchange Points (IXP). By using the archives from the RIS collectors, the appearance of BGP zombies can be tracked

by simply identifying, from all RIS peers, RIBs that retain the beacon prefix after this has been withdrawn more than 1.5 hours ago. Such 1.5 hour threshold is set empirically to avoid late withdrawals due to BGP convergence [144], route flap damping [146], or stale routes [147]. We also monitor the beacon's visibility by means of a RIPEstat looking glass [148], so that every time a zombie outbreak is seen, we launch traceroute measurements towards beacon prefixes from Atlas probes located in zombie ASes. We conducted measurements during the three periods listed in the following table:

Start	End	#IPv4 outbreaks	#IPv6 outbreaks
2017-03-01	2017-04-28	1732	591
2017-10-01	2018-12-28	384	1202
2018-07-19	2018-08-31	520	686

Table 5.1: Measurement periods and number of detected outbreaks for the 27 monitored beacons.

5.2.2 G-SSL to identify zombies

From the RIS collectors, one can only detect zombies appearing in ASes peering with the RIS. In this subsection, we show that G-SSL can accurately infer zombies for the remainder of ASes.

Towards this goal, we retrieve, for each outbreak, the AS path of zombie entries and the last valid path for peers that have correctly withdrawn the beacon. Then, we construct a graph in which the ASes represent nodes and consecutive ASes in the AS paths get connected by an edge. Figure 5.1 illustrates the resulting graph for an outbreak occurring for beacon 84.205.71.0/24 on September 9th 2017 between 22:00 and 00:00. The green nodes represent RIS peers that have correctly withdrawn the prefix at 22:00. The red nodes represent zombie peers observed from 22:00 to 00:00. The grey nodes represent ASes that are not peering with RIS collectors, hence no observations are available for these ASes, albeit they appear in the collected AS paths.

As it can be seen from Figure 5.1, the RIS measurements do not reveal if the grey nodes are part of the zombie outbreak or not. Therefore, G-SSL arises as a tool that is tailor-made to infer the state of such ASes. For this reason, we propose to employ G-SSL with the result from the RIS collectors as the expertized data: i.e., ASes that correctly withdraw the beacon (green nodes) constitute the annotated examples of one class, and the observed zombies (red nodes) constitute the annotated examples of a second class. To perform the inferences, we use as G-SSL method the standard PageRank algorithm ($\gamma = 1$) in the multi-class setting (as it will be shown in our evaluation below, there is no need to increase the complexity of the task by including the extra degree of freedom γ, since standard PageRank solves this inference task with almost exact precision). In addition, to systematically detect the scope of each zombie outbreak, we automatically tune the regularization parameter of G-SSL, μ, by means of a leave-one-out cross validation procedure: from the set of documented vertices, one element, per class, is selected as a labeled

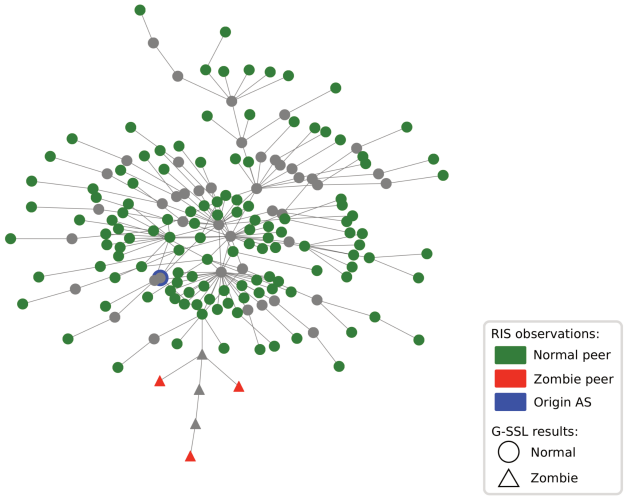
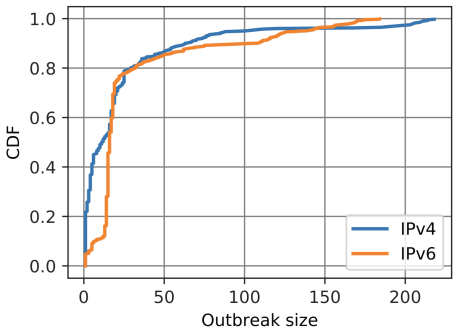


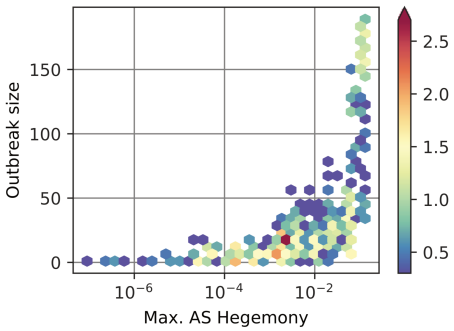
Figure 5.1: Graph from AS paths for an outbreak occurring for beacon 84.205.71.0/24 on September 9th, 2017 between 22:00 and 00:00. Each node is an AS. Green represents ASes that correctly withdraw the prefix at 22:00. Red represents zombie ASes observed from 22:00 to 00:00. Grey nodes are ASes not peering with RIS collectors. Shape of nodes indicate G-SSL prediction: circle is normal AS; triangle is zombie AS.

example, while the rest is added to the group of not documented and used for performance assessment. The procedure is repeated and μ is selected, from a fine grid of values, as the one maximizing average detection performance. In Figure 5.1, the result of G-SSL is displayed according to the shape of nodes: triangles represent detected zombies; circles represent normal ASes. For the outbreak documented in the figure, the three ground truth zombies share the same upstream provider and the zombies inferred by G-SSL correspond to all the downstream ASes of the same provider. This gives a glimpse on the accuracy of the G-SSL prediction.

Evaluation data: To evaluate the prediction accuracy of G-SSL, we construct a ground truth validation set for a significant amount of the ASes not peering with the RIS. This is done by means of traceroute measurements performed every time we detect an outbreak. Such traceroute measurements are done with RIPE Atlas measurement platform [149]: a platform composed of 10k devices constantly doing traceroute measurements to multiple destinations. We select five Atlas proves for each AS found in the zombie paths and perform traceroutes towards the corresponding prefix every 5 minutes until the prefix is re-announced. To identify zombies from the traceroute measurements, we discard the first public IP found in the tracerouts as it usually stands for a gateway with a default route. Then, we group all the traceroutes initiated from the same AS and if these consist only of ICMP network unreachable errors and unresponsive routers, then we consider the AS as normal since such events imply that packets are not being forwarded and the route has been correctly withdrawn. For traceroutes with responsive routers, we retrieve the router's ASN using longest prefix match and compute F_A: the number of IP addresses from ASN A that forwarded packets, and, E_A: the number of IP addresses from ASN A that sent an ICMP error. We consider ASN A as a zombie if $F_A > E_A$. Lastly, we establish our ground truth evaluation set by merging the results from both the traceroutes measurements and



(a) Distribution of outbreak size



(b) Relation of outbreak size and maximum AS hegemony affected

Figure 5.2: Characterization of the number of zombie ASes per outbreak

the RIS collectors.

Evaluation results: For the measurement periods detailed in Table 5.1, G-SSL accurately identifies 97% of the zombie ASes and 99% of the normal ASes, which indicate that G-SSL delivers inferences that are reliable enough for the characterization of zombie outbreaks that we present the next subsection. Lastly, we stress that the traceroute measurements and the RIS peers lead to an evaluation set that does not include all ASes in the graph, thus, in our experiments, G-SSL returned 35% more classified ASes than the ASes contained in the evaluation set. These ASes are not used for the evaluation.

5.2.3 Characterization of zombie outbreaks via G-SSL

In this subsection, we use the G-SSL predictions to characterize the scope of zombie outbreaks for our three measurement periods detailed in Table 5.1.

For each zombie outbreak, we count the total number of zombie ASes returned by G-SSL. Then, we compute the distribution of outbreak sizes, which is displayed in Figure 5.2a. Our results indicate that outbreaks have a wider scope on IPv6 than IPv4, as, on average, an outbreak affects 30 ASes for IPv6 and 24 ASes for IPv4, which correspond to 10% and 17% of the total monitored ASes, respectively. Moreover, the median outbreak size is 16 ASes for IPv6 and 11 ASes for IPv4. Indeed, 63% of the outbreaks contain between 12 and 19 ASes for IPv6, while 18% of outbreaks of IPv4 have a number of zombie ASes within such range. For IPv4, 45% of the outbreaks have between 1 and 6 ASes.

In addition, we study the relationship between the importance of the transit networks affected by zombies and the outbreak sizes. In Figure 5.2b, we show the relationship between the size of an outbreak and the largest AS hegemony from the affected ASes. The AS hegemony is a score that measures the centrality of an AS in the Internet: a large value indicates an important transit network (Tier-1 ISPs). The figure displays a correlation between large outbreaks and large transit networks, and small outbreaks with networks having low AS hegemony. This observation permits to reinterpret Figure 5.2a to conclude that zombies are more prone to affect networks with large AS hegemony under IPv6 than IPv4. To illustrate the large scope of outbreaks in transit networks with large

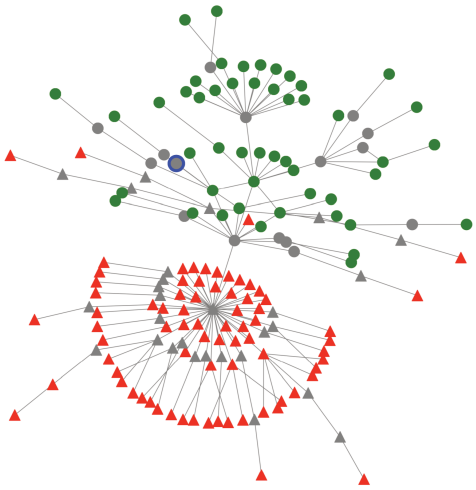


Figure 5.3: Outbreak for beacon 84.205.70.0/24 on December 6th 2017 in which a zombie was detected in Tier-1 network Level(3).

hegemony, we display in Figure 5.3 an outbreak where zombies arise in a Tier-1 network called Level(3). As it can be seen, since a zombie impacts an important network, it causes almost half of the ASes to become afflicted with stuck routes.

Table 5.2 lists the transit networks that more frequently appeared in zombie outbreaks. For the table, only ASes with hegemony larger than 0.001 are considered. As it can be seen, the top affected ASes under IPv4 heavily vary in the three measurement periods, which is not the case for IPv6 where Init7 and Atom86 appear at the top for the three measurement periods. Manual inspection of the data reveals that Atom86 is downstream of Init7, implying that it is affected every time Init7 has zombies. The zombies of Init7 usually propagate to 14 downstream ASes. This can be verified in Figure 5.4, where we display one of the outbreaks in which a zombie was detected in Init7. This, along with the frequency in which Init7 is affected by zombies, explain why Figure 5.2a reveals that it is very likely to find outbreaks of around 15 ASes in our data.

Let us stress that the goal of this section is to show that G-SSL is an effective tool to detect and characterize BGP zombies, permitting us to obtain the first characterization of BGP zombies. Yet, it remains to be verified if our characterization is reminiscent of what occurs in the wild, as we obtained it from data collected in a controlled setting. The main limitation for applying G-SSL in the real setting is the collection of the expertized data. This is because, in the wild, it is difficult to distinguish zombies from local routing issues or from routing configuration changes aiming to limit the visibility of prefixes. Despite such challenges, our characterization seems to pinpoint issues already observed in practice. For example, network operators at Init7 acknowledge to have experienced issues with IPv6 routes (as inferred by G-SSL in Table 5.2), indicating that they cause customer complains every few months. Lastly, we stress that our results do not imply that the detected outbreaks are caused by the transit network listed in Table 5.2. To find the root cause of zombie outbreaks, additional measurements within these networks and their peers are needed.

Mar/Apr 2017	Oct/Dec 2017	Jul/Aug 2018
	IPv4	
AS3303 Swisscom 46.13%	AS6939 HE 14.84%	AS6667 Elisa 19.81%
AS12874 Fastweb 46.07%	AS1103 SURFnet 9.90%	AS680 DFN 17.69%
AS8359 MTS 9.93%	AS7575 AARNet 9.38%	AS7018 AT&T 16.73%
AS680 DFN 9.18%	AS286 KPN 9.38%	AS3549 Level3 GBLX 15.96%
AS7018 AT&T 8.60%	AS6453 TATA 9.11%	AS7575 AARNet 15.19%
	IPv6	
AS8455 Atom86 39%	AS13030 Init7 57%	AS13030 Init7 74%
AS13030 Init7 39%	AS8455 Atom86 55%	AS8455 Atom86 73%
AS5580 Hibernia 36%	AS8928 Interoute 36%	AS7018 AT&T 15.69%
AS7018 AT&T 8%	AS9002 RETN 35%	AS23106 CEMIG 13%
AS7018 Fiord 6%	AS33891 Core-Backbone 22%	AS1916 RNP 13%

Table 5.2: Top 5 affected transit ASes in IPv4 and IPv6. Each cell reports the frequency in which the AS appeared in the outbreaks of the corresponding measurement period.

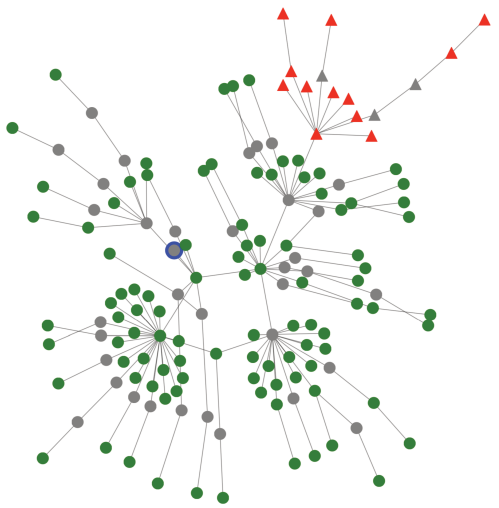


Figure 5.4: Outbreak for beacon 2001:7fb:fe06::/48 on March 1st, 2017 in which a zombie was detected in Init7.

5.3 L^γ-PageRank for IP to AS mapping

5.3.1 Experimental Setup and goals

In this section, our goal is to use G-SSL to solve the issue of identifying the managing ASes of the nodes in a network of IP addresses. We aim to show that, for this challenge, the standard PageRank algorithm lacks the predictive power to deliver reliable inferences, while, on the other hand, L^γ-PageRank can solve the task reliably.

We recall that the objective of operating at the IP network level is because such level of granularity may help reveal and understand better various issues affecting internet traffic [138, 139, 140]. For this experiment, we perform traceroute measurements by means of the Atlas measurement platform [149]. Then, we build a graph by stitching together the

multiple paths of IP addresses collected from the traceroutes.

As mentioned in the Introduction, the main challenge in inferring ASes from the network if IP addresses is that, albeit one can recover the publicly available ASN for the prefixes conforming the network, such IP to ASN mapping is misleading for routers at the boundaries of ASes. Yet, such mapping remains correct for intra-AS routers and for some of the routers at the boundaries. Therefore, we propose to use this information as the expertized data for G-SSL, which we then employ to estimate the true AS for all the nodes in the graph. The possibility to retrieve the public ASN for IP prefixes gives us the flexibility to construct labelled datasets with the following varying degrees of confidence:

- **Strict expert:** a node is labelled to belong to AS X if all of its neighbors are mapped to AS X via the IP prefix to ASN mapping. As a result, only intra-AS nodes get labelled, but we can be confident that the AS assigned to the annotated examples is the correct one.

- **Loose expert:** all nodes are labelled via the IP prefix to ASN mapping. This implies that some of our labelled examples are incorrect (for some nodes at the boundaries of ASes).

- **Weighted expert:** a node is labelled with the ASes obtained for its own IP and the ASes for the IPs of its neighboring nodes. Therefore, a node may have multiple labels if its own IP prefix to ASN mapping is different from the IP prefix to ASN mapping of its neighbors. If a labelled point normally has a unit weight in G-SSL, then the weighted expert divides the unit weight into the multiple labels it assigns to a node.

In this work, we assess the advantages/drawbacks that each of type of semi-supervision brings when solving the task. Figure 5.5 displays the graph arising from our traceroute measurements and the annotations given by the experts to vertices. To easy identification and reference to the graph, we refer to nodes in the graph by means of an ID written in blue. Green, magenta and brown colored text encode annotations by experts and the true AS of nodes. Green and magenta, together, coincide with the true AS of nodes. Green alone constitutes the annotations given the strict expert and magenta stands for the nodes not annotated by the strict expert. The loose expert tags all nodes, albeit some are incorrect. In this context, green and brown, together, constitute the annotations given by loose expert. Notice that brown tags are sometimes incorrect (this can be seen by comparing them with magenta). The annotations given by the weighted expert are difficult to represent in the figure, they will be made precise below.

For the experiments, we compare the standard PageRank algorithm and L^γ-PageRank with $\gamma = 2$. Let us highlight that this task is tailor-made to be addressed with the topology arising from L^2. The nodes that we aim to accurately classify lie at the boundaries of ASes. These nodes always have the same class of their 1-hop intra-AS neighbours (nodes for which we know their true AS label) and always have a different class of nodes that lie at 2-hop distance from them, which correspond to the intra-AS nodes of the other ASes (nodes for which we also know their true AS label). Therefore, if we use the L^2-graph, which places a negative edge between 2-hop distant nodes while preserving the positive

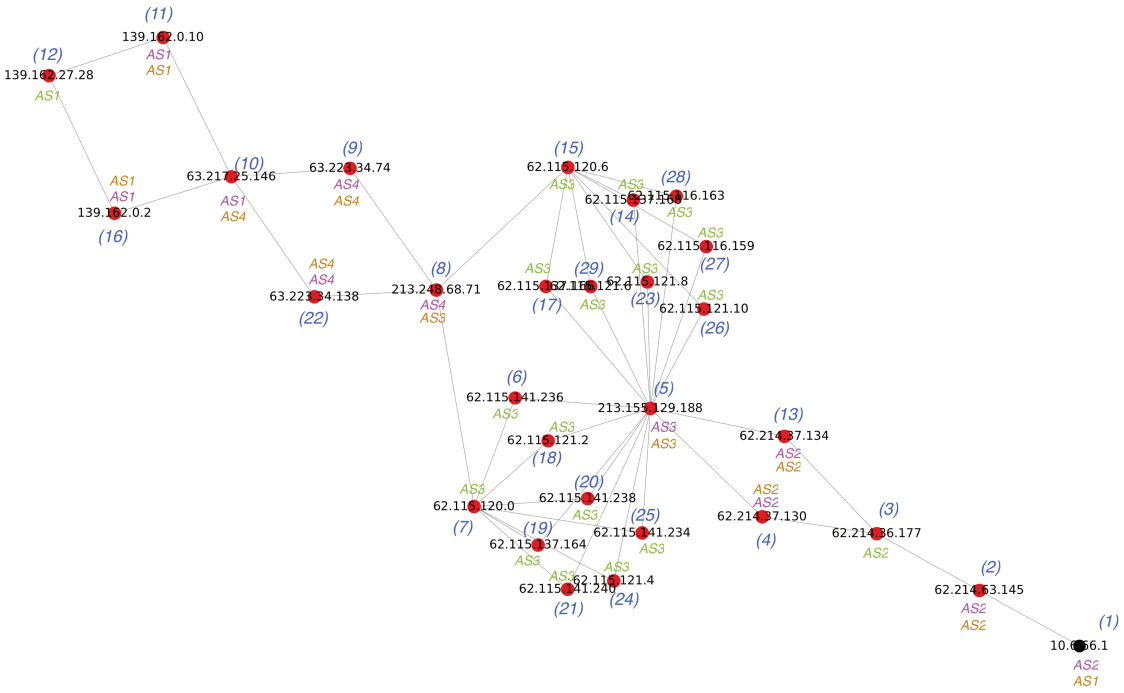


Figure 5.5: Graph of IP addresses from tracerout measurements. Nodes are given in ID (blue). Green and magenta coincide with the true AS of nodes. Green stands for the annotations given by the strict expert and magenta referts to nodes not annotated by the strict expert. Green and brown together constitute the annotations given by the loose expert.

original connections, then the nodes that we aim to classify always remain positively connected with nodes that have their same AS, and become negatively connected with nodes that have different AS labels. AS a result, nodes at the boundaries should now repulse (negative edge) the incorrect AS labels and favor (positive edge) the correct AS label.

Lastly, we observe that the ground truth data given by the different experts are suitable to be addressed by both sweep-cuts and the multi-class approach. For example, we can see from Figure 5.5 that the strict expert, albeit being very reliable, it does not provide any annotated example for AS4. Thus, if we classify data via the multi-class approach, the nodes originally belonging to AS4 will get assigned by G-SSL to either AS1, AS2, or AS3, which, in any case, is incorrect. On the other hand, if classify the data via sweep-cuts, since they retrieve individual classes, then they have the potential to never assign the nodes belonging to AS4 to any class, which avoids drawing incorrect conclusions on them. Notice that the multi-class approach may be more convenient than the sweeps in some situations. For example, the weighted experts may give more than one annotation to some nodes. Thus, if we use the sweep-cuts to classify the data, then such nodes are prone to be assigned nodes into more than one class by the sweep. On the contrary, the multi-class approach is designed to assign nodes to only one class, making it more suitable for this type of expertized data. For our experiments, we classify the data by means of these two approaches and assess their benefits/drawbacks.

ID	IP	True AS	Labels				Sweep-cut		Multi-class	
			AS1	AS2	AS3	AS4	$\gamma = 1$	$\gamma = 2$	$\gamma = 1$	$\gamma = 2$
(1)	10.6.66.1	AS2	0	0	0	0	AS2	AS2	AS2	AS2
(2)	62.214.63.145	AS2	0	0	0	0	AS2	AS2	AS2	AS2
(3)	62.214.36.177	AS2	0	1	0	0	AS2	AS2	AS2	AS2
(4)	62.214.37.130	AS2	0	0	0	0	AS2	AS2	AS2	AS2
(5)	213.155.129.188	AS3	0	0	0	0	AS2, AS3	AS3	AS3	AS3
(6)	62.115.141.236	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(7)	62.115.120.0	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(8)	213.248.68.71	AS4	0	0	0	0	AS1	n.a	AS3	AS3
(9)	63.223.34.74	AS4	0	0	0	0	AS1	n.a	AS3	AS1
(10)	63.217.25.146	AS1	0	0	0	0	AS1	AS1	AS1	AS1
(11)	139.162.0.10	AS1	0	0	0	0	AS1	AS1	AS1	AS1
(12)	139.162.27.28	AS1	1	0	0	0	AS1	AS1	AS1	AS1
(13)	62.214.37.134	AS2	0	0	0	0	AS2	AS2	AS2	AS2
(14)	62.115.137.168	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(15)	62.115.120.6	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(16)	139.162.0.2	AS1	0	0	0	0	AS1	AS1	AS1	AS1
(17)	62.115.137.166	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(18)	62.115.121.2	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(19)	62.115.137.164	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(20)	62.115.141.238	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(21)	62.115.141.240	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(22)	63.223.34.138	AS4	0	0	0	0	AS1	n.a	AS3	AS1
(23)	62.115.121.8	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(24)	62.115.121.4	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(25)	62.115.141.234	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(26)	62.115.121.10	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(27)	62.115.116.159	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(28)	62.115.116.163	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(29)	62.115.121.6	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3

Table 5.3: Best classification attained by G-SSL using the *strict expert* labelled data. No labelled data is available for AS4. Green cells refer to correct inferences and red cells to incorrect inferences. Cells having n.a. indicate that the node is not assigned to any class by G-SSL. Cells with more than one tag indicate that G-SSL assigns the node to more than one class.

5.3.2 Results and discussion

Classification using strict experts

In Table 5.3, we report the classification given by PageRank and L^γ-PageRank ($\gamma = 2$) in the estimation of the ASes from the IP network displayed in Figure 5.5 when the labelled data is given by the strict expert. The table reports the best classification attained by each method for a grid of μ values. Green cells refer to correct inferences and red cells to incorrect inferences. Cells having n.a. indicate that the node is not assigned to any class by G-SSL. Cells having more than one AS tag indicate the node was assigned to more than one class by G-SSL.

Table 5.3 indicates that, for the strict expert, L^γ-PageRank ($\gamma = 2$) with sweep-cuts delivers the best possible prediction permitted by the labelled data. Notice that we lack

annotated examples for AS4, therefore G-SSL does not know about the existence of such class and can be prone to assign nodes of AS4 (IDs: 8,9,22) to one of the other classes. Notably, L^γ-PageRank with $\gamma = 2$ and the sweep-cuts (designed to identify classes individually), classify the data in a way that all the nodes belonging to either AS1, AS2, or AS3 are assigned into their correct class, and nodes belonging to AS4 (IDs: 8,9,22) are left unclassified, leaving open the possibility that another class whose label was not collected exists. On the contrary, doing sweeps on the standard PageRank method ($\gamma = 1$) leads to various errors. First, it retrieves some classes with numerous incorrect ASes, leading to 58% of nodes to be assigned into more than one class. One of such multiple classes is always the correct one, yet, in practice, there is no way to remove the ambiguity arising from two classes being assigned to a node, thus counting as a miss-classification. In addition, standard PageRank is not able to realize that nodes with IDs: 8,9,22 all belong to a missing class (AS4), incorrectly assigning them to class AS1.

To have a better grasp on why standard PageRank is prone to miss-classify the data and why L^γ-PageRank delivers such reliable inferences, let us show next the sorting of nodes attained by the classification functions (degree-normalized PageRank vectors) of both methods when they are computed using the labelled points of AS1:

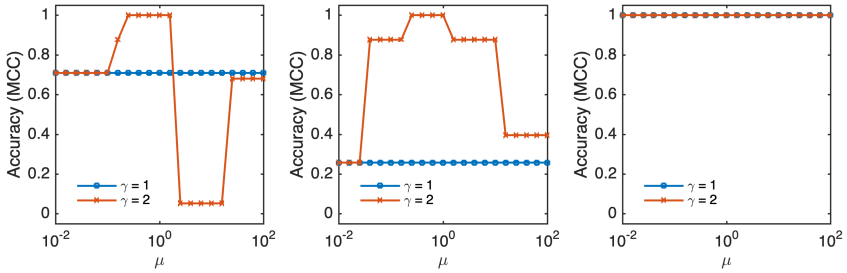
Sorting by standard PageRank ($\gamma = 1$):

$$q = [\underbrace{12, 11, 16, 10}_{\text{AS1}}, \underbrace{9, 22, 8}_{\text{AS4}}, \underbrace{7, 15, 6, 14, 17, 23, 26, 27, 28, 29, 25, 18, 20, 19, 21, 24, 5}_{\text{AS3}}, \underbrace{13, 4, 3, 2, 1}_{\text{AS2}}]$$

Sorting by L^γ-PageRank ($\gamma = 2$):

$$q = [\underbrace{12, 16, 11, 10}_{\text{AS1}}, \underbrace{5, 6, 14, 19, 20, 21, 26, 17, 18, 23, 24, 25, 27, 28, 29, 7, 15, 1, 2, 3, 13, 4}_{\text{AS3}}, \underbrace{8, 9, 22}_{\text{AS4}}]$$

Please recall that this sorting step is key in the sweep-cut partitioning procedure (see Algorithm 1 in Section 2.3, page 36): the first element in the listing defines a set, the first and second elements define a second set, this is done successively until N sets are collected, and upon which Cheeger ratios are computed and the set one with smallest is returned as the G-SSL classification. Thus, we can see that standard PageRank retrieves a classification function that accurately places the nodes belonging to AS1 at the begining, but just right next to them it places the nodes of AS4. If we observe in detail the graph from Figure 5.5, we can see that, while nodes of AS1 (IDs: 12,11,16,10) define a cluster with small Cheeger ratio, the nodes of AS1 together with those of AS4 (IDs:8,9,22) define an even better cluster that is more balanced and has a smaller Cheeger ratio. Hence, the sweep retrieves $AS1 \cup AS4$ as a partition, which results in the the miss-classification of the nodes of AS4. On the other hand, if we look at the sorting given by L^γ-Pagerank with $\gamma = 2$, we note the nodes of AS1 are placed correctly at the beginning, then, next to AS1, we see the nodes of AS3, and the nodes of AS4 have now been deferred to the tail of the sorting. Therefore, the partition $AS1 \cup AS4$ is no longer among the possible partitions. Indeed, when we consider any mixture of AS1 with the nodes of AS3, such sets never define a cluster of small Cheeger ratio. The only possible set of small Cheeger ratio in the search space is AS4, hence it is accurately retrieved. Clearly, the effect of placing AS4 at the tail of the sorting is due to the emergence of negative edges between the nodes



(a) AS 1 (b) AS 2 (c) AS 3

Figure 5.6: Classification accuracy of each AS as a function of μ using sweep-cuts on the annotated examples of the *strict expert*. No annotated examples are available for AS4.

of AS1 and AS4, which indicate that they should constitute opposite classes.

Now, let us focus on the performance obtained by classifying the data using the multi-class approach. Table 5.3 shows that both standard PageRank and L^γ-PageRank accurately classify all nodes of AS1, AS2 and AS3, but they miss-classify the nodes of AS4. As discussed above, one of the drawbacks of the multi-class approach is that it always assigns a class to nodes, forcing us to collect annotated examples of all classes to avoid entire classes from being miss-classified, which, in this example, is what happens with the nodes of AS4.

For a more fair and complete comparison, we display in Figure 5.6 the performance accuracy of classifications given by the sweep-cuts (individually, for each class) as a function of the G-SSL regularization parameter μ. Figure 5.9a, does the same for the multi-class classification. From these figures, we observe that the standard PageRank algorithm is, in general, not very sensitive to changes in μ. On the other hand, $\gamma = 2$ displays more sensitivity to variations in μ, implying that efforts must be employed for its optimal selection. Despite the need for its tuning, the figure indicates that L^γ-PageRank has a superior performance for a not so narrow regime of μ values.

Classification using loose experts

Table 5.4 reports the classification obtained by standard PageRank and L^γ-PageRank ($\gamma = 2$) in the estimation of the ASes from the IP network displayed in Figure 5.5 when the labelled data is given by the loose expert. The table reports the best classification attained by each method for a grid of μ values. Green cells refer to correct inferences and red cells to incorrect inferences. Cells having more than one AS tag indicate that the node was assigned to more than one class by G-SSL.

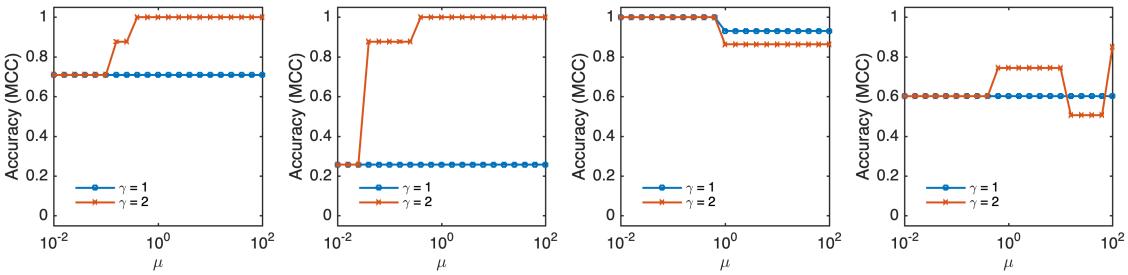
The table indicates that the introduction of the additional (non-necessarily reliable) labelled points, which make expertized data for AS4 available, allows L^γ-PageRank, in the multi-class setting, to achieve a perfect a classification result, and, in the sweep-cut setting, to only fail in the assignment of one single node. Indeed, it can be seen that the sole node for which the sweep-cut fails (ID: 10) is due to it being incorrectly annotated by the loose expert. Nevertheless, it is remarkable that the multi-class approach profits from the loose experts and is able to relabel all the nodes incorrectly labelled into their correct

ID	IP	True AS	Labels				Sweep-cut		Multi-class	
			AS1	AS2	AS3	AS4	$\gamma = 1$	$\gamma = 2$	$\gamma = 1$	$\gamma = 2$
(1)	10.6.66.1	AS2	0	0	0	0	AS2	AS2	AS2	AS2
(2)	62.214.63.145	AS2	0	1	0	0	AS2	AS2	AS2	AS2
(3)	62.214.36.177	AS2	0	1	0	0	AS2	AS2	AS2	AS2
(4)	62.214.37.130	AS2	0	1	0	0	AS2	AS2	AS2	AS2
(5)	213.155.129.188	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(6)	62.115.141.236	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(7)	62.115.120.0	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(8)	213.248.68.71	AS4	0	0	1	0	AS1, AS4	AS4	AS3	AS4
(9)	63.223.34.74	AS4	0	0	0	1	AS1, AS4	AS4	AS4	AS4
(10)	63.217.25.146	AS1	0	0	0	1	AS1, AS4	AS1, AS4	AS4	AS1
(11)	139.162.0.10	AS1	1	0	0	0	AS1, AS4	AS1	AS1	AS1
(12)	139.162.27.28	AS1	1	0	0	0	AS1, AS4	AS1	AS1	AS1
(13)	62.214.37.134	AS2	0	1	0	0	AS2	AS2	AS2	AS2
(14)	62.115.137.168	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(15)	62.115.120.6	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(16)	139.162.0.2	AS1	1	0	0	0	AS1, AS4	AS1	AS1	AS1
(17)	62.115.137.166	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(18)	62.115.121.2	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(19)	62.115.137.164	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(20)	62.115.141.238	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(21)	62.115.141.240	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(22)	63.223.34.138	AS4	0	0	0	1	AS1, AS4	AS4	AS4	AS4
(23)	62.115.121.8	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(24)	62.115.121.4	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(25)	62.115.141.234	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(26)	62.115.121.10	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(27)	62.115.116.159	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(28)	62.115.116.163	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(29)	62.115.121.6	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3

Table 5.4: Best classification attained by G-SSL using the *loose expert* labelled data. Green cells refer to correct inferences and red cells to incorrect inferences. Cells with more than one tag indicate that G-SSL assigns the node to more than one class.

class. Concerning the standard PageRank algorithm, we can see that it does not really takes advantage from the loose experts. On the one hand side, the two nodes incorrectly labelled by the expert (ID: 8,10) are the ones where PageRank fails (multi-class approach), hence lacks the ability that L^γ-PageRank has of to relabeling data into their correct class. On the other hand, we see that the loose expert worsens, with respect to the strict expert, the classification via sweep-cuts as we see an increment of nodes assigned into more than one class from 58% to 82%.

Lastly, we display in Figure 5.7 the accuracy of classifications given by the sweep-cuts (individually, for each class) as a function of the G-SSL regularization parameter μ. In Figure 5.9b, we display the same for the multi-class classification approach. These figures still verify that the standard PageRank algorithm tends to not display changes in its performance with changes in μ. In the case of $\gamma = 2$, it can be seen that for the sweep on the labels of AS4, more care needs to be taken in the tuning of μ with respect to other ASes. On the multi-class setting, $\gamma = 2$ attains perfect accuracy but the regime of μ values that



(a) AS 1 (b) AS 2 (c) AS 3 (d) AS 4

Figure 5.7: Classification accuracy of each AS as a function of μ using sweep-cuts on the annotated examples of the *loose expert*.

permits for it is narrow. In general, other values of μ do not attain perfect accuracy, but for most values $\gamma = 2$ still displays a superior performance over $\gamma = 1$.

Classification using weighted experts

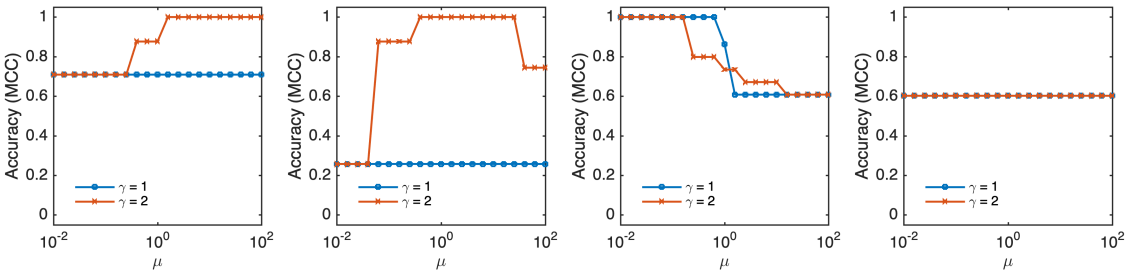
Table 5.5 reports the classification obtained by standard PageRank and L^γ-PageRank ($\gamma = 2$) in the estimation of the ASes from the IP network displayed in Figure 5.5 when the labelled data is given by the weighted expert. The table reports the annotations given by the weighted expert and the weight assigned to each labelled example. The table reports the best classification attained by each method for a grid of μ values. Green cells refer to correct inferences and red cells to incorrect inferences. Cells having more than one AS tag indicate that the node was assigned to more than one class by G-SSL.

The table indicates that L^γ-PageRank ($\gamma = 2$) still outperforms standard PageRank, yet the weighted expert does not permit L^γ-PageRank to attain predictions without errors. Indeed, the weighted expert worsens the performance of L^γ-PageRank with respect to the loose expert. For instance, the classification via sweep-cuts now increases the number of nodes assigned to more than one class. We note that such miss-classifications arise from the sweep for AS4, which detects all the nodes from AS1 as being of class AS4. The error is due to unreliable annotated examples of AS4, in which the weighted expert incorrectly says that 3 out of the 4 nodes belonging to AS1 are of class AS4. This illustrates the limits of G-SSL to errors in annotated data. It is remarkable that L^γ-PageRank, in the multi-class approach, is able to override most of the incorrectly labelled points by just failing to accurately detect the AS of one node (ID: 10). Standard PageRank displays no difference between the loose expert and the weighted expert. The number of nodes assigned to more than one class via sweeps-cuts remains in 58%, and the multi-class approach still miss-classifies the same two vertices (IDs: 8,10).

The accuracy of classifications given by the sweep-cuts (individually, for each class) as a function of the G-SSL regularization parameter μ is shown in Figure 5.8. Figure 5.9c reports the same for the multi-class approach. The figures indicate that, for the weighted expert, standard PageRank remains not very sensitive to μ and L^γ-PageRank is more sensitive, but displays a superior performance for a wide range of μ values.

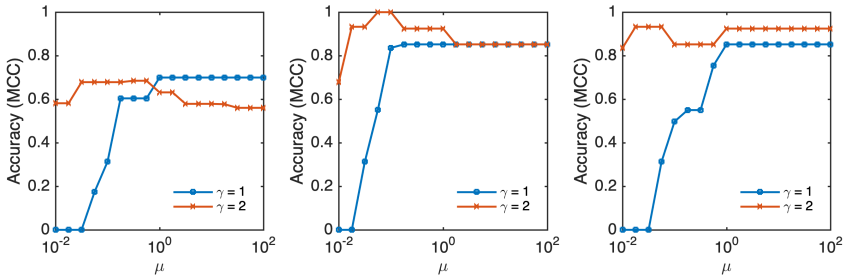
ID	IP	True AS	Labels				Sweep-cut		Multi-class	
			AS1	AS2	AS3	AS4	$\gamma = 1$	$\gamma = 2$	$\gamma = 1$	$\gamma = 2$
(1)	10.6.66.1	AS2	0	0	0	0	AS2	AS2	AS2	AS2
(2)	62.214.63.145	AS2	0	0.5	0	0	AS2	AS2	AS2	AS2
(3)	62.214.36.177	AS2	0	1	0	0	AS2	AS2	AS2	AS2
(4)	62.214.37.130	AS2	0	0.5	0.5	0	AS2	AS2	AS2	AS2
(5)	213.155.129.188	AS3	0	0.5	0.5	0	AS2, AS3	AS3	AS3	AS3
(6)	62.115.141.236	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(7)	62.115.120.0	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(8)	213.248.68.71	AS4	0	0	0.5	0.5	AS1, AS4	AS4	AS3	AS4
(9)	63.223.34.74	AS4	0	0	0.5	0.5	AS1, AS4	AS4	AS4	AS4
(10)	63.217.25.146	AS1	0.5	0	0	0.5	AS1, AS4	AS1, AS4	AS4	AS4
(11)	139.162.0.10	AS1	0.5	0	0	0.5	AS1, AS4	AS1, AS4	AS1	AS1
(12)	139.162.27.28	AS1	1	0	0	0	AS1, AS4	AS1, AS4	AS1	AS1
(13)	62.214.37.134	AS2	0	0.5	0.5	0	AS2	AS2	AS2	AS1
(14)	62.115.137.168	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(15)	62.115.120.6	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(16)	139.162.0.2	AS1	0.5	0	0	0.5	AS1, AS4	AS1, AS4	AS1	AS1
(17)	62.115.137.166	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(18)	62.115.121.2	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(19)	62.115.137.164	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(20)	62.115.141.238	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(21)	62.115.141.240	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(22)	63.223.34.138	AS4	0	0	0.5	0.5	AS1, AS4	AS4	AS4	AS4
(23)	62.115.121.8	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(24)	62.115.121.4	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(25)	62.115.141.234	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(26)	62.115.121.10	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(27)	62.115.116.159	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(28)	62.115.116.163	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3
(29)	62.115.121.6	AS3	0	0	1	0	AS2, AS3	AS3	AS3	AS3

Table 5.5: Best classification attained by G-SSL using the *weighted expert* labelled data. The table reports the annotations given by the expert and their weights. Green cells refer to correct inferences and red cells to incorrect inferences. Cells with more than one tag indicate that G-SSL assigns the node to more than one class.



(a) AS 1 (b) AS 2 (c) AS 3 (d) AS 4

Figure 5.8: Classification accuracy of each AS as a function of μ using sweep-cuts on the annotated examples of the *weighted expert*.



(a) Strict expert (b) Loose expert (c) Weighted expert

Figure 5.9: Classification accuracy of all ASes as a function of μ using the multi-class approach.

Chapter 6

Conclusions

Summary

In this thesis we started with the aim of addressing some of the limitations of G-SSL, which tends to only be reliable under simple data settings. Towards this aim, in Chapter 1, we recalled essential results from graph theory, random walk theory and graph signal processing. Special emphasis was given to the diffusion aspect of random walkers and the heat equation. Moreover, graph signals were introduced and motivations for the different definitions of Laplacian matrices were given along with their implications on the definition and impact on the graph Fourier transform. It was also highlighted that these results permit to extend the concept of filter from signal processing to graph signals.

In chapter 2, we gave a thorough review of the field of G-SSL that we believe was missing. In the first part, we covered the traditional approach to G-SSL as a Tikhonov regularization problem. We reviewed the different structural forcing proposed in the literature and discussed their motivations and relationships. From the reviewed methods, it was argued that the PageRank algorithm is the most performing proposition, yet it suffers from biased outputs when the labelled data is unbalanced. We covered works showing that G-SSL suffers from the curse of dimensionality issue when the unlabelled data grows infinitely large. We introduced results indicating that the source of the issue is the Laplacian kernel and we gave special emphasis to the solution based on Laplacian iterations as it was a source of motivation for this work. In the second part of Chapter 2, we pointed that G-SSL can also be addressed from the perspective of graph partitioning. We introduced graph-cut problems and stressed that they are unfeasible to solve exactly, but that can be relaxed by G-SSL. Then, we revisited and re-interpreted results originally developed in the context of local graph clustering to show that the PageRank method, in the G-SSL framework, can also be used to confine diffusion processes and reveal clusters in the graph. Then, it was stressed that the so-called sweep-cut technique complements those results and impacts G-SSL overriding the need to collect balanced labelled points for all classes by allowing to find them individually.

Chapter 3 proposed our main contribution: the L^γ-PageRank G-SSL method, an extension of PageRank based on (non-necessarily integers) powers of the (combinatorial) Laplacian matrix. The analysis given in the Chapter shows that the added degree of freedom offers more versatility than standard PageRank, providing the potential to address some of the limitations of G-SSL. Precisely, we showed that when clusters are obtained via

the sweep-cut procedure, L^γ-PageRank can significantly outperform standard PageRank. Furthermore, we showed that the multi-class approach also benefits from our proposition, as our method significantly overrides the issue of unbalanced labelled data. These improvements were possible due to a novel interpretation we did of the L^γ operator, in which we showed that for each value of γ, it gives rise to new graphs with the potential to improve G-SSL. We showed that two regimes of new graphs appear: (i) $\gamma < 1$ in which graphs of positive edges are created and (ii) $\gamma > 1$ that leads to signed graphs. Concerning the regime $\gamma < 1$, we showed that our L^γ-PageRank corresponds to an extension of the regular PageRank algorithm to Lévy processes, in which random walkers are given the ability to perform long-distant jumps in a single step. We showed by means of numerical experimentation that the Lévy flights can improve the classification functions when the data presents complex local structures. Concerning the regime of $\gamma > 1$, we showed that the richness of such graphs comes from the sign of edges, allowing to code for similarities but also to emphasize dissemblance between nodes in the graph. Thus, while 2 nodes can only be disconnected on the initial graph, they can 'repulse' themselves in these topologies. Notably, we have shown that there is an optimal graph (related to an optimal γ) on which the classification will lead to a maximal performance. We proposed a simple yet efficient algorithm to estimate the optimal γ and hence determine the best topology for analyzing a given dataset.

Chapter 4 investigated fast and efficient implementations that avoid the costly matrix inversion demanded by G-SSL methods. We contributed with an extension of the successful algorithms of power iteration and Gauss-Southwel, state-of-the-art methods for fast PageRank computation, to L^γ-PageRank. Moreover, we extended the dynamic versions of such algorithms to L^γ-PageRank, allowing us to update the solution of L^γ-PageRank when facing evolving graph structures at a much smaller cost than re-computing the solution from scratch every time the graph changes. These extensions were possible because L^γ-PageRank was shown to be a low pass graph signal filter, for which the field of Graph Sinal Processing had proposed efficient implementations via ARMA recursions and Chebyshev polynomials. We elaborated on the ARMA recursions to extend the aforementioned algorithms. Our assessment showed that, while the Chebyshev polynomials remain the most effective approach to compute the solution of L^γ-PageRank from scratch, they do not have the key feature of the ARMA-based algorithms: the warm restart, which proves instrumental to accelerate the computation of L^γ-PageRank by several orders of magnitude when the graph evolves. In addition, we explored the feasibility of using neural networks to solve the update problem. We addressed the question that if we show a sequence of graphs with their exact G-SSL solutions to a neural network, then can this learn the mapping to update the G-SSL solution when the graph evolves? We proposed a procedure to encode for graph changes as features living on the graph vertices that are later used to train a Multi-Layer Perception. Our preliminary results indicate that the neural network bears the potential to effectively update the solution of G-SSL as it was able to outperform the analytic ARMA expression for an equal computational complexity.

Chapter 5 used G-SSL to address current issues in Internet routing. We used G-SSL to provide the first characterization of BGP zombies. Then, we used G-SSL to solve the IP to AS mapping challenge. For our characterization of zombies, our study spans for over a year and a half of data collected in a controlled setting via RIS beacons and RIS

collectors. From RIS collectors we tracked zombie outbreaks and constructed a set of annotated examples that were fed into G-SSL to predict the scope of outbreaks beyond RIS peers. G-SSL (std. PageRank) inferred zombie ASes with 97% accuracy and normal ASes with 99% accuracy according to a validation dataset build from tracerout measurements. G-SSL inferences were used to then characterize the scope of zombie outbreaks. Results indicate that oubreaks affect, on average, 10% (IPv4) and 17% (IPv6) of the total ASes in our dataset. Moreover, we found that the scope of outbreaks is related to the importance of transit networks affected. Then, we used L^γ-PageRank to solve the IP to AS mapping challenge. We built a graph from tracerout measruements and constructed labelled datasets using the publicly available ASNs with varying degrees of reliability. We showed that standard PageRank always miss-classified data, while L^γ-PageRank ($\gamma = 2$) can solve the task without errors. We showed that a strict expert may be very reliable but may not collect annotated examples from all classes. In such cases, we showed that sweep-cuts offer the best solution to classify data, allowing to leave unclassified nodes whose label was not collected. Then, we showed that when the expert is unreliable, classifying data in the multi-class setting can override incorrect labelled points, while sweep-cuts are likely to assign nodes to more than one class.

Future research

The procedures proposed in this work open various research directions. The first of them would be the extension of other standard clustering tools, such as Unsupervised Learning via Spectral Clustering, to exploit the richer L^γ topologies proposed in this work. Indeed, the potential attained by the L^γ-graphs to be applied in other contexts calls for a more in-depth study on what determines their optimal topology. We gave initial results indicating why $\gamma = 2$, which creates negative edges between two-hop distant nodes, normally constitutes a reliable topology, yet larger γ values minimize the Cheeger ratio even more before it starts increasing. This lack of insights on what topological properties determine the optimal γ opens a research direction. In the same line of the optimal topology, improvements over our algorithm for the optimal estimation of γ should also be sought. For instance, our algorithm has to find a set via random walks with an empirically chosen threshold of 0.7, which is far from being optimal. In addition, we recall that the notion of optimal γ remains valid only for the sweep-cut partition technique and that insights on what determines the optimal γ for the multi-class approach are missing. Another calling research direction is the interplay between γ and σ in the generalized optimization framework. It may be interesting to see if such framework can lead to novel partition algorithms that find clusters with other metrics more significant than the Cheeger ratio. Still, on the parameter side, we stress that G-SSL also depends on the selection of the μ parameter for which we lack insights on how to optimally tune. It may be interesting to see if one can address the optimal μ tuning from the results on the sweep-cut, suggesting that one may chose the best μ as the one that leads to the sharpest drop in the sorted scores. In a different direction, we showed that G-SSL as a diffusion process is driven by a Helmholtz PDE equation. Therefore, it is calling to explore if other diffusion processes driven by other master equations can be more meaningful for certain applications. Indeed, still on the diffusion side, it is an open problem to see if the L^γ-PageRank, for the regime $\gamma > 1$, can be given an interpretation as random walkers operating on signed graphs. Lastly, we recall that a fundamental open problem in G-SSL is the relationship between the other G-SSL propositions covered in this thesis with the graph topology and if the sweeps can

be reliably extended to them.

From the algorithmic side, we have left various open problems that call for immediate further research. The principal is the efficient computation of the L^γ-graphs. The fact that L^γ-PageRank requires to re normalize by the generalized degree matrix D_γ implies that one needs to estimate the latter via recursive matrix-matrix products. We showed that $\gamma = 2$ overrides this issue because it has a closed form expression consisting on information only from the one-hop vicinity of nodes, however, for larger powers one needs information from far-distant nodes. One possible way to amend the issue can be to find closed form expressions for D_γ ($\gamma \in \mathbb{Z}$) and then design distributed techniques so that each node collects the necessary information to compute D_γ. Indeed, another concern is that we are not aware of any way to estimate the L^γ-graphs for fractional powers without relying on eigendecompositions. This challenges pave the way to explore other approaches beyond the closed form solutions to compute G-SSL. At the end, G-SSL is just a function on the graph vertices that is estimated from the labelled points according to some optimization problem, thus it may be worth exploring other approaches to reach such function, such as neural networks. Concerning neural networks, we have proposed one to solve the update problem and, while the idea is delivering promising results, it still operates at the entire graph level. Thus, it calls for a local extension that allows to obtain sublinear complexity in order to be competitive with the ARMA-based updating algorithms. Indeed, it would be of utmost importance to find a way to recast the Chebyshev method in a way that it can use warm restarts due to its high efficiency.

Concerning our characerization of BGP zombies, the immediate work to perform is to verify how much our results are reminiscent of what occurs in wild. It is important to assess if detection of zombies in the wild demands to go beyond the standard PageRank algorithm. In such case, L^γ-PageRank arises as a natural method to consider. With respect to the IP to AS mapping problem, the immediate research direction is to extrapolate our experiments to a larger dataset as we only provided a proof of concept in a small of graph of 29 IP addresses. Larger datasets may better reveal differences/advantages of the various experts and may better expose the limits of using $\gamma = 2$ (which we stuck to it due to its easy interpretation and excellent results in our dataset). In such case, we must consider other γ values and also assess the relevance our algorithm for the automatic estimation of the optimal γ in such real world scenario.

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[2] R. C. Deo, "Machine learning in medicine," *Circulation*, vol. 132, no. 20, pp. 1920–1930, 2015.

[3] B. C. Bonoto, V. E. de Araújo, I. P. Godói, L. L. P. de Lemos, B. Godman, M. Bennie, L. M. Diniz, and A. A. G. Junior, "Efficacy of mobile apps to support the care of patients with diabetes mellitus: a systematic review and meta-analysis of randomized controlled trials," *JMIR mHealth and uHealth*, vol. 5, no. 3, p. e4, 2017.

[4] J. E. Mück, B. Ünal, H. Butt, and A. K. Yetisen, "Market and patent analyses of wearables in medicine," *Trends in biotechnology*, vol. 37, no. 6, pp. 563–566, 2019.

[5] K. Avrachenkov, P. Gonçalves, A. Legout, and M. Sokol, "Classification of content and users in bittorrent by semi-supervised learning methods," in *International Wireless Communications and Mobile Computing Conference (3rd International Workshop on Traffic Analysis and Classification)*, (Cyprus), August 2012. Best paper award.

[6] A. Subramanya and J. Bilmes, "Soft-supervised learning for text classification," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, (Stroudsburg, PA, USA), pp. 1090–1099, Association for Computational Linguistics, 2008.

[7] W. Hu, J. Gao, J. Xing, C. Zhang, and S. Maybank, "Semi-supervised tensor-based graph embedding learning and its application to visual discriminant tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 1, pp. 172–188, 2016.

[8] H. Cecotti, "Active graph based semi-supervised learning using image matching: application to handwritten digit recognition," *Pattern Recognition Letters*, vol. 73, pp. 76–82, 2016.

[9] F. De Morsier, M. Borgeaud, V. Gass, J.-P. Thiran, and D. Tuia, "Kernel low-rank and sparse graph for unsupervised and semi-supervised classification of hyperspectral images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 6, pp. 3410–3420, 2016.

[10] M. Sokol, *Graph-based semi-supervised learning methods and quick detection of central nodes*. Theses, Université Nice Sophia Antipolis, Apr. 2014.

[11] R. Andersen, F. Chung, and K. Lang, "Local graph partitioning using pagerank vectors," in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pp. 475–486, IEEE, 2006.

[12] M. Belkin, I. Matveeva, and P. Niyogi, "Regularization and semi-supervised learning on large graphs," in *International Conference on Computational Learning Theory*, pp. 624–638, Springer, 2004.

[13] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, pp. 321–328, 2004.

[14] D. Zhou and C. J. Burges, "Spectral clustering and transductive learning with multiple views," in *Proceedings of the 24th international conference on Machine learning*, pp. 1159–1166, ACM, 2007.

[15] X. Zhou and M. Belkin, "Semi-supervised learning by higher order regularization," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (G. Gordon, D. Dunson, and M. Dudk, eds.), vol. 15 of *Proceedings of Machine Learning Research*, (Fort Lauderdale, FL, USA), pp. 892–900, PMLR, 11–13 Apr 2011.

[16] R. Andersen, F. R. K. Chung, and K. J. Lang, "Using pagerank to locally partition a graph," *Internet Mathematics*, vol. 4, pp. 35–64, 01 2007.

[17] R. Andersen and F. Chung, "Detecting sharp drops in pagerank and a simplified local partitioning algorithm," in *Theory and Applications of Models of Computation* (J.-Y. Cai, S. B. Cooper, and H. Zhu, eds.), (Berlin, Heidelberg), pp. 1–12, Springer Berlin Heidelberg, 2007.

[18] G. Jeh and J. Widom, "Scaling personalized web search," in *Proceedings of the 12th international conference on World Wide Web*, pp. 271–279, Acm, 2003.

[19] T. Maehara, T. Akiba, Y. Iwata, and K.-i. Kawarabayashi, "Computing personalized pagerank quickly by exploiting graph structures," *Proceedings of the VLDB Endowment*, vol. 7, no. 12, pp. 1023–1034, 2014.

[20] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[21] P. Berkhin, "Bookmark-coloring algorithm for personalized pagerank computing," *Internet Mathematics*, vol. 3, no. 1, pp. 41–62, 2006.

[22] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Transactions on Signal Processing*, vol. 65, no. 2, pp. 274–288, 2016.

[23] D. I. Shuman, P. Vandergheynst, and P. Frossard, "Chebyshev polynomial approximation for distributed signal processing," in *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1–8, IEEE, 2011.

[24] M. Yoon, W. Jin, and U. Kang, "Fast and accurate random walk with restart on dynamic graphs with guarantees," in *Proceedings of the 2018 World Wide Web Conference*, pp. 409–418, International World Wide Web Conferences Steering Committee, 2018.

[25] N. Ohsaka, T. Maehara, and K.-i. Kawarabayashi, "Efficient pagerank tracking in evolving networks," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 875–884, ACM, 2015.

[26] B. Hayes, "Computing science: Graph theory in practice: Part i," *American Scientist*, vol. 88, no. 1, pp. 9–13, 2000.

[27] J. M. Harris, J. L. Hirst, and M. J. Mossinghoff, *Combinatorics and graph theory*, vol. 2. Springer, 2008.

[28] R. Albert, H. Jeong, and A.-L. Barabási, "Internet: Diameter of the world-wide web," *nature*, vol. 401, no. 6749, p. 130, 1999.

[29] J. Scott, "Social network analysis," *Sociology*, vol. 22, no. 1, pp. 109–127, 1988.

[30] P. Gai and S. Kapadia, "Contagion in financial networks," *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 466, no. 2120, pp. 2401–2423, 2010.

[31] F. Wang, U. Srinivasan, S. Uddin, and S. Chawla, "Application of network analysis on healthcare," in *Proceedings of the 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pp. 596–603, IEEE Press, 2014.

[32] J.-F. Rual, K. Venkatesan, T. Hao, T. Hirozane-Kishikawa, A. Dricot, N. Li, G. F. Berriz, F. D. Gibbons, M. Dreze, N. Ayivi-Guedehoussou, *et al.*, "Towards a proteome-scale map of the human protein–protein interaction network," *Nature*, vol. 437, no. 7062, p. 1173, 2005.

[33] Z. Zhang and M.-Y. Chow, "Convergence analysis of the incremental cost consensus algorithm under different communication network topologies in a smart grid," *IEEE Transactions on Power Systems*, vol. 27, no. 4, pp. 1761–1768, 2012.

[34] J. Costantine, S. Al-Saffar, C. G. Christodoulou, K. Y. Kabalan, and A. El-Hajj, "The analysis of a reconfigurable antenna with a rotating feed using graph models," *IEEE Antennas and Wireless Propagation Letters*, vol. 8, pp. 943–946, 2009.

[35] B. Krishnamachari, D. Estrin, S. B. Wicker, *et al.*, "The impact of data aggregation in wireless sensor networks.," in *ICDCS workshops*, vol. 578, 2002.

[36] R. M. Gray *et al.*, "Toeplitz and circulant matrices: A review," *Foundations and Trends® in Communications and Information Theory*, vol. 2, no. 3, pp. 155–239, 2006.

[37] A. Condon and R. M. Karp, "Algorithms for graph partitioning on the planted partition model," *Random Structures & Algorithms*, vol. 18, no. 2, pp. 116–140, 2001.

[38] D. Ghoshdastidar and A. Dukkipati, "Consistency of spectral partitioning of uniform hypergraphs under planted partition model," in *Advances in Neural Information Processing Systems*, pp. 397–405, 2014.

[39] C. Tsourakakis, "Streaming graph partitioning in the planted partition model," in *Proceedings of the 2015 ACM on Conference on Online Social Networks*, pp. 27–35, ACM, 2015.

[40] E. Mossel, J. Neeman, and A. Sly, "Reconstruction and estimation in the planted partition model," *Probability Theory and Related Fields*, vol. 162, pp. 431–461, Aug 2015.

[41] 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*, vol. 84, no. 6, p. 066106, 2011.

[42] E. Abbe, "Community detection and stochastic block models: Recent developments," *Journal of Machine Learning Research*, vol. 18, no. 177, pp. 1–86, 2018.

[43] A. N. Langville and C. D. Meyer, *Google's PageRank and beyond: The science of search engine rankings*. Princeton University Press, 2011.

[44] M. E. Newman, "A measure of betweenness centrality based on random walks," *Social networks*, vol. 27, no. 1, pp. 39–54, 2005.

[45] M. H. Ribeiro, P. H. Calais, V. A. Almeida, and W. Meira Jr, "" everything i disagree with is# fakenews": Correlating political polarization and spread of misinformation," *arXiv preprint arXiv:1706.05924*, 2017.

[46] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *International symposium on computer and information sciences*, pp. 284–293, Springer, 2005.

[47] W. Wei and B. Selman, "Accelerating random walks," in *International Conference on Principles and Practice of Constraint Programming*, pp. 216–232, Springer, 2002.

[48] M. Molloy and B. Reed, "A critical point for random graphs with a given degree sequence," *Random structures & algorithms*, vol. 6, no. 2-3, pp. 161–180, 1995.

[49] F. Neumann and C. Witt, "Ant colony optimization and the minimum spanning tree problem," *Theoretical Computer Science*, vol. 411, no. 25, pp. 2406–2413, 2010.

[50] D. Aldous and J. A. Fill, "Reversible markov chains and random walks on graphs, 2002. unfinished monograph, recompiled 2014," 2002.

[51] L. Lovász *et al.*, "Random walks on graphs: A survey," *Combinatorics, Paul erdos is eighty*, vol. 2, no. 1, pp. 1–46, 1993.

[52] S. Chen, A. Sandryhaila, G. Lederman, Z. Wang, J. M. Moura, P. Rizzo, J. Bielak, J. H. Garrett, and J. Kovačević, "Signal inpainting on graphs via total variation minimization," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8267–8271, IEEE, 2014.

[53] G. Arvanitis, A. S. Lalos, K. Moustakas, and N. Fakotakis, "Feature preserving mesh denoising based on graph spectral processing," *IEEE transactions on visualization and computer graphics*, vol. 25, no. 3, pp. 1513–1527, 2019.

[54] M. Onuki, S. Ono, M. Yamagishi, and Y. Tanaka, "Graph signal denoising via trilateral filter on graph spectral domain," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 137–148, 2016.

[55] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Autoregressive moving average graph filtering," *IEEE Transactions on Signal Processing*, vol. 65, no. 2, pp. 274–288, 2017.

[56] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *IEEE Transactions on Signal Processing*, vol. 65, no. 22, pp. 5911–5926, 2017.

[57] S. Chen, A. Sandryhaila, and J. Kovačević, "Distributed algorithm for graph signal inpainting," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3731–3735, IEEE, 2015.

[58] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5445–5449, IEEE, 2013.

[59] M. Tsitsvero, S. Barbarossa, and P. Di Lorenzo, "Signals on graphs: Uncertainty principle and sampling," *IEEE Transactions on Signal Processing*, vol. 64, no. 18, pp. 4845–4860, 2016.

[60] N. Perraudin and P. Vandergheynst, "Stationary signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3462–3477, 2017.

[61] N. Tremblay and P. Borgnat, "Graph wavelets for multiscale community mining," *IEEE Transactions on Signal Processing*, vol. 62, no. 20, pp. 5227–5239, 2014.

[62] M. Rizkallah, X. Su, T. Maugey, and C. Guillemot, "Graph-based transforms for predictive light field compression based on super-pixels," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1718–1722, IEEE, 2018.

[63] C. Zhang, D. Florencio, and C. Loop, "Point cloud attribute compression with graph transform," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 2066–2070, IEEE, 2014.

[64] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in neural information processing systems*, pp. 3844–3852, 2016.

[65] M. Zhang, Z. Cui, M. Neumann, and Y. Chen, "An end-to-end deep learning architecture for graph classification," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[66] A. Elmoataz, O. Lezoray, and S. Bougleux, "Nonlocal Discrete Regularization on Weighted Graphs: A Framework for Image and Manifold Processing," *IEEE Transactions on Image Processing*, vol. 17, pp. 1047–1060, July 2008.

[67] 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," *arXiv preprint arXiv:1211.0053*, 2012.

[68] X. Zhu and A. B. Goldberg, "Introduction to semi-supervised learning," *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.

[69] A. Subramanya and P. P. Talukdar, "Graph-based semi-supervised learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 4, pp. 1–125, 2014.

[70] M. Zhao, R. H. M. Chan, T. W. S. Chow, and P. Tang, "Compact graph based semi-supervised learning for medical diagnosis in alzheimers disease," *IEEE Signal Processing Letters*, vol. 21, pp. 1192–1196, Oct 2014.

[71] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[72] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[73] Y. Bengio, O. Delalleau, and N. Le Roux, *Label Propagation and Quadratic Criterion*, pp. 193–216. MIT Press, semi-supervised learning ed., January 2006.

[74] X. Mai, *Methods of random matrix for large dimensional statistical learning.* PhD thesis, Université Paris-Saclay, oct 2019.

[75] K. Avrachenkov, A. Mishenin, P. Gonçalves, and M. Sokol, "Generalized optimization framework for graph-based semi-supervised learning," in *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 966–974, 2012.

[76] X. Mai and R. Couillet, "Revisiting and improving semi-supervised learning: A large dimensional approach," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3547–3551, May 2019.

[77] B. Nadler, N. Srebro, and X. Zhou, "Statistical analysis of semi-supervised learning: The limit of infinite unlabelled data," in *Advances in Neural Information Processing Systems*, pp. 1330–1338, 2009.

[78] F. R. Chung and F. C. Graham, *Spectral graph theory.* No. 92, American Mathematical Soc., 1997.

[79] L. Lovász and M. Simonovits, "The mixing rate of markov chains, an isoperimetric inequality, and computing the volume," in *Proceedings [1990] 31st annual symposium on foundations of computer science*, pp. 346–354, IEEE, 1990.

[80] D. A. Spielman and S.-H. Teng, "Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems," in *Proceedings of the STOC*, vol. 4, 2004.

[81] Z. Xiaojin and G. Zoubin, "Learning from labeled and unlabeled data with label propagation," *Tech. Rep., Technical Report CMU-CALD-02–107, Carnegie Mellon University*, 2002.

[82] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

[83] M. Szummer and T. Jaakkola, "Partially labeled classification with markov random walks," in *Advances in neural information processing systems*, pp. 945–952, 2002.

[84] M. Belkin and P. Niyogi, "Using manifold stucture for partially labeled classification," in *Advances in neural information processing systems*, pp. 953–960, 2003.

[85] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly, "Video suggestion and discovery for youtube: taking random walks through the view graph," in *Proceedings of the 17th international conference on World Wide Web*, pp. 895–904, ACM, 2008.

[86] P. P. Talukdar and K. Crammer, "New regularized algorithms for transductive learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 442–457, Springer, 2009.

[87] M. Orbach and K. Crammer, "Graph-based transduction with confidence," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 323–338, Springer, 2012.

[88] A. Subramanya and J. Bilmes, "Semi-supervised learning with measure propagation," *Journal of Machine Learning Research*, vol. 12, no. Nov, pp. 3311–3370, 2011.

[89] T. H. Haveliwala, "Topic-sensitive pagerank," in *Proceedings of the 11th international conference on World Wide Web*, pp. 517–526, ACM, 2002.

[90] K. Avrachenkov, P. Gonçalves, and M. Sokol, "On the choice of kernel and labelled data in semi-supervised learning methods," in *International Workshop on Algorithms and Models for the Web-Graph*, pp. 56–67, Springer, 2013.

[91] B. Girault, *Signal Processing on Graphs-Contributions to an Emerging Field*. PhD thesis, Lyon, École normale supérieure, 2015.

[92] C. Hu, L. Cheng, J. Sepulcre, G. El Fakhri, Y. M. Lu, and Q. Li, "A graph theoretical regression model for brain connectivity learning of alzheimer's disease," in *2013 IEEE 10th International Symposium on Biomedical Imaging*, pp. 616–619, IEEE, 2013.

[93] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[94] X. Mai and R. Couillet, "The counterintuitive mechanism of graph-based semi-supervised learning in the big data regime," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, (New Orleans, France), 2017.

[95] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[96] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.

[97] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.

[98] M. Stoer and F. Wagner, "A simple min-cut algorithm," *Journal of the ACM (JACM)*, vol. 44, no. 4, pp. 585–591, 1997.

[99] P. Elias, A. Feinstein, and C. Shannon, "A note on the maximum flow through a network," *IRE Transactions on Information Theory*, vol. 2, no. 4, pp. 117–119, 1956.

[100] Y.-C. Wei and C.-K. Cheng, "Towards efficient hierarchical designs by ratio cut partitioning," in *1989 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pp. 298–301, IEEE, 1989.

[101] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Departmental Papers (CIS)*, p. 107, 2000.

[102] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak mathematical journal*, vol. 23, no. 2, pp. 298–305, 1973.

[103] F. Chung, "Four proofs for the Cheeger inequality and graph partition algorithms," in *Proceedings of ICCM*, (Hiroshima, Japan), Citeseer, 2007.

[104] F. Chung, "Pagerank as a discrete green?s function," *Geometry and Analysis I ALM*, vol. 17, pp. 285–302, 2010.

[105] A. Tsiatas, *Diffusion and Clustering on Large Graphs*. PhD thesis, La Jolla, CA, USA, 2012. AAI3513269.

[106] K. Avrachenkov, V. Dobrynin, D. Nemirovsky, S. K. Pham, and E. Smirnova, "Pagerank based clustering of hypertext document collections," in *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 873–874, ACM, 2008.

[107] F. Chung, P. Horn, and A. Tsiatas, "Distributing antidote using pagerank vectors," *Internet Mathematics*, vol. 6, no. 2, pp. 237–254, 2009.

[108] F. C. Graham and A. Tsiatas, "Finding and visualizing graph clusters using pagerank optimization," in *International Workshop on Algorithms and Models for the Web-Graph*, pp. 86–97, Springer, 2010.

[109] F. Chung, P. Horn, and J. Hughes, "Multi-commodity allocation for dynamic demands using pagerank vectors," in *International Workshop on Algorithms and Models for the Web-Graph*, pp. 138–152, Springer, 2012.

[110] F. Chung, A. Tsiatas, and W. Xu, "Dirichlet pagerank and trust-based ranking algorithms," in *International Workshop on Algorithms and Models for the Web-Graph*, pp. 103–114, Springer, 2011.

[111] A. Z. Broder, R. Lempel, F. Maghoul, and J. Pedersen, "Efficient pagerank approximation via graph aggregation," *Information Retrieval*, vol. 9, no. 2, pp. 123–138, 2006.

[112] F. Chung and W. Zhao, "A sharp pagerank algorithm with applications to edge ranking and graph sparsification," in *International Workshop on Algorithms and Models for the Web-Graph*, pp. 2–14, Springer, 2010.

[113] A. P. Riascos and J. L. Mateos, "Fractional dynamics on networks: Emergence of anomalous diffusion and lévy flights," *Phys. Rev. E*, vol. 90, p. 032809, Sep 2014.

[114] A. P. Riascos and J. L. Mateos, "Long-range navigation on complex networks using lévy random walks," *Physical Review E*, vol. 86, no. 5, p. 056110, 2012.

[115] P. Zhang, C. Moore, and L. Zdeborova, "Phase transitions in semisupervised clustering of sparse networks," *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 90, 04 2014.

[116] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[117] D. Hond and L. Spacek, "Distinctive descriptions for face processing.," in *BMVC* (A. F. Clark, ed.), 1997.

[118] D. Greene and P. Cunningham, "Practical solutions to the problem of diagonal dominance in kernel document clustering," in *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, (New York, NY, USA), pp. 377–384, ACM, 2006.

[119] "The phoneme database: https://www.openml.org/d/1489, accessed 1 feb 2019.,"

[120] S. H. Rice, "A stochastic version of the price equation reveals the interplay of deterministic and stochastic processes in evolution," *BMC evolutionary biology*, vol. 8, no. 1, p. 262, 2008.

[121] Z. Zhan, R. Hu, X. Gao, and N. Huai, "Fast incremental pagerank on dynamic networks," in *International Conference on Web Engineering*, pp. 154–168, Springer, 2019.

[122] T. Haveliwala, "Efficient computation of pagerank," tech. rep., Stanford, 1999.

[123] Y. Fujiwara, M. Nakatsuji, T. Yamamuro, H. Shiokawa, and M. Onizuka, "Efficient personalized pagerank with accuracy assurance," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 15–23, ACM, 2012.

[124] B. Bahmani, K. Chakrabarti, and D. Xin, "Fast personalized pagerank on mapreduce," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, pp. 973–984, ACM, 2011.

[125] D. Fogaras, B. Rácz, K. Csalogány, and T. Sarlós, "Towards scaling fully personalized pagerank: Algorithms, lower bounds, and experiments," *Internet Mathematics*, vol. 2, no. 3, pp. 333–358, 2005.

[126] K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova, "Monte carlo methods in pagerank computation: When one iteration is sufficient," *SIAM Journal on Numerical Analysis*, vol. 45, no. 2, pp. 890–904, 2007.

[127] B. Bahmani, A. Chowdhury, and A. Goel, "Fast incremental and personalized pagerank," *Proceedings of the VLDB Endowment*, vol. 4, no. 3, pp. 173–184, 2010.

[128] I. C. Ipsen and R. S. Wills, "Mathematical properties and analysis of google?s pagerank," *Bol. Soc. Esp. Mat. Apl*, vol. 34, pp. 191–196, 2006.

[129] N. Tremblay, P. Gonçalves, and P. Borgnat, "Design of graph filters and filterbanks," in *Cooperative and Graph Signal Processing*, pp. 299–324, Elsevier, 2018.

[130] B. N. Parlett, H. Simon, and L. Stringer, "On estimating the largest eigenvalue with the lanczos algorithm," *Mathematics of computation*, vol. 38, no. 157, pp. 153–165, 1982.

[131] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[132] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[133] F. Wu, T. Zhang, A. H. d. Souza Jr, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," *arXiv preprint arXiv:1902.07153*, 2019.

[134] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International conference on machine learning*, pp. 2014–2023, 2016.

[135] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[136] "Dynamic sbm python class." `https://github.com/estbautista/DynSBM`.

[137] J. W. Stewart III, *BGP4: inter-domain routing in the Internet*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[138] A. Dhamdhere, D. D. Clark, A. Gamero-Garrido, M. Luckie, R. K. Mok, G. Akiwate, K. Gogia, V. Bajpai, A. C. Snoeren, and K. Claffy, "Inferring persistent interdomain congestion," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 1–15, ACM, 2018.

[139] M. S. Kang and V. D. Gligor, "Routing bottlenecks in the internet: Causes, exploits, and countermeasures," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 321–333, ACM, 2014.

[140] A. Milolidakis, R. Fontugne, and X. Dimitropoulos, "Detecting network disruptions at colocation facilities," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pp. 2161–2169, IEEE, 2019.

[141] A. Marder and J. M. Smith, "Map-it: Multipass accurate passive inferences from traceroute," in *Proceedings of the 2016 Internet Measurement Conference*, pp. 397–411, ACM, 2016.

[142] M. Luckie, A. Dhamdhere, B. Huffaker, D. Clark, *et al.*, "Bdrmap: Inference of borders between ip networks," in *Proceedings of the 2016 Internet Measurement Conference*, pp. 381–396, ACM, 2016.

[143] A. Marder, M. Luckie, A. Dhamdhere, B. Huffaker, J. M. Smith, *et al.*, "Pushing the boundaries with bdrmapit: Mapping router ownership at internet scale," in *Proceedings of the Internet Measurement Conference 2018*, pp. 56–69, ACM, 2018.

[144] Z. M. Mao, R. Bush, T. G. Griffin, and M. Roughan, "Bgp beacons," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pp. 1–14, ACM, 2003.

[145] "Ripe ncc, ris raw data." `https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-raw-data`.

[146] C. Villamizar, R. Govindan, and R. Chandra, "Bgp route flap damping," tech. rep., (No. RFC 2439), 1998.

[147] S. Sangli, Y. Rekhter, R. Fernando, J. Scudder, and E. Chen, "Graceful restart mechanism for bgp," tech. rep., (No. RFC 4724), 2007.

[148] "Ripe ncc, ripestat: Bgp looking glass." `https://stat.ripe.net/widget/looking-glass`.

[149] "Ripe ncc, atlas." `https://atlas.ripe.net`.