Characterizing Edges in Signed and Vector-Valued Graphs

Géraud Le Falher

Université de Lille

École Doctorale Sciences Pour l'Ingénieur

THÈSE DE DOCTORAT

préparée au sein de CRIStAL Centre de Recherche en Informatique, Signal et Automatique de Lille
et du centre de recherche Inria Lille - Nord Europe
financée par Inria

Spécialité : **Informatique**

présentée par

Géraud LE FALHER

CHARACTERIZING EDGES IN SIGNED AND VECTOR-VALUED GRAPHS

sous la direction de **Dr. Marc TOMMASI**
et l'encadrement de **Fabio VITALE**

Soutenue publiquement à **Villeneuve d'Ascq**, le **16 avril 2018** devant le jury composé de :

M. Liva **RALAIVOLA**	Univ. Marseille	Rapporteur
M. Alessandro **PROVETTI**	Birbeck College in London	Rapporteur
Mme Elisa **FROMONT**	Univ. Rennes	Examinateur
Mme Sophie **TISON**	Univ. Lille	Examinateur
M. Claudio **GENTILE**	Inria	Invité
M. Fabio **VITALE**	Univ. Lille	Encadrant
M. Marc **TOMMASI**	Univ. Lille	Directeur

Abstract

In this thesis, we develop methods to efficiently and accurately characterize edges in complex networks. In simple graphs, nodes are connected by a single semantic. For instance, two users are friends in a social networks, or there is a hypertext link from one webpage to another. Furthermore, those connections are typically driven by node similarity, in what is known as the homophily mechanism. In the previous examples, users become friends because of common features, and webpages link to each other based on common topics. By contrast, complex networks are graphs where every connection has one semantic among k possible ones. Those connections are moreover based on both partial homophily and heterophily of their endpoints. This additional information enable finer analysis of real world graphs. However, it can be expensive to acquire, or is sometimes not known beforehand. We address the problems of inferring edge semantics in various settings. First, we consider graphs where edges have two opposite semantics, and where we observe the label of some edges. These so-called *signed graphs* are a convenient way to represent polarized interactions. We propose two learning biases suited for directed and undirected signed graphs respectively. This leads us to design several algorithms leveraging the graph topology to solve a binary classification problem that we call EDGE SIGN PREDICTION. Second, we consider graphs with $k \geq 2$ available semantics for edge. In that case of *multilayer graphs*, we are not provided with any edge label, but instead are given one feature vector for each node. Faced with such an unsupervised EDGE ATTRIBUTED CLUSTERING problem, we devise a quality criterion expressing how well an edge k-partition and k semantical vectors explains the observed connections. We optimize this *goodness of explanation* criterion in vectorial and matricial forms, and show how those two methods perform on synthetic data.

Résumé

Dans cette thèse, nous proposons des méthodes pour caractériser efficacement et précisément les arêtes au sein de réseaux complexes. Dans les graphes simples, les nœuds sont liés au travers d'une sémantique unique. Par exemple, deux utilisateurs sont amis dans un réseau social, ou une page web contient un lien hypertexte pointant vers un autre page. De plus, ces connexions sont généralement guidées par la similarité entre les nœuds, au travers d'un mécanisme appelé homophilie. Dans les exemples précédents, les utilisateurs deviennent amis à cause de caractéristiques communes, et les pages web sont reliées les unes aux autres sur la base de sujets communs. En revanche, les réseaux complexes sont des graphes où chaque connexion possède une sémantique parmi k possibles. Ces connexions sont en outre basées à la fois sur une homophilie et une hétérophilie partielle des nœuds à leurs extrémité. Cette information supplémentaire permet une analyse plus fine des graphes issus d'applications réelles. Cependant, elle peut être coûteuse à acquérir, ou n'est pas toujours disponible a priori. Nous abordons donc le problème d'inférer la sémantique des arêtes dans plusieurs contextes. Tout d'abord, nous considérons les graphes où les arêtes ont deux sémantiques opposées, et où nous observons l'étiquette de certaines arêtes. Ces « graphes signés » sont une façon élégante de représenter des interactions polarisées. Nous proposons deux biais d'apprentissage, adaptés respectivement aux graphes signés dirigés et non dirigés. Ceci nous amène à concevoir plusieurs algorithmes utilisant la topologie du graphe pour résoudre un problème de classification binaire que nous appelons EDGE SIGN PREDICTION. Deuxièmement, nous considérons les graphes avec $k \geq 2$ sémantiques possibles pour les arêtes. Dans ce cas, nous ne recevons pas d'étiquette d'arêtes, mais plutôt un vecteur de caractéristiques pour chaque nœud. Face à ce problème non supervisé d'EDGE ATTRIBUTED CLUSTERING, nous concevons un critère de qualité exprimant dans quelle mesure une k-partition des arêtes et k vecteurs sémantiques expliquent

les connexions observées. Nous optimisons ce critère « qualité explicative » sous une forme vectorielle et matricielle et illustrons le comportement de ces deux méthodes sur des données synthétiques.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Pr. Marc Tommasi. From the day we first got in touch in that spring four years ago, and all the way towards my defense, he has provided me with thorough guidance. In terms of research of course, but on many other topics as well, from professional development to folkloric music. Moreover, this has always been done in the kindest way possible. Therefore, I'm doubly indebted to him, for not only I wouldn't have complete my PhD without him, but he made it a very pleasant experience. Second, such feelings carries over to include my advisor, Dr. Fabio Vitale. There has been time when his sense of rigor has challenged me. But at the end, it was an important learning experience, and his ability to crack a joke at the least expected moment of a long research meeting has proved very useful: the fact we went to bars in at least three different countries speaks for it. I would also like to thank Pr. Claudio Gentile for hosting me in Varese, and giving me sharp advice along those three years.

Then I want to thank Alessandro Provetti and Liva Ralaivola who kindly agreed to review my manuscript, as well as Elisa Fromont and Sophie Tison for having accepted to be part of my committee. To conclude this academic paragraph, I'd like to sincerely acknowledge how lucky I was to collaborate with all my talented co-authors: Nicolò Cesa-Bianchi, Emre Çelikten, Claudio Gentile, Aristides Gionis, Michael Mathioudakis and Fabio Vitale.

Although doing a PhD is a rather personal enterprise, I was fortunate to be part of the awesome Magnet team[1], whose past and current members I met include, in seating order: Thomas, Pauline, David, Mathieu (Thursday French Fries forever), Thanh, Carlos, Pierre, Paul, Juhi, Quentin, William, Mikaela, Pascal (special thanks for great accommodation and sportive spirit), Rémi, Aurélien and Hippolyte. Thanks for stimulating discussions on many topics, constant readiness to help and answer my questions, life long lessons of table football, and quite importantly, thanks for patiently bearing with me while I was finishing lunches (and/or fiddling with my phone)! Of course, this wouldn't be proper acknowledgment without mentioning the best office in the world, namely B224, and the best possible office mates: Nathalie (for all the cookies) and Thibault (because there were some genuinely good jokes among all of them). Whereas I've always been slightly worried about logistics, I never had to think about it thanks to the fantastic work of Julie. Finally, one last Magnet member, Antonino, left before I joined but nonetheless deserve my gratitude for he helped me find my next job. Same credit goes to Michal, in addition to being a great company in all occasions. At this stage, I might as well thanks the rest of the Sequel team, and especially Julien, Alexandre, Émilie, Frédéric, Romaric, Matteo, Daniele, Julien, Bilal, Marta, Tomáš, Daniele, Florian, Jean-Bastien, Olivier and Ronan (too bad my defense is not on Friday, for I know you wouldn't be long to hit the dance floor). Finally I want to thank all the support crews from Inria, as they made it a great place for doing research, as well as the University Lille 3 for making my short teaching experience so pleasant.

Doing a PhD is also supposed to be a rather labor intensive enterprise, but again, I was fortunate to live in the student city of Lille. There I met many wonderful people, each of them who contributed in a way to this manuscript. It all started when a young version of myself attended a student event organized by Tilda and later had to pleasure to collaborate with Guillaume, Antoine, Fabien, Alexandre, and by extension with Carmelo, Jason, Cindy, Antoine (Dujardin), Clément, Ilkay, Benjamin

[1] https://www.inria.fr/en/teams/magnet

and Émilie. On my way to organize an elusive café lingua, I also enjoyed the company of many members of a meetup group, including Samir, Nathalie, Marie-Pierre, Anne Charlotte, Léa, Géraldine, Fabienne, George, Norida and Thomas, as well as Rashida, Mairead and Yash, who bore with me during circumstances better not written here. Another great event in Lille is undeniably the "apéro culture", where I was privileged to meet Cyrille, Élise, Audrey, Wasilla and Olmo. This picture wouldn't be complete without Stéfana, Arthur, Esmeralda, Daniel, and my flatmate Édouard for some nice video games nights.

I also received support from outside Lille, thanks to Tristan, Olivier (may the force be with you guys), Aloïs, Juliette, Florent, Isabelle, Camille and Claire. In these cold days of February, I also have a warm thought for people I met during my visits in Aalto: Luiza, Annika, Eric, Polina and Klaudia as well as Kiran, Vera and Sanja for their long-standing support and comments on some part of this thesis.

You astute reader may at this point grow suspicious of whether a single man can really be so lucky. Yet this is not over and I was again fortunate to met just before starting my PhD a wonderful friend. Over those years, Nataša has given me support that words probably can't describe. But it's safe to say that without her, this thesis and even myself would both be lacking something essential.

Enfin, j'aimerais remercier ma mère et mon frère, pour leur soutien inconditionel sans lequel toute cette aventure n'aurais pas été possible.

Berlin, Germany, February 27, 2018
Géraud Le Falher

Contents

1 Introduction **1**
- 1.1 Learning in graphs . . . 2
- 1.2 Graph with several edge semantics . . . 3
  - 1.2.1 Signed graphs . . . 4
  - 1.2.2 Multilayer graphs . . . 7
- 1.3 Predicting edge type . . . 8
- 1.4 Outline . . . 10

2 On the Troll-Trust Model for edge sign prediction in Social Networks **11**
- 2.1 Notation and Preliminaries . . . 13
- 2.2 Generative Model for Edge Labels . . . 16
- 2.3 Algorithms in the Batch Setting . . . 17
  - 2.3.1 Approximation to Bayes via dense sampling . . . 17
  - 2.3.2 Approximation to Maximum Likelihood via Label Propagation 19
- 2.4 Related work . . . 21
- 2.5 Experimental Analysis . . . 25
  - 2.5.1 Datasets . . . 25
  - 2.5.2 Synthetic signs . . . 27
  - 2.5.3 Real signs . . . 29
  - 2.5.4 Additional experiments . . . 32
- 2.6 Algorithms in the Online Setting . . . 33
- 2.7 Open questions . . . 35
- 2.8 Summary . . . 36
- 2.9 Additional material . . . 37
  - 2.9.1 Proofs from Section 2.3 . . . 37
    - 2.9.1.1 Proof of Theorem 1 . . . 37
    - 2.9.1.2 Derivation of the maximum likelihood equations . . 40
    - 2.9.1.3 Label propagation on G'' . . . 40
  - 2.9.2 Proof from Section 2.6 . . . 41
  - 2.9.3 Further Experimental Results . . . 43

3 Edge sign prediction in general graphs and Correlation Clustering **45**
- 3.1 A bias for general graphs . . . 45
  - 3.1.1 Sign generative model and behavior . . . 45
  - 3.1.2 Directed edges requirement . . . 47
  - 3.1.3 Social balance as a learning bias . . . 48
- 3.2 Correlation Clustering . . . 51
  - 3.2.1 Problem setting and connection to Edge Sign Prediction . 52
  - 3.2.2 State of the art . . . 54
    - 3.2.2.1 Exact methods . . . 54
    - 3.2.2.2 Hardness and approximations . . . 56
    - 3.2.2.3 Heuristics . . . 61
    - 3.2.2.4 Active and online settings . . . 63
  - 3.2.3 Beyond worst case instances . . . 64
  - 3.2.4 Variants and extensions . . . 65

3.3 Low stretch trees and spanners . . . 70
3.3.1 GALAXY TREE: a spanning tree designed for sign prediction . 70
3.3.2 Related work . . . 79
3.3.3 Empirical evaluation . . . 82
3.3.3.1 Graph topology . . . 83
3.3.3.2 Stretch . . . 84
3.3.3.3 Sign prediction . . . 84
3.4 Conclusions . . . 85
3.4.1 Summary . . . 85
3.4.2 Future work . . . 85

4 Edge clustering in node attributed graphs 91
4.1 Attributed graphs and problem definition . . . 93
4.1.1 Setting and modelling . . . 93
4.1.2 Learning problem and additional constraints . . . 95
4.2 Proposed approaches . . . 99
4.2.1 k-MEANS baseline and improvement . . . 99
4.2.2 Convex relaxation . . . 100
4.2.3 Matrix optimization . . . 101
4.2.3.1 FRANK–WOLFE method . . . 101
4.2.3.2 EXPLICIT low rank factorization . . . 103
4.3 Synthetic experiments . . . 104
4.3.1 Data generation . . . 104
4.3.2 Results . . . 107
4.4 Related work . . . 109
4.5 Open directions . . . 113

5 Conclusion 117
5.1 Contributions . . . 117
5.2 Future work . . . 118
5.2.1 Reciprocal recommendation . . . 118
5.2.2 Representation learning . . . 119

List of Figures

2.1 Part of a DSSN centered on node u. 14
2.2 Illustration of the graph transformations turning EDGE SIGN PREDICTION into node classification 14
2.3 The node-labeling of G' illustrating the relation between the edge label complexity and the cutsize. Here the four edges part of the cut are in heavy black. 16
2.4 The sign $y_{u,v}$ of the edge $u \to v$ is positive with probability $\frac{1}{2}(p_u + q_v)$. . . 16
2.5 Synthetic distributions on WIKIPEDIA. 27
2.6 Results on WIKIPEDIA with BLC(tr, un) on a training set of size 40% . 29

3.1 A two-clustering of a complete strongly balanced graph 49
3.2 The four possible undirected triads, as classified by the two structural balance theories introduced in the main text 50
3.3 Small example of CORRELATION CLUSTERING 53
3.4 The transformation from CORRELATION CLUSTERING on G to k-MINIMUM MULTICUT on H (reproduced from [Dem+06]) 57
3.5 A positive star with few negative edges 64
3.6 A sample star 71
3.7 Cross edges between two stars 73
3.8 Unfolding stars to recover spanning trees 76
3.9 The hierarchical structure of stars created by GALAXY TREE 77
3.10 The other iterations of GALAXY TREE 77
3.11 A line graph with stars in blue. 78
3.12 Real world pictures and their binarized version 87
3.13 Stretch over graphs of increasing size 88
3.14 MCC over various graphs 89
3.15 *A sym exp* over various graphs 89

4.1 A small instance of EDGE ATTRIBUTED CLUSTERING and an hand-crafted solution, albeit non-optimal. 96
4.2 Finding the k_{local} directions of node when generating synthetic graph 106

List of Tables

1 List of notations used in this thesis x

2.1 The label regularity values for the nodes of the example G graph of Figure 2.2a. 16
2.2 Directed Signed Social Networks dataset properties 26
2.3 Comparison of BLC(tr, un) with the Bayes optimal predictor 28
2.4 MCC with increasing training set size 31
2.5 Training time on a 15% training set 32
2.6 MCC difference when using reciprocal edges 32
2.7 Same as Table 2.4, but when selecting the training set by the timestamp of the edges. 33
2.8 The effect of not sampling edges at random 34
2.9 Values of $s_j(i)$ for $i \leq 7$ and $j \leq 3$. 42
2.10 Values of $m_{r,c}$ for $r \leq 7$, $c \leq 4$ and $|E| \to \infty$. 42
2.11 Normalized logistic regression coefficients averaged over 12 runs (with one standard deviation) 43

3.1 Biological dataset properties. The columns have the same meaning as in Table 2.2 on page 26 46
3.2 This table is the same as in Table 2.4 on page 31, but this time on three smaller, directed biological networks. 47
3.3 MCC on the six previous datasets with direction removed 48
3.4 Hardness results of CORRELATION CLUSTERING 57
3.5 Best current results on CORRELATION CLUSTERING problems 59
3.6 Approximation results for CORRELATION CLUSTERING on general graphs with K clusters 60
3.7 Length of the paths not in the resulting spanning tree. 78
3.8 Reproduction of Table 1 from [AN12], showing the evolution of the best asymptotic average stretch over time. 81
3.9 A summary of the lowest stretches achievable for various problems. 83
3.10 Dataset description 83

4.1 Real world attributed graphs 92
4.2 Combining node profiles through the Hadamard product 94
4.3 Qualite of our synthetic instances according to generation parameters 108
4.4 Performance of our proposed methods on synthetic instances 109
4.5 Same as Table 4.4, but reporting $d(\mathcal{D}_k, \widehat{\mathcal{D}}_k)$, which should be as close as possible to 0. 109
4.6 Generative model of node attributed graphs with community structure 114

In this thesis we will sometimes write remarks in a smaller font and with a light blue edging.

> Such remarks provide additional information to the topic discussed above, but can be skipped without harming the comprehension of the main material.

Table 1 – List of notations used in this thesis

Symbol	Meaning
$[\![K]\!]$	The natural integers from 1 to K, i.e. $\{1, 2, \dots, K\}$
$\mathbb{I}\{p\}$	An indicator function evaluating to 1 if the predicate p is true and 0 otherwise
$x_{;i}$	For a vector $x \in \mathbb{R}^d$, $x_{;i}$ denotes the value of its i^{th} component.
$\boldsymbol{S}^d$	The sphere of ℓ_2 unit vectors in dimension d, i.e. $\left\{x \in \mathbb{R}^d \,:\, \|x\|_2 = 1\right\}$
G	An unweighted graph. It should be clear from the context whether it is directed or not
V	The set of all the nodes of a graph, with $\lvert V\rvert = n$. Unless noted otherwise, nodes are indexed from 1 to n
u	A generic node of G. When referring to several nodes, we naturally use u, v, w and so on.
E	The set of all the edges of a graph, with $\lvert E\rvert = m$
(u, v)	An undirected edge between nodes u and v
$u \to v$	A directed edge from node u to node v
$y_{u,v}$	The sign of the edge (u, v), which can be either $+1$ or -1
$Y(E)$	The labeling of E, that is the set of all signs of E: $Y(E) = \{y_{u,v} : (u, v) \in E\}$
E_{train}	A subset of E, given or chosen, of which we observe the signs
$\deg(u)$	The total degree of node u (that is, the number of edges incident to u, regardless of their direction)
$\mathcal{N}(u)$	The set of all neighbors of u, regardless of edge direction. It thus holds that $\lvert\mathcal{N}(u)\rvert = \deg(u)$
$\mathcal{N}^+(u)$	The set of all positive neighbors of u, regardless of edge direction. That is, $\forall v \in \mathcal{N}^+(u),\ y_{u,v} = +1$.
$\mathcal{N}^-(u)$	The set of all negative neighbors of u, regardless of edge direction. That is, $\forall v \in \mathcal{N}^+(u),\ y_{u,v} = -1$.
T	An unweighted and undirected tree
$\text{path}^T(u, v)$	The unique path between u and v in the tree T, represented by an ordered list of edges
$\lvert\,\text{path}^T(u, v)\rvert$	The length of the path between u and v in the tree T, that is its number of edges
$\text{diam}(G)$	The diameter of G, that is the longest shortest path between any two nodes.

Chapter 1

Introduction

Graphs are a natural way to represent the relationships over a set of entities. Because of the simplicity and flexibility of this formalism, graphs are ubiquitous and have been used in countless fields. To make the rest of our explanations more concrete, we now give three examples of graphs that we consider later on.

1. In a social network[1], the nodes of the graph are human users, and relationships between nodes denote interactions between the corresponding users. One social network we study in Chapter 2 summarizes the connections between the editors of WIKIPEDIA. These editors can be promoted to administrators after a vote from their peers. An edge $u \to v$ in this network means that user u has voted on the possible promotion of v.
2. In computer vision, we can represent an image as a graph. Each pixel is a node, and those nodes are connected to four neighbors, namely the adjacent pixels from the top, bottom, left and right sides. The relationship in this graph is therefore adjacency in an image.
3. In e-commerce, we can consider a co-purchase network. Nodes are products being sold on a website, and two nodes are connected if the corresponding products are frequently bought together by customers, for instance a phone and a memory card.

Not only are there graphs in many domains, but the progress of technology in the last few decades has made it easier to collect many graphs in every single domain, sometimes with up to billions of nodes and hundreds of billions of edges. The availability of such large amounts of structured data has prompted the development of automated methods to extract insights from them. For instance, it is possible to cluster nodes into coherent groups, predict the category of the nodes or study how to best propagate information within a graph. We describe in more details these possibilities and others in Section 1.1. By doing so, our goal is to illustrate the wide potential of *learning in graphs*.

At this point of the discussion though, we have only considered the most common and simple kind of graph, one representing a single type of relation between nodes.[2] However, in many situations, there are two dimensions along which graphs exhibit more complexity. First, nodes have more than one type of relation among each other. Second, two nodes are not only connected because of their global similarity but also for more nuanced reasons. As a case in point, let us look again at our three examples. In the WIKIPEDIA network, a vote can support the promotion or oppose it. This additional information enrich our understanding of the relationships among the nodes in the graph. It also points out that two editors can be connected because they share a common topic of interest, but come from different socio-economic backgrounds and therefore cannot agree on this topic.

[1] In the rest of this thesis, we use the terms *graph* and *network* interchangeably.

[2] Note that by "type of relation", we do not refer to some graph-theoretical characteristic of an edge, such as being directed or weighted. Indeed, all the graphs we consider have homogeneous edges with that respect. Instead, we mean *domain specific semantic*, as we will make clear in examples.

Likewise in an image, an edge between two pixels is positive if the two pixels belong to the same object (say a car or a building) and negative otherwise. This information could be used to segment the image. In the co-purchase network, there are even more than two types of relation. Assuming the products are movies, two movies can be frequently bought together because they are part of a series (like Star Wars), because they have the same director but different genre, because they receive the same prestigious award in different years, and so on. The second type of relation ("same director" and "different genre") is actually an example of a mixed relation, for it combines similarity and dissimilarity over several features of the nodes.

We call *complex networks* graphs where edges have different *semantics*, or types, and where connections are explained by more than simple global similarity. In Section 1.2, we give additional examples of such complex networks. As showed in our three previous examples, we distinguish between two cases. The first is when there are two types of edge having opposite semantics. Such graphs are called *signed graphs* and have been extensively studied since the fifties, for we shall see they have many applications. Their name comes from the fact that edges are typically labeled $+1$ and -1. The second case is when there are more than two types of edge. Such graphs have also been studied for a long time under different names and we refer to them as *multilayer graphs*.

It is natural to expect that we can extract finer insights from graphs with edge semantics. However, in many cases, the information about edge types is not available, at least not in a convenient form. We therefore present in Section 1.3 the problem of characterizing edges in complex networks. Informally, given an input graph and possibly some extra information, we want to predict the type of every edge. This problem can take several forms depending on what information is available as input. First, the graph may be directed or undirected. Second, there might be two edge types (in the case of signed graphs) or more (for general multilayer graphs). Third, the problem can be supervised or not. In the supervised setting, we are provided with labels for some of the edges. In the unsupervised setting, there is no label at all. We thus make the additional assumption that we observe some *attributes* of the nodes. Once again, we return to our three examples to illustrate what those attributes can be. In the WIKIPEDIA graph, we could have for each user data about her age, experience and area of expertise. In an image, each pixel is associated with a color, along with higher order visual features. Each product of a co-purchase network comes with information about, e.g., its price, popularity, category and availability. In the absence of label, our intuition is that these attributes can inform us about the types of edge among nodes.

We list in Section 1.4 three concrete instantiations of this general problem that we consider in this thesis. Our solutions to these three problems offer evidence in the defense of our thesis statement:

> **There exist efficient and accurate methods to predict edge type in complex networks, relying only on the graph topology or also on node attributes.**

1.1 Learning in graphs

The birth of graph theory is credited to Euler [Eul41] in 1736 for his elegant solution of the Seven Bridges of Königsberg problem[3]. Since then, it has been a very active branch of mathematics [BLW86]. Indeed, it provides a conceptually simple yet immensely rich framework to model phenomena where entities are connected with each other [dFon+11]. Coupled with the increasing availability of large amount

[3] https://en.wikipedia.org/wiki/Seven_Bridges_of_Königsberg

of relational data, learning on graphs has recently spurred a lot of interest across various lines of research, with tangible benefits.

community detection [For10] The goal is to cluster nodes in tightly connected groups that are loosely connected with the rest of the graph. This allows for a better understanding of the graph organization, and present a higher level view than looking at the individual node relationships. For instance, it has been used to identify proteins functional groups [SM03], or to see how different scientific fields relate based on publication data [RB08].

semi-supervised learning [CSZ06; ST14] In addition to labeled data, the learner is also provided with unlabeled data at training time, and its goal is to classify nodes. Connecting similar instances allow propagating information along the graph. This has found applications in classifying text documents [TC09] or aligning categories and relations across multiple knowledge bases [WTM13].

node embedding [Wil17] The goal is to find a low dimensional representation of the nodes, based on their structural patterns. This usually performed in an unsupervised way, although it is also possible to include problem supervision when available. Such representation can then be used in downstream tasks, for instance visualization [Tan+15] or the aforementioned node classification, even in the inductive setting where new nodes can join the graph after training [HYL17].

link prediction [MBC16] Given a snapshot of the graph at time t, the goal of link prediction is to return a set of links that do not exist at time t but will be created by time $t + \Delta_t$. Most methods are based on the assumption that link creation is driven by node similarity. It has been successfully applied to inferring potential interactions between proteins without expensive experiments [QBK06] and uncovering hidden associations in criminal networks [CMN08].

information and influence propagation [CLC13] The study of processes by which content is spread across networks, and how such processes can be influenced to speed them up or slow them down. Two representative applications are selecting the best seeds in a social network to promote a viral marketing campaign [KKT15] and containing more effectively the diffusion of actual biological viruses [Pra+13].

network evolution [AS14] These methods focus on the mechanisms and consequences of the growth of networks. They also seek ways to keep the results of some data mining algorithms up to date and relevant. Monitoring the changes in the interactions of proteins can indeed be used as an early indicator for some kind of diabetes [Bey+10]. Furthermore, sudden changes in a network of computers are usually worth investigating, for they might signal external attacks [IK04].

This list of graph learning problems and their applications to real world scenario is incomplete. Yet it already demonstrates the impact of inferring patterns in relational data over many aspects of our lives. However, we argue that more can be done. Indeed, all the methods presented above only consider graphs with a single type of edge and where nodes are connected based on some domain-dependant notion of similarity.

1.2 Graph with several edge semantics

The three graphs we described in the introduction are examples of what we call complex networks. Such graphs have more than one type of edge, and edges are not explained by mere global node similarity. The point of this section is to illustrate that this notion is not simply a mathematical variation of a "simple" graph, where the adjacency matrix would take values in $\{-1, 0, 1\}$ or $\{0, 1, \ldots, k\}$ instead of $\{0, 1\}$.

Rather, we review many applications in various domains where complex networks are the right model to represent rich networked data. First, we showcase many uses of signed networks. In that case, we do not insist on the fact that similarity is not the only driver of node connections, for it is implicit that negative edges actually denote dissimilarity. Second, we present multilayer graphs, that is graphs with more than two types of edges. When applicable, we highlight how the connections in such graphs indeed rely on a nuanced similarity or dissimilarity across some of the nodes attributes.

To elaborate on this last point, our hypothesis is that connections in complex networks are the results of both partial homophily (that is, nodes are connected when they are similar on a subset of the attributes) and partial heterophily (that is, nodes are connected when they are dissimilar on a subset of the attributes). As examples of the latter, think of dating websites —where most users are linked with users of the opposite gender [The09a; Tys+16]; diffusing innovations —where meeting people with different backgrounds and point of views is crucial to favor diversity and creativity [Rog03]; and online news consumption —where connecting people from different sides of the political spectrum helps to avoid echo chambers and instead fuel a democratic debate [Gar+17].

1.2.1 Signed graphs

In this section, we present a list of signed graph usages, sorted by domains [Tan+16b]. Many of those signed graphs are the input of some clustering algorithms. In the context of signed graph, the clustering task can be captured by the CORRELATION CLUSTERING problem. We provide a thorough overview of this problem in Chapter 3. Here we simply give a broad, informal definition. The objective in CORRELATION CLUSTERING is to cluster the nodes of a signed graph in a way that minimizes the number of positive edges across clusters and the number of negative edges within clusters. Such edges are called disagreements.

Computer Vision The ubiquitous task of segmenting an image into homogeneous regions is a prerequisite for many further processing. As we mentioned earlier in one example, building a signed graph can help, although it might be costly to do it at the pixel level. For instance, to segment cells in microscopy imagery, Zhang *et al.* [ZYH14] first use generic image features to classify pixel in belonging to region boundaries or not. Then, they extract small scale regions called superpixels. After building the adjacency graph of these superpixels, they assign edge weights by averaging the boundary probabilities of the pixels separating adjacent superpixels. They also add strong negative constraints between distant superpixels, and lastly cluster these superpixels according to the CORRELATION CLUSTERING objective to obtain the final segmentation. A similar approach was used earlier in Kim *et al.* [Kim+11], who stress the importance of considering such higher order constraints between distant superpixels in order to achieve good performance. This was also extended to 3D segmentation [And+12], where additional tuning allows to segment a volume image of a mouse cortex with up to billions voxels. Beier *et al.* [BHK15] segment 2D and 3D images with an energy based formulation of CORRELATION CLUSTERING and iteratively improve their solution by merging it with another clustering given by a proposal generator. By developing another scalable energy based optimization procedure, and with the help of few user-provided cues, Bagon *et al.* [BG11] are able to apply CORRELATION CLUSTERING directly at the pixel level.

Beyond image segmentation, Gori *et al.* [Gor+17] develop a method to extract a network of descriptive curves from 3D shapes. After an initial stage of generating many such *flowlines*, they describe in Section 6 a CORRELATION CLUSTERING formulation to extract *reliable* representative flowlines, using geometric constraints to obtain positive or negative cues that two flowlines are from the same reliable representative.

Finally, in order to track several targets across sequential video frames, Solera *et al.* [SCC15] propose a multistage framework. One step revolves around a matrix A that defines the cost of assigning an object tracked in previous frames to an object detected in the current frame. This matrix is turned into a symmetric affinity matrix $\bar{A}_{sym}$ that can be seen as a signed graph adjacency matrix. CORRELATION CLUSTERING is then used to extract clusters (called zones), in which local processing is performed. This is beneficial since the complexity of these local methods can be adapted to the difficulty of each zone.

Natural Language Processing Coreference resolution is the task of finding all expressions that refer to the same entity in a text. Like image segmentation, it is a preprocessing step that can later be used in document summarization, question answering, and information extraction. Furthermore, in that case, it is also natural to build a graph of words. One then add negative edges between words that cannot refer to the same entity (for instance because they are of different gender) and positive edges between words with linguistic cues indicating they might refer to the same entity. McCallum *et al.* [MW05, Section 2.3] instead tackle coreference resolution by using an undirected graphical model on which performing inference is equivalent to CORRELATION CLUSTERING. On small scale instances, Elsner *et al.* [ES09] use the signed graph procedure outlined above. They first obtain an upper bound of the optimal solution by solving a SDP relaxation of the problem. They then compare various heuristics and show that the best performing ones are within few percents of the optimum, provided they are followed by a local search step, such as the Best One Element Move [GMT07]. Further NLP tasks amenable to a signed graph representation are referenced in their paper. Another task is to cluster words based on distributional embedding vectors while adding antonym constraints [SPU17].

Biology Signed graphs are also abundant in biology. A typical input is a similarity matrix between genes expression level in various experimental conditions, and the goal is to cluster those genes into groups that react similarly. Ben-Dor *et al.* [BSY99, Section 4] gives three examples: 112 genes involved in the rat central nervous system, 1246 genes of the roundworm *C. elegans* and 2000 human genes obtained from 40 tumorous and 22 normal colon samples. Mason *et al.* [Mas+09] analyze a signed co-expression network of genes involved in embryonic stem cells to find which genes are related to pluripotency (the ability to differentiate into any type of cell in the body) or self-renewal (the ability to replicate indefinitely). Another application is to study the variation of one individual DNA [DV15]. In the human organism, chromosomes are organized in pair, and both chromosomes of a pair have similar but not identical DNA sequences. This is mostly because of single nucleotide polymorphisms (SNPs), where a single base differs between the two DNA sequences, leading to different alleles of the corresponding gene. A haplotype is the list of all alleles at a contiguous region of a single chromosome, and this information is used in several medical applications. The high-throughput sequencing of one individual genome yields many short *reads* that provides information about the order of nucleotides in a fragment of one chromosome and that can be used to assemble haplotypes. To do so, the authors build a graph of reads and define a similarity function between reads to assign weights on the edges. The clusters of that graph correspond to haplotypes, and are obtained by a SDP relaxation of the CORRELATION CLUSTERING objective. DasGupta *et al.* [Das+07] also consider graphs whose nodes are genes, but in a different context. In this case, positive edges represent an activating connection, while negative edges represent inhibiting connection. They also define a *monotone system* as a balanced subgraph, that is a subgraph which does not contain a cycle with an odd number of edges. Such monotone system are stable, in the sense that modifying the concentration of

one gene will have a predictable effect, even ignoring the precise kinematics of the chemical reactions involved. Their goal is to find the minimum number of edges to remove in order to decompose a dynamics system into a collection of monotone system. This allows to study the complete system more easily. More applications of weighted complete signed graphs in biology are presented in [BB13, Section 6].

Network science One early use of signed graphs was to model social interactions [Har53; Hei58]. Here we present some recent references within this line of research. For instance, one can extract all the votes of the members of a political parliament and form a graph whose nodes are politicians and edge weights quantify how much they agree or disagree on various issues they have been voting on. This can be used to study various social science questions such as loyalty, leadership, coalitions, political crisis and polarization. It has been applied to the European parliament [Men+15], Slovenian parliament [Jia15] and the Brazilian parliament [LF17]. This can also be used at international level. For instance, by considering a dataset of military alliances and disputes, Traag *et al.* [TB09] cluster countries into blocks that resemble those identified by Huntington in his *Clash of Civilizations* book. Another source of data is the vote on resolutions during the United Nations General Assembly [MMP12]. Finally, one can also study how to exploit the information contained within the negative links to enhance the visualization of social networks [Kun+10; KT11].

Others

- Deduplication, also called duplicate detection or entity resolution, is the process of identifying objects from a real-world, noisy database that refer to the same entity. On a high level, a solution to this problem is to build a graph of all the available objects, define a similarity between them and run a CORRELATION CLUSTERING algorithm. The main challenge thus lies in devising an appropriate similarity measure, given that objects can have very different features from one database to another. Arasu *et al.* [ARS09] propose a declarative language, expressing both hard constraints (that have to be satisfied) and soft constraints (that can be seen as cues guiding the process). Because of these hard constraints that admissible clusterings have to respect, the authors have to modify in nontrivial ways an existing CORRELATION CLUSTERING algorithm. This was extended to weighted and partial constraints by Shen *et al.* [SW14]. Another example is given by Gael *et al.* [GZ07], who cluster together news articles in different languages covering the same event. CORRELATION CLUSTERING was also evaluated among other solutions to that problem by Hassanzadeh *et al.* [Has+09], who note that their non optimized implementation does not perform best.

- Given an electrical circuit layout, Chiang *et al.* [Chi+07] extract a graph of its components (called shifter) that must be assigned one of two possible phases. Because two shifters next to some specific shape must be in opposite phase and two shifters separated by less than a specified distance must be of the same phase, the authors look for a two-clustering of the nodes that will minimize the number of disagreements.

- In finance, one can represent an investment portfolio as a signed graph [HLW02]. Each node is a security, and the edge between two securities is weighted by their correlation, which can be negative. For instance, a graph with only positive edges is speculative, as all the securities move in the same direction, either up or down. On the other hand, if the securities can be partitioned in groups without disagreement, the risk is limited, for two clusters will move in opposite directions, providing the investors with a hedging guarantee.

- In wireless networks, signed graphs can be used to solve optimization problems involved in determining energy-saving routes [Rat+12] or to exchange cryptographic keys in a secure and efficient manner [SC17].

1.2.2 Multilayer graphs

Besides signed graphs, in this thesis we also consider multilayer graphs [Kiv+14; Boc+14]. Those are graphs with k edge types, and the name refer to the fact we can see them as the superposition of the k subgraphs induced by each edge type. Even when $k = 2$, we make a distinction between signed graphs and multilayer graphs. In signed graphs, the two types of edge have reverse semantic, whereas in a 2-layers graph, they simply denote two possible interactions, for instance advisor-advisee or regular coauthors relation in a citation network [Wan+10]. In general though, we focus on cases where k is larger than 2, and not larger than a few dozen in order to preserve interpretability. Like signed graphs, these multilayer graphs are versatile enough to be used in many fields.

Social networks Szell *et al.* [ST10] model the interactions of the players of *Pardus*, a massively multilayer online game. These players can be friend or enemy, send private messages, trade resources, attack each other and set a bounty on the head of another player. These six types of interactions are either positive or negative, but their nuances cannot simply by explained by global similarity and dissimilarity. Another example is photo sharing website Flickr and its users. They can interact in eleven ways, either directly or through comments, shared tags, groups membership and so on [KMK11]. Again, while being part of a common group denotes shared interests between two users, overall they must also differ in some other attributes (for instance location) in order to bring diversity to this group.

Citation networks In these networks, nodes are research papers or authors, and edges typically connect two nodes whenever one cites the other. For instance, using the DBLP dataset, Cai *et al.* [Cai+05] connect two authors if they have co-authored a paper in one the 1 000 conferences appearing in the data. One can then consider that the resulting graph was obtained as the superposition of these 1 000 subgraphs. Besides direct citations, Dunlavy *et al.* [DKK11] consider four others reasons to connect 5 000 SIAM papers, based on their similarity in terms of abstract words, title words, keywords and authors.

Economic networks In our ever increasingly globalized economy, entities around the world are getting more and more tightly connected. However, those connections take on many different forms simultaneously. For something seemingly as simple as the connections between the largest 951 ports in the world, one must already notice that these connections can be implemented by any combinations of three kinds of ships: bulk dry carriers, container ships and oil tankers [Kal+10]. Likewise, the 162 countries of the International Trade Network are connected by 96 kinds of commodities they can exchange [BFG10]. Finally, Cranmer *et al.* [CMM15] studies international relations through the lens of Kant's three folded program for peace, based on democracy, economic dependence and supra national governance. They build the graph of all countries and connect them in three layers. All democracies form a clique in the first layer, countries are connected with weight proportional to the amount of yearly trade in the second layer and with weight proportional to the number of international organizations they belong to together in the third layer. While the essence of commerce is to exploit differential between the partners involved (i.e. heterophily), trade intensity and participating in common institutions also involve geographical, historical and cultural ties (i.e. homophily).

Biological networks The interactions in biology also take several forms and studying them as a whole has proved fruitful. One example is a genes co-expression network, where each connection was tested under 130 different experimental conditions, providing as many layers [Li+11]. Multilayer graphs are also a relevant way to represent ecological networks [Pil+17]. For instance, Kéfi *et al.* [Kéf+15] build the graph of more than 100 species living on the Chilean coast[4] and divide their interactions in three categories: trophic (i.e. one specie eating another one), positive non-trophic (e.g. refuge providing) and negative non-trophic (e.g. competition for shared resources). Finally, in neuroscience, multilayer graphs have recently emerged as a useful tool to better understand the human brain [MB16]. The nodes of such graphs are neurons, and the edges can be labeled in various ways: some correspond to actual physical links while others are functional (i.e. neurons responding in the same way to external stimuli), some are present in healthy subjects and others in treated patients, some are acquired through MRI and others by EEG.

1.3 Predicting edge type

Let us summarize in one sentence the two previous sections. Learning in graphs provides many insights, and many graphs are complex, in the sense of having edges expressing different semantics and created for mixed reasons. The logical conclusion is that we want to learn in complex graphs. However, this requires the edges of such graphs to be labeled with one of k types of relationship. In the examples presented above, this was already done. Yet we argue that efficiently labeling the edges is an interesting problem, especially in the following three situations:

1. There exists a function that can perfectly label any edge because it is tailored to this specific graph, but calling it is expensive. For instance, say we want to know whether two connected users of a social network are tied to each other through family, work, school or hobby. The labeling function in this case simply asks users to label their relationships, assuming the answers do not contain any noise. At the scale of Facebook, this would require asking hundred of questions to each user on average, which is quite time-consuming. Furthermore, it would also cost marketing resource to convince users this is beneficial for them, and engineering time to ensure this information remain confidential. Likewise, in a biological network, determining whether two proteins interact positively or negatively with each other is achieved by a lab experiment, which requires time and material.

2. All edges are already labeled, perfectly and without any cost, but the number k of edge types is very large. Indeed, we have seen examples where they are hundreds or even thousands of edge types. Similarly to what happen in dimensionality reduction [Bur10], to make intuitive sense of such data, we want to reduce the number of edge types to less than ten. A natural way is to ask a domain expert to cluster edge types together. However, this is again time consuming, and does not necessarily make use of the topology of the graph.

3. The input graph is actually unlabeled. That is, we observe interactions between the nodes, and we assume from domain knowledge that these interactions fall into k categories. However, there is no reasonable way to come up with a specific labeling function. In Table 4.1 on page 92, we present several such examples, but for now we simply recall our earlier co-purchase network. As we mentioned, there are several reasons why two products could be bought together. Yet it is unlikely that customers will provide this type of feedback, for it does not bring them immediate advantage.

[4] http://app.mappr.io/play/chile-marine-intertidal-network

In this thesis, we consider three versions of the problem of predicting edge types.

1. The EDGE SIGN PREDICTION problem, which takes as input the topology of a directed signed graph, and the label of some of the edges. The output is the label of the remaining edges. Therefore, it can be seen as a supervised binary classification problem.

2. The same EDGE SIGN PREDICTION problem, where the input graph is undirected. In this case, we consider an active variant where, instead of being given a random training set of labels, we have a budget of queries we can use to request arbitrary labels.

3. The EDGE ATTRIBUTED CLUSTERING problem, which takes as input an unlabeled, undirected multilayer graph, the attributes of all its nodes, and a number of edge types k. The output is a k clustering of the edges, and a set of k vectors describing the clusters in terms of node attributes. Therefore, it can be seen as an unsupervised clustering problem with side information.

We now briefly review existing approaches addressing these problems, in order to highlight our contribution in the next section.

The modern formulation of the EDGE SIGN PREDICTION on directed graphs can be attributed to Leskovec *et al.* [LHK10]. Their idea is to compute local features of the nodes based on the training labels. Such features include variations on the node degree, such as the number of positive outgoing edges or the number of negative incoming edges. These node features are combined into edge features, and a supervised classification algorithm is trained. Several works have devised additional features [SM15; Yua+17], most notably based on scoring the nodes in a Page Rank fashion [TNV10; SJ14; WAS16]. These methods are accurate, fast —for they are local, and interpretable —for the features are hand crafted. The drawback is that this feature engineering is done mostly in an ad hoc way. On the other hand, it is also possible to look at the problem from a global point of view, by completing the signed adjacency matrix through low rank factorization [Chi+14; Wan+17a]. These methods are also accurate, and bypass feature engineering, but by nature, they require careful algorithms to scale with larger networks. Finally, during the writing of this manuscript, several papers have been published, which use embedding of the nodes in a low dimensional space, based on the training labels [IPR17; YWX17]. This also avoids feature engineering but is global in nature and not so interpretable.

One way to solve the problem on undirected graphs would be to apply the previous methods, having first replaced every edge (u, v) by two directed edges $u \to v$ and $v \to u$. This is not quite satisfactory, both from a theoretical but also practical point of view (see Section 3.1.2). Instead, Cesa-Bianchi *et al.* [Ces+12b] draw a connection between the EDGE SIGN PREDICTION problem and the CORRELATION CLUSTERING problem. Recall that given a fully labeled signed graph, the solution to CORRELATION CLUSTERING is a partition of the nodes that minimize the number of disagreement edges [BBC02]. Cesa-Bianchi *et al.* [Ces+12b] assume that the signs are originally consistent with an underlying, hidden 2-clustering, but that we only observe the signs after they have been flipped uniformly at random. In this case, they show that the optimal number of disagreements is a lower bound of the number of mistakes made by any active EDGE SIGN PREDICTION algorithm. One natural approach would then be to solve CORRELATION CLUSTERING based on the observed signs, and predict the remaining signs consistently with the inferred clusters. At first, it seems hopeless, as CORRELATION CLUSTERING is difficult to approximate on general graphs [CGW03], even when there are only two clusters [GG06]. However, this worst case analysis does not forbid more positive results on signed graphs that exhibit stability under perturbation [Ben15; BB09; NJ09] or are obtained through perturbations from an ideal case [AL09; MMV15]. Furthermore, in the active setting,

the learner gets to choose which signs are observed. The general idea of Cesa-Bianchi *et al.* [Ces+12b] is thus to use the query budget to build fully labeled paths between each connected nodes. Those paths must be as short as possible, since the predicted sign of (u, v) is the product of the signs along the path from u to v. Therefore, the shorter the path and the less influence of the random perturbations. At the same time, those paths must span the whole graph. This is the topic of an active research area [AN12; EN17].

As for the EDGE ATTRIBUTED CLUSTERING problem, to the best of our knowledge, it has not been studied under our assumptions. A more common problem in attributed graphs is to cluster nodes into communities [LM12; YML13; Xu+14a; ZLZ16]. However, it is not immediate how such methods, generally based on generative models, could be adapted to our problem. Direct approaches to classify edges have been proposed, based on graphical models [TZT11], nearest neighbors with a customized distance [AHZ16] and edge embedding in knowledge graphs [Bor+13]. Yet they all rely on having training labels as supervision. A simple unsupervised method is to cluster the line graph of the input graph [EL09], but this does not take advantage of attributes. Another method uses topological features [Ahm+17] but is rather complex and thus not very interpretable. If we further assume that each edge type is associated with a Euclidean space, and that the position of nodes in one space are not correlated with their position in another space, then it is possible to recover an approximation of these Euclidean metrics in polynomial time without any supervision [Abr+15].

1.4 Outline

The contributions of this thesis are organized as follow:

- We start in Chapter 2 by addressing the EDGE SIGN PREDICTION problem in Directed Signed Social Networks. Our goal is to design a method that is scalable, principled and accurate. For that, we start by introducing a generative model for signs, and derive approximations of the optimal Bayes predictor and of the maximum likelihood predictor. We confirm the theoretical soundness of these approaches by performing extensive synthetic and real world experiments. Finally, to the best of our knowledge, we are the first to give an online algorithm for the EDGE SIGN PREDICTION problem on directed graphs. This chapter is based on an existing publication [Le +17].

- We then move from directed to undirected signed graphs, and explain in Chapter 3 why this requires another learning bias. Namely, we assume that the nodes belong to k groups, and that the sign between two nodes is positive when both nodes belong to the same group, negative otherwise. We describe how this new bias is related to the CORRELATION CLUSTERING problem, and present a thorough overview of existing approaches. Finally, we provide the first implementation of an existing spanner construction [Vit14], and give a preview of its performance on synthetic and real signed graphs.

- In Chapter 4, we consider the EDGE ATTRIBUTED CLUSTERING problem on multilayer graphs with node attributes. Our approach is to seek a small number of vectors that, once assigned to every edge, best explain the graph (i.e. maximize a score function between the vector assigned to an edge and the profiles of this edge's endpoints). From this initial formulation, we derive two optimization problems, and show how to solve them on synthetic data.

- Finally, reflecting on our treatment of the three previous problems, we discuss in Chapter 5 other settings and methods that might extend the problem of characterizing edges in complex networks beyond the frame of this thesis.

Chapter 2

On the Troll-Trust Model for edge sign prediction in Social Networks

As we saw in the introduction, many situations in various fields can be modeled by signed graphs. Nonetheless, one of the most natural usage of signed graphs is the study of social networks. It is indeed what first motivated their inception. Quite naturally, we thus begin our exploration of the role of edges in complex networks by a binary classification problem in signed graphs. Namely, we want to predict whether an observed interaction between two nodes is positive or negative, and we call this problem EDGE SIGN PREDICTION.[1] To motivate this problem from a practical point of view, let us first define precisely the type of social networks we consider. We call *Directed Signed Social Network* (DSSN) a directed graph whose nodes are either human beings (often called users if they are the members of an online community) or artifacts directly created by individuals[2]. Furthermore, a directed edge from node u to node v denote an interaction initiated by the user u and whose object is user v. Such an interaction can be positive (to praise, to support, to trust, to befriend) or negative (to criticize, to oppose, to distrust, to make enemy with).

Known examples are EBAY, where users trust or distrust agents in the network based on their personal interactions, SLASHDOT, where each user can tag another user as friend or foe, BLABLACAR, a carpooling website where users can evaluate drivers and passengers as pleasant travelers or not, and EPINION, where users can rate positively or negatively not only products, but also other users. In such examples, the sign of the interaction is explicit. However, there are other cases where such interactions are only displayed in their raw form and need further processing to be given a sign. The typical situation where this happens is when interactions are mediated through text, for instance in Twitter or in the comment section of some online content. It is then possible to extract signs using sentiment analysis methods [HAR12].

Another distinction is that in most cases, only the owner of the network can observe the detail of all those interactions. Think for instance of YOUTUBE, where users can like or dislike the video of another user but where only aggregate anonymous statistics are available publicly. The situation is similar on the STACK OVERFLOW community, where users can upvote or downvote answers according to their perceived quality, but where only the total score are available.

With this definition of social networks at hand, we now look at how the learning problem of EDGE SIGN PREDICTION can be used in real world applications. More precisely, we present four possible applications, and order them by how severely negative interactions impact the well-being of users of the network. First, it could help improve the quality of link recommendation in solely positive networks.

[1] This chapter is closely based on a paper written with several co-authors [Le +17].

[2] This semantic detail allows us to treat citation networks as social networks.

Such networks indeed contain implicit negative links [Yan+12]. By asking users to label a small proportion of existing links as truly positive or actually negative, recommended links could later be discarded if they are predicted to be negative. Second, this could be used to monitor news textual comments in a scalable fashion and thus ensure that online debates remain courteous and constructive. For instance, Manosevitch *et al.* [MSL14] show that by visually reminding comment authors of the importance of respecting the plurality of opinions, the quality of debates is improved (other references can be found in [Dia15]). In a more extreme case, banning users with aggressive and hateful behavior has proved effective in the case of Reddit [Cha+17]. Third, like many other fields, human resource management is undergoing a transformation through the increasing use of Machine Learning [Tom16]. Among other tools, assessing personal relationships between employees is crucial to maintain a productive work environment [MMB02]. Whereas this can be done by invasive wearable sensors [OGP09], our methods could predict which employees would form the tighter teams from a small amount of labeled data. Fourth, and most critical, predicting the sign of interactions could be a tool to detect users with a high proportion of negative interactions. Indeed, because people tend to be less inhibited in their online interactions [Sul04], some users may join an online community with the main goal to disrupt it, by engaging into anti-social behavior and creating conflictual relationships with other members. This kind of attitude expressed publicly on social media leads to the following definition of *trolls*: "users whose real intentions are to cause disruption and/or to trigger or exacerbate conflict for the purposes of their own amusement" [Har10]. Shachaf *et al.* [SH10] elaborate on their motives, adding that boredom, attention seeking, and revenge motivate trolls; they find pleasure from causing damage to other people or to the community as a whole. An extreme form of trolling is bullying (i.e. the behavior of someone intentionally and repeatedly harming a victim that is unable to defend himself or herself) and just like anything else, it has an online version called cyber bullying [SSF13]. Such behavior is rather widespread, one study revealing that 72% of 1 454 surveyed teenagers reported at least one occurrence of cyberbullying, most of them through instant messaging [JG08]. While we have not conducted experimental study on this topic, we believe that being able to predict such harmful interactions would be beneficial for the majority of users.

Such applications motivate studying EDGE SIGN PREDICTION, which is the problem of classifying the positive or negative nature of the links based on the network topology. Prior knowledge of the network topology is often a realistic assumption, for in several situations the discovery of the link sign can be more costly than acquiring the topological information of the network. For instance, when two users of an online social network communicate on a public web page, we immediately detect a link. Yet, the classification of the link sign as positive or negative may require more involved techniques.

From the modeling and algorithmic viewpoints, because of the huge amount of available networked data, a major concern in developing learning methods for EDGE SIGN PREDICTION is algorithmic scalability. Many successful, yet simple heuristics for EDGE SIGN PREDICTION are based on the troll-trust features, i.e. on the fraction of outgoing negative links (trollness) and incoming positive links (trustworthiness) at each node. We first define these notions and more notations in Section 2.1, along with others tools and precise problem statements. We also introduce suitable graph transformations defining reductions from EDGE SIGN PREDICTION to node sign prediction problems. Then we study such troll-trust heuristics by defining in Section 2.2 a probabilistic generative model for the signs on the directed links of a given network. We also show that these heuristics can be understood and analyzed as approximators to the Bayes optimal classifier for our generative model. In this context, we design our first batch algorithm and show in Section 2.3.1 that it provably approximates the Bayes classifier on dense graphs.

We furthermore gather empirical evidence supporting our probabilistic model by observing that a logistic model trained on trollness and trustworthiness features alone is able to learn weights that, on all datasets considered in our experiments, consistently satisfy the properties predicted by our model.

Our graph transformations opens up the possibility of using the arsenal of known algorithmic techniques developed for node classification. In particular, we introduce in Section 2.3.2 our second batch algorithm. It takes the form of a Label Propagation algorithm that, combined with our problem reduction, approximates the maximum likelihood estimator of our probabilistic generative model. In order to compare our two algorithms with existing work, we then describe with more details previous heuristics and related methods in Section 2.4. We then experimentally evaluate our proposed approach in Section 2.5. On synthetic data, we confirm the quality of our approximation of the Bayes predictor. More importantly, on real-world data, we show the competitiveness of our approach in terms of both prediction performance (especially in the regime where training data are scarce) and scalability.

Finally, in Section 2.6, we point out that the notions of trollness and trustworthiness naturally define a measure of complexity, or learning bias, for the signed network that can also be used to design *online* (i.e. sequential) learning algorithms for the EDGE SIGN PREDICTION problem. The learning bias encourages settings where the nodes in the network have polarized features (e.g., trollness/trustworthiness are either very high or very low). Our online analysis holds under adversarial conditions, namely, without any stochastic assumption on the assignment of signs to the network links.

2.1 Notation and Preliminaries

In this chapter[3], we let $G = (V, E)$ be a *directed* graph, whose edges $(u, v) \in E$ carry a binary label $y_{u,v} \in \{-1, +1\}$. The edge labeling will sometimes be collectively denoted by the $|V| \times |V|$ matrix $Y = [y_{u,v}]$, where $Y_{u,v} = y_{u,v}$ if $(u, v) \in E$, and $y_{u,v} = 0$ otherwise. The corresponding edge-labeled graph will be denoted by $G(Y) = (V, E(Y))$. We use $\mathcal{E}_{\text{in}}(u)$ and $\mathcal{E}_{\text{out}}(u)$ to denote, respectively, the set of edges incoming to and outgoing from node $u \in V$, with $d_{\text{in}}(u) = \left|\mathcal{E}_{\text{in}}(u)\right|$ and $d_{\text{out}}(u) = \left|\mathcal{E}_{\text{out}}(u)\right|$ being the in-degree and the out-degree of u. Moreover, $d^+_{\text{in}}(u)$ is the number of edges $(w, u) \in \mathcal{E}_{\text{in}}(u)$ such that $y_{w,u} = +1$. We define $d^-_{\text{in}}(u)$, $d^+_{\text{out}}(u)$, and $d^-_{\text{out}}(u)$ similarly, so that, for instance, $d^-_{\text{out}}(u)/d_{\text{out}}(u)$ is the fraction of outgoing edges from node u whose label in $G(Y)$ is -1. We call $tr(u) = d^-_{\text{out}}(u)/d_{\text{out}}(u)$ the *trollness* of node u, and $un(u) = d^-_{\text{in}}(u)/d_{\text{in}}(u)$ the *untrustworthiness* of node u. Finally, we also use the notation $\mathcal{N}_{\text{in}}(u)$ and $\mathcal{N}_{\text{out}}(u)$ to represent, respectively, the in-neighborhood and the out-neighborhood of node $u \in V$. Most of these notations are illustrated on Figure 2.1.

Given the directed graph $G = (V, E)$, we define two *edge-to-node reductions* transforming the original graph G into other graphs. As we see later, these reductions are useful in turning the EDGE SIGN PREDICTION problem into a *node* sign prediction problem (often called node classification problem), for which many algorithms are indeed available [BC01; ZGL03; BDL06; HP07; HLP09; Vit+11; HPV12; Ces+13; HPG15]. Although any node classification method could in principle be used, the reductions we describe next are essentially aimed at preparing the ground for quadratic energy-minimization approaches computed through a *Label Propagation* algorithm [ZGL03; BDL06].

The first reduction builds an *undirected* graph $G' = (V', E')$ as follows. Each node $u \in V$ has two copies in V', call them u_{in} and u_{out}. Each directed edge (u, v) in E is associated with one node, call it $e_{u,v}$, in V', along with the two undirected edges $(u_{\text{out}}, e_{u,v})$ and $(e_{u,v}, v_{\text{in}})$. Hence $|V'| = 2|V| + |E|$ and $|E'| = 2|E|$. Moreover, if $G =$

[3] And in the next chapter when applicable.

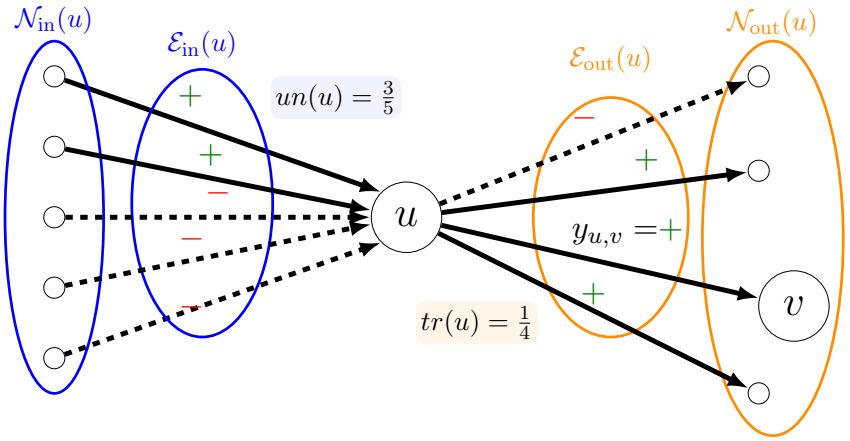


Figure 2.1 – Part of a DSSN centered on node u.

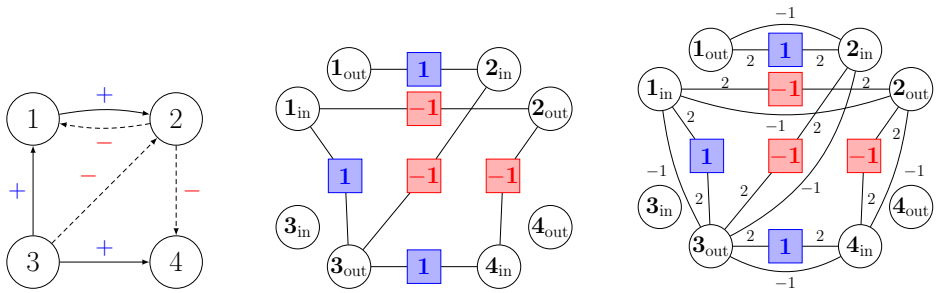


Figure 2.2 – **(a)** A directed edge-labeled graph G. **(b)** Its corresponding graph G' resulting from the first reduction we describe. The square nodes in G' correspond to the edges in G, and carry the same labels as their corresponding edges. On the other hand, the $2|V|$ circle nodes in G' are unlabeled. Observe that some nodes in G' are isolated and thus unimportant. These are exactly the nodes in G' corresponding to the nodes having in G no outgoing or no incoming edges: for instance nodes 3 and 4 in G. **(c)** The weighted graph resulting from the second reduction we describe.

$G(Y)$ is edge labeled, then this labeling transfers to the subset of nodes $e_{u,v} \in V'$, so that G' is a graph $G'(Y) = (V'(Y), E')$ with partially-labeled nodes. The second reduction builds an *undirected and weighted* graph $G'' = (V'', E'')$. Specifically, we have $V'' \equiv V'$ and $E'' \supset E'$, where the set E'' also includes edges $(u_{\text{out}}, v_{\text{in}})$ for all u and v such that $(u, v) \in E$. The edges in E' have weight 2, whereas the edges in $E'' \setminus E'$ have weight -1. We provide in Section 2.9.1.3 an analytic justification for this weights choice in the context of Label Propagation. Finally, as in the first reduction, if $G = G(Y)$ is edge labeled, then this labeling transfers to the subset of nodes $e_{u,v} \in V''$. Graph G', which will not be used in the following, is an intermediate structure between G and G'' and provides a conceptual link to the standard cutsize measure in node sign classification, as we will describe shortly. Figure 2.2 illustrates these two reductions. Note that because $|V''| = 2|V|$ and $|E''| = 3|E|$, the reduction yielding G'' can be computed in linear time, and does not require the knowledge of the edge label, meaning it can be done in parallel with potential label querying. Furthermore, it can be updated incrementally if new nodes or edges are added to the original graph G.

These reductions serve two purposes: First, they allow us to use the many algorithms designed for the better studied problem of node sign prediction. Second, the reduction yielding G'', along its specific choice of edge weights, is designed to make the Label Propagation solution approximate the maximum likelihood estimator associated with our generative model (see Section 2.3.2). Note also that efficient Label Propagation implementations exist that can leverage the sparsity of

G''.

We consider two learning settings associated with the problem of EDGE SIGN PREDICTION: a batch setting and an online setting. In the batch setting, we assume that a training set of edges E_0 has been drawn uniformly at random *without replacement* from E, we observe the labels in E_0, and we are interested in predicting the sign of the remaining edges $E \setminus E_0$ by making as few prediction mistakes as possible. The specific batch setting we study here assumes that labels are produced by a generative model which we describe in the next section, and our label regularity measure is a quadratic function (denoted by $\Psi^2_{G''}(Y)$ and defined in Section 2.5 as a regularized energy function of G''[4]) related to this model. $\Psi^2_{G''}(Y)$ is small just when all nodes in G tend to be either troll or trustworthy.

On the other hand, the *online* setting we consider is the standard mistake bound model of online learning [Lit88] where all edge labels are assumed to be generated by an adversary and sequentially presented to the learner according to an arbitrary permutation. More precisely, in this setting, learning is split into a sequence of rounds: At each round $t = 1, 2, \ldots$ a learning algorithm A outputs a prediction $\widehat{y}_t \in \{-1, +1\}$ for the label $y_t \in \{-1, +1\}$ of an edge arbitrarily selected by the adversary. After the prediction is made, label y_t is revealed to the algorithm, hence allowing it to change its internal state. In the next round, a new edge is selected by the adversary, and so on. Notice that since the underlying labeling over the edges is decided by the adversary once and for all,[5] all edges occur exactly once within the sequence to be predicted (so that this game lasts exactly $|E|$ rounds). The adversary decides both the underlying labeling Y over the edges of G and the order of their presentation to the learning algorithm. We say that A has made a mistake at time t if $\widehat{y}_t \neq y_t$, and we measure A's prediction performance simply through the total number of mistakes $M_A(Y)$ it makes over $G(Y)$ when the worst possible presentation order of the edge labels in Y is selected by the adversary. As it is standard practice, we contrast $M_A(Y)$ to some kind of *regularity* measure $\Psi_G(Y)$ of the labeling Y over G, so that we are in fact aimed at bounding the (cumulative) regret $M_A(Y) - \Psi_G(Y)$.

We want to design our label regularity measure such that it is small when nodes in G tend to be either troll or trustworthy. Indeed, when this happens, few labels from the incoming and outgoing edges of each node are sufficient to predict the labels on the remaining edges and the EDGE SIGN PREDICTION problem is to some extent "easy". Formally, for fixed G and Y, let

$$\Psi_{\text{in}}(v, Y) = \min\left\{d^-_{\text{in}}(v), d^+_{\text{in}}(v)\right\} \qquad \text{and} \qquad \Psi_{\text{out}}(u, Y) = \min\left\{d^-_{\text{out}}(u), d^+_{\text{out}}(u)\right\}$$

be respectively the number of the least used label in the incoming edges to v and the outgoing edges from u. Let also

$$\Psi_{\text{in}}(Y) = \sum_{v \in V} \Psi_{\text{in}}(v, Y) \qquad \text{and} \qquad \Psi_{\text{out}}(Y) = \sum_{u \in V} \Psi_{\text{out}}(u, Y)$$

be the sum of irregularity over all the nodes of the graph. Then we define $\Psi_G(Y) = \min\left\{\Psi_{\text{in}}(Y), \Psi_{\text{out}}(Y)\right\}$. The two measures $\Psi_G(Y)$ and $\Psi^2_{G''}(Y)$ are conceptually related and their value on real data is quite similar (see Table 2.4 on page 25).

The reductions presented above are meaningful only if they are able to approximately preserve the two label regularity measures $\Psi_G(Y)$ and $\Psi^2_{G''}(Y)$ when moving from edges to nodes. That is, if the EDGE SIGN PREDICTION problem is easy for a given $G(Y) = (V, E(Y))$, then the corresponding node sign prediction problems on $G'(Y) = (V'(Y), E')$ and $G''(Y) = (V''(Y), E)$ are also easy, and vice versa.

[4] If we denote the value of the square node $e_{u,v}$ of G'' as $\frac{1+y_{u,v}}{2}$, the value of the circle node u_{out} as p_u and the value of the circle node v_{in} as q_v, we have that $\Psi^2_{G''}(Y) = \min_{(\boldsymbol{p},\boldsymbol{q})} \sum_{(u,v)\in E} \left(\frac{1+y_{u,v}}{2} - \frac{p_u+q_v}{2}\right)^2$, where $(\boldsymbol{p}, \boldsymbol{q}) = \{p_u, q_u\}_{u=1}^{|V|}$.

[5] For simplicity, we assume the adversary is deterministic.

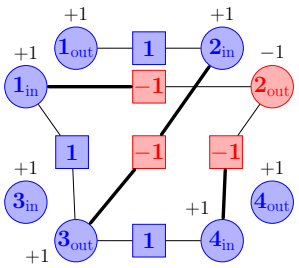


Figure 2.3 – The node-labeling of G' illustrating the relation between the edge label complexity and the cutsize. Here the four edges part of the cut are in heavy black.

Let us first introduce the notion of cutsize of an undirected node-labeled graph $G'(Y)$, which is the number of edges in G' connecting nodes that have mismatching labels. Now, because the only nodes in G' we are interested in predicting are those corresponding to the edges in G (the colored squares in Figure 2.2b), the online prediction problem on the edges of G translates to a node sign prediction problem on a subset of V'. As for the remaining nodes in V' (the circles in Figure 2.2b), we are free to assign arbitrary labels so as to minimize the corresponding mistake bound over G'. Such an assignment is constructed as follow: we set the labeling Y on V' in such a way that $y_{u_{\text{out}}} = +1$ if $tr(u) \leq 1/2$ and -1 otherwise. Similarly, we have $y_{u_{\text{in}}} = +1$ if $un(u) \leq 1/2$, and -1 otherwise. For instance, given the trollness and untrustworthiness values for the graph G of Figure 2.2a reported in Table 2.1, following this construction would result in a labeled version of G' shown in Figure 2.3. We see that by labeling circle nodes by the majority label of the square nodes there are connected to, we have that the cutsize of $G'(Y)$ equals $\Psi_{\text{in}}(Y) + \Psi_{\text{out}}(Y)$.[6] A similar reasoning apply in the batch case with the $\Psi^2_{G''}(Y)$ measure, which can be seen as a soft quadratic version of the cutsize.

Table 2.1 – The label regularity values for the nodes of the example G graph of Figure 2.2a.

node u	1	2	3	4	total
$\Psi_{\text{out}}(u, Y)$	0	0	1	0	1
$\Psi_{\text{in}}(u, Y)$	1	1	0	1	3
$tr(u)$	0	1	1/3	1	—
$un(u)$	1/2	1/2	1/2	1/2	—

2.2 Generative Model for Edge Labels

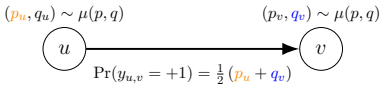


Figure 2.4 – The sign $y_{u,v}$ of the edge $u \to v$ is positive with probability $\frac{1}{2}(p_u + q_v)$.

We now define the stochastic generative model for edge labels we use in the batch learning setting. Given the graph $G = (V, E)$, let the label $y_{u,v} \in \{-1, +1\}$ of directed edge $(u, v) \in E$ be generated as follows. Each node $u \in V$ is endowed with two latent parameters $p_u, q_u \in [0, 1]$, which we assume to be generated, for each node u, by an independent draw from a fixed but unknown joint prior distribution $\mu(p, q)$ over $[0, 1]^2$. Each label $y_{u,v} \in \{-1, +1\}$ is then generated by an independent draw from the mixture of p_u and q_v

$$\Pr\left(y_{u,v} = 1\right) = \frac{p_u + q_v}{2}$$

This process is illustrated in Figure 2.4.

The basic intuition is that the nature $y_{u,v}$ of a relationship $u \to v$ is stochastically determined by a mixture between how much node u tends to like other people (p_u) and how much node v tends to be liked by other people (q_v). In a certain sense, $1 - tr(u)$ is the empirical counterpart to p_u, and $1 - un(v)$ is the empirical counterpart to q_v. One might view our model as reminiscent of standard models for

[6] In fact, for the sake of this specific argument, nothing prevents from retaining of G' either only the edges $(u_{\text{out}}, e_{u,v})$ or only the edges $(e_{u,v}, v_{\text{in}})$, resulting in a cutsize of $\Psi_{\text{out}}(Y)$ and $\Psi_{\text{in}}(Y)$, respectively.

link generation in social network analysis, like the classical p_1 model from [HL81]. However, note that the similarity falls short, for all these models aim at representing the likelihood of the network topology, rather than the probability of edge signs, once the topology is *given*.

Notice that the Bayes optimal prediction for $y_{u,v}$ is $y^*(u,v) = \operatorname{sign}\left(\eta(u,v) - \frac{1}{2}\right)$, where $\eta(u,v) = \Pr\left(y_{u,v} = 1\right)$. Moreover, once all the signs have been generated, we can compute the expected number of positive edges outgoing from u as the expected value of the random variable $P_u = \sum_{v \in \mathcal{N}_{\text{out}}(u)} \mathbb{I}\{y_{u,v} = +1\}$. Using this, we have that the probability of drawing at random a $+1$-labeled edge from $\mathcal{E}_{\text{out}}(u)$ equals

$$\frac{\mathbb{E}\left(P_u\right)}{d_{\text{out}}(u)} = \frac{1}{d_{\text{out}}(u)} \sum_{v \in \mathcal{N}_{\text{out}}(u)} \frac{p_u + q_v}{2} = \frac{1}{2}\left(p_u + \frac{1}{d_{\text{out}}(u)} \sum_{v \in \mathcal{N}_{\text{out}}(u)} q_v\right) = \frac{1}{2}\left(p_u + \overline{q}_u\right), \tag{2.1}$$

where $\overline{q}_u = \frac{1}{d_{\text{out}}(u)}\left(\sum_{v \in \mathcal{N}_{\text{out}}(u)} q_v\right)$ is the average q value of u out neighbors. Similarly, the probability of drawing at random a $+1$-labeled edge from $\mathcal{E}_{\text{in}}(v)$ equals

$$\frac{1}{2}\left(q_v + \frac{1}{d_{\text{in}}(v)} \sum_{u \in \mathcal{N}_{\text{in}}(v)} p_u\right) = \frac{1}{2}\left(q_v + \overline{p}_v\right) \tag{2.2}$$

2.3 Algorithms in the Batch Setting

Given $G(Y) = (V, E(Y))$, in the batch setting we have at our disposal a training set E_0 of labeled edges from $E(Y)$. Formally, we have a training set $E_0 = \left((u_1, v_1), y_{u_1,v_1}\right), \left((u_2, v_2), y_{u_2,v_2}\right), \ldots, \left((u_m, v_m), y_{u_m,v_m}\right)$, that has been drawn from $E \times Y$ uniformly at random without replacement, with $m = |E_0|$. We want to build a predictive model for the labels of the remaining edges. We present two algorithms to do so, which both compute estimates of the all parameters p and q of our generative model. They differ in the approximation guarantees they provide, and in the class of graphs to which they apply. The first algorithm runs on the graph G and estimates locally the parameters by their empirical means in the training set, which under some density assumptions are showed to concentrate around their true values. The second algorithm, on the other hand, exploits the reduced graph G'' and computes a maximum likelihood estimation of the parameters through a global label propagation approach, without making any density assumption.

2.3.1 Approximation to Bayes via dense sampling

Our first algorithm is an approximation to the Bayes optimal predictor $y^*(u,v)$. Let us denote by $\widehat{tr}(u)$ and $\widehat{un}(u)$ the trollness and the untrustworthiness of node u when both are computed on the subgraph induced by the training edges. Recall that the Bayes optimal predictor classifies an edge $u \to v$ using the following rule: $y^*(u,v) = \operatorname{sign}\left(\frac{p_u + q_v}{2} - \frac{1}{2}\right)$. Our approximation thus consists in using the quantities $\widehat{tr}(u)$ and $\widehat{un}(u)$ to estimate $\frac{p_u + q_v}{2}$. This results in the following rule:

$$\operatorname{sign}\left(\left(1 - \widehat{tr}(u)\right) + \left(1 - \widehat{un}(v)\right) - \tau - \frac{1}{2}\right), \tag{2.3}$$

where $\tau \geq 0$ is the only parameter to be trained. We now give an intuition to justify this equation, and defer the technical arguments to end of the chapter, on page 37, where we will formalize what it means for a quantity to be "close" to another. Note first that $1 - \widehat{tr}(u) = \frac{\widehat{d}^{+}_{\text{out}}(u)}{\widehat{d}_{\text{out}}(u)}$ is the empirical mean[7] of the probability of a random edge outgoing from u to be positive, and is therefore "close" to $\frac{1}{2}\left(p_u + \overline{q}_u\right)$

[7] The hat symbol denote quantities computed solely on the training set.

according to (2.1). By the same reasoning, $1 - \widehat{un}(v) \approx \frac{1}{2}(q_v + \overline{p}_v)$ in accordance with (2.2). At this stage, $\big(1 - \widehat{tr}(u)\big) + \big(1 - \widehat{un}(v)\big) - \frac{p_u+q_v}{2} \approx \frac{1}{2}(\overline{p}_v + \overline{q}_u)$. Since $\overline{p}_v$ is a sample mean of i.i.d. $[0,1]$-valued random variables independently drawn from the prior marginal $\int_0^1 \mu(\cdot, q)dq$, it concentrates around its expectation μ_p. Likewise, $\overline{q}_u \approx \mu_q$. Now we seek an estimation of $\frac{1}{2}(\overline{p}_v + \overline{q}_u) \approx \frac{1}{2}(\mu_p + \mu_q)$. For that, we compute the expected number of positive edges in the graph, which, as in the previous section, is the expected value of the random variable $Z = \sum_{u \to v \in E} \mathbb{I}\{y_{u,v} = +1\}$. Let V_{out} be the subset of node of V with at least one outgoing edge, and define similarly V_{in} as $\{v \in V : d_{\text{in}}(v) > 0\}$. According to our generative model, $\mathbb{E}(Z) = \sum_{u \to v \in E} \frac{p_u+q_v}{2} = \sum_{u \in V_{\text{out}}} \left(\sum_{v \in \mathcal{N}_{\text{out}}(u)} \frac{p_u+q_v}{2} \right)$. Observe that

$$\sum_{u \in V_{\text{out}}} \left(\sum_{v \in \mathcal{N}_{\text{out}}(u)} p_u \right) = \sum_{v \in V_{\text{in}}} \left(\sum_{u \in \mathcal{N}_{\text{in}}(v)} p_u \right) = \sum_{v \in V_{\text{in}}} d_{\text{in}}(v)\overline{p}_v \approx \sum_{v \in V_{\text{in}}} d_{\text{in}}(v)\mu_p = |E|\mu_p$$

and

$$\sum_{u \in V_{\text{out}}} \left(\sum_{v \in \mathcal{N}_{\text{out}}(u)} q_v \right) = \sum_{u \in V_{\text{out}}} d_{\text{out}}(u)\overline{q}_u \approx \sum_{u \in V_{\text{out}}} d_{\text{out}}(u)\mu_q = |E|\mu_q \,.$$

Therefore, letting τ be the fraction of positive edge in the graph, we have that $\tau = \frac{\mathbb{E}(Z)}{|E|} \approx \frac{1}{2}(\mu_p + \mu_q)$. This means that if $\widehat{\tau}$ is the quantity τ computed on the training, then $\widehat{\tau}$ is close to $(\overline{p}_v + \overline{q}_u)$. Putting together all this approximations thus provide an informal justification of the rule (2.3).

We are now ready to describe our algorithm, which follows naturally from the formula derived above. It takes as input a training set E_0 drawn at random without replacement and outputs a sign prediction for all the edges in $E \setminus E_0$. We call it BLC(tr, un), which stands for Bayes Learning Classifier based on *tr*ollness and *un*trustworthiness.

1. For each $u \in V$, let $\widehat{tr}(u) = \widehat{d}^{-}_{\text{out}}(u)/\widehat{d}_{\text{out}}(u)$, i.e. the fraction of negative edges found in $\mathcal{E}_{\text{out}}(u) \cap E_0$.
2. For each $v \in V$, let $\widehat{un}(v) = \widehat{d}^{-}_{\text{in}}(v)/\widehat{d}_{\text{in}}(v)$, i.e. the fraction of negative edges found in $\mathcal{E}_{\text{in}}(v) \cap E_0$.
3. Let $\widehat{\tau}$ be the fraction of positive edges in $E_L \cap E_0$ (this set E_L is a technical requirement that we shall define shortly).
4. Any edge $(u, v) \in E \setminus E_0$ is predicted as

$$\widehat{y}(u, v) = \text{sign}\left(\big(1 - \widehat{tr}(u)\big) + \big(1 - \widehat{un}(v)\big) - \widehat{\tau} - \tfrac{1}{2} \right)$$

Despite its apparent simplicity, BLC(tr, un) works reasonably well in practice, as demonstrated by our experiments (see Section 2.5). Moreover, unlike previous edge sign prediction methods for directed graphs, our classifier comes with a rigorous theoretical motivation, since it approximates the Bayes optimal classifier $y^*(u, v)$ with respect to the generative model defined in Section 2.2. Indeed, we quantify this approximation on nodes whose in-degree and out-degree are not too small in the next result. However, this first requires a few extra assumption on the training set. Namely, given parameters Q (a positive integer) and $\alpha = |E_0|/|E| \in (0, 1)$, we assume there exists a set $E_L \subseteq E$ of size $\frac{2Q}{\alpha}$ where each vertex $u \in V$ appearing as an endpoint of some edge in E_L occurs at most once as origin —i.e. $u \to v$— and at most once as destination —i.e. $v \to u$. While the definition of E_L is not immediately intuitive, this set is needed to find an estimate $\widehat{\tau}$ of τ in (2.3) during Step 3 of BLC(tr, un), and its definition allows us to applying an Hoeffding bound on independent variables. Any undirected matching of G of size $O(\log |V|)$ can be

used, obtained for instance by the blossom algorithm [Edm65]. In practice, however, we never computed E_L, and estimated τ on the entire training set E_0 (instead of $E_L \cap E_0$).

Theorem 1. *Let $G(Y) = (V, E(Y))$ be a directed graph with labels on the edges generated according to the model in Section 2.2. If the algorithm is run with parameter $Q = \Omega(\log |V|)$, and $\alpha \in (0,1)$ such that the above assumption about E_L is satisfied, then $\widehat{y}(u,v) = y^*(u,v)$ holds with high probability simultaneously for all test edges $(u,v) \in E$ such that $d_{\text{out}}(u), d_{\text{in}}(v) = \Omega(\log |V|)$, and $\eta(u,v) = \Pr(y_{u,v} = 1)$ is bounded away from $\frac{1}{2}$.*

While we defer the full proof of Theorem 1 to the additional material at the end of this chapter (Section 2.9.1.1), here we give a sketch of the method used. First, by Lemma 2 on negatively associated random variables (stated on page 37), we show that with our choice of Q, for $\theta = \frac{2Q}{\alpha}$ and for any u having enough out neighbors (i.e. $d_{\text{out}}(u) \geq \theta$), at least Q edges of $\mathcal{E}_{\text{out}}(u)$ are in the training set with high probability. Likewise, for any v with enough in neighbors (i.e. $d_{\text{in}}(v) \geq \theta$), at least Q edges of $\mathcal{E}_{\text{in}}(v)$ are in $E_0$with high probability. With this number of samples, we then show a chain of concentration results roughly following the outline given earlier to justify the equation (2.3), culminating in proving that for any $0 < \varepsilon < \frac{1}{16}$,

$$\left| \left(1 - \widehat{tr}(u)\right) + \left(1 - \widehat{un}(v)\right) - \widehat{\tau} - \frac{p_u + q_v}{2} \right| \leq 8\varepsilon$$

simultaneously holds with high probability for each

$$(u,v) \in \{(u,v) \in E \,:\, d_{\text{in}}(v) \geq \theta,\, d_{\text{out}}(u) \geq \theta\} \setminus E_0.$$

The approach leading to Theorem 1 requires the graph to be sufficiently dense. At first sight, having $Q = \Omega(\log |V|)$ training edges per nodes appears to be a reasonable assumption. Consider for instance Facebook —which is neither signed nor directed though. It has two billion users as of 2017 [8], each of them having $155 \approx 7.2 \log |V|$ friends on average [Dun16]. However, the constant in the Ω notation is a trade off between the number of edges to sample per node, and the quality guarantee of the $\frac{p_u+q_v}{2}$ estimation. In the Facebook example, having a good guarantee that holds simultaneously for all the test edges might require more than $7.2 \log |V|$ samples per node. We will nonetheless see in the experiments that we can still apply $\text{BLC}(tr, un)$ with satisfying results, especially since its simplicity makes it very scalable. Additionally, and in order to sidestep this density limitation, we now introduce a second method based on label propagation.

2.3.2 Approximation to Maximum Likelihood via Label Propagation

Remember we suppose that the training set E_0 has been drawn uniformly at random without replacement, with $m = |E_0|$. Then a reasonable approach to approximate $y^*(u,v)$ would be to resort to a maximum likelihood estimator of the parameters $\{p_u, q_u\}_{u=1}^{|V|}$ based on E_0. If we further assume, in order to make the computation more tractable, that the joint prior distribution $\mu(p,q)$ is uniform over $[0,1]^2$ with independent marginals,[9] we show in the supplementary material on page 40 that for $\ell \in \{1, \ldots, |V|\}$ the gradient of the log-likelihood function with respect to a given p_ℓ and q_ℓ satisfies

$$\frac{\partial \log \Pr\left(E_0 \,\middle|\, \{p_u, q_u\}_{u=1}^{|V|}\right)}{\partial p_\ell} = \sum_{k=1}^{m} \frac{\mathbb{I}\{u_k = \ell, y_{\ell,v_k} = +1\}}{p_\ell + q_{v_k}} - \sum_{k=1}^{m} \frac{\mathbb{I}\{u_k = \ell, y_{\ell,v_k} = -1\}}{2 - p_\ell - q_{v_k}} \tag{2.4}$$

[8] https://investor.fb.com/investor-news/press-release-details/2017/Facebook-Reports-Second-Quarter-2017-Results/

[9] As we will see, in real data, around 80% if the edges are positive, meaning this assumption of uniformity over $[0,1]^2$ is unlikely to fully holds, for otherwise the signs would more balanced.

$$\frac{\partial \log \Pr\left(E_0 \,\middle|\, \{p_u, q_u\}_{u=1}^{|V|}\right)}{\partial q_\ell} = \sum_{k=1}^{m} \frac{\mathbb{I}\{v_k = \ell, y_{u_k,\ell} = +1\}}{p_{u_k} + q_\ell} - \sum_{k=1}^{m} \frac{\mathbb{I}\{v_k = \ell, y_{u_k,\ell} = -1\}}{2 - p_{u_k} - q_\ell}\,, \tag{2.5}$$

where $\mathbb{I}\{\cdot\}$ is the indicator function of the event at argument. Unfortunately, equating (2.4) and (2.5) to zero, and solving for parameters $\{p_u, q_u\}_{u=1}^{|V|}$ gives rise to a hard set of nonlinear equations. Moreover, some such parameters may never occur in these equations, namely whenever $\mathcal{E}_{\text{out}}(u)$ or $\mathcal{E}_{\text{in}}(v)$ are not represented in E_0 for some $u, v \in V$.

Our *first approximation* is therefore to replace the nonlinear equations resulting from (2.4) and (2.5) by a set of linear equations. In the case of (2.4), for a given $\ell \in V$, we make the assumption that $(p_\ell + q_{v_k})(2 - p_\ell - q_{v_k})$ is constant for every $k \in [1, \dots, m]$. Multiplying (2.4) by this constant quantity and setting the resulting equation to zero, we obtain for each $\ell \in V$:

$$\sum_{k=1}^{m} \mathbb{I}\{u_k = \ell, y_{\ell,v_k} = +1\}\,(2 - p_\ell - q_{v_k}) = \sum_{k=1}^{m} \mathbb{I}\{u_k = \ell, y_{\ell,v_k} = -1\}\,(p_\ell + q_{v_k}) \tag{2.6}$$

We apply a similar transformation to (2.5) in order to obtain, for each $\ell \in V$:

$$\sum_{k=1}^{m} \mathbb{I}\{v_k = \ell, y_{u_k,\ell} = +1\}\,(2 - p_{u_k} - q_\ell) = \sum_{k=1}^{m} \mathbb{I}\{v_k = \ell, y_{u_k,\ell} = -1\}\,(p_{u_k} + q_\ell) \tag{2.7}$$

At this point, we find convenient to take a step back and define one of the label regularity measure we introduced in Section 2.1. Namely, recall we said $\Psi^2_{G''}(Y)$ could be seen as a soft quadratic version of the cutsize. With the notation of our generative model, we can write it as

$$\Psi^2_{G''}(Y) = \min_{(\boldsymbol{p},\boldsymbol{q})} \sum_{(u,v)\in E} \left(\frac{1 + y_{u,v}}{2} - \frac{p_u + q_v}{2}\right)^2 = \min_{(\boldsymbol{p},\boldsymbol{q})} f_E(\boldsymbol{p}, \boldsymbol{q})\,,$$

where $(\boldsymbol{p}, \boldsymbol{q}) = \{p_u, q_u\}_{u=1}^{|V|}$ is the set of the model parameters. Intuitively, minimizing f_E with respect to $(\boldsymbol{p}, \boldsymbol{q})$ is similar to the maximum likelihood approach, as we seek the parameters $(\boldsymbol{p}, \boldsymbol{q})$ that "best agree" with the observed signs. It also turns out that if we restrict the minimization problem to the observed edges (i.e. $\min_{(\boldsymbol{p},\boldsymbol{q})} f_{E_0}(\boldsymbol{p}, \boldsymbol{q})$) and set the derivative of f_{E_0} with respect to p_ℓ and q_ℓ to zero, we recover the equations (2.6) and (2.7) respectively.

To include the full topology of the graph and not restrain ourselves to the observed edges, we follow a label propagation approach by adding to f_{E_0} the corresponding test set function $f_{E\setminus E_0}$, and treat the sum of the two as the function to be minimized during training w.r.t. both $(\boldsymbol{p}, \boldsymbol{q})$ and all $y_{u,v} \in [-1, +1]$ for $(u, v) \in E \setminus E_0$:

$$\min_{(\boldsymbol{p},\boldsymbol{q}), y_{u,v}\in[-1,+1],\,(u,v)\in E\setminus E_0} \left(f_{E_0}(\boldsymbol{p}, \boldsymbol{q}) + f_{E\setminus E_0}(\boldsymbol{p}, \boldsymbol{q})\right) \tag{2.8}$$

Binary ± 1 predictions on the test set $E \setminus E_0$ are then obtained by thresholding the computed values $y_{u,v}$.

We now proceed to solve (2.8) via label propagation [ZGL03] on the graph G'' obtained through the second reduction of Section 2.1. Indeed, one can show[10] that this objective is equal —up to a regularization term— to the quadratic energy objective minimized by label propagation methods. However, because of the presence of negative edge weights in G'', we first have to symmetrize[11] variables

[10] as we do in Section 2.9.1.3.

[11] While we note here that such linear transformation of the variables does not change the problem, we provide more details in Section 2.9.1.3 of the supplementary material.

p_i, q_i and $y_{u,v}$ so as they all lie in the interval $[-1, +1]$. After this step, one can see that, once we get back to the original variables, label propagation computes the harmonic solution minimizing the function

$$\widehat{f}\big(\boldsymbol{p}, \boldsymbol{q}, y_{u,v\,(u,v)\in E\setminus E_0}\big) = f_{E_0}(\boldsymbol{p}, \boldsymbol{q}) + f_{E\setminus E_0}(\boldsymbol{p}, \boldsymbol{q}) + \frac{1}{2}\sum_{u\in V}\left(d_{\text{out}}(u)\Big(p_u - \frac{1}{2}\Big)^2 + d_{\text{in}}(u)\Big(q_i - \frac{1}{2}\Big)^2\right)$$

The function $\widehat{f}$ is thus a regularized version of the target function $f_{E_0} + f_{E\setminus E_0}$ in (2.8), where the regularization term tries to enforce the extra constraint that whenever a node u has a high out-degree then the corresponding p_u should be close to $1/2$. Thus, on any edge (u, v) departing from u, the Bayes optimal predictor $y^*(u, v) = \text{sign}(p_u + q_v - 1)$ will mainly depend on q_v being larger or smaller than $1/2$ (assuming v has small in-degree). Similarly, if u has a high in-degree, then the corresponding q_u should be close to $1/2$, implying that on any edge (v, u) arriving at u, the Bayes optimal predictor $y^*(v, u)$ will mainly depend on p_v (assuming v has small out-degree). Put differently, a node having a huge out-neighborhood makes each outgoing edge "count less" than a node having only a small number of outgoing edges, and similarly for in-neighborhoods.

The label propagation algorithm operating on G'' does so (see again Figure 2.2c) by iteratively updating as follows:

$$\begin{aligned}
p_u &\leftarrow \frac{-\sum_{v\in\mathcal{N}_{\text{out}}(u)} q_v + \sum_{v\in\mathcal{N}_{\text{out}}(u)}(1 + y_{u,v})}{3\, d_{\text{out}}(u)} \qquad \forall u \in V \\
q_v &\leftarrow \frac{-\sum_{u\in\mathcal{N}_{\text{in}}(v)} p_u + \sum_{u\in\mathcal{N}_{\text{in}}(v)}(1 + y_{u,v})}{3\, d_{\text{in}}(v)} \qquad \forall v \in V \\
y_{u,v} &\leftarrow \frac{p_u + q_v}{2} \qquad \forall (u, v) \in E \setminus E_0\ .
\end{aligned}$$

The algorithm is guaranteed to converge [ZGL03] to the minimizer of $\widehat{f}$. Notice that the presence of negative weights on the edges of G'' does not prevent label propagation from converging. In fact, any node classification algorithm handling both positive and negative weights on the edges of G'' could be used instead of label propagation. One alternative would thus be the WTA algorithm from [Ces+13]. However, our label propagation algorithm is the one we will be championing in our experiments of Section 2.5.

2.4 Related work

Interest in signed networks can be traced back to the psychological theory of structural balance [CH56; Hei58] and its weak version [Dav67], that we will describe with more details in Section 3.1.3 on page 48. The advent of online signed social networks has enabled a more thorough and quantitative understanding of that phenomenon. In this section, we provide an overview of methods tackling the same EDGE SIGN PREDICTION problem as us. Along the way, we give five of them a name in small capitals, for they are recent and effective. Therefore, we will compare our approaches with those methods in the experiments of the next section. At the end, we mention some closely related variants of the original EDGE SIGN PREDICTION problem.

Existing methods can be broadly divided into three strategies, which share some similarities:

1) embedding the nodes of the graph in a low dimensional space using spectral or neural techniques, before using node positions as features for a classifier;

2) completing the adjacency matrix through a global optimization algorithm; and
3) computing local features of the edges with several heuristics and train a classifier such as logistic regression or SVM.

In the first direction, the spectral embedding is illustrated by works from Kunegis *et al.* [KLB09] and Zheng *et al.* [ZS15]. We shall describe them more thoroughly in Section 3.2.4, but here we note that, following the natural orientation of the study of graph spectrum, they focus more on clustering than EDGE SIGN PREDICTION. Furthermore, the use of the adjacency matrix usually requires a quadratic running time in the number of nodes, which makes those methods hardly scalable to large graphs. More recently, there has been great interest in adapting word embedding techniques such as word2vec [Mik+13] to a "corpus" of random walks that are considered as documents, while nodes play the role of word. This allows the unsupervised extraction of node features that can then used to train downstream classifier, see [CO16; Wil17; CZC17] for three recent surveys. It can also be tailored to exploit the specificity of signed graphs. The goal is to find for every node u a vector $x_u \in \mathbb{R}^d$ in such a way that in this new space, nodes are close to their positive neighbors and far from their negative neighbors. For instance in SiNE [Wan+17d], for a node triplet u, v, w such that the edge (u, v) is positive and the edge (u, w) is negative, the objective is to find vectors maximizing $f(x_u, x_w) + \delta - f(x_u, x_v)$. The similarity function f is chosen to be a Siamese multilayers neural network, whose parameters are learned by back propagation. In SIGNet [IPR17], the similarity function is a sigmoid whose argument $x_u^T x_v$ is weighted by the sign of the corresponding edge, with an optimization technique closer to word2vec. Finally, SNE [YWX17] uses a log-bilinear model, and the training objective is to predict, given a path $\mathcal{P}$, which node should follow $\mathcal{P}$, weighting differently the vector nodes in $\mathcal{P}$ depending of whether they are the source of a positive or negative edge in $\mathcal{P}$.

Next we look at matrix completion approaches, which are global by nature. For instance, Chiang *et al.* [Chi+14] tackle the EDGE SIGN PREDICTION problem through this lens, restricting themselves to undirected graphs. They consider the observed adjacency matrix A, made of the edges in the training set E_0, as a noisy sampling of the adjacency matrix $A^\star$ of an underlying complete graph satisfying the weakly balance condition (that is with no cycle containing only one negative edge, see Definition 3.1.4 on page 49). This condition implies the existence of a small number k of node clusters with positive links within clusters and negative links across clusters, which in turn implies $\operatorname{rank} A^\star = k$. By recovering a complete matrix $\tilde{A}$ that matches the non-zeros entries of A, it is possible to predict the sign of $(u, v) \notin E_0$ as $\widehat{y}_{u,v} = \operatorname{sign}(\tilde{A}_{u,v})$. Although the exact version of this problem is NP-HARD, the authors assume that k is an hyperparameter known beforehand and look for two matrices $W, H \in \mathbb{R}^{k \times |V|}$ that minimise a sigmoidal reconstruction loss of A, subject to a nuclear norm regularization term. The minimization is carried out by Stochastic Gradient Descent and we refer to this method as LOWRANK. The approach of Wang *et al.* [Wan+17a] is similar, but they consider directed graphs, use the logistic loss and compute at each iteration a threshold optimizing the F_1-score on the observed signs. Furthermore, they argue that to better handles class imbalance, it is preferable to minimize the maxnorm of the recovered matrix, which is a tighter approximation to the rank function than the nuclear norm. We thus refer to this method as MAXNORM.

Finally, the last set of methods are based on the computation of local features of the graph. These features are evaluated on the subgraph induced by the training edges, and the resulting values are used to train a supervised classification algorithm. The most basic set of local features used to classify a given edge $u \to v$ are defined by $d^+_{\text{in}}(v), d^-_{\text{in}}(v), d^+_{\text{out}}(u), d^-_{\text{out}}(u)$ computed over the training set E_0, and by the embeddedness coefficient $\left|\mathcal{E}_{\text{out}}(u) \cap \mathcal{E}_{\text{in}}(v)\right|$, which is the number of common neighbors of u and v. In turn, these degree features can be used to define more

complicated features, such as a notion of similarity between two nodes based on how they rate and are rated by their neighbors [Yua+17]. Another way of looking at neighborhoods and degrees is to mine ego networks[12] with a SVM [Pap+14]. Bachi *et al.* [Bac+12b] also use an approach based on ego networks, in a data mining fashion. Namely, they extract frequent small subgraphs from the collection of all ego networks of G. Then, they construct rules, which are made of two frequent subgraphs differing by a single edge.

A sophisticated take on degree features is presented by Song *et al.* [SM15], who note that a node can belong to one of the 16 node-types based of whether the number of its positive (respectively negative) outgoing (respectively incoming) edges is zero or not. The number of unobserved incoming and outgoing edges of each node u let us define a 16-dimensional vector V_u containing the probability of transitioning to any other type once the unobserved signs are revealed. Then each edge $u \to v$ is associated with a feature vector consisting of the outer product of V_u with V_v and also including additional features such as triads count and degree information before training a Logistic Regression model. We refer to this method as BAYESIAN.

Other types of features are derived from social status theory, which posits that a positive link from u to v denotes that user u considers user v as having a higher status or skill than herself [LHK10]. This has implications on the distribution of the so called *triads* in the network. A triad is a triangle formed by $u \to v$ together with $u \to w$ (or $w \to u$) and $v \to w$ (or $w \to v$) for any $w \in \mathcal{N}_{\text{out}}(u) \cap \mathcal{N}_{\text{in}}(v)$. Taking signs and directions into account, there are 16 possible triads but according to the status theory some must be more represented than others. The 16 TRIADS method [LHK10] exploits this fact by counting for each edge in the training set how frequently it is involved in each of the 16 triad types. It also adds 7 degree features before training a Logistic Regression model.

A third group of features is based on node ranking scores. These scores (usually one or two per node) are computed using a variety of methods, including

- PageTrust [dKD08], which adapts the random walk of PageRank [Pag+99], by making walkers keep in memory a list of nodes they do not like. If they ever reach such a node, their walk stop. The final PageTrust is computed iteratively until convergence.
- Prestige [ZA10], which can be seen as a compounded degree feature, as it is defined by $P(u) = \frac{d^+_{\text{in}}(u) - d^-_{\text{in}}(u)}{d_{\text{in}}(u)}$.
- exponential ranking [TNV10], which is the fixed point of the equation $\pi = A^T \frac{\exp \frac{1}{\mu}\pi}{||\exp \frac{1}{\mu}\pi||_1}$. This follows from the discrete choice theory and by assuming that we observe the reputation with some noise that is double exponentially distributed with parameter μ.
- Bias and Deserve [MB11], which are defined in terms of each other in weighted graphs. The bias is the tendency of a node u to trust/mistrust others, that is the difference between its opinion of a neighbor v and what v truly deserves according to the network:

$$bias(u) = \frac{1}{d_{\text{out}}(u)} \sum_{v \in \mathcal{N}_{\text{out}}(u)} (w_{u,v} - deserve(v)).$$

  The deserve of a node v is the aggregated opinion of its in-neighbors, discounted by their bias:

$$deserve(v) = \frac{1}{d_{\text{in}}(v)} \sum_{u \in \mathcal{N}_{\text{in}}(v)} (w_{u,v}(1 - \max\{0, \text{sign}(w_{u,v}) \times bias(u)\})) .$$

[12] Recall that the ego network of a node u is the subgraph induced by the neighbors $\mathcal{N}(u)$ of u.

- Reputation and Optimism [SJ14], defined for a node u by $\frac{\sum_{v\in\mathcal{N}_{\text{in}}(u)} y_{v,u}\sigma(v)}{\sum_{v\in\mathcal{N}_{\text{in}}(u)}\sigma(v)}$ and $\frac{\sum_{v\in\mathcal{N}_{\text{out}}(u)} y_{u,v}\sigma(v)}{\sum_{v\in\mathcal{N}_{\text{out}}(u)}\sigma(v)}$, where $\sigma(v)$ is the ranking score assigned to node v. The former can be seen as a weighted version of Prestige, while the latter is its outgoing counterpart.
- TrollTrust [WAS16], which builds upon [SJ14] but defines the ranking $\sigma(u)$ as the trustworthiness of u. It follows from a recursive definition of trollness based on the opinion of one node's neighbors weighted by their own trollness, which allow to assign a $\sigma(u)$ to each node u through a set of non linear equations solved by an iterative method. These σ values are used to compute Reputation and Optimism scores, thus providing four features for each edge, which are in turn used to train a Logistic Regression model for the classification task. We refer to this method as RANKNODES.

Other works have also considered versions of the problem where side information related to the network is available to the learning system. For instance, [PKV15] uses the product purchased on EPINION in conjunction with a neural network, [CDL15] identifies trolls by analysing the textual content of their post, [Wan+17c] improves the embedding approach of Wang *et al.* [Wan+17d] by considering the words written in reviews as attributes of the users, and [Ye+13] uses SVM to perform transfer learning from one network to another. [Tan+13] uses a matrix completion approach, approximating A by PCP^T, where the row P_u of the matrix $P \in \mathbb{R}^{n\times d}$ is the low rank representation of the node u and $C \in \mathbb{R}^{d\times d}$ is the correlation matrix between these representations. They also assume they are given a symmetric matrix Z of the homophily coefficients $\zeta_{u,v}$ between u and v and add the following regularization term to be minimized: $\sum_{u<v}\zeta_{u,v}||P_u - P_v||_2^2 = Tr(U^T L_Z U)$, where L_Z is the Laplacian of Z. While many of these approaches have interesting performances, they often require extra information which is not always available (or reliable) and, in addition, may face severe scaling issues.

Whereas our focus is on *binary* prediction, researchers have also considered a weighted version of the problem, where edges measure the amount of trust or distrust between two users. Note that typically, the embedding methods we discussed at the beginning of this section are able to handle weighted networks. One of the early and influential work on modeling how distrust propagate among users is [Guh+04]. They propose to represent atomic conclusions (such as if u trusts v and v trusts w then u is likely to trusts w) as matrix operators and define four of them that are assembled by a weighted linear combination. Starting from the observed adjacency matrix A, they repeatedly apply this operator (with potentially a discount factor) to obtain a final weigh matrix F (that can also be rounded to provide sign prediction). Qian *et al.* [QA14] extend the binary case with categorical relationships (such as strongly positive or weakly negative) and describe how every unbalanced triads experience some stress depending on the strength of its contradictory relationships. Arguing that the network should converge to a balanced state with as little change as possible, they express finding this minimal transformation as a Multidimensional Scaling problem, and the resulting graph to characterize unlabeled edges. Finally, Kumar *et al.* [Kum+16] explicitly consider the weight prediction problem in signed graphs. They use a procedure similar to the ranking methods discussed above. Namely, they defined two scores for each node that are computed iteratively from a uniform initialization. Assuming that weights lie in $[-1, 1]$, the fairness $f(u)$ of u is a measure of how fair or reliable u is when rating others nodes, while the goodness $g(v)$ of v measure how trustworthy is v when evaluated with complete fairness. Formally, $f(u) = 1 - \frac{1}{2\mathcal{N}_{\text{out}}(u)}\sum_{v\in\mathcal{N}_{\text{out}}(u)}\frac{|w_{u,v}-g(v)|}{2}$ and $g(v) = \frac{1}{\mathcal{N}_{\text{in}}(v)}\sum_{u\in\mathcal{N}_{\text{in}}(v)} f(u)w_{u,v}$. Note that these definitions are very close to bias and deserve from Mishra *et al.* [MB11], but the absolute value in the factor 2

in the expression of fairness allows for better convergence property. Finally, the weight of a test edge $u \to v$ is predicted as $f(u) \times g(v)$.

While we presented methods that operate in the batch setting, and will present an online algorithm as well in Section 2.6, other works have addressed the EDGE SIGN PREDICTION problem from an active learning point of view [Ces+12a; Ces+12b]. Recall that in the active setting, we are given a budget of edge labels to observe, and are free to select them the way we want within $E(Y)$. Again, the goal is to make as few mistakes as possible when predicting the sign of the remaining edges. These two methods build a spanning subgraph of G and query all its edge, but they differ in its the construction. Cesa-Bianchi *et al.* [Ces+12a] partition G into stars and connect them in a tree, whereas Cesa-Bianchi *et al.* [Ces+12b] cover the graph with cycles, each containing one test edge and being queried for the other, while respecting the user specified query budget. This relies on a different bias than our generative model, and this will be the subject of our next chapter. One might also consider an online version of the problem where the topology is not known in advance but discovered as prediction are made [GHP13]. This naturally increases and the difficulty of the problem, as reflected by the computational cost of the solution proposed, which is quadratic in $|V|$ *at each prediction*.

The survey [Tan+16b] contains pointers to many papers on signed networks, in particular for the EDGE SIGN PREDICTION problem.

2.5 Experimental Analysis

We now evaluate our EDGE SIGN PREDICTION methods on representative real-world datasets of varying density and label regularity. After presenting the data and our evaluation criterion, we proceed in two steps. First, we simulate our generative model on real networks to give them signs, then we study to which extent we can recover the parameters p and q of each node, and how predictions based on these estimation compare with the Bayes optimal. Second, we select random training sets from the actual signs. This shows that our methods compete well against existing approaches in terms of both predictive and computational performance. We are especially interested in small training set regimes, and have restricted our comparison to the batch learning scenario since all competing methods we are aware of have been developed in that setting only.

2.5.1 Datasets

We consider six real-world classification datasets. The first four are Directed Signed Social Networks widely used as benchmarks for this task [LHK10; SJ14; WAS16; Wan+17a]. In ADVOGATO, a trust-based social network for open source developers, a user u can certify another user v with different degrees of trust: "Observer", "Apprentice" (both of which we consider negative), "Journeyer" and "Master" (both of which we consider positive).[13] A full description of this trust metric, and its resistance to attacks, is available in the PhD thesis of the website's creator [Lev02, Section 4]). In WIKIPEDIA, there is an edge from user u to user v if v applies for an admin position and u votes for or against that promotion. In SLASHDOT, a news sharing and commenting website, member u can tag other members v as friends or foes. Finally, in EPINION, an online shopping website, user v reviews products and, based on these reviews, another user u can display whether he considers v to be reliable or not. In addition to these four datasets, we considered two other signed social networks where the signs are inferred automatically, rather than given explicitly by the users. In WIK. EDITS [MAC11], an edge from Wikipedia user u to user v indicates whether they edited the same article in a constructive manner or

[13] We download the 7th of July, 2014 version from http://www.trustlet.org/datasets/advogato/.

Table 2.2 – Dataset properties. The 5th column gives the fraction of positive labels. The next two columns provide two different measures of label regularity, while the last two columns give the proportion of reciprocal edges, and among them the fraction with different signs.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\frac{\lvert E\rvert}{\lvert V\rvert}$	$\frac{\lvert E^+\rvert}{\lvert E\rvert}$	$\frac{\Psi^2_{G''}(Y)}{\lvert E\rvert}$	$\frac{\Psi_G(Y)}{\lvert E\rvert}$	reciprocal edges	reciprocal disagreement
CITATIONS	4 831	39 452	8.2	72.3%	.076	.191	5.1%	27.1%
ADVOGATO	5 417	51 312	9.5	75.1%	.061	.132	33.6%	28.6%
WIKIPEDIA	7 114	103 108	14.5	78.8%	.063	.142	5.6%	10.0%
SLASHDOT	82 140	549 202	6.7	77.4%	.059	.143	17.7%	4.0%
EPINION	131 580	840 799	6.4	85.3%	.031	.074	30.8%	2.1%
WIK. EDITS	138 587	740 106	5.3	87.9%	.034	.086	6.5%	14.6%

not.[14] Finally, in the CITATIONS [Kum16] network, an author u cites another author v by either endorsing or criticizing v's work. The edge sign is derived by classifying the citation sentiment with a technique using a list of positive and negative words; see [Kum16] for more details.[15]

Table 2.2 summarizes statistics for these datasets. We note that most edge labels are positive. Hence, test set accuracy is not an appropriate measure of prediction performance. We instead evaluated our performance using the so-called Matthews Correlation Coefficient (MCC) [Bal+00], defined as

$$\text{MCC} = \frac{tp \times tn - fp \times fn}{\sqrt{(tp + fp)(tp + fn)(tn + fp)(tn + fn)}} \tag{2.9}$$

MCC combines all the four quantities found in a binary confusion matrix ($t$rue $p$ositive, $t$rue $n$egative, $f$alse $p$ositive and $f$alse $n$egative) into a single metric which ranges from -1 (when all predictions are incorrect) to $+1$ (when all predictions are correct) through 0 (when predictions are made uniformly at random).

Although the semantics of the edge signs is not the same across these networks, we can see from Table 2.2 that our generative model essentially fits all of them. Specifically, two columns of the table report the rate of label (ir)regularity, as measured by $\Psi^2_{G''}(Y)/|E|$ (6th column) and $\Psi_G(Y)/|E|$ (7th column), where

$$\Psi^2_{G''}(Y) = \min_{(\boldsymbol{p},\boldsymbol{q})} \sum_{(u,v)\in E} \left(\frac{1 + y_{u,v}}{2} - \frac{p_u + q_v}{2} \right)^2$$

as first described in Section 2.3.2 and $\Psi_G(Y)$ is the label regularity measure adopted in the online setting, as defined in Section 2.1. It is reasonable to expect that higher label irregularity corresponds to lower prediction performance. This trend is in fact confirmed by our experimental findings: whereas EPINION tends to be easy, CITATIONS tends to be hard, and this holds for all algorithms we tested, even if they do not explicitly comply with our inductive bias principles. Moreover, $\Psi^2_{G''}(Y)/|E|$ tends to be proportional to $\Psi_G(Y)/|E|$ across datasets, hence confirming the anticipated connection between the two regularity measures.

Finally, there is a low fraction of reciprocal edges (i.e. both $u \to v \in E$ and $v \to u \in E$), which is a common mechanism of link formation in directed networks [GL04; Squ+13]). Moreover, in most cases, such reciprocal edges do not disagree, i.e. they have the same sign. In practice, we can use this fact to improve our prediction at no additional computational cost: when predicting $u \to v \in E_{\text{test}}$, if the reciprocal edge $v \to u$ is part of the training set, we set $\widehat{y}_{u,v} = y_{v,u}$. For clarity, when comparing our methods with existing approaches, we dot not use that heuristic. But afterwards, we show in Table 2.6 when it can be beneficial.

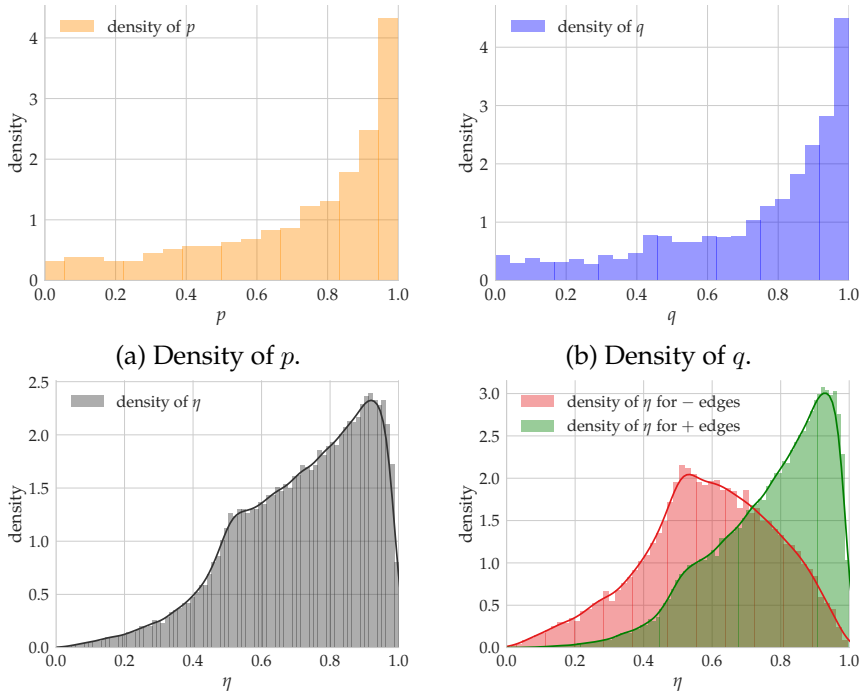
[14] This is the KONECT version of the "Wikisigned" dataset, from which we removed self-loops.
[15] We again removed self-loops and merged multi-edges which are all of the same sign.

2.5.2 Synthetic signs

Recall that according to our generative model of Section 2.2, each node u has a parameter p_u governing its sending behavior and each node v has a parameter q_v governing its receiving behavior, such that the sign of the edge $u \to v$ is positive with probability $\frac{p_u+q_v}{2} = \eta(u,v)$. Given a topology $G = (V, E)$, we start by assigning a p and q values to each node. If we want $\eta(u,v)$ to be uniform over E, we have to take into account the out-degree of nodes with at least one outgoing edge, and likewise the in-degree of nodes with at least one incoming edge. For that, in the case of p, we partition the interval $[0,1]$ into $|V|$ segments of size proportional to $d_{\text{out}}(u_1), d_{\text{out}}(u_2), \ldots, d_{\text{out}}(u_{|V|})$. We shuffle these segments and draw a number uniformly at random from each of them, which we set as the initial p' value of the corresponding node. Then, because we want to model real sign distribution and have more positive than negative edges, we apply an exponential transformation

$$p = \frac{1}{1-e^{-\lambda}}\left(1 - e^{-\lambda p'}\right),$$

where we choose $\lambda = 3$. We do the same for q, then for every edge $u \to v$, we set it positive with probability $\eta(u,v)$. An example of the distributions we obtain for the WIKIPEDIA graph is showed on Figure 2.5[16], giving 71.9% of positive edges.



(a) Density of p.

(b) Density of q.

(c) Density of η.

(d) Density of η for both signs.

Figure 2.5 – Synthetic distributions on WIKIPEDIA.

Once the signs are generated, we select a training set E_0 uniformly at random, and predict the signs of the remaining edges in the testing set $E \setminus E_0$. The Bayes optimal predictor classifies an edge $u \to v$ as positive if $\eta(u,v) \geq 1/2$ and negative otherwise. We compare it with the BLC(tr, un) algorithm analyzed in Section 2.3.1. Recall that BLC(tr, un) proceeds as follow: after computing $\widehat{tr}(u)$ and $\widehat{un}(u)$ on training set E_0 for all $u \in V$ (or setting those values to $1/2$ in case there is no outgoing or incoming edges for some node), we use the equation (2.3)[17], having estimated τ on E_0.

[16] More precisely, these figures are normalized histograms, meaning the counts are scaled down so that the bars total area sums up to one.

[17] We reproduce this equation here for convenience: $\text{sign}\left(\left(1-\widehat{tr}(u)\right) + \left(1-\widehat{un}(v)\right) - \tau - \frac{1}{2}\right)$.

Table 2.3 – Comparing $\text{BLC}(tr, un)$ with the Bayes optimal on three metrics: Accuracy, MCC and ability to estimate η. The results are averaged over 25 random sampling of E_0 for each dataset and each training size, and we report one standard deviation after every number.

Dataset	$\frac{\lvert E_0\rvert}{\lvert E\rvert}$	$\text{BLC}(tr, un)$ Accuracy	Bayes Accuracy	$\text{BLC}(tr, un)$ MCC	Bayes MCC	MAE$(\eta, \widehat{\eta})$ for the whole testing set	MAE$(\eta, \widehat{\eta})$ for highly sampled edges
CITATIONS	20%	66.1 ± 0.6	75.1 ± 0.1	16.2 ± 0.5	29.7 ± 0.2	$.256 \pm .00$	$.119 \pm .01$
	40%	68.2 ± 0.4	74.7 ± 0.2	19.1 ± 0.6	29.6 ± 0.4	$.205 \pm .00$	$.085 \pm .00$
	80%	71.4 ± 0.4	75.1 ± 0.5	23.4 ± 0.9	29.7 ± 1.1	$.152 \pm .00$	$.059 \pm .00$
ADVOGATO	20%	66.5 ± 0.6	75.1 ± 0.1	17.1 ± 0.6	30.3 ± 0.2	$.242 \pm .00$	$.116 \pm .00$
	40%	69.6 ± 0.4	75.4 ± 0.1	21.0 ± 0.5	30.8 ± 0.3	$.192 \pm .00$	$.083 \pm .00$
	80%	71.6 ± 0.5	75.4 ± 0.4	23.5 ± 0.9	30.5 ± 1.0	$.142 \pm .00$	$.060 \pm .00$
WIKIPEDIA	20%	70.5 ± 0.4	75.3 ± 0.1	21.8 ± 0.4	30.4 ± 0.2	$.169 \pm .00$	$.108 \pm .00$
	40%	72.3 ± 0.2	75.3 ± 0.1	25.3 ± 0.5	30.7 ± 0.3	$.122 \pm .00$	$.075 \pm .00$
	80%	73.3 ± 0.3	75.3 ± 0.3	28.0 ± 0.6	30.9 ± 0.6	$.090 \pm .00$	$.054 \pm .00$
SLASHDOT	20%	66.9 ± 0.3	75.2 ± 0.0	18.3 ± 0.3	30.2 ± 0.1	$.224 \pm .00$	$.098 \pm .00$
	40%	69.2 ± 0.2	75.2 ± 0.0	20.6 ± 0.2	30.1 ± 0.1	$.182 \pm .00$	$.068 \pm .00$
	80%	70.9 ± 0.2	75.1 ± 0.1	22.8 ± 0.3	30.1 ± 0.2	$.144 \pm .00$	$.048 \pm .00$
EPINION	20%	67.9 ± 0.3	75.2 ± 0.0	19.5 ± 0.2	30.4 ± 0.1	$.194 \pm .00$	$.089 \pm .00$
	40%	69.7 ± 0.2	75.1 ± 0.0	21.2 ± 0.3	30.0 ± 0.1	$.159 \pm .00$	$.063 \pm .00$
	80%	71.2 ± 0.1	75.2 ± 0.1	23.5 ± 0.2	30.3 ± 0.2	$.128 \pm .00$	$.044 \pm .00$
WIK. EDITS	20%	67.8 ± 0.3	75.3 ± 0.0	19.2 ± 0.2	30.5 ± 0.1	$.221 \pm .00$	$.097 \pm .00$
	40%	69.6 ± 0.2	75.2 ± 0.0	21.4 ± 0.2	30.3 ± 0.1	$.180 \pm .00$	$.067 \pm .00$
	80%	71.1 ± 0.2	75.3 ± 0.1	23.7 ± 0.2	30.4 ± 0.2	$.144 \pm .00$	$.047 \pm .00$

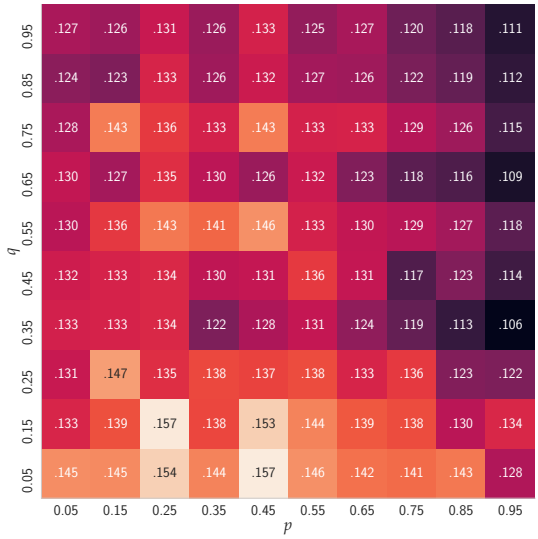
The result of this comparison on the six datasets is showed in Table 2.3. For each network, we generated the signs once. Then, for different training size (20%, 40% or 80% of E), we sampled a training set and predict using the knowledge of the true η (Bayes predictor) or its estimated value ($\text{BLC}(tr, un)$). The MCC of the Bayes predictor is the same on all datasets (around 30[18]), and so is its accuracy (around 75%). The corresponding values for $\text{BLC}(tr, un)$ are close on all datasets (although some are easier) and the gap naturally decreases as the training size increases. Another interesting quantity is how close can $\text{BLC}(tr, un)$ estimates η using $\widehat{\eta} = \big(1 - \widehat{tr}(u)\big) + (1 - \widehat{un}(v)) - \widehat{\tau}$ as defined in Section 2.3.1. We compute the mean absolute error (MAE) between η and $\widehat{\eta}$ for all testing edges, and then specifically for the testing edges whose both endpoints have been sampled above a certain threshold in the training set. As expected, the estimation is more accurate with increasing training set size, and with increasing number of samples for a given edge.

Finally, we can also look in more details at the role of p and q by building a 2D histogram of the testing edges $u \to v$ based on their coordinate (p_u, q_v). For instance on Figure 2.6a, we see that the MAE between η and $\widehat{\eta}$ is not uniform over the p, q-space. It seems to decrease along the diagonal from $(0, 0)$ to $(1, 1)$. Note though that Figure 2.6b suggests this is likely due to the higher number of edges in the top right corner (reflecting the imbalance toward positive edges), which allows better estimation. In this case, we cannot use the MCC because outside the diagonal, the Bayes predictor classifies edges either all positive or all negative, resulting in a division by zero in the definition of MCC. We thus fall back on accuracy and display in Figure 2.6c the difference between the accuracy of of $\text{BLC}(tr, un)$ and the accuracy of the Bayes predictor. This time, the anti diagonal where $\frac{p+q}{2} = \frac{1}{2}$ seems to play a special role. Indeed, as showed more clearly in Figure 2.6d, the gap in accuracy between $\text{BLC}(tr, un)$ and the Bayes predictor increases symmetrically as $\frac{p+q}{2}$ moves away from $1/2$. The gap also tends to disappear when $\frac{p+q}{2} = \frac{1}{2}$, for in that region,

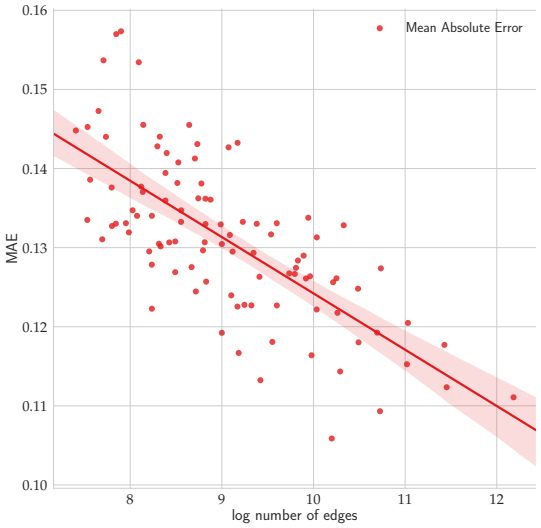
[18] Here and in the following, we multiply all MCC value by 100 to improve readability.

both predictors can only rely on random predictions.

While we show those patterns on WIKIPEDIA with a 40% training size, they are consistent across datasets and training size.

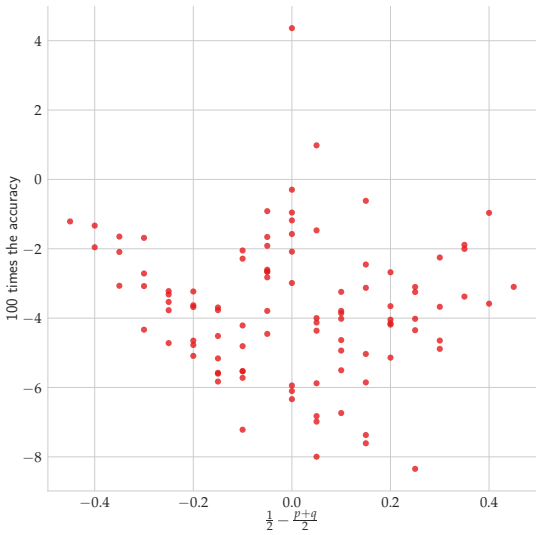


(a) Mean absolute error (MAE) of the $\eta(u, v)$ estimation.



(b) MAE in each cell as a function of the number of sampled edges.

(c) 100 time the difference of accuracy between BLC(tr, un) and Bayes.



(d) Difference of accuracy between BLC(tr, un) and Bayes as $\frac{p+q}{2}$ moves away from $\frac{1}{2}$.

Figure 2.6 – Results on WIKIPEDIA with BLC(tr, un) on a training set of size 40%

2.5.3 Real signs

We compared the following algorithms:

1. The label propagation algorithm of Section 2.3.2 (referred to as L. PROP.). The actual binarizing threshold was set by cross-validation on the training set.

2. The BLC(tr, un) algorithm described in Section 2.3.1.

3. A logistic regression model where each edge (u, v) is associated with the features $[1 - \widehat{tr}(u), 1 - \widehat{un}(v)]$ computed again on E_0 (we call this method LOGREG). The best binary thresholding is again computed on E_0. Experimenting with this logistic model serves to support the claim we made in the introduction that our generative model in Section 2.2 is a good fit for the data.

4. The solution obtained by directly solving the unregularized problem (2.8) through a constrained minimization algorithm (referred to as UNREG.). Again, the actual binarizing threshold was set by cross-validation on the training set.[19]

5. The matrix completion method from [Chi+14] based on LOWRANK matrix factorization. Since the authors showed their method to be robust to the choice of the rank parameter k, we picked $k = 7$ in our experiments.

6. The other MAXNORM matrix completion method from [Wan+17a], setting the parameter λ to 1.2 as advised in their paper.

7. A logistic regression model built on 16 TRIADS features derived from status theory [LHK10].

8. The TrollTrust algorithm from [WAS16], naming it RANKNODES. As for hyperparameter tuning (β and λ_1 in [WAS16]), we closely followed the authors' suggestion of doing cross validation.

9. The last competitor is the logistic regression model whose features have been build according to [SM15]. We call this method BAYESIAN.

The above methods can be roughly divided into *local* and *global* methods. A local method hinges on building local predictive features, based on neighborhoods: BLC(tr, un), LOGREG, 16 TRIADS, and BAYESIAN essentially fall into this category. The remaining methods (L. PROP., LOWRANK, MAXNORM and RANKNODES) are global in that their features are designed to depend on global properties of the graph topology.

Our main results are summarized in Table 2.4, reporting MCC test set performance after training on sets of varying size (from 5% to 25%, plus 50% and 90%). Results have been averaged over 12 repetitions. Because scalability is a major concern when training on sizeable datasets, we also give an idea of relative training times by reporting separately in Table 2.5 the time (in milliseconds) it took to train a single run of each algorithm on a training set of size[20] 15% of $|E|$, and then predict on the test set. Some conclusions emerge from those experiments:

1. Global methods tend to outperform local methods in terms of prediction performance, but are also significantly (or even much) slower (running times can differ by as much as three orders of magnitude). This is not surprising, and is in line with previous experimental findings (e.g., [SJ14; WAS16]). BAYESIAN looks like an exception to this rule, but its running time is indeed in the same ballpark as global methods.

2. L. PROP. almost always ranks first or at least second in this comparison when MCC is considered, at least in the small training set regime (that is when having access to at most 25% of the labels, which might be more realistic in many real world situations). On top of it, L. PROP. is fastest among the global methods (one or even two orders of magnitude faster), thereby showing the benefit of our approach to EDGE SIGN PREDICTION.

3. Two methods are competitive with us, BAYESIAN and RANKNODES. BAYESIAN achieves its best results on the larger dataset (SLASHDOT, EPINION and WIK. EDITS) when the training set size is large. Indeed, it learns a much more

[19] We have also tried to minimize (2.8) by removing the $[-1, +1]$ constraints, but got similar MCC results as the ones we report for UNREG.

[20] Comparison of training time performances is fair since all algorithms have been carefully implemented using the same stack of Python libraries, and run on the same machine (16 Xeon cores and 192GB Ram).

complex model than our methods, with 256 parameters[21], which also requires a lot of time to be trained. RANKNODES also shines in those large datasets, even at small training size. Note however that the difference with L. PROP. is usually less than one point, at the expense of a five times longer learning time.

4. The regularized solution computed by L. PROP. is always better than the unregularized one computed by UNREG. in terms of both MCC and running time.

5. As claimed in the introduction, our Bayes approximator BLC(tr, un) closely mirrors in performance the more involved LOGREG model. In fact, supporting our generative model of Section 2.2, the logistic regression weights for features $1 - \widehat{tr}(i)$ and $1 - \widehat{un}(j)$ are almost equal (see Table 2.11 in the supplementary material), thereby suggesting that predictor (2.3), derived from the theoretical results in Section 2.3.1, is *also* the best logistic model based on trollness and untrustworthiness.

Table 2.4 – MCC with increasing training set size, with one standard deviation over 12 random sampling of E_0. The last five columns refer to the methods we took from the literature. For the sake of readability, we multiplied all MCC values by 100. The best number in each row is highlighted in **bold brown** and the second one in *italic red*. If the difference is statistically significant (p-value of a paired Student's t-test less than 0.005), the best score is underlined.

	$\frac{\lvert E_0\rvert}{\lvert E\rvert}$	L. PROP.	BLC(tr, un)	LOGREG	UNREG.	LOWRANK	MAXNORM	16 TRIADS	RANKNODES	BAYESIAN
CITATIONS	5%	**24.2** ± 0.9	*19.8* ± 0.5	19.8 ± 0.5	15.9 ± 0.5	12.4 ± 0.7	1.2 ± 1.4	11.4 ± 1.1	17.5 ± 1.0	15.2 ± 1.4
	10%	**31.7** ± 0.8	*28.0* ± 0.6	27.9 ± 0.7	26.0 ± 0.7	17.9 ± 0.7	12.6 ± 0.8	17.2 ± 1.0	25.1 ± 0.9	25.5 ± 0.9
	15%	**36.1** ± 0.7	33.1 ± 0.8	*33.2* ± 0.7	31.6 ± 0.7	22.0 ± 0.6	22.2 ± 0.9	21.0 ± 1.0	31.2 ± 1.0	32.0 ± 0.8
	20%	**38.9** ± 0.8	*37.1* ± 0.6	36.9 ± 0.6	35.6 ± 0.5	25.7 ± 0.9	30.3 ± 0.8	24.3 ± 0.7	35.2 ± 0.7	36.7 ± 0.5
	25%	**41.1** ± 0.7	39.7 ± 0.8	39.7 ± 0.7	38.4 ± 0.9	29.0 ± 0.6	36.5 ± 0.9	27.0 ± 0.6	37.8 ± 0.9	*39.8* ± 1.0
	50%	47.2 ± 0.6	48.1 ± 0.6	47.7 ± 0.5	46.7 ± 0.7	44.9 ± 1.4	**55.9** ± 0.9	35.4 ± 1.0	46.0 ± 0.6	*50.9* ± 0.8
	90%	51.6 ± 2.1	52.6 ± 2.2	52.5 ± 1.7	52.3 ± 1.9	60.4 ± 1.6	**66.7** ± 1.0	47.2 ± 1.9	51.5 ± 2.1	*61.7* ± 1.3
ADVOGATO	5%	**40.9** ± 0.7	36.5 ± 0.7	*36.8* ± 0.8	30.2 ± 0.8	25.1 ± 1.0	4.7 ± 3.3	29.5 ± 3.7	32.2 ± 1.0	19.8 ± 0.8
	10%	**46.8** ± 0.6	44.7 ± 0.6	*45.3* ± 0.7	42.8 ± 0.9	28.3 ± 0.7	24.2 ± 1.6	37.6 ± 1.4	38.8 ± 1.0	30.7 ± 0.9
	15%	*50.4* ± 0.5	49.5 ± 0.6	**50.4** ± 0.6	47.9 ± 0.6	30.3 ± 0.8	30.1 ± 1.1	42.0 ± 0.8	45.1 ± 1.6	38.5 ± 0.8
	20%	*52.2* ± 0.6	51.9 ± 0.5	**53.0** ± 0.5	51.1 ± 0.7	31.5 ± 0.7	33.5 ± 1.3	44.4 ± 0.6	48.9 ± 0.9	43.4 ± 0.6
	25%	*54.2* ± 0.7	54.1 ± 0.5	**55.3** ± 0.5	53.7 ± 0.6	32.7 ± 0.7	35.6 ± 1.1	46.7 ± 0.9	51.7 ± 1.1	47.6 ± 0.7
	50%	59.0 ± 0.5	59.8 ± 0.5	**60.7** ± 0.4	*60.3* ± 0.4	37.1 ± 0.8	42.8 ± 1.2	52.2 ± 0.9	58.2 ± 0.6	56.5 ± 0.5
	90%	63.3 ± 1.3	64.0 ± 1.2	*64.5* ± 1.2	**65.0** ± 1.1	44.0 ± 2.0	49.5 ± 1.9	57.9 ± 1.3	63.3 ± 1.1	63.0 ± 1.2
WIKIPEDIA	5%	**39.8** ± 0.7	38.2 ± 1.0	*39.0* ± 0.7	36.0 ± 0.7	24.7 ± 0.8	24.6 ± 0.9	9.7 ± 1.0	33.4 ± 0.6	26.2 ± 1.4
	10%	*46.7* ± 0.6	45.9 ± 0.5	**46.8** ± 0.5	44.4 ± 0.6	31.5 ± 0.7	34.9 ± 0.4	26.8 ± 1.5	43.2 ± 0.6	40.3 ± 0.6
	15%	*50.3* ± 0.5	49.6 ± 0.4	**50.5** ± 0.4	48.8 ± 0.4	35.4 ± 0.7	39.4 ± 0.6	34.2 ± 0.6	48.4 ± 0.5	46.7 ± 0.4
	20%	*52.5* ± 0.4	51.9 ± 0.5	**52.8** ± 0.5	51.7 ± 0.5	38.0 ± 0.6	42.2 ± 0.5	38.2 ± 0.7	51.2 ± 0.5	50.2 ± 0.4
	25%	*54.2* ± 0.6	53.6 ± 0.6	**54.6** ± 0.5	53.6 ± 0.4	40.2 ± 0.6	44.2 ± 0.6	41.2 ± 0.6	53.5 ± 0.5	53.0 ± 0.6
	50%	*57.8* ± 0.3	56.9 ± 0.5	**57.9** ± 0.4	57.6 ± 0.4	47.3 ± 0.7	48.8 ± 0.7	48.9 ± 0.5	57.8 ± 0.4	57.4 ± 0.5
	90%	**60.5** ± 0.8	59.5 ± 0.9	60.4 ± 0.9	59.9 ± 1.1	53.5 ± 0.9	51.3 ± 0.8	55.6 ± 0.8	*60.4* ± 1.0	59.6 ± 1.2
SLASHDOT	5%	*41.0* ± 0.2	36.3 ± 0.5	37.1 ± 0.3	33.6 ± 0.2	36.9 ± 0.4	14.9 ± 0.8	27.8 ± 1.0	**45.3** ± 0.4	29.4 ± 0.3
	10%	*46.6* ± 0.2	42.2 ± 0.2	43.1 ± 0.2	40.9 ± 0.3	39.7 ± 0.3	25.7 ± 0.7	40.3 ± 1.1	**47.8** ± 0.5	38.2 ± 0.2
	15%	**49.7** ± 0.2	45.4 ± 0.3	46.5 ± 0.2	45.4 ± 0.2	41.3 ± 0.3	32.5 ± 0.3	45.5 ± 1.3	*48.7* ± 0.7	43.4 ± 0.2
	20%	*51.8* ± 0.2	47.6 ± 0.3	48.7 ± 0.1	48.3 ± 0.2	42.9 ± 0.2	36.6 ± 0.3	49.0 ± 0.7	**52.0** ± 0.3	47.0 ± 0.3
	25%	*53.3* ± 0.2	49.4 ± 0.2	50.2 ± 0.1	50.0 ± 0.1	44.3 ± 0.6	39.7 ± 0.3	50.6 ± 0.4	**53.3** ± 0.3	49.4 ± 0.2
	50%	**57.2** ± 0.1	54.2 ± 0.2	55.0 ± 0.1	54.6 ± 0.1	52.9 ± 0.5	48.2 ± 0.1	55.2 ± 0.4	*56.8* ± 0.2	56.3 ± 0.1
	90%	*59.6* ± 0.5	57.4 ± 0.5	58.0 ± 0.5	57.7 ± 0.4	59.0 ± 0.4	53.5 ± 0.4	58.0 ± 0.4	59.2 ± 0.6	**61.1** ± 0.5
EPINION	5%	*54.6* ± 0.3	46.9 ± 0.6	48.9 ± 0.3	42.8 ± 0.4	39.9 ± 0.3	28.9 ± 0.3	41.4 ± 2.0	**56.0** ± 0.6	37.9 ± 0.4
	10%	*59.0* ± 0.4	54.1 ± 0.4	55.9 ± 0.2	53.3 ± 0.4	44.4 ± 0.3	36.7 ± 0.4	50.8 ± 1.5	**60.5** ± 0.3	49.9 ± 0.4
	15%	*61.5* ± 0.3	57.9 ± 0.3	59.3 ± 0.1	58.7 ± 0.5	48.6 ± 0.7	41.0 ± 0.5	54.5 ± 1.8	**62.7** ± 0.2	56.5 ± 0.7
	20%	*63.0* ± 0.3	60.3 ± 0.3	61.4 ± 0.1	61.8 ± 0.2	52.0 ± 0.9	43.9 ± 0.4	56.3 ± 1.2	**64.1** ± 0.3	61.4 ± 0.4
	25%	64.2 ± 0.2	61.8 ± 0.3	62.9 ± 0.1	63.5 ± 0.1	55.0 ± 0.7	46.3 ± 0.4	58.6 ± 1.4	**65.2** ± 0.6	*64.7* ± 0.3
	50%	67.3 ± 0.2	66.3 ± 0.2	66.7 ± 0.1	67.2 ± 0.2	63.4 ± 0.3	52.8 ± 0.2	64.2 ± 1.4	*69.9* ± 0.2	**72.5** ± 0.1
	90%	69.7 ± 0.3	69.2 ± 0.3	69.1 ± 0.3	69.7 ± 0.3	68.0 ± 0.7	57.6 ± 0.6	67.6 ± 0.7	*71.6* ± 0.3	**77.4** ± 0.4
WIK. EDITS	5%	**36.5** ± 0.4	*31.0* ± 0.2	30.8 ± 0.2	21.9 ± 0.3	23.2 ± 0.5	15.8 ± 0.5	3.2 ± 0.7	26.8 ± 0.4	26.8 ± 0.7
	10%	**38.9** ± 0.5	35.6 ± 0.2	*35.8* ± 0.1	29.7 ± 0.3	27.3 ± 0.3	23.9 ± 0.5	11.7 ± 0.9	33.8 ± 0.5	35.1 ± 0.3
	15%	*38.8* ± 0.8	37.6 ± 0.2	38.1 ± 0.2	33.5 ± 0.2	30.0 ± 0.4	28.5 ± 0.2	18.2 ± 0.6	36.7 ± 0.4	**40.0** ± 0.2
	20%	39.0 ± 0.7	38.8 ± 0.1	*39.6* ± 0.2	35.0 ± 0.2	32.3 ± 0.7	31.5 ± 0.4	21.5 ± 0.4	38.5 ± 0.3	**43.3** ± 0.2

[21] corresponding to 16 × 16 feature per edge

$\frac{\lvert E_0\rvert}{\lvert E\rvert}$	L. Prop.	BLC(tr, un)	LogReg	Unreg.	LowRank	MaxNorm	16 Triads	RankNodes	Bayesian
25%	38.8 ± 0.6	39.6 ± 0.4	*40.5* ± 0.1	36.2 ± 0.1	34.3 ± 0.8	34.2 ± 0.4	23.2 ± 0.7	39.8 ± 0.3	**45.8** ± 0.3
50%	39.5 ± 0.5	42.8 ± 0.2	42.6 ± 0.2	38.9 ± 0.2	41.0 ± 0.7	41.0 ± 0.3	27.4 ± 0.5	*42.9* ± 0.3	**53.2** ± 0.2
90%	41.7 ± 0.5	44.5 ± 0.6	43.9 ± 0.4	41.0 ± 0.5	45.3 ± 0.4	45.8 ± 0.3	31.6 ± 0.5	*46.0* ± 0.5	**58.5** ± 0.4

Table 2.5 – The time taken (in milliseconds) to train, for each dataset, on a 15% training set and predict the remaining 85%. The experimental setting is the same as in Table 2.4.

Dataset	L. Prop.	BLC(tr, un)	LogReg	Unreg.	LowRank	MaxNorm	16 Triads	RankNodes	Bayesian
Citations	19,2	0,6	3,6	2 827	3 222	23 229	7,5	157	4 787
Advogato	21,4	0,8	5,7	5 538	4 094	37 440	10,9	224	6 597
Wikipedia	42,2	1,7	8,0	10 613	8 277	59 992	17,7	255	12 583
Slashdot	645	8,2	42,9	78 658	67 454	350 742	139	2 446	68 008
Epinion	1 447	10,6	72,5	142 507	127 641	692 560	234	3 381	111 435
Wik. Edits	897	9,6	60,2	208 676	125 884	632 663	188	4 233	95 374

2.5.4 Additional experiments

We perform two other sets of experiments. The first one evaluates the effect of predicting reciprocal edges by their value if available. As expected, the results in Table 2.6 demonstrate that it improves MCC when the network has enough reciprocal edges with low disagreements, like in Slashdot and Epinion. It has no effect when there are few reciprocal edges, like in Citations, Wikipedia and Wik. Edits. And it is detrimental when there are many reciprocal edges that do not agree enough, like in Advogato, in which case it is better to rely solely on the learned model.

Table 2.6 – 100 time the difference of MCC between using the twin heuristic and not using it for our two main methods. Those values are computed on the same training/testing split as of Table 2.4.

	5%	10%	15%	20%	25%	50%	90%
				Citations			
L. Prop.*	0.0	0.0	0.0	0.0	0.0	−0.1	−0.7
BLC$^\star(tr, un)$	0.0	0.0	0.0	0.0	0.1	−0.2	−0.7
				Advogato			
L. Prop.*	−0.2	−0.6	−1.0	−1.6	−2.2	−5.4	−10.8
BLC$^\star(tr, un)$	−0.1	−0.5	−1.0	−1.6	−2.2	−5.7	−11.1
				Wikipedia			
L. Prop.*	0.0	0.0	0.0	0.0	0.0	−0.1	−0.2
BLC$^\star(tr, un)$	0.0	0.0	0.0	−0.1	−0.1	−0.2	−0.2
				Slashdot			
L. Prop.*	0.3	0.5	0.8	0.9	1.1	2.0	3.3
BLC$^\star(tr, un)$	0.3	0.6	0.9	1.1	1.3	2.2	3.5
				Epinion			
L. Prop.*	0.3	0.5	0.7	0.8	1.0	1.9	3.0
BLC$^\star(tr, un)$	0.4	0.6	0.8	1.0	1.2	1.8	2.8
				Wik. Edits			
L. Prop.*	0.0	0.0	0.0	0.0	−0.1	−0.1	−0.3
BLC$^\star(tr, un)$	0.0	0.0	0.0	0.0	0.0	−0.2	−0.5

In the second set of experiments, we study the effect of our hypothesis that the labeled edges of the training set are chosen uniformly at random. In two of our datasets (Wikipedia and Epinion), edges come with the timestamp at which they were created. A more realistic way of choosing the training set is

therefore, for a given training size m, to let E_0 be the m oldest edges, and try to predict the remaining, newest ones. This is a common experimental setting in link prediction, where the goal is to infer which pair of nodes will be connected in the future [MBC16]. However, as showed in Table 2.7, this makes the problem much more challenging, both for our methods and our competitors. It is quite surprising that even when the training set is as large as 90% of all edges, the MCC are so low compared with those reported in Table 2.4.

Table 2.7 – Same as Table 2.4, but when selecting the training set by the timestamp of the edges.

	$\frac{\lvert E_0\rvert}{\lvert E\rvert}$	L. Prop.	BLC(tr, un)	LogReg	Unreg.	LowRank	MaxNorm	16 Triads	RankNodes	Bayesian
Wikipedia	5%	2.2 ± 0.0	4.0 ± 0.0	4.0 ± 0.0	-1.0 ± 1.0	3.0 ± 0.4	0.5 ± 0.4	1.5 ± 0.1	2.8 ± 0.2	2.0 ± 0.2
	10%	3.1 ± 0.0	5.6 ± 0.0	5.9 ± 0.0	0.4 ± 0.9	5.8 ± 0.5	0.8 ± 0.3	2.2 ± 0.3	6.3 ± 0.5	4.2 ± 0.2
	15%	5.2 ± 0.0	5.7 ± 0.0	6.5 ± 0.1	0.7 ± 0.7	6.2 ± 0.5	1.4 ± 0.4	3.4 ± 0.2	7.4 ± 0.5	3.4 ± 0.1
	20%	5.5 ± 0.0	6.9 ± 0.0	7.4 ± 0.1	0.7 ± 0.7	5.3 ± 0.6	1.6 ± 0.2	3.9 ± 0.2	10.1 ± 0.2	2.7 ± 0.1
	25%	5.8 ± 0.0	6.5 ± 0.0	7.2 ± 0.0	2.6 ± 0.9	7.2 ± 0.7	1.9 ± 0.3	6.6 ± 0.3	9.4 ± 0.1	3.9 ± 0.3
	50%	7.0 ± 0.0	7.3 ± 0.0	8.0 ± 0.0	4.7 ± 0.6	8.1 ± 1.1	3.3 ± 0.5	17.1 ± 1.7	12.4 ± 0.0	5.2 ± 0.4
	90%	20.9 ± 0.0	14.7 ± 0.0	13.3 ± 0.0	13.5 ± 0.9	10.5 ± 0.7	7.4 ± 1.7	20.2 ± 0.5	18.7 ± 0.1	13.2 ± 0.3
Epinion	5%	26.8 ± 0.0	23.2 ± 0.0	23.9 ± 0.0	20.9 ± 0.5	23.0 ± 0.3	13.9 ± 0.2	28.9 ± 0.2	31.9 ± 0.2	29.8 ± 0.8
	10%	26.3 ± 0.0	25.0 ± 0.0	26.0 ± 0.1	22.4 ± 0.5	28.4 ± 0.4	16.7 ± 0.3	29.3 ± 0.3	34.3 ± 0.0	33.7 ± 0.5
	15%	23.2 ± 0.0	23.0 ± 0.0	25.2 ± 0.0	21.0 ± 0.4	28.7 ± 0.4	17.1 ± 0.5	28.1 ± 0.3	32.6 ± 0.2	33.1 ± 0.3
	20%	18.4 ± 0.0	21.3 ± 0.0	24.2 ± 0.0	19.5 ± 0.3	26.2 ± 0.4	16.6 ± 0.3	26.8 ± 0.3	29.3 ± 0.0	32.7 ± 0.3
	25%	30.4 ± 0.0	25.1 ± 0.0	27.9 ± 0.0	20.5 ± 0.4	25.7 ± 0.3	17.0 ± 0.4	26.4 ± 0.3	35.6 ± 0.1	30.4 ± 0.3
	50%	21.2 ± 0.0	25.5 ± 0.0	29.1 ± 0.0	19.2 ± 0.5	19.9 ± 1.0	12.2 ± 0.8	31.3 ± 0.4	34.7 ± 0.0	22.3 ± 0.1
	90%	38.6 ± 0.0	32.1 ± 0.0	33.9 ± 0.0	25.2 ± 0.8	24.9 ± 0.8	8.2 ± 1.2	40.8 ± 0.5	42.5 ± 0.0	36.8 ± 0.3

The explanation of why these two sampling strategies produce so different results can be inferred from Table 2.8. Letting V_{out} be the subset of node of V with at least one outgoing edge, and defining similarly V_{in}, we see that the node coverage is naturally larger when sampling at random. Yet, when the training set reaches 90% size, both sampling strategies cover roughly 90% of the nodes in V_{out} and in V_{in}. The difference in MCC is thus explained by the last three columns. It shows, for testing edges $u \to v$, the breakdown between three situations, from the least to the most informative:

(i) no sampled edge outgoing from u nor incoming to v;
(ii) some sampled edges, but only either outgoing from u or incoming to v; and
(iii) some sampled edges, both outgoing from u and incoming to v;

When the training set is build uniformly at random, the vast majority of testing edges falls into the case (iii). This is no longer true with temporal training set, where situations (ii) and even (i) are more common, making prediction more difficult. The difference of repartition is particularly strong in Wikipedia, justifying the better performance in Epinion.

This conclusion is not in contradiction with our previous experiments, since our algorithms were developed under the assumption that both the revealed labels and the ones to predict were distributed uniformly within the graph. Recall for instance that Theorem 1 requires to have $\Omega(\log |V|)$ labels outgoing from u and incoming to v to guarantee an accurate prediction of the sign of $u \to v$. Therefore in the next section, we present an online algorithm, whose goal is to guarantee good performance when the labeling is regular enough, despite the adversarial order in which the predictions are asked.

2.6 Algorithms in the Online Setting

In the online scenario, recall we do not assume anymore that the signs are generated by the model of Section 2.2. Instead, they are chosen adversarially, and presented sequentially in order to force as many mistakes as possible. This is achieved by deviating from the regular labeling where every user is either perfectly troll or

Table 2.8 – The effect of not sampling edges at random on WIKIPEDIA and EPINION. In case of random sampling, all values are averaged over 20 trials. Refer to the main text for an interpretation of how this affects the predictive performance.

Sampling	Fraction of V_{out} sampled	Fraction of V_{in} sampled	Testing edges with no endpoint sampled	Testing edges with one endpoint sampled	Testing edges with both endpoints sampled
			WIKIPEDIA		
20% oldest	28.84%	27.76%	70.06%	27.18%	2.76%
20% random	53.23%	91.45%	0.10%	7.17%	92.73%
50% oldest	57.51%	52.46%	48.71%	45.22%	6.07%
50% random	76.48%	97.46%	0.04%	3.62%	96.34%
90% oldest	92.44%	90.09%	14.82%	68.37%	16.81%
90% random	95.98%	99.65%	0.03%	2.47%	97.50%
			EPINION		
20% oldest	7.81%	36.39%	20.94%	70.84%	8.22%
20% random	40.00%	43.87%	3.24%	19.00%	77.75%
50% oldest	33.71%	61.48%	15.95%	65.55%	18.50%
50% random	67.81%	71.40%	1.57%	12.82%	85.62%
90% oldest	90.09%	91.13%	13.34%	45.92%	40.75%
90% random	94.34%	95.16%	0.94%	9.72%	89.35%

trustworthy, as measured by the regularity measure $\Psi_G(Y)$ described at the end of Section 2.1. We start by presenting an algorithm that combines randomized Weighted Majority instances and makes little more than $\Psi_G(Y)$ mistakes on any edge-labeled graph $G(Y) = (V, E(Y))$. We then show there is not much room for improvement, for as long as $\Psi_G(Y)$ is within a budget of K irregularities, any online algorithm is condemned to err at least $K/2$ times.

Theorem 2. *There exists a randomized online prediction algorithm A whose expected number of mistakes satisfies $\mathbb{E}M_A(Y) = \Psi_G(Y) + O\left(\sqrt{|V|\Psi_G(Y)} + |V|\right)$ on any edge-labeled graph $G(Y) = (V, E(Y))$.*

Proof. Let each node $u \in V$ host two instances of the randomized Weighted Majority (RWM) algorithm [LW94] with an online tuning of their learning rate [Ces+97; ACG02]: one instance for predicting the sign of outgoing edges (u, v), and one instance for predicting the sign of incoming edges (v, u). Both instances simply compete against the two constant experts, predicting always $+1$ or always -1. Denote by $M(u, v)$ the indicator function (zero-one loss) of a mistake on edge (u, v). Then the expected number of mistakes of each RWM instance satisfy [Ces+97; ACG02]:

$$\sum_{v \in \mathcal{N}_{\text{out}}(u)} \mathbb{E}\, M(u, v) = \Psi_{\text{out}}(u, Y) + O\left(\sqrt{\Psi_{\text{out}}(u, Y)} + 1\right)$$

and

$$\sum_{u \in \mathcal{N}_{\text{in}}(v)} \mathbb{E}\, M(u, v) = \Psi_{\text{in}}(v, Y) + O\left(\sqrt{\Psi_{\text{in}}(v, Y)} + 1\right) .$$

We then define two meta-experts: an ingoing expert, which predicts $y_{u,v}$ using the prediction of the ingoing RWM instance for node v, and the outgoing expert, which predicts $y_{u,v}$ using the prediction of the outgoing RWM instance for node u. The

number of mistakes of these two experts satisfy

$$\sum_{u\in V}\sum_{v\in\mathcal{N}_{\text{out}}(u)}\mathbb{E}\,M(u,v)=\Psi_{\text{out}}(Y)+O\left(\sqrt{|V|\Psi_{\text{out}}(Y)}+|V|\right)$$
$$\sum_{v\in V}\sum_{u\in\mathcal{N}_{\text{in}}(v)}\mathbb{E}\,M(u,v)=\Psi_{\text{in}}(Y)+O\left(\sqrt{|V|\Psi_{\text{in}}(Y)}+|V|\right)\ ,$$

where we used $\sum_{v\in V}\sqrt{\Psi_{\text{in}}(v,Y)}\leq\sqrt{|V|\Psi_{\text{in}}(Y)}$, and similarly for $\Psi_{\text{out}}(Y)$. Finally, let the overall prediction of our algorithm be a RWM instance run on top of the ingoing and the outgoing experts. Then the expected number of mistakes of this predictor satisfies

$$\begin{aligned}\sum_{(u,v)\in E}\mathbb{E}\,M(u,v)&=\Psi_G(Y)+O\left(\sqrt{|V|\Psi_G(Y)}+|V|+\sqrt{\left(\Psi_G(Y)+|V|+\sqrt{|V|\Psi_G(Y)}\right)}\right)\\&=\Psi_G(Y)+O\left(\sqrt{|V|\Psi_G(Y)}+|V|\right)\end{aligned}$$

as claimed. □

We complement the above result by providing a mistake lower bound. Like Theorem 2, the following result holds for all graphs, and for all label irregularity levels $\Psi_G(Y)$.

Theorem 3. *Given any edge-labeled graph $G(Y)=(V,E(Y))$ and any integer $K\leq\left\lfloor\frac{|E|}{2}\right\rfloor$, a randomized labeling $Y\in\{-1,+1\}^{|E|}$ exists such that $\Psi_G(Y)\leq K$, and the expected number of mistakes that any online algorithm A make can be forced to satisfies $\mathbb{E}M_A(Y)\geq\frac{K}{2}$.*

Proof. Let $\mathcal{Y}_K$ be the set of all labelings Y such that the total number of negative and positive edges are K and $|E|-K$, respectively (without loss of generality we will focus on negative edges). Consider the randomized strategy that draws a labeling $Y\in\{-1,+1\}^{|E|}$ of the edges of the input graph uniformly at random from $\mathcal{Y}_K$. For each node $u\in V$, we have $\Psi_{\text{in}}(u,Y)\leq d^-_{\text{in}}(u)$, which implies $\Psi_{\text{in}}(Y)\leq K$. A very similar argument applies to the outgoing edges, leading to $\Psi_{\text{out}}(Y)\leq K$. The constraint $\Psi_G(Y)\leq K$ is therefore always satisfied.

The adversary will force on average $1/2$ mistakes in each one of the first K rounds of the online protocol by repeating K times the following: (i) A label value $\ell\in\{-1,+1\}$ is selected uniformly at random. (ii) An edge (u,v) is sampled uniformly at random from the set of all edges that were not previously revealed and whose labels are equal to ℓ.

The learner is required to predict $y_{u,v}$ and, in doing so, $1/2$ mistakes will be made on average because of the randomized labeling procedure. Observe that this holds even when A knows the value of K and $\Psi_G(Y)$. Hence, we can conclude that the expected number of mistakes that A can be forced to make is always at least $K/2$, as claimed. □

In fact, we can refine the above statement by proving that, as $\frac{K}{|E|}\to 0$, the lower bound gets arbitrarily close to K for any $G(Y)$, hence asymptotically matching the upper bound of Theorem 2. A sketch of the proof can be found on page 41.

2.7 Open questions

Given the fundamental role of our generative model, we would like to adapt it in order to handle the three problem extensions presented in the related works of Section 2.4.

1. When available, using **side information** about users is an alluring option to alleviate the cold start problem. Recall that in our experiments, we saw that if we learn our model on historical data, we cannot successfully predict the interactions involving new users, for their trollness and trustworthiness are not yet known to us. However, we expect that for existing users, side information is correlated with their value of p and q. This knowledge thus provides a parameterised prior for the p and q values of news users. As we observe their behavior in the network, our estimation of p and q would rely less and less on prior information and more and more on observed data, as it is common in a Bayesian approach. Note that this introduces an additional learning stage.

2. Our generative model, and in particular the prior distribution $\mu(p, q)$, could be exploited to guide label queries in an adaptive **active learning** setting. Given a budget of queries B, and after an initial phase of querying edge labels uniformly at random, one might use concentration results on μ[22] to devise an informativeness criterion, indicating for which node the value of p and q are the most uncertain. Combined with the graph topology, this would suggest which edge is the most important to query next, until the budget is exhausted.

3. So far we discussed binary classification of edges. In **weighted graphs** though, a refined problem would be to predict the weight of unlabeled edges. Let us assume for simplicity that the weights are bounded and within $[-1, 1]$. The most immediate solution would be, for an edge $u \to v$, to linearly shift our estimation of $\frac{p_u+q_v}{2} = \eta(u, v)$ from $[0, 1]$ to $[-1, 1]$ and use this quantity as our prediction. We could also change the semantic of $\eta(u, v)$ from being the probability of $u \to v$ to be positive to the mean of a narrow Gaussian from which the weight of $u \to v$ is drawn. Finally, by letting p_u range from -1 to $+1$, we note a similarity with fairness and goodness, suggesting $p_u \times q_v$ could approximate the weights.

2.8 Summary

In this chapter, we started our characterization of edges in complex networks by studying the EDGE SIGN PREDICTION problem in Directed Signed Social Networks. As mentioned in the introduction, given a network topology and a few labeled interactions, being able to infer the polarity of the remaining interactions is valuable to improve user experience. Our main insight is that two features of users behavior, trollness and (un)trustworthiness, are key to understand their interactions. This leads us to design a simple sign generative model. Such a model serves both as a theoretical justification for many successful heuristics and as the underpinning of our BLC(tr, un) algorithm. This algorithm is fast, trivially parallelizable, provably close to the Bayes optimal predictor on dense graphs and experimentally accurate even on sparse graphs. We further exploit this model in the batch setting through an approximation of the maximum likelihood estimation procedure. While making this approximation, we moreover cast the problem as a node prediction problem. This can be efficiently tackled by standard Label Propagation algorithms.

We validate our theoretical results by experimentally assessing these two methods in the small training set regime, on synthetic and real-world datasets. We draw two main conclusions from our experiments. First, our generative model is robust across domains. Indeed, it produces predictors that are empirically both close to the Bayes optimal and to general linear models trained on trollness and trustworthiness features. This is notable as this result is achieved by simply counting negative edges, without resorting to Stochastic Gradient Descent methods typically involved

[22] As done in the proof of Theorem 1, on the facing page.

in training large scale linear models. Second, our methods are in practice either strictly better than their competitors in terms of prediction quality or, when they are not, they are faster.

Finally, we study the online adversarial setting, where trollness and (un)trustworthiness naturally lend themselves to define a notion of edge sign regularity. Based on this regularity, we provide an upper and an (almost matching) lower bounds on the expected number of prediction mistakes. We conclude by presenting three directions in which our generative model could be extended: namely use side information as priors, guide budgeted queries in active learning and generate weights in addition to signs.

In the next chapter, we will go even further than this generative model and explore another learning bias for EDGE SIGN PREDICTION, which is not based on trollness and (un)trustworthiness, but rather on the social balance theory.

2.9 Additional material

2.9.1 Proofs from Section 2.3

2.9.1.1 Proof of Theorem 1

The following ancillary results will be useful.

Lemma 1 (Hoeffding's inequality for sampling without replacement)**.** *Let* $\mathcal{X} = \{x_1, \dots, x_N\}$ *be a finite subset of* $[0,1]$ *and let*

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \,.$$

If $X_1, \dots, X_n$ *is a random sample drawn at random from* $\mathcal{X}$ *without replacement, then, for every* $\varepsilon > 0$*,*

$$\Pr\left(\left|\frac{1}{n}\sum_{t=1}^{n} X_t - \mu\right| \geq \varepsilon\right) \leq 2e^{-2n\varepsilon^2} \,.$$

Lemma 2. *Let* $\mathcal{N}_1, \dots, \mathcal{N}_n$ *be* n *subsets of a finite set* E*. Let* $E_0 \subseteq E$ *be sampled uniformly at random* without *replacement from* E*, with* $|E_0| = m$*. Then, for* $\delta \in (0,1)$*,* $Q > 0$*, and* $\theta \geq 2 \times \max\left\{Q, 4\log\frac{n}{\delta}\right\}$*, we have*

$$\Pr\Big(\exists i \,:\, |\mathcal{N}_i| \geq \theta, |\mathcal{N}_i \cap E_0| < Q\Big) \leq \delta$$

provided $|E| \geq m \geq \frac{2|E|}{\theta} \times \max\left\{Q, 4\log\frac{n}{\delta}\right\}$.

Proof of Lemma 2. Set for brevity $p_i = {}^{|\mathcal{N}_i|}/_{|E|}$. Then, due to the sampling without replacement, each random variable $|\mathcal{N}_i \cap E_0|$ is the sum of m dependent Bernoulli random variables $X_{i,1}, \dots, X_{i,m}$ such that $\Pr(X_{i,t} = 1) = p_i$, for $t = 1, \dots, m$. Let i be such that $|\mathcal{N}_i| \geq \theta$. Then, since $\theta \geq 2Q$, we may assume that $m\frac{2Q}{\theta} \leq |E|$, which implies

$$Q \leq \frac{m\theta}{2|E|} \leq \frac{m\,p_i}{2} = \frac{\mathbb{E}\big[|\mathcal{N}_i \cap E_0|\big]}{2} \,.$$

Since the variables $X_{i,j}$ are negatively associated, we may apply a (multiplicative) Chernoff bound [DP09, Section 3.1]. This gives

$$\Pr\left(|\mathcal{N}_i \cap E_0| < Q\right) \leq e^{-\frac{m\,p_i}{8}} \leq e^{-\frac{m\,\theta}{8|E|}}$$

so that $\Pr\big(\exists i \,:\, |\mathcal{N}_i| \geq \theta, |\mathcal{N}_i \cap E_0| < Q\big) \leq n\,e^{-\frac{m\,\theta}{8|E|}}$, which is in turn upper bounded by δ whenever $m \geq \frac{8|E|}{\theta}\log\frac{n}{\delta}$ (again a natural assumption since $\theta \geq 8\log\frac{n}{\delta}$). □

Let now $E_\theta = \{(u,v) \in E : d_{\text{in}}(v) \geq \theta,\, d_{\text{out}}(u) \geq \theta\} \setminus E_0$, where $E_0 \subseteq E$ is the set of sampled edges provided to the learning algorithm of Section 2.3.1. Then Theorem 1 in the main paper is an immediate consequence of the following lemma.

Lemma 3. *Let $G(Y) = (V, E(Y))$ be a directed graph with labels on the edges generated according to the model in Section 2.2. For all $0 < \alpha, \delta < 1$ and $0 < \varepsilon < \frac{1}{16}$, if the learning algorithm of Section 2.3.1 is run with parameters $Q = \frac{1}{2\varepsilon^2} \log \frac{4|V|}{\delta}$ and α, then with probability at least $1 - 9\delta$ the predictions $\widehat{y}(u,v)$ satisfy $\widehat{y}(u,v) = y^*(u,v)$ for all $(u,v) \in E_\theta$ such that $\left|\eta(u,v) - \frac{1}{2}\right| > 8\varepsilon$.*

Proof of Lemma 3. We apply Lemma 2 with $\theta = \frac{2Q}{\alpha} \geq 2 \times \max\left\{Q, 4 \log \frac{2|V|+1}{\delta}\right\}$ to the $2|V|+1$ subsets of E consisting of E_L and, for all $u \in V$, of $\mathcal{E}_{\text{in}}(u)$ and $\mathcal{E}_{\text{out}}(u)$. We have that, with probability at least $1-\delta$, at least Q edges of E_L are sampled, at least Q edges of $\mathcal{E}_{\text{in}}(v)$ are sampled for each v such that $\mathcal{N}_{\text{in}}(v) \geq \theta$, and at least Q edges of $\mathcal{E}_{\text{out}}(u)$ are sampled for each u such that $\mathcal{N}_{\text{out}}(u) \geq \theta$. For all $(u,v) \in E_\theta$ let

$$\overline{p}_v = \frac{1}{d_{\text{in}}(v)} \sum_{u \in \mathcal{N}_{\text{in}}(v)} p_u \quad \text{and} \quad \overline{q}_u = \frac{1}{d_{\text{out}}(u)} \sum_{v \in \mathcal{N}_{\text{out}}(u)} q_v$$

and set for brevity $\widehat{\delta}_{\text{in}}(v) = 1 - \widehat{un}(v)$ and $\widehat{\delta}_{\text{out}}(u) = 1 - \widehat{tr}(u)$. We now prove that $\widehat{\delta}_{\text{in}}(v)$ and $\widehat{\delta}_{\text{out}}(u)$ are concentrated around their expectations for all $(u,v) \in E_\theta$. Consider $\widehat{\delta}_{\text{out}}(u)$ (the same argument works for $\widehat{\delta}_{\text{in}}(v)$). Let $V_1, \ldots, V_Q$ be the first Q draws in $E_0 \cap \mathcal{N}_{\text{out}}(u)$ and define

$$\widehat{\mu}_p(u) = \frac{1}{Q} \sum_{t=1}^{Q} \frac{p_u + q_{V_t}}{2} \,.$$

Applying Lemma 1 to the set $\left\{\frac{p_u+q_v}{2} : v \in \mathcal{N}_{\text{out}}(u)\right\}$, and using our choice of Q, we get that $\left|\widehat{\mu}_p(u) - \mu_p(u)\right| \leq \varepsilon$ holds with probability at least $1 - \delta/(2|V|)$, where

$$\mu_p(u) = \frac{1}{d_{\text{out}}(u)} \sum_{v \in \mathcal{N}_{\text{out}}(u)} \frac{p_u + q_v}{2} = \frac{p_u + \overline{q}_u}{2} \,.$$

Now consider the random variables $Z_t = \mathbb{I}\{y_{u,v_t} = 1\}$, for $t = 1, \ldots, Q$. Conditioned on $V_1, \ldots, V_Q$, these are independent Bernoulli random variables with $\mathbb{E}[Z_t \mid V_t] = \frac{p_u + q_{V_t}}{2}$. Hence, applying a standard Hoeffding bound for independent variables and using our choice of Q, we get that

$$\left| \frac{1}{Q} \sum_{t=1}^{Q} Z_t - \widehat{\mu}_p(u) \right| \leq \varepsilon$$

with probability at least $1 - \delta/(2|V|)$ for every realization of $V_1, \ldots, V_Q$.

Since $\widehat{\delta}_{\text{out}}(u) = (Z_1 + \cdots + Z_Q)/Q$, we get that $\left|\widehat{\delta}_{\text{out}}(u) - \mu_p(u)\right| \leq 2\varepsilon$ with probability at least $1 - 2\delta/(2|V|)$. Applying the same argument to $\widehat{\delta}_{\text{in}}(v)$, and the union bound[23] on the set $\left\{\widehat{\delta}_{\text{in}}(v), \widehat{\delta}_{\text{out}}(u) : (u,v) \in E_\theta\right\}$, we get that

$$\left| \widehat{\delta}_{\text{out}}(u) + \widehat{\delta}_{\text{in}}(v) - \frac{\overline{p}_v + \overline{q}_u}{2} - \frac{p_u + q_v}{2} \right| \leq 4\varepsilon \tag{2.10}$$

simultaneously holds for all $(u,v) \in E_\theta$ with probability at least $1 - 4\delta$. Now notice that $\overline{p}_v$ is a sample mean of Q i.i.d. $[0,1]$-valued random variables drawn from the prior marginal $\int_0^1 \mu(\cdot, q)\, dq$ with expectation μ_p. Similarly, $\overline{q}_u$ is a sample mean of Q

[23] The sample spaces for the ingoing and outgoing edges of the vertices occurring as endpoints in E_θ overlap. Hence, in order to prove a uniform concentration result, we need to apply the union bound over the random variables defined over these sample spaces.

i.i.d. $[0,1]$-valued random variables independently drawn from the prior marginal $\int_0^1 \mu(p,\cdot)dp$ with expectation μ_q. By applying Hoeffding bound for independent variables, together with the union bound to the set of pairs of random variables whose sample means are $\overline{p}_v$ and $\overline{q}_u$ for each $(u,v) \in E_\theta$ (there are at most $2|V|$ of them) we obtain that

$$\left|\overline{p}_v - \mu_p\right| \leq \varepsilon \qquad \text{and} \qquad \left|\overline{q}_u - \mu_q\right| \leq \varepsilon$$

hold simultaneously for all $(u,v) \in E_\theta$ with probability at least $1-2\delta$. Combining with (2.10) we obtain that

$$\left|\widehat{\delta}_{\text{out}}(u) + \widehat{\delta}_{\text{in}}(v) - \frac{\mu_p + \mu_q}{2} - \frac{p_u + q_v}{2}\right| \leq 5\varepsilon \tag{2.11}$$

simultaneously holds for each $(u,v) \in E_\theta$ with probability at least $1-6\delta$. Next, let E'_L be the set of the first Q edges drawn in $E_L \cap E_0$. Then

$$\mathbb{E}\left[\widehat{\tau}\right] = \frac{1}{Q} \sum_{(u,v)\in E'_L} \Pr\left(y_{u,v} = 1\right) = \frac{1}{Q} \sum_{(u,v)\in E'_L} \frac{p_u + q_v}{2}\,,$$

where the expectation is w.r.t. the independent draws of the labels $y_{u,v}$ for $(u,v) \in E'_L$. Hence, by applying again Hoeffding bound (this time without the union bound) to the $Q = \frac{1}{2\varepsilon^2}\log\frac{4|V|}{\delta}$ independent Bernoulli random variables $\mathbb{I}\{y_{u,v}=1\}$, $(u,v) \in E'_L$, the event $\left|\widehat{\tau} - \mathbb{E}\left[\widehat{\tau}\right]\right| \leq \varepsilon$ holds with probability at least $1-\delta$. Now, introduce the function

$$F(\boldsymbol{p},\boldsymbol{q}) = \mathbb{E}\left[\widehat{\tau}\right] = \frac{1}{Q} \sum_{(u,v)\in E'_L} \frac{p_u + q_v}{2}\,.$$

For any realization $\boldsymbol{q}_0$ of $\boldsymbol{q}$, the function $F_1(\boldsymbol{p}) = F(\boldsymbol{p},\boldsymbol{q}_0)$ is a sample mean of $Q = \frac{1}{2\varepsilon^2}\log\frac{4|V|}{\delta}$ i.i.d. $[0,1]$-valued random variables $\{p_u : (u,v) \in E'_L\}$ (recall that if $u \in V$ is the origin of an edge $(u,v) \in E'_L$, then it is not the origin of any other edge $(u,v') \in E'_L$). Using again the standard Hoeffding bound, we obtain that

$$\left|F(\boldsymbol{p},\boldsymbol{q}) - E_{\boldsymbol{p}}\left[F(\boldsymbol{p},\boldsymbol{q})\right]\right| \leq \varepsilon$$

holds with probability at least $1-\delta$ for each $\boldsymbol{q} \in [0,1]^{|V|}$. With a similar argument, we obtain that

$$\left|E_{\boldsymbol{p}}\left[F(\boldsymbol{p},\boldsymbol{q})\right] - E_{\boldsymbol{p},\boldsymbol{q}}\left[F(\boldsymbol{p},\boldsymbol{q})\right]\right| \leq \varepsilon$$

also holds with probability at least $1-\delta$. Since

$$E_{\boldsymbol{p},\boldsymbol{q}}\left[F(\boldsymbol{p},\boldsymbol{q})\right] = \frac{\mu_p + \mu_q}{2}$$

we obtain that

$$\left|\widehat{\tau} - \frac{\mu_p + \mu_q}{2}\right| \leq 3\varepsilon \tag{2.12}$$

with probability at least $1-3\delta$. Combining (2.11) with (2.12) we obtain

$$\left|\widehat{\delta}_{\text{out}}(u) + \widehat{\delta}_{\text{in}}(v) - \widehat{\tau} - \frac{p_u + q_v}{2}\right| \leq 8\varepsilon$$

simultaneously holds for each $(u,v) \in E_\theta$ with probability at least $1-9\delta$. Putting together concludes the proof. □

2.9.1.2 Derivation of the maximum likelihood equations

Recall that the training set $E_0 = \left\{(u_k, v_k), y_{u_k,v_k}\right) : k = 1, \dots, m\}$ is drawn uniformly at random from E without replacement. We can write

$$\begin{aligned}\Pr\left(E_0 \,\middle|\, \{p_u, q_u\}_{u=1}^{|V|}\right) =& \frac{1}{\binom{|E|}{m} m!} \prod_{k=1}^{m} \left(\frac{p_{u_k} + q_{v_k}}{2}\right)^{\mathbb{I}\{y_{u_k,v_k}=+1\}} \\ & \times \prod_{k=1}^{m} \left(1 - \frac{p_{u_k} + q_{v_k}}{2}\right)^{\mathbb{I}\{y_{u_k,v_k}=-1\}} \\ =& \frac{1}{\binom{|E|}{m} m!} \prod_{\ell=1}^{|V|} \Bigg(\prod_{k=1}^{m} \left(\frac{p_\ell + q_{v_k}}{2}\right)^{\mathbb{I}\{u_k=\ell,\, y_{\ell,v_k}=+1\}} \times \\ & \prod_{k=1}^{m} \left(1 - \frac{p_\ell + q_{v_k}}{2}\right)^{\mathbb{I}\{u_k=\ell,\, y_{\ell,v_k}=-1\}} \Bigg)\end{aligned}$$

so that $\log \Pr\left(E_0 \,\middle|\, \{p_u, q_u\}_{u=1}^{|V|}\right)$ is proportional to

$$\begin{aligned}\sum_{\ell=1}^{|V|} \Big(& \sum_{k=1}^{m} \mathbb{I}\{u_k = \ell, y_{\ell,v_k} = +1\} \log\left(\frac{p_\ell + q_{v_k}}{2}\right) + \\ & \sum_{k=1}^{m} \mathbb{I}\{u_k = \ell, y_{\ell,v_k} = +1\} \log\left(1 - \frac{p_\ell + q_{v_k}}{2}\right) \Big)\end{aligned}$$

and

$$\frac{\partial \log \Pr\left(E_0 \,\middle|\, \{p_u, q_u\}_{u=1}^{|V|}\right)}{\partial p_\ell} = \sum_{k=1}^{m} \frac{\mathbb{I}\{u_k = \ell, y_{\ell,v_k} = +1\}}{p_\ell + q_{v_k}} - \sum_{k=1}^{m} \frac{\mathbb{I}\{u_k = \ell, y_{\ell,v_k} = -1\}}{2 - p_\ell - q_{v_k}}.$$

By a similar argument,

$$\begin{aligned}\Pr\left(E_0 \,\middle|\, \{p_u, q_u\}_{u=1}^{|V|}\right) = \frac{1}{\binom{|E|}{m} m!} \prod_{\ell=1}^{|V|} \Bigg(& \prod_{k=1}^{m} \left(\frac{p_{u_k} + q_\ell}{2}\right)^{\mathbb{I}\{v_k=\ell,\, y_{u_k,\ell}=+1\}} \times \\ & \prod_{k=1}^{m} \left(1 - \frac{p_{u_k} + q_\ell}{2}\right)^{\mathbb{I}\{v_k=\ell,\, y_{u_k,\ell}=-1\}} \Bigg)\end{aligned}$$

so that

$$\frac{\partial \log \Pr\left(E_0 \,\middle|\, \{p_u, q_u\}_{u=1}^{|V|}\right)}{\partial q_\ell} = \sum_{k=1}^{m} \frac{\mathbb{I}\{v_k = \ell, y_{u_k,\ell} = +1\}}{p_{u_k} + q_\ell} - \sum_{k=1}^{m} \frac{\mathbb{I}\{v_k = \ell, y_{u_k,\ell} = -1\}}{2 - p_{u_k} - q_\ell}.$$

2.9.1.3 Label propagation on G''

Here we provide more details on the choice of weight for the edges of G'', as well as an explanation on why we temporarily use symmetrized variables lying in $[-1, 1]$ (which we will denote with primes, so that for instance $p'_u = 2p_u - 1$). Since only the ratio between the negative and positive weights matters, we fix the negative weight of the edges in $E'' \setminus E'$ to be -1 and we denote by ϵ the weight of edges in E'. With these notations, Label Propagation on G'' seeks the harmonic minimizer of the following expression

$$\frac{1}{16}\sum_{u,v\in E}\Big[\epsilon\left(y_{u,v}-p'_u\right)^2+\epsilon\left(y_{u,v}-q'_v\right)^2+(p'_u+q'_v)^2\Big]$$

which can be successively rewritten as

$$\begin{aligned}
&\frac{1}{16}\sum_{u,v\in E}\Big[\epsilon\left(y_{u,v}+1-2p_u\right)^2+\epsilon\left(y_{u,v}+1-2q_v\right)^2+(2p_u+2q_v-2)^2\Big]\\
&=\frac{1}{8}\sum_{u,v\in E}\left[2\epsilon\left(\frac{y_{u,v}+1}{2}-p_u\right)^2+2\epsilon\left(\frac{y_{u,v}+1}{2}-q_v\right)^2+8\left(\frac{p_u+q_v-1}{2}\right)^2\right]\\
&=\frac{1}{8}\sum_{u,v\in E}\left[2\epsilon\left(\left(\frac{y_{u,v}+1}{2}\right)^2-p_u(1+y_{u,v})+p_u^2\right)+2\epsilon\left(\left(\frac{y_{u,v}+1}{2}\right)^2-q_v(1+y_{u,v})+q_v^2\right)+\right.\\
&\qquad\left.8\left(\left(\frac{p_u+q_v}{2}\right)^2-\frac{p_u+q_v}{2}+\frac{1}{4}\right)\right]\\
&=\frac{1}{8}\sum_{u,v\in E}4\left(\epsilon\left(\frac{y_{u,v}+1}{2}\right)^2-2\epsilon\left(\frac{y_{u,v}+1}{2}\right)\left(\frac{p_u+q_v}{2}\right)+2\left(\frac{p_u+q_v}{2}\right)^2\right)\\
&\qquad+\sum_{u,v\in E}\Big[\left(2\epsilon p_u^2-4p_u+1\right)+\left(2\epsilon q_v^2-4q_v+1\right)\Big]
\end{aligned}$$

By setting $\epsilon=2$, we can factor this expression into

$$\sum_{u,v\in E}\left(\frac{y_{u,v}+1}{2}-\frac{p_u+q_v}{2}\right)^2+\frac{1}{2}\sum_{u,v\in E}\left(\left(p_u-\frac{1}{2}\right)^2+\left(q_v-\frac{1}{2}\right)^2\right)\,.$$

2.9.2 Proof from Section 2.6

Proof sketch that, in the context of Theorem 3, as $\frac{K}{|E|}\to 0$ *then* $\mathbb{E}M_A(Y)=K$. Let $\mathcal{E}$ be the following event: There is at least one unrevealed negative label. The randomized iterative strategy used to force a number of mistakes arbitrarily close to K is identical to the one described in the proof of Theorem 3 on page 35, except for the stopping criterion. Namely, the adversary draws a labeling Y uniformly at random from $\mathcal{Y}_K$, the set of all labelings Y such that the total number of negative and positive edges are K and $|E|-K$. Then the adversary repeats the following two steps: (i) A label value $\ell\in\{-1,+1\}$ is selected uniformly at random. (ii) An edge (u,v) is sampled uniformly at random from the set of all edges that were not previously revealed and whose labels are equal to ℓ. This is done until $\mathcal{E}$ is true.

Let $m_{r,c}$ be defined as follows: For $c=1$ it is equal to the expected number of mistakes forced in round r when $K=1$. For $c>1$ it is equal to the difference between the expected number of mistakes forced in round r when $K=c$ and $K=c-1$. One can see that $m_{r,c}$ is null when $r<c$. When $K=1$, the probability that $\mathcal{E}$ is true in round r is clearly equal to $\frac{1}{2^{r-1}}$. Hence, the expected number of mistakes made by A when $K=1$ in any round r is equal to $\frac{1}{2}\,\frac{1}{2^{r-1}}=\frac{1}{2^r}$. We can therefore conclude that $m_{r,1}=\frac{1}{2^r}$ for all r.

A simple calculation shows that if $r=c$ then $m_{r,c}=\frac{1}{2^r}$. Furthermore, when $r>1$ and $c>1$, we have the following recurrence:

$$m_{r,c}=\frac{m_{r-1,c}+m_{r-1,c-1}}{2}\,.$$

In order to calculate $m_{r,c}$ for all r and c, we will rest on the ancillary quantity $s_j(i)$, defined recursively for any positive integer variables i and j by $s_0(i)=1$ and

$$s_j(i)=\sum_{k=1}^{i}s_{j-1}(k)\,.$$

(see Table 2.9) for examples)

i \ j	**0**	**1**	**2**	**3**
1	1	1	1	1
2	1	2	3	4
3	1	3	6	10
4	1	4	10	20
5	1	5	15	35
6	1	6	21	56
7	1	7	28	84

Table 2.9 – Values of $s_j(i)$ for $i \leq 7$ and $j \leq 3$.

It is not difficult to verify that

$$m_{r,c} = \frac{s_{c-1}(r-c+1)}{2^r}.$$

Since $s_j(i) = \frac{\langle i \rangle_j}{j!}$, where $\langle i \rangle_j$ is the rising factorial $i(i+1)(i+2)\dots(i+j-1)$, we have

$$m_{r,c} = \frac{\langle r-c+1 \rangle_{c-1}}{(c-1)!2^r}.$$

(see Table 2.10) for examples)

r \ c	**1**	**2**	**3**	**4**
1	$1/2^1$	0	0	0
2	$1/2^2$	$1/2^2$	0	0
3	$1/2^3$	$2/2^3$	$1/2^3$	0
4	$1/2^4$	$3/2^4$	$3/2^4$	$1/2^4$
5	$1/2^5$	$4/2^5$	$6/2^5$	$4/2^5$
6	$1/2^6$	$5/2^6$	$10/2^6$	$10/2^6$
7	$1/2^7$	$6/2^7$	$15/2^7$	$20/2^7$

Table 2.10 – Values of $m_{r,c}$ for $r \leq 7$, $c \leq 4$ and $|E| \to \infty$.

When $\frac{K}{|E|} \to 0$, given any integer $K' > 1$, the difference between the expected number of mistakes forced when $K = K'$ and $K = K' - 1$ is equal to

$$\begin{aligned}\sum_{r=K'}^{\infty} m_{r,K'} &= \frac{1}{(K'-1)!} \sum_{r=K'}^{\infty} \frac{\langle r-K'+1 \rangle_{K'-1}}{2^r} \\ &= \frac{1}{(K'-1)!2^{K'-1}} \sum_{r'=1}^{\infty} \frac{\langle r' \rangle_{K'-1}}{2^{r'}},\end{aligned}$$

where we set $r' = r - K' + 1$. Setting $i' = i - 1$ and recalling that

$$\langle i \rangle_j = j! \binom{i+j-1}{i-1},$$

we have

$$\frac{1}{j!} \sum_{i=1}^{\infty} \frac{\langle i \rangle_j}{2^i} = \sum_{i=1}^{\infty} \frac{\binom{i+j-1}{i-1}}{2^i} = \sum_{i'=0}^{\infty} \frac{\binom{i'+j}{i'}}{2^{i'+1}}.$$

Now, using the identity

$$\binom{i'+j+1}{i'} = \binom{i'+j}{i'} + \binom{i'+j}{i'-1},$$

we can easily prove by induction on j that

$$\sum_{i'=0}^{\infty} \frac{\binom{i'+j}{i'}}{2^{i'+1}} = 2^j.$$

Hence, we have

$$\sum_{r=K'}^{\infty} m_{r,K'} = 1.$$

Moreover, as shown earlier, $m_{r,1} = \frac{1}{2^r}$ for all r. Hence we can conclude that when $\frac{K}{|E|} \to 0$

$$\mathbb{E} M_A(Y) \geq \sum_{r=1}^{\infty} \frac{1}{2^r} + \sum_{K'=2}^{K} \sum_{r=K'}^{\infty} m_{r,K'} = K$$

for any edge-labeled graph $G(Y)$ and any constant K, as claimed. □

2.9.3 Further Experimental Results

This section contains more evidence related to the experiments in Section 2.5. In particular, we experimentally demonstrate the alignment between BLC(tr, un) and LOGREG.

After training on the two features $1 - \widehat{tr}(u)$ and $1 - \widehat{un}(v)$, LOGREG has learned three weights w_0, w_1 and w_2, which allow to predict $y_{u,v}$ according to

$$\text{sign}\Big(\big(w_1(1 - \widehat{tr}(u)\big) + w_2\big(1 - \widehat{un}(v)\big) + w_0\Big).$$

This can be rewritten as

$$\text{sign}\Big(\big(1 - \widehat{tr}(u)\big) + w_2'\big(1 - \widehat{un}(v)\big) - \tfrac{1}{2} - \tau'\Big),$$

with $w_2' = \frac{w_2}{w_1}$ and $\tau' = -\left(\frac{1}{2} + \frac{w_0}{w_1}\right)$.

As shown in Table 2.11, and in accordance with the predictor built out of Equation (2.3), w_2' is almost 1 on *all datasets*, while τ' tends to be always close the fraction of positive edges in the dataset.

Table 2.11 – Normalized logistic regression coefficients averaged over 12 runs (with one standard deviation)

	$\frac{\lvert E_0 \rvert}{\lvert E \rvert}$	w_2	τ
CITATIONS	5%	0.929 ± 0.05	0.637 ± 0.04
	10%	0.980 ± 0.02	0.700 ± 0.02
	15%	0.993 ± 0.04	0.728 ± 0.03
	20%	1.016 ± 0.02	0.747 ± 0.02

	25%	1.013 ± 0.03	0.750 ± 0.02
	50%	1.037 ± 0.01	0.776 ± 0.01
	90%	1.044 ± 0.01	0.788 ± 0.01
	5%	1.109 ± 0.06	0.724 ± 0.04
	10%	1.059 ± 0.04	0.718 ± 0.02
ADVOGATO	15%	1.067 ± 0.02	0.740 ± 0.01
	20%	1.043 ± 0.01	0.735 ± 0.01
	25%	1.047 ± 0.02	0.745 ± 0.02
	50%	1.020 ± 0.01	0.736 ± 0.01
	90%	1.006 ± 0.01	0.735 ± 0.01
	5%	0.914 ± 0.03	0.693 ± 0.02
	10%	0.942 ± 0.03	0.730 ± 0.02
WIKIPEDIA	15%	0.951 ± 0.02	0.748 ± 0.01
	20%	0.956 ± 0.01	0.755 ± 0.01
	25%	0.971 ± 0.01	0.771 ± 0.01
	50%	0.976 ± 0.01	0.779 ± 0.01
	90%	0.984 ± 0.01	0.787 ± 0.00
	5%	1.019 ± 0.02	0.686 ± 0.01
	10%	1.019 ± 0.01	0.707 ± 0.01
SLASHDOT	15%	1.005 ± 0.01	0.706 ± 0.01
	20%	1.000 ± 0.01	0.710 ± 0.00
	25%	0.998 ± 0.01	0.713 ± 0.00
	50%	0.984 ± 0.00	0.716 ± 0.00
	90%	0.975 ± 0.00	0.717 ± 0.00
	5%	1.099 ± 0.02	0.790 ± 0.02
	10%	1.058 ± 0.01	0.779 ± 0.00
EPINION	15%	1.030 ± 0.01	0.769 ± 0.01
	20%	1.016 ± 0.01	0.765 ± 0.01
	25%	1.007 ± 0.01	0.762 ± 0.00
	50%	0.981 ± 0.00	0.752 ± 0.00
	90%	0.963 ± 0.00	0.745 ± 0.00
	5%	1.047 ± 0.02	0.852 ± 0.01
	10%	1.038 ± 0.01	0.872 ± 0.01
WIK. EDITS	15%	1.030 ± 0.01	0.879 ± 0.01
	20%	1.012 ± 0.01	0.873 ± 0.01
	25%	1.009 ± 0.01	0.877 ± 0.01
	50%	0.980 ± 0.00	0.868 ± 0.00
	90%	0.947 ± 0.00	0.851 ± 0.00

Chapter 3

Edge sign prediction in general graphs and Correlation Clustering

In the previous chapter, we presented a learning bias for the EDGE SIGN PREDICTION problem, namely our sign generative model. By leveraging its rigorous theoretical guarantees, we have demonstrated empirically its good performance on real Directed Signed Social Networks. However, in this chapter, we consider other kinds of signed networks and show they require another bias and different, more combinatorial algorithms. More precisely, we start by pointing in Section 3.1 that our previous bias may not apply to directed graphs from domains other than social science, or to undirected graphs. Motivated by balance theories, we then introduce a new bias, assuming that ideally, nodes are grouped in K clusters such that all edges within clusters are positive and all edges between cluster are negative. Recovering such clusters from a given signed graph has been studied extensively in the last decades under the CORRELATION CLUSTERING name. In Section 3.2, we show how this difficult combinatorial clustering problem is connected to our learning bias. We survey a wide range of methods to solve it exactly, approximately or heuristically, paying special attention to settings where such recovery is easier, such as noisy or stable instances. Finally, focusing on the important and convenient special case $K = 2$, we develop in Section 3.3 an algorithm that, given an undirected general graph, produce a spanning tree designed to support fast and accurate edge sign prediction.

3.1 A bias for general graphs

Let us recall first the sign model we used for Directed Signed Social Networks. Each node u is endowed with two parameters p_u and q_u drawn from an arbitrary joint distribution μ over $[0, 1] \times [0, 1]$. p_u can be interpreted as the tendency of u to send positive edges to other nodes (i.e. the "niceness" of u), while q_u can be interpreted as the tendency of u to receive positive edges from other nodes (i.e. the "popularity" of u). The sign of an edge $u \to v$ is then chosen to be positive with probability $\frac{p_u+q_v}{2}$. This suggests that nodes in the graph have a form of agency, and further imposes that edges are directed. We show experimentally that failing to meet these two assumptions harms the performance of our previous methods. This is not surprising, for our bias is no more justified in that case. Therefore, we suggest a different bias, drawing heavily on social balance theories, although we shall see later this holds for many nonsocial graphs as well.

3.1.1 Sign generative model and behavior

We can interpret the description of our sign model as if, when establishing a new link, nodes were deciding the sign of this link based on their individual preferences. While this makes sense in social networks where nodes represent human beings,

Table 3.1 – Biological dataset properties. The columns have the same meaning as in Table 2.2 on page 26

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\frac{\lvert E\rvert}{\lvert V\rvert}$	$\frac{\lvert E^+\rvert}{\lvert E\rvert}$	$\frac{\Psi^2_{G''}(Y)}{\lvert E\rvert}$	$\frac{\Psi_G(Y)}{\lvert E\rvert}$	reciprocal edges	reciprocal disagreement
HIPPOCAMPAL	501	1 046	2.1	69.5%	.056	.124	0.2%	0.0%
CANCER	1 240	3 065	2.5	78.4%	.047	.108	5.5%	27.1%
REGULONDB	1 700	2 570	1.5	50.3%	.060	.148	0.0%	0.0%

one can imagine contexts where this model is not applicable. Another model, e.g. for proteins, could be that proteins belongs to functional groups and that two proteins interact positively if they belong to the same group, negatively otherwise. We presented a way to circumvent this notion of node behavior with our online algorithm, where this time, signs are generated by an arbitrary adversary. However, even in that case, our bias remains that the labeling is regular. Recall this means informally that all the outgoing signs from a given node tend to be the same, and likewise for the incoming signs. Indeed, irregularities are the cost payed by the adversary to make our algorithm mispredict. In other words, regularity hints at a consistent sign behavior from nodes, that we cannot assume carry out outside of social networks. Pursuing the proteins example, we can imagine that proteins interact half of the time with proteins of their own group, and half of time with proteins from other groups. This would correspond to maximum irregularity, yet it is a plausible situation. We now show experimentally, that in fact, biological networks do not lend themselves to our sign model bias.

Namely, we consider *gene regulatory networks*. The nodes of such directed graphs are various chemicals (such as genes, proteins or messenger RNAs) and a directed edge $u \to v$ indicates that one node u influences the concentration of another node v through a chemical reaction. This influence can be positive (that is, an increase of u concentration results in an increase of v concentration) or negative (that is, an increase of u concentration results in a decrease of v concentration). In this context, we could now interpret the "niceness" of u as it ability to accelerate chemical reactions, and its "popularity" as its propensity to take part in reactions accelerated by other nodes. Intuitively, this is quite far-fetched. To demonstrate this, we borrow the following three gene regulatory networks from [KvK09, Table 1], and display their statistics in Table 3.1:

1. The signaling pathways of the HIPPOCAMPAL CA1 neuron [Maa+05].
2. The interactions between genes and their products that are involved in CANCER development in humans and mouses [Cui+07].
3. The REGULONDB database [Gam+16], summarizing the connection between transcription factors and their targets in the Escherichia coli K-12 bacteria. We download four files[1] containing experimental evidence and keep only the positive and negative edges with Strong support.

Compared with the Directed Signed Social Networks of the previous chapter, those biological networks are smaller, have lower average degree and little to no reciprocal edges. Trying to solve the EDGE SIGN PREDICTION problem using the same procedure as in Section 2.5, we read in Table 3.2 that our methods have lower absolute MCC performance (for instance, L. PROP. with a 15% training size achieves 20, 24 and 45 compared with values between 36 and 61 on Directed Signed Social Networks). To further illustrate the mismatch between the troll bias and biological networks, notice that the standard deviation figures are several times larger than those reported in Table 2.4.

[1] http://regulondb.ccg.unam.mx/menu/download/datasets/index.jsp

Table 3.2 – This table is the same as in Table 2.4 on page 31, but this time on three smaller, directed biological networks.

	$\frac{\lvert E_0\rvert}{\lvert E\rvert}$	L. PROP.	BLC(tr, un)	LOGREG	UNREG.	LOWRANK	MAXNORM	16 TRIADS	RANKNODES	BAYESIAN
HIPPOCAMPAL	5%	**14.1** ± 5.7	*11.5* ± 6.3	9.6 ± 6.7	8.5 ± 4.7	10.9 ± 6.3	2.3 ± 3.4	3.7 ± 4.0	11.4 ± 7.4	5.4 ± 5.2
	9%	*18.1* ± 6.1	16.9 ± 6.0	15.4 ± 4.2	12.2 ± 4.2	15.3 ± 4.9	1.8 ± 4.6	7.0 ± 3.3	**21.2** ± 4.8	15.0 ± 5.7
	15%	*20.3* ± 5.5	18.6 ± 4.9	17.6 ± 4.4	14.3 ± 4.2	18.0 ± 4.5	2.7 ± 2.6	12.1 ± 4.9	**24.4** ± 3.7	19.5 ± 4.5
	20%	22.0 ± 4.3	21.1 ± 4.0	19.0 ± 3.1	17.7 ± 4.0	17.2 ± 2.4	3.0 ± 3.4	15.9 ± 3.6	**26.7** ± 3.5	*23.2* ± 3.5
	25%	24.4 ± 3.8	23.2 ± 3.2	21.5 ± 3.6	19.2 ± 2.6	21.5 ± 3.5	3.2 ± 3.4	18.5 ± 3.6	**28.6** ± 4.7	*24.7* ± 3.7
	50%	30.5 ± 4.5	*32.1* ± 4.4	32.0 ± 4.8	28.1 ± 4.2	31.9 ± 4.0	16.0 ± 3.9	31.5 ± 4.1	**36.4** ± 3.9	31.2 ± 4.1
	90%	34.4 ± 5.8	35.7 ± 7.1	35.5 ± 7.4	31.3 ± 7.8	**40.8** ± 12.7	*39.9* ± 11.4	36.3 ± 9.0	35.9 ± 6.6	37.2 ± 9.3
CANCER	5%	**15.6** ± 4.0	12.9 ± 4.1	10.6 ± 3.9	9.1 ± 2.9	13.0 ± 2.4	0.7 ± 1.9	5.7 ± 4.6	14.2 ± 5.1	*14.7* ± 3.3
	10%	*21.7* ± 2.6	18.0 ± 3.2	15.9 ± 3.1	12.1 ± 2.6	16.9 ± 2.7	1.0 ± 2.2	10.3 ± 4.3	**21.7** ± 2.9	15.6 ± 3.4
	15%	*24.6* ± 2.3	21.3 ± 3.5	20.1 ± 2.2	15.5 ± 2.0	19.6 ± 2.7	0.5 ± 2.0	14.8 ± 2.6	**24.8** ± 3.2	20.3 ± 3.0
	20%	*26.5* ± 2.0	24.0 ± 2.3	24.2 ± 2.2	19.7 ± 2.4	22.7 ± 3.4	3.5 ± 2.5	18.8 ± 3.2	**27.4** ± 2.7	22.4 ± 2.5
	25%	*29.3* ± 2.1	25.9 ± 1.6	26.3 ± 2.1	21.4 ± 1.2	24.8 ± 2.9	7.2 ± 2.6	21.2 ± 3.0	**29.6** ± 2.0	24.8 ± 3.0
	50%	34.6 ± 1.7	34.7 ± 1.9	*34.8* ± 1.7	32.0 ± 2.9	31.3 ± 3.2	22.4 ± 2.6	31.7 ± 1.8	**35.3** ± 3.3	33.7 ± 2.0
	90%	39.8 ± 4.4	42.4 ± 4.5	42.8 ± 4.7	39.3 ± 5.3	34.4 ± 5.3	34.3 ± 4.3	40.6 ± 6.5	**43.9** ± 3.9	*42.8* ± 4.9
REGULONDB	5%	26.5 ± 8.9	31.5 ± 4.7	20.4 ± 13.6	29.0 ± 2.9	**32.6** ± 4.1	−0.5 ± 2.2	7.7 ± 7.6	*31.7* ± 7.3	5.0 ± 7.7
	10%	39.3 ± 4.9	**43.6** ± 2.9	38.5 ± 5.2	37.2 ± 2.3	40.1 ± 2.3	−2.3 ± 2.2	19.6 ± 7.2	*43.3* ± 3.7	8.2 ± 5.6
	15%	44.9 ± 2.1	*45.6* ± 2.4	40.9 ± 6.0	40.2 ± 2.1	44.9 ± 2.9	0.5 ± 2.3	29.9 ± 4.9	**49.2** ± 3.0	17.4 ± 9.6
	20%	*46.8* ± 2.5	46.1 ± 2.3	42.0 ± 3.3	42.0 ± 2.2	46.1 ± 2.9	3.0 ± 2.7	35.3 ± 5.1	**51.4** ± 3.5	20.3 ± 11.5
	25%	*49.1* ± 2.4	48.4 ± 3.1	44.4 ± 2.6	43.8 ± 2.7	48.2 ± 2.0	5.6 ± 2.1	38.6 ± 5.6	**54.5** ± 2.1	32.7 ± 8.5
	50%	52.4 ± 2.8	48.7 ± 2.3	44.6 ± 3.2	47.4 ± 2.8	*55.8* ± 1.8	26.0 ± 2.6	50.3 ± 4.6	**59.7** ± 1.7	40.0 ± 6.7
	90%	56.4 ± 6.2	50.4 ± 5.6	47.9 ± 6.1	47.2 ± 5.6	**65.8** ± 5.5	47.2 ± 6.4	59.7 ± 6.1	*64.0* ± 4.7	49.5 ± 6.5

3.1.2 Directed edges requirement

Having two parameters per node, one for outgoing edges and another one for incoming edges, clearly targets directed graphs. Many online interactions are inherently directed, for instance friendship, trust or communication. On the other hand, predicting edge signs in undirected graphs is an equally relevant objective. A prime example of such a situation is when we are given n objects, some pairwise similarities them, and the similarity function itself, which unfortunately takes an exorbitant time to be evaluated. There is an underlying graph and being able to predict the sign of its undirected edges would save us expensive evaluations of the symmetric similarity function.

A trivial way to turn an undirected graph G into a directed graph G' is to let $V' = V$ and, for every edge (u, v) in E, to add both $u \to v$ and $v \to u$ to E' with the same sign as (u, v). In terms of our generative model, this corresponds to putting all the probability mass of μ on the $p = q$ diagonal. This is clearly not a very satisfactory solution, for it removes one degree of freedom from the model. To illustrate this point, we conduct the following experiment. We use our previous Directed Signed Social Networks datasets and remove the edge direction. As mentioned earlier, for a few pair of nodes, there are reciprocal edges of different signs, in which case we pick a sign arbitrarily. Given those undirected graphs, we orient them using the approach described above and compare our method with the LOWRANK approach ran directly on the undirected graphs. This is a fair comparison, for LOWRANK is designed to works solely on undirected graphs.

As we can see in Table 3.3, our methods perform worse than when running on the original directed graphs. Looking at L. PROP. on a 15% training set as an example, we observe that the MCC decreases by one (WIK. EDITS) to almost ten (EPINION) points. On the other hand, the LOWRANK seems to perform better. Recall however that because it is an undirected method in the first place and because we roughly double the number of signed edges, a 15% training size means the algorithm gets to observe more signs and therefore perform better. This small experiment thereby shows that although the performance remains decent, not having direction information hurts our bias.

Table 3.3 – MCC results on the six datasets from Chapter 2, after removing the directions as described in the main text. These results are presented like in Table 2.4, except we have transposes the rows and the columns, and we show only three relevant methods.

$\frac{\lvert E_0\rvert}{\lvert E\rvert}$	5%	10%	15%	20%	25%	50%	90%
				CITATIONS			
L. PROP.	**22.2** ± 0.7	**28.6** ± 0.5	**32.0** ± 0.5	*34.4* ± 0.3	*36.9* ± 0.5	41.7 ± 0.4	44.9 ± 0.9
BLC(tr, un)	*18.1* ± 0.7	25.3 ± 0.8	29.7 ± 0.4	32.6 ± 0.4	35.8 ± 0.4	*42.2* ± 0.4	*45.0* ± 1.0
LOWRANK	17.3 ± 0.5	*25.3* ± 0.3	*31.4* ± 0.5	**36.3** ± 0.8	**41.2** ± 1.2	**55.4** ± 0.8	**60.6** ± 3.9
				ADVOGATO			
L. PROP.	**36.5** ± 0.4	**41.4** ± 0.5	**43.9** ± 0.4	**45.7** ± 0.5	**47.0** ± 0.5	**50.4** ± 0.4	**52.7** ± 0.7
BLC(tr, un)	*32.8* ± 0.5	*38.9* ± 0.7	*41.8* ± 0.8	*44.3* ± 0.8	*46.3* ± 0.7	*49.8* ± 0.5	*52.6* ± 0.8
LOWRANK	28.8 ± 0.6	32.2 ± 0.4	33.9 ± 0.6	35.0 ± 0.5	36.3 ± 0.7	41.1 ± 1.0	46.2 ± 3.4
				WIKIPEDIA			
L. PROP.	**36.3** ± 0.3	**42.6** ± 0.4	**45.8** ± 0.2	**47.8** ± 0.4	**49.2** ± 0.2	**52.5** ± 0.3	*54.1* ± 0.5
BLC(tr, un)	*35.0* ± 0.6	*41.4* ± 0.4	*44.9* ± 0.3	*46.9* ± 0.5	*48.2* ± 0.4	51.3 ± 0.4	52.7 ± 0.6
LOWRANK	30.6 ± 0.7	37.2 ± 0.4	40.7 ± 0.5	43.4 ± 0.6	45.8 ± 0.6	*51.8* ± 0.7	**54.6** ± 4.0
				SLASHDOT			
L. PROP.	**39.4** ± 0.1	**44.9** ± 0.2	**47.8** ± 0.1	**49.7** ± 0.1	**51.0** ± 0.1	*54.4* ± 0.1	*56.4* ± 0.2
BLC(tr, un)	35.5 ± 0.3	40.7 ± 0.1	43.8 ± 0.2	46.0 ± 0.2	47.3 ± 0.2	51.5 ± 0.1	54.0 ± 0.4
LOWRANK	*39.0* ± 0.3	*41.8* ± 0.3	*44.8* ± 0.5	*48.0* ± 0.6	*50.5* ± 0.4	**57.2** ± 0.3	**59.1** ± 0.6
				EPINION			
L. PROP.	**45.1** ± 0.4	*50.0* ± 0.4	*52.6* ± 0.3	*54.4* ± 0.2	*55.8* ± 0.3	*59.3* ± 0.1	*62.0* ± 0.2
BLC(tr, un)	40.7 ± 0.6	47.2 ± 0.5	50.2 ± 0.6	53.1 ± 0.2	54.6 ± 0.2	58.6 ± 0.1	60.9 ± 0.2
LOWRANK	*44.1* ± 0.1	**50.8** ± 0.7	**55.9** ± 0.5	**59.3** ± 0.4	**61.7** ± 0.2	**66.8** ± 0.2	**68.4** ± 0.8
				WIK. EDITS			
L. PROP.	**34.4** ± 0.3	**37.1** ± 0.4	**37.9** ± 0.3	**38.4** ± 0.6	*38.5* ± 0.4	*38.9* ± 0.3	*39.4* ± 0.3
BLC(tr, un)	*28.5* ± 0.3	*32.9* ± 0.2	*34.8* ± 0.1	35.5 ± 0.2	36.1 ± 0.1	37.2 ± 0.1	37.6 ± 0.3
LOWRANK	26.7 ± 0.4	30.9 ± 0.4	34.2 ± 0.6	*36.6* ± 0.8	**38.7** ± 0.7	**44.0** ± 0.4	**46.3** ± 1.8

3.1.3 Social balance as a learning bias

From a narrow practical point of view, balance theories specify which sign assignments of a triangle are possible. By forbidding triangles with a single negative edge, weak balance gives rise to a *K-consistent* clusters structure on complete graphs. After making these notions more detailed in the rest of this section, we conclude by precisely exposing our learning bias.

> We focus here on balance theory, for it is both compelling and well-established. Indeed, as illustrated by the account of Zheng *et al.* [ZZW15], it has been used extensively in the last half century to study signed networks and their dynamics (an example of this far reaching influence is the analysis of the relation between characters in fairy tales [Aus80]). However, it has shortcomings, the main one being its inability to deal with directed graphs. Therefore, alternative theories governing signs formation have been proposed, as surveyed in [YH15].

In his seminal work on interpersonal relations, Heider [Hei46] posits through psychological and sociological arguments that in order to reduce their cognitive dissonance, three people always interact in a way that preserve social balance. These social balance requirements can be succinctly summarized by four statements [Hei58]:

1. my friend's friend is my friend 
2. my friend's enemy is my enemy 

3. my enemy's friend is my enemy 

4. my enemy's enemy is my friend 

This can be readily translated into graph properties. Given a path of length two over three nodes, note that the first part of each statement (in blue and orange) defines the four possible sign assignments of such a path. The last part of the statement (in green) then prescribes which sign should close the triangle to respect the social balance. Such triangles are called strongly balanced.

Definition 3.1.1 (strongly balanced triangle)**.** *A triangle is strongly balanced if it has zero or two negative signs. Otherwise it is unbalanced.*

We use triangles as the building block of our definition of strongly balanced graphs.

Definition 3.1.2 (strongly balanced complete graphs)**.** *A complete signed graph is strongly balanced if all its triangles are strongly balanced.*

It is interesting to note that this local property of triangles gives raise to a global structure. Namely, a strongly balanced complete graph can be split in two (possibly empty) clusters L and R such that all edges within L and R are positive, while all edges between L and R are negative. Indeed, let us pick an arbitrary node u and let $L = \mathcal{N}^+(u) \setminus \{u\}$ and $R = V \setminus L$ (where $\mathcal{N}^+(u)$ denotes the set of all positive neighbors of u). Now let $v, w \in L \setminus \{u\}$ and $x, y \in R$. From Figure 3.1 and the fact that every triangle is balanced, we can see that (v, w) and (x, y) are positive edges, while (w, y) is a negative edge. We say that L and R are 2-consistent clusters and more generally, we define

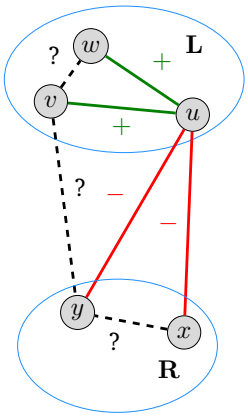


Figure 3.1 – A two-clustering of a complete strongly balanced graph

Definition 3.1.3 (K-consistent clustering)**.** *Given a graph $G = (V, E)$, a clustering $\mathcal{C} = \{C_1, \dots, C_K\}$ of V is* consistent *with the signs of E if for every edge (u, v) in E:*
(i) (u, v) is positive if $\mathcal{C}(u) = \mathcal{C}(v)$, and
(ii) (u, v) is negative if $\mathcal{C}(u) \neq \mathcal{C}(v)$.

We just saw that if a complete graph is strongly balanced, then it admits a 2-consistent clustering (Theorem 4 proves that the converse is true). It is natural to ask under which conditions a complete signed graph admits a $K > 2$ consistent clustering. This is where the notion of weak balance comes handy. It was noted by sociologists that among triangles with odd number of negative edges, one is more stable and common than that the other. Therefore, Davis [Dav67] relax the strong balance into the weak balance by considering triangles with three negative edges to be balanced, as illustrated on Figure 3.2. Formally, we can modify our two previous definitions of balance to arrive at weak balance:

Definition 3.1.4 (Weak balance)**.** *A triangle is weakly balanced if it has zero, two or three negative signs. Otherwise it is unbalanced.*
A complete signed graph is weakly balanced if all its triangles are weakly balanced.

In that case as well, the local property of triangles has structural implication for the whole graph. Namely, we can partition a complete weakly balanced graph into K consistent clusters. Consider the same construction as in Figure 3.1, putting a node u and its positive neighborhood in one cluster L and the rest of the nodes in R. For $v, w \in L \setminus \{u\}$ and $x, y \in R$, we can still conclude that (v, w) is positive and (v, y) is negative. On the other hand, this time weak balance allows (x, y) to be either positive or negative. We can split R further into the positive neighborhood of

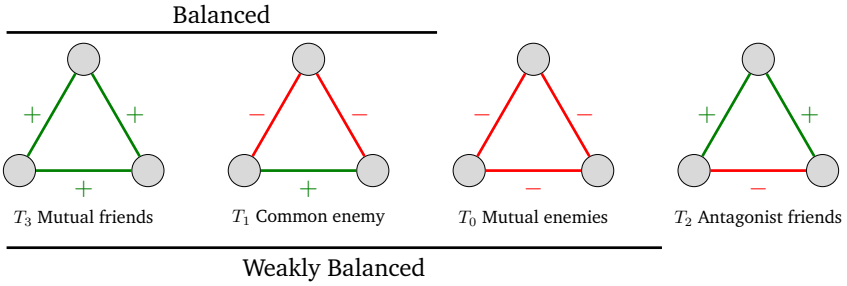


Figure 3.2 – The four possible undirected triads, as classified by the two structural balance theories introduced in the main text

x and the rest, and keep doing so until we form the k clusters. In the following we will make this argument more formal.

We gave the definitions of balance in the context of complete graphs in order to build intuition about the partition consequences and because it easier to reason on triangles. Furthermore, if we are given a set of objects and a similarity function, we can indeed build a complete graph. However, they are situations were it is not realistic to make the completeness assumption, especially for social networks that are typically very sparse. We therefore extend the concept of balance to general graphs in the following way:

Definition 3.1.5 (balanced general graphs)**.** *A general graph is strongly (respectively weakly) balanced if there is a sign assignment of all its missing edges such that the resulting complete signed graph is strongly (respectively weakly) balanced.*

To recover the characterization of signed graphs by consistent clustering, we need to consider longer cycles than triangles.

Definition 3.1.6 (balanced cycles)**.** *A cycle is strongly (respectively weakly) unbalanced if it has an odd number of negative edges (respectively exactly one negative edge). Otherwise it is strongly (respectively weakly) balanced.*

Being strongly balanced is equivalent to having a 2-consistent clustering, as proved in the landmark theorem of Harary [Har53, Theorem 3]:

Theorem 4 (Structural Theorem)**.** *A graph $G = (V, E)$ is strongly balanced if and only if V admits a 2-consistent clustering.*

> According to Hüffner *et al.* [HBN10], a similar theorem was proved earlier by König [Kön36, Theorem X.11], although Zaslavsky [Zas12] notes that it was stated "without the terminology of signs, while [Har53] has the first recognition of the crucial fact that labelling edges by elements of a group—specifically, the sign group—can lead to a general theory."

A useful characterization, whose proof can be found in [EK10, page 122], is the following:

Theorem 5. *A graph $G = (V, E)$ is strongly balanced if and only if all its cycles are strongly balanced.*

Similar results also hold for weak balance.

Theorem 6. *A graph $G = (V, E)$ is weakly balanced if and only if V admits a K-consistent clustering.*

Proof. Let first assume that G is weakly balanced. According to the definition, we can choose sign for the missing edges such that it becomes a complete graph with all its triangles weakly balanced. As mentioned earlier, we can then pick a node u_1

and let $C_1 = \mathcal{N}^+(u) \cup \{u\}$. All nodes in C_1 are connected positively with each other and negatively with nodes in $V \setminus C_1$. Note also that $|V \setminus C_1| < |V|$ and therefore we can repeat this procedure until V is exhausted, at which point we have obtained our K consistent clusters.

Conversely, assume we have K consistent clusters. We can complete the graph G by letting the edge (u, v) be positive if u and v are in the same cluster and negative otherwise. Let us the pick three arbitrary nodes $u, v, w \in V$. There are three cases: they are either all in the same cluster, all in different clusters or two of them are in a first cluster and the third node is in a second cluster. In every case, one can check they form a weakly balanced triangle and therefore G is weakly balanced. □

Theorem 7 (Theorem 1 of [Dav67])**.** *A graph $G = (V, E)$ is weakly balanced if and only if all its cycles are weakly balanced.*

In real data though, we do not expect either strong or weak balance to hold, for they are fairly demanding model. Indeed, three of the real networks considered in Chapter 2 have been repeatedly shown to be unbalanced, although the extent of this unbalance depends on the importance given to longer cycles compared with triads [FIA11; EB14; SA17]. Easley *et al.* [EK10, Section 5.5] nonetheless prove that even when only a fraction of the triangle are strongly balanced in a complete graph, the two clusters structure is still present, although it does not cover the whole graph any more.

Theorem 8. *Let $\epsilon < 1/8$ and $\delta = \sqrt[3]{\epsilon}$. If at least $1 - \epsilon$ of all triangles in a signed complete graph are strongly balanced, then either*

(a) there is a set consisting of at least $1 - \delta$ of the nodes in which at least $1 - \delta$ of all edges are positive, or else

(b) the nodes can be divided into two groups L and R such that

(i) at least $1 - \delta$ of the edges in L are positive,

(ii) at least $1 - \delta$ of the edges in R are positive, and

(iii) at least $1 - \delta$ of the edges between L and R are negative.

Using these results on weak balance, we can now make our new bias explicit. Drawing parallel with the previous generative model, we assume that each node u is endowed with an integer $\mathcal{C}(u) \in \{1, \dots, K\}$ that specifies its cluster and we let the undirected edge u, v be positive if $\mathcal{C}(u) = \mathcal{C}(v)$ and negative otherwise. In other words, this corresponds to picking an arbitrary partition of V and setting the signs of E in such a way that the partition becomes a consistent clustering. This can still be seen as a generative process because of the initial integer drawn uniformly at random, but one that avoids the two pitfalls discussed before. Indeed, there is no decision to be made by nodes, since the previous probability has been replaced by a binary function (either both nodes are in the same cluster or not). Furthermore, since the clustering function is symmetric, the model is inherently suited to undirected graphs.

3.2 CORRELATION CLUSTERING

As mentioned earlier, most real signed graphs are not fully balanced and thus do not have a perfectly consistent clustering. The problem of quantifying how much a given graph departs from this ideal situation is called CORRELATION CLUSTERING. After defining it formally in Section 3.2.1, we show that CORRELATION CLUSTERING is a learning problem on its own, and discuss how it relates to the binary classification problem of predicting edge signs, among many others applications. We then present numerous methods to solve CORRELATION CLUSTERING in Section 3.2.2, ranging from exact methods (that implicitly leverage our bias) to heuristics (whose greedy and energy minimization frameworks are classic in unsupervised graph cut problems). We also devote a large section to approximation methods, in order to get

a sense of why the problem is NP-COMPLETE and even hard to approximate. This allows us to finish in Section 3.2.3 by describing situations related to our bias where is possible to solve the problem efficiently and optimally. In this section, we thus update existing surveys on the same topic [BGL14; IIK16; Wir17] and highlight the importance of CORRELATION CLUSTERING to solve our binary learning problem.

3.2.1 Problem setting and connection to EDGE SIGN PREDICTION

Like other clustering frameworks, in CORRELATION CLUSTERING, we are given a set of objects and we want to gather them into groups (called clusters) so that objects belonging to one cluster are similar to each other while being dissimilar to objects from all the other clusters.

In CORRELATION CLUSTERING, we formalize this problem by considering objects as the nodes of a graph G, whose edge weights encode similarity. Namely, in the most general case, for nodes u and v, the edge between u and v is associated with two positive numbers: $w^+_{u,v}$ denotes the strength of the similarity between u and v, while $w^-_{u,v}$ denotes the strength of the dissimilarity between u and v. Note, however, that in many applications, only one of these two numbers is nonzero, in which case we more conveniently set $w_{u,v} = \begin{cases} w^+_{u,v} & \text{if } w^+_{u,v} > 0 \text{ and } w^-_{u,v} = 0 \\ -w^-_{u,v} & \text{if } w^+_{u,v} = 0 \text{ and } w^-_{u,v} > 0 \end{cases}$

Now consider a clustering $\mathcal{C}$ of V, that is a function from V to $\mathbb{N}^{|V|}_{>0}$ that assigns to each node a cluster index. For instance, $\mathcal{C}(u) = 3$ means that u belongs to the third cluster. We will also abuse the notation and let $\mathcal{C}$ be a set of clusters $\{C_1, C_2, \dots, C_K\}$ that form a partition of V.[2] We can evaluate how $\mathcal{C}$ fits our clustering paradigm in two ways, either by the number of *agreements*, that is the weighted number of positive edges inside clusters plus the weighted number of negative edges across clusters; or by the number of *disagreements*, that is the weighted number of negative edges inside clusters plus the weighted number of positive edges across clusters. Given a cost function c, which is usually the identity, CORRELATION CLUSTERING can then be seen as a graph optimization problem, either of maximizing agreements (MAXAGREE):

$$\max_{\mathcal{C}} \sum_{\mathcal{C}(u)=\mathcal{C}(v)} c(w^+_{uv}) + \sum_{\mathcal{C}(u)\neq\mathcal{C}(v)} c(w^-_{uv}) \tag{3.1}$$

or minimizing disagreements (MINDISAGREE)[3]:

$$\min_{\mathcal{C}} \sum_{\mathcal{C}(u)=\mathcal{C}(v)} c(w^-_{uv}) + \sum_{\mathcal{C}(u)\neq\mathcal{C}(v)} c(w^+_{uv}) \tag{3.2}$$

Although an optimal clustering $\mathcal{C}^\star$ achieves the same value on both (3.1) and (3.2), we will see in Section 3.2.2 that the latter objective is in some sense "easier". Another interesting feature of the CORRELATION CLUSTERING problem is that contrary to other clustering formulations, it does not require us to set the number of clusters K beforehand. Instead, K emerges as a natural property of the solution. Since clustering is an unsupervised problem, this is generally handy. However, in some situations, we may have prior knowledge on how many clusters are the data, or external constraints. This can be handled with parametrized version of CORRELATION CLUSTERING, namely MAXAGREE$[K]$ and MINDISAGREE$[K]$ where the optimization is over clustering with exactly K clusters.

In Figure 3.3, we show a simple instance of CORRELATION CLUSTERING and one of its optimal solution.

[2] That is, $\forall i\ C_i \subset V$, $\bigcup_{i=1}^K C_i = V$ and $\forall i \neq j$, $C_i \cap C_j = \emptyset$.

[3] Note that in the data mining literature, CORRELATION CLUSTERING may refer to another problem. Namely, it is a special case of high-dimensional data clustering, where features are locally correlated in various ways across different clusters that reside in arbitrarily oriented subspaces [KKZ09].

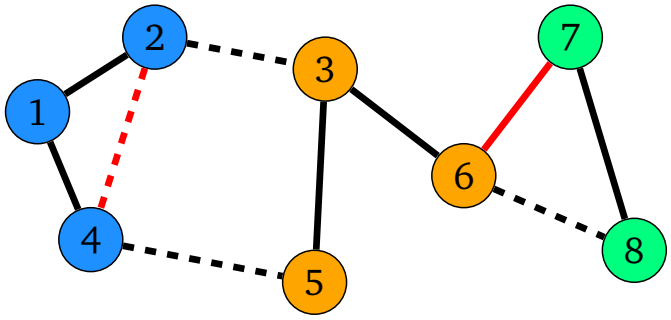


Figure 3.3 – A small graph with eight nodes and ten edges. Solid edges represent positive edges and dashed edges represent negative edges. A clustering $\mathcal{C}$ is showed with 3 clusters: $\{1, 2, 4\}$, $\{3, 5, 6\}$ and $\{7, 8\}$. $\mathcal{C}$ incurs two disagreements: the negative edge between nodes 1 and 2 within the blue cluster, and the positive edge $(6, 7)$ between the orange and green clusters. Those disagreements are created by two cycles with one negative edge and thus cannot be avoided, meaning that $\mathcal{C}$ is optimal. However, it is not the unique solution: for instance, merging the orange and green clusters would also yield two disagreements.

Connection with EDGE SIGN PREDICTION and other applications EDGE SIGN PREDICTION can clearly be casted as a supervised classification problem where, given a labelled training set of m_0 edges $\mathcal{X} = \{(e_1, y_1), \ldots, (e_{m_0}, y_{m_0})\}$, we can extract for every edge a feature vector in $\mathbb{R}^d$, pick an hypothesis class like linear models, find among this class the hypothesis minimizing the empirical risk, and use it to predict the sign of edges in the testing set. In contrast, CORRELATION CLUSTERING in an agnostic problem. More precisely, the hypothesis class is fixed, as it consists of all possible partitions of the nodeset. We are trying to find the partition that best approximates the observed signs, but each weakly unbalanced cycle will cost us at least one unavoidable mistake. Despite this fundamental difference in their nature, CORRELATION CLUSTERING is connected with EDGE SIGN PREDICTION at two levels, theoretical and practical. On the theoretical side, computing the minimum number of disagreements gives a measure of the EDGE SIGN PREDICTION problem complexity. Intuitively, given a labelled subgraph H of G (that is the training set), if there are few disagreements, the "quasi" connected components of H^+ should give a reasonable estimation of the underlying consistent clustering. Predicting the signs based on this clustering should results in few errors. More formally, one can show that by finding a clustering $\mathcal{C}^\star$ of H that has less disagreements than κ times the smallest number of disagreements Δ, the number of errors on the testing set will be at most of order $\kappa\Delta + \sqrt{mn \log n}$ [Ces+12b, Theorem 6]. These theoretical considerations naturally lead to a practical batch algorithm, which first finds a clustering minimizing as much as possible the number of disagreements on the training set and then predicts signs according to this clustering. Our assumption in doing so is that, according to our bias, our graph has few disagreements. Therefore, provided we can find a clustering close to the optimal, we will not make too many prediction mistakes.

Besides the special case of predicting edge sign, Demaine *et al.* [Dem+06, Section 5] point out that CORRELATION CLUSTERING is well suited to several more general situations:

- when the items to be clustered do not belong to a natural metric space (preventing approaches such as k-means) but we still know for some pairs whether they are similar or not.

- when we do not know the number of clusters beforehand but we have a similarity measure. In that case, we can select a problem-specific similarity

threshold and set all edges with a similarity larger than the threshold to be positive while the others are set to negative.

- when we have a classic clustering problem (that is a set of objects, a distance between them and an objective function to minimize) with additional pairwise constraints of the form *must-link/cannot-link*. Instead of restraining a clustering algorithm to the space of feasible solution, we convert the distances between objects and the constraints into signed edges and solve the resulting CORRELATION CLUSTERING problem.

Furthermore, as mentioned in the introduction of this thesis on page 4, CORRELATION CLUSTERING have been used in many domains. For instance in computer vision, it is used to segment images in 2D [Kim+11; BG11; ZYH14] or 3D [And+12; BHK15], to simplify 3D shapes [Gor+17] and to track targets across sequential video frames [SCC15]. When the nodes are words, CORRELATION CLUSTERING is employed to identify coreference [MW05; ES09] or cluster synonym words [SPU17]. It plays a large role in biology, to clusters genes [BSY99], identify mutation regions in chromosomes [DV15] or finding stable subsystems [Das+07]. Applied to social networks, it helps analyze political assemblies [Men+15; Jia15; LF17] and international relations [TB09; MMP12]. Finally, given large databases with duplicated records, it naturally models entity resolution [GZ07; Has+09; ARS09; SW14].

3.2.2 State of the art

As we saw, solving CORRELATION CLUSTERING on a training set would give us a principled way of predicting edge signs. In this section, we thus look at existing methods for CORRELATION CLUSTERING. We start with exact methods and fixed parameter algorithms on complete graphs, noting they implicitly assume our learning bias. However, in practice, these methods do not scale well, especially if their assumptions are not met. Indeed, we next present hardness results about CORRELATION CLUSTERING, namely that it is NP-HARD to approximate with some constant factor greater than one, and then describe some proposed approximations. Although this appears rather negative at first, we show that heuristics can solve large instances with satisfactory results. We conclude by further highlighting the learning aspect of CORRELATION CLUSTERING through approaches in the active and online settings.

3.2.2.1 Exact methods

Because of the complexity of the CORRELATION CLUSTERING, one has to rely on approximations to solve large instances of the problem. However, we can imagine offline signed social networks with only few nodes, in which case it is reasonable to expect to find the optimal clustering. Furthermore, this can also be useful to evaluate in practice the quality of heuristic methods, albeit in non asymptotic settings. The methods used to solve CORRELATION CLUSTERING exactly depend on the type of graph considered. More precisely, the fixed parameter algorithms we describe now only handle complete graphs, while the next set of methods also work on general graphs.

Complete graphs Let us first describe the cluster editing problem, where one is given a general unsigned input graph $H = (V, E)$ and wants to find the smallest number of edges that have to be added or deleted to turn H into node-disjoint union of cliques. It is possible to alternate between an instance H of cluster editing and an instance G of CORRELATION CLUSTERING on complete graphs in polynomial time. To see why, let $G = (V, (E^+, E^-))$ be a complete signed graph where V is the nodeset of h, E^+ is the set E of edges of H and E^- is the set of all edges that are not in H. The optimal clusters of G are the cliques of H, the negative disagreements

within clusters are edges added to H and the positive disagreements between clusters are deleted from H. To the best of our knowledge, the problem was first introduced under this name by Shamir *et al.* [SST02].[4] They show the problem is NP-COMPLETE, even if the number of clusters $K \geq 2$ is set beforehand[5] and provide a 0.878-approximation in the weighted $K = 2$ case using the celebrated Goemans-Williamson SDP relaxation [GW95].

If we parameterize the problem by the number d of edges that need to be added or deleted (that is, the number of disagreements), then it can be solved in polynomial time in the size of the input graph (but not in d). The best known approximation so far is $O(1.62^d + m + n)$ [Böc12], which search for conflict triples (i.e. unbalanced triangles in the sign language) and branch by either deleting or merging one positive edge. If we additionally look for exactly K clusters (i.e. MINDISAGREE$[K]$), there is a fixed parameters algorithm running in $O(2^{O(\sqrt{kd})} + n + m)$ [Fom+14]. If at most a edges can be added and at most b edges can be deleted at each node, and if the minimum size of a cluster is at least $2(a + b)$, then the problem can be solved in polynomial time [Abu17] by using various reduction rules. Finally, for planar graph, there is a PTAS[6] running in $O(n2^{\epsilon^{-1}\log(\epsilon^{-1})})$ obtained by dividing the graph into independent components of bounded treewidth [BGW17]. In other words, if we are given a complete signed graph whose positive subgraph is planar, then we can minimize the number of disagreements arbitrarily close to the optimal, provided we have enough time to do so.

General graphs On general graphs, a natural idea is to formulate the clustering problem as an integer problem and solve it optimally in polynomial time. Assign a binary variable x_{uv} to each pair of nodes (so that $x_{uv} = x_{vu}$). For a given clustering $\mathcal{C}$, let $x_{uv} = 0$ if u and v are in the same cluster and $x_{uv} = 1$ is u and v are in different clusters. Noting that $1 - x_{uv}$ is 1 if the edge (u, v) is within a cluster and 0 otherwise, the weighted number of disagreements is then $w(\mathcal{C}) = \sum_{(u,v)\in E^-} w_{uv}(1 - x_{uv}) + \sum_{(u,v)\in E^+} w_{uv}x_{uv}$. By construction, if edges (u, v) and (v, w) are within the same cluster, then (v, w) is also within that cluster. In terms of x variable, we have that $x_{uv} = 0 \wedge x_{vw} = 0 \implies x_{uw} = 0$. For x to be a valid cluster assignment, we thus require that all variables are either 0 or 1 and respect the triangle inequality.

$$\text{minimize} \quad \sum_{(u,v)\in E^-} w_{uv}(1 - x_{uv}) + \sum_{(u,v)\in E^+} w_{uv}x_{uv} \tag{3.3}$$

$$\begin{aligned} \text{subject to} \quad & x_{uw} \leq x_{uv} + x_{vw} \\ & \leq x_{uv} \in \{0, 1\} \\ & x_{uv} = x_{vu} \end{aligned} \tag{3.4}$$

Figueiredo *et al.* [FM13] solve this binary integer program on random instances and show that depending on the negative edge density, the FICO Xpress solver starts to be unable to finish within a one-hour time limit when $n \geq 40$. Aref *et al.* [AMW16] describe four linear and quadratic binary integer problems to model CORRELATION CLUSTERING with two clusters along with some preprocessing optimizations. With the Gurobi software, they solve instances with up to 3200 edges in less than a second. They also sketch extensions to weighted graphs and more than two clusters. In a different direction, Berg *et al.* [BJ17], encode the linear and quadratic integer

[4] However, the same problem was studied before and we refer the reader to the comprehensive survey of Böcker *et al.* [BB13] for additional details, whereas we shall only give important and more recent pointers here.

[5] The reduction is from 3-exact 3 cover, see [SST02, Theorems 1, 2 and Corollary 1].

[6] An algorithm $\mathcal{A}$ is a *polynomial-time approximation scheme (PTAS)* for a minimization (respectively maximization) problem $\mathcal{P}$ in NP if given any $\epsilon > 0$ and any instance x of $\mathcal{P}$ of size n, $\mathcal{A}$ produces, in time polynomial in n, a solution that is within a factor $1 + \epsilon$ (respectively $1 - \epsilon$) of being optimal with respect to x. Note that the time is not necessarily polynomial in ϵ, so that a running time of $O(n^{\frac{1}{\epsilon}})$ would qualify [Aus+99, Definition 3.10].

formulations into weighted MaxSAT instances and use the state of the art solver MaxHS[7] [DB13] to get the exact solution on instances with at most 1000 nodes in less than a few hours.

There is also a fixed parameter algorithm to solve MINDISAGREE for general unweighted graphs. Indeed, we shall see later than the problem is equivalent to k-MINIMUM MULTICUT, which in this case asks for the minimal number d of edges to remove in order to separate every pair of nodes in $S = \{(s_i, t_i)\}_{i=1}^{k} \in V^2$. It turns out that k-MINIMUM MULTICUT can be solved exactly in time $2^{O(d^3)} \cdot n^{O(1)}$ [Dán14].

After defining the matrix A by $A_{uv} = w^+_{uv} - w^-_{uv}$, Veldt *et al.* [VWG17] show that the MAXAGREE problem can be written up to constant factor by defining one vector x_u per node as:

$$\begin{aligned} \text{maximize} \quad & \sum_{u<v} A_{uv} x_u^T x_v \\ \text{subject to} \quad & x_{uw} \in \{e_1, \dots, e_n\} \end{aligned} \tag{3.5}$$

where e_i are the canonical vectors of $\mathbb{R}^n$. They show that when the matrix A is positive semidefinite of rank k, the CORRELATION CLUSTERING problem can be solved exactly in $O(n^{k^2})$ time. More practically, they also give an algorithm that closely approximate the objective of (3.5) in $O(nk)$ time.

Reflecting on these methods, we can conclude that CORRELATION CLUSTERING is solvable in (near) polynomial time when the problem instance follow our learning bias. Indeed, fixed parameter algorithms are efficient when the number of disagreements is low, which corresponds to a quasi consistent clustering. Likewise, the small rank assumption of the adjacency matrix is equivalent to assuming a consistent clustering [Chi+14, Theorem 13]. Even when solving integer programs, one can use the bias to add inequalities and thus tighten the polytope of admissible solutions. On the other hand, we will now see that in the worst case, for example if an instance does not obey our bias, then there is little hope to obtain an optimal solution.

3.2.2.2 Hardness and approximations

Although the same problem was considered earlier [DM96; BSY99], Bansal *et al.* [BBC02] coin the term CORRELATION CLUSTERING and are the first to study this problem complexity. Namely, for complete, unweighted signed graphs, they show that both MINDISAGREE and MAXAGREE are NP-COMPLETE. Along the way, they give a 17429-approximation of MINDISAGREE and a PTAS that, for any $\epsilon \in [0, 1]$, runs in $O(n^2 e^{O(\epsilon^{-10} \log \epsilon^{-1})})$ and returns with probability $1 - \epsilon/3$ a solution with at most ϵn^2 fewer agreements than the optimal solution of MAXAGREE.

The next year, several authors independently strengthened these results and extended them to weighted and general graphs, as summarized in Table 3.4. In the most complete paper, Charikar *et al.* [CGW03] show that on complete graphs, in sharp contrast with MAXAGREE that admits a PTAS, MINDISAGREE is APX-HARD, "that is, is NP-hard to approximate within some constant factor greater than one". Unfortunately, it is not easy to give intuition why, for this follows from a "somewhat intricate reduction from max 2-colorable subgraph problem on bounded degree 3-uniform hypergraphs". On general graphs though, both versions of the optimization problem are APX-HARD. It is showed for MAXAGREE by a reduction from MAX 3SAT [CGW03, Theorem 9] and for MINDISAGREE by using a reduction from the k-MINIMUM MULTICUT problem [CGW03, Theorem 8]. In the case of MINDISAGREE, Emanuel *et al.* [EF03] show a reduction in the other direction. Because k-MINIMUM MULTICUT is an interesting graph cut problem, and

[7] http://maxhs.org

Table 3.4 – Hardness results of CORRELATION CLUSTERING

	MINDISAGREE		MAXAGREE	
graph	weighted	unweighted	weighted	unweighted
Complete	APX-HARD [CGW03]	NP-COMPLETE [BBC02], APX-HARD [CGW03]		NP-COMPLETE [BBC02]
General	APX-HARD [CGW03; DI03]	APX-HARD [CGW03; EF03]	APX-HARD [CGW03, Thm. 9]	

its complexity has been well studied, we here give more details on its equivalence with CORRELATION CLUSTERING.

For that we follow [Dem+06] but omit the full proofs to avoid redundancy. In k-MINIMUM MULTICUT, given a weighted graph $H = (V_H, E_H, w_H)$ and a collection of k sources and targets $S = \{(s_i, t_i)\}_{i=1}^{k} \in V_H^2$, we want to find the lightest set of edges $T \in E_H$ whose removal disconnects every pair s_i from the corresponding t_i. Let us first describe a polynomial transformation from k-MINIMUM MULTICUT to CORRELATION CLUSTERING. Namely, given an instance (H, S) of k-MINIMUM MULTICUT, we let $W_H = \sum_{e \in E_H} w_H(e)$ be the total weight of H and G_H be the same graph as H with all its edges labeled positively. We then connect every pair (s_i, t_i) by a negative edge of weight $W_H + 1$. One can check ([Dem+06, Theorem 4.7]) that a multicut T in H with weight W induces a clustering in G_H of weighted disagreement W by the connected components of $G^+ \setminus T$. Likewise, a clustering of G_H with weight W induces a multicut on H with weight as most W. In the unweighted case, the reduction is similar but the heavy negative edges are simulated in the following way: include every source and target in a clique of n nodes and connect that clique to the corresponding source or target by n negative edges.

We next present the polynomial transformation from CORRELATION CLUSTERING to k-MINIMUM MULTICUT. Given $G = (V, (E^+, E^-), w)$ we let H_G be the graph induced by E^+ with the same weight. Then for every negative edge $(u, v) \in E^-$ of weight w^-_{uv}, we create a new vertex $v_{\widehat{uv}}$, connect u to $v_{\widehat{uv}}$ with weight w^-_{uv} and let $(v_{\widehat{uv}}, v)$ be a source-target pair added to S_G. This construction is depicted in Figure 3.4 and one can show ([Dem+06, Theorem 4.4]) that it takes a linear time to construct a multicut of weight W in H_G from a clustering of weight W in G, and vice versa.

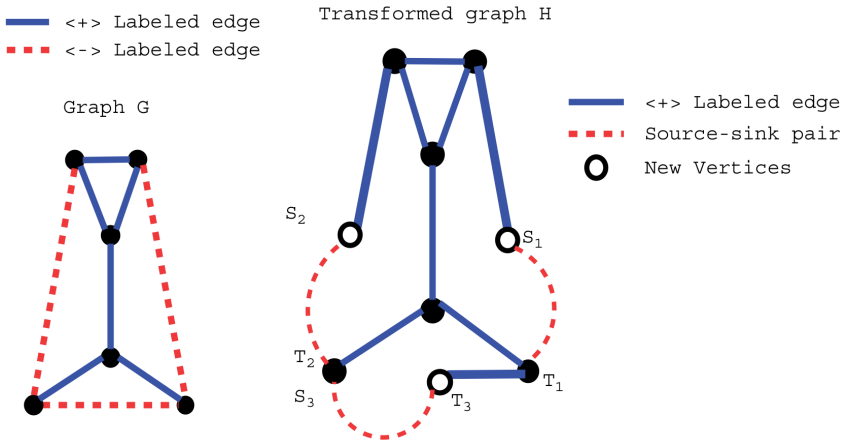


Figure 3.4 – The transformation from CORRELATION CLUSTERING on G to k-MINIMUM MULTICUT on H (reproduced from [Dem+06])

If a certain conjecture in computational complexity is true[8], then for every constant $c > 0$, it is NP-HARD to approximate k-MINIMUM MULTICUT within

[8] Namely, the Unique Game Conjecture of Khot [Kho02].

a factor c [Cha+06]. On general graphs, the best known approximation factor in $O(\log k)$ [GVY93], which according to the presented reduction translates to $O(\log |E^-|) = O(\log n^2) = O(\log n)$. This approximation ratio is achieved by first solving the relaxation of the integer problem (3.3), where the constraint (3.4) is replaced by $x_{uv} \in [0, 1]$. From now on, we refer to that relaxation as the *canonical* MINDISAGREE *LP*. Once this canonical MINDISAGREE LP is solved, we interpret x_{uv} as a distance: the larger it is and the more we want u and v to be in different clusters. We can then use the REGIONGROWING method [GVY93]. The idea is to pick a random center u and to add to u's cluster all the nodes at distance less than r from u before removing that cluster and repeating the process. $r < 1/2$ is chosen adaptively such that the weight of positive edges leaving the cluster is less than $c \log(n + 1)$ times the fractional weights of the positive edges inside the cluster.

However, Charikar *et al.* [CGW03, Theorem 2] note that the LP formulation has a poor integrality gap when it comes to MAXAGREE, thus they turn to a Semi Definite Program. Say that each cluster is associated with a basis vector, then for each node u in a cluster, we set a_u to be the corresponding basis vector. If u and v are in the same cluster, we then have $a_u \cdot a_v = 1$ while if they belong to different clusters, $a_u \cdot a_v = 0$. The weighted number of agreements can then be represented by

$$\begin{aligned} \text{maximize} \quad & \sum_{(u,v)\in E^+} w_{uv}(a_u \cdot a_v) + \sum_{(u,v)\in E^-} w_{uv}(1 - a_u \cdot a_v) \\ \text{subject to} \quad & a_u \cdot a_u = 1 \\ & a_u \cdot a_v \geq 0 \end{aligned} \tag{3.6}$$

After solving the SDP, a clustering can be obtained by a general rounding technique H_t: pick t random hyperplanes and divide the nodes in 2^t clusters. Charikar *et al.* [CGW03, Theorem 3] prove that taking the best results of H_2 and H_3 gives in a 0.7664 approximation on general graph. This was slightly improved to 0.7666 by Swamy [Swa04] with a different rounding: pick k random unit vectors (called *spokes*) and assign each a_u to the closest spoke.

Combining MINDISAGREE and MAXAGREE, Charikar *et al.* [CW04, Section 4] give a $\Omega(\frac{1}{\log n})$ approximation of the MAXCORR problem, which is maximizing (3.1) $-$ (3.2) and can be formulated as a quadratic programming problem solved in polynomial time.

In complete graphs, Charikar *et al.* [CGW03, Section 3] also give an improved 4-approximation to MINDISAGREE, by rounding the same LP and using a simpler version of REGIONGROWING with fixed radius. Namely, we pick a ball center u uniformly at random with radius $1/2$: if the average distance of the nodes in the ball to u is less than $1/4$, the ball forms a cluster, otherwise $\{u\}$ forms a singleton cluster. We then remove the corresponding nodes from the graph and repeat until all nodes are clustered.

Not long after, Ailon *et al.* [ACN05] came up with a better approximation. To explain it, we first describe their randomized combinatorial algorithm KWIKCLUSTER, which gives a 3-approximation of MINDISAGREE on complete unweighted graphs (and was later derandomized while preserving its approximation guarantee [vZW08]). At each iteration, we pick a node u uniformly at random (called the pivot) and we create a cluster containing u and all its neighbors linked by a positive edges. On weighted complete graphs, they tweak this algorithm by using the solution of the canonical LP to obtain different approximation factor depending on the constraints imposed on the weight. Recall that in the general formulation of the problem, each edge carries two positive numbers: $w^+_{u,v}$ and $w^-_{u,v}$. If the weights respect the probability constraints stating that for all edge (u, v) in E, $w^+_{u,v} + w^-_{u,v} = 1$, this tweaking provide a 2.5-approximation. Note that unweighted graphs naturally fit into that case, as each edge is either labeled $+$ or $-$. If the weights additionally respect the triangular inequality constraints stating that $w^-_{u,v} \leq w^-_{u,w} + w^-_{w,v}$, this

Table 3.5 – Best current results on CORRELATION CLUSTERING problems. The "easiest" setting is MAXAGREE on complete graphs, for it admits PTASs. All others cases are APX-HARD. However, we see that on the diagonal (that is MINDISAGREE on complete graphs and MAXAGREE on general graphs), there exists constant factor approximations. This is not the case for the most "difficult" problem, MINDISAGREE on general graphs.

G	MINDISAGREE	MAXAGREE
Complete	2.06 (and 1.5 if G is weighted and weights obey the triangular inequality) [Cha+15]	PTAS from Bansal *et al.* [BBC02] and by setting $k = \Omega(1/\epsilon)$ in [GG06]
General	$O(\log n)$ [CGW03]	0.7666 [Swa04]

become a 2-approximation. After solving the canonical LP with additional probability constraints, when picking a node u, each of its neighbors v is added to the cluster of u with probability x_{uv}. Chawla *et al.* [Cha+15] improve these two factors to respectively 2.06 and 1.5 by exploiting the same idea but setting the probability to include each neighbor v of u in the cluster of u to be $1 - f^+(x_{uv})$ if $(u, v) \in E^+$ and $1 - f^-(x_{uv})$ if $(u, v) \in E^-$, with a careful choice of f^+ and f^-. They also give a derandomized version of their algorithm in the full version of the paper [Cha+14, Theorem 23].

This concludes the presentation of the known approximation results on CORRELATION CLUSTERING, that we gather in Table 3.5. To summarize, except for maximizing the number of agreements on a complete graphs, we cannot hope for exact solutions for the other problems in the worst case, since they are APX-HARD. Moreover, even constant factor approximation are ruled out for MINDISAGREE on general graphs, as showed by the equivalence with k-MINIMUM MULTICUT. Currently there are three main approaches to approximate MINDISAGREE. First, on a general graph, we can solve the canonical LP, and use the REGIONGROWING method to round the fractional solution, resulting in an $O(\log n)$ approximation. Second, on complete graphs, the simplest solution is the KWIKCLUSTER algorithm, that grows clusters out of randomly selected pivots, in a way reminiscent of the proof of Theorem 6. Third, the approximation factor of this general idea can be improved by interpreting the fractional solution of the linear program as probability to be included in the pivot's cluster. The cost of solving linear programs with so many variables and constraints being prohibitive, real world general graphs requires to move from methods with guarantees to more heuristic approaches we describe next. Before that, we make two remarks. The first one regards a line of work improving scalability of these approximations by presenting distributed solutions. The second one is concerned with the case where the number of clusters is set beforehand, and proves that the problem becomes easier.

> To handle the massive size of some large real world dataset, parallelizing existing algorithms is a natural approach. Due to its simplicity and approximation guarantee, KWIKCLUSTER has been adapted three times for that purpose. First, Chierichetti *et al.* [CDK14] describe how to uniformly sample several pivots in the same round, remove pivots that are adjacent through positive edges and then grow the corresponding clusters with potential conflicts solved according to the node order in a global permutation drawn at the beginning of the algorithm. On general graphs, this requires $O(\log n \operatorname{diam}(G))$ rounds, which on complete graphs reduces to $O(\log n)$. Furthermore, this almost preserves the approximation factor, which is $3 + \frac{14\epsilon}{1-7\epsilon}$, where ϵ is a parameter smaller than 1. Finally, this allows to experimentally cluster graphs with millions of nodes and billions of edges. Second, Pan *et al.* [Pan+15] describe an equivalent version of KWIKCLUSTER where a permutation π of the nodes is drawn at the start of the algorithm and the pivots are chosen sequentially in the order of π. The exact same partition can be obtained when

several pivots are chosen at the same time by different threads as long as they respect two concurrency rules: (i) two nodes u and v can become pivot at the same time if they are not connected with a positive edge, otherwise only the one with the smallest index in π becomes pivot; (ii) if w is a positive neighbor of two pivots u and v, it is affected to the cluster of the pivot with the smallest index in π. Enforcing these rules preserves the factor 3 approximation and the algorithms terminate after $O(\log n \operatorname{diam}(G))$ rounds. The authors also present a version where each round is faster as it does not enforce rule (i). However, this weaken the approximation guarantee to $(3+\epsilon)OPT + O(\epsilon n \log^2 n)$. In practice, the solution is very close the one of KWIKCLUSTER, although it degrades as the number of threads increases.

The number of rounds preserving the 3-approximation is lowered to $O(\log \log n)$ by Ahn *et al.* [Ahn+15]. They also present results obtained in a single pass, which corresponds to the streaming model: the algorithm receives the edges of G one by one and upon seeing the last one outputs its result. The additional constraint is that this algorithm can only use $O(n \operatorname{polylog} n)$ space. In this setting, the authors show a polynomial time $(1-\epsilon)$-approximation of MAXAGREE if the weights are bounded (and $0.766(1-\epsilon)$ if the weights are arbitrary); and an $O(\log |E^-|)$ approximation of MINDISAGREE in polynomial time with arbitrary weights. This is done by combining graph sketching and a method to solve convex programs in a space efficient manner.

Bonchi *et al.* [BGK13] suggest a different paradigm to solve CORRELATION CLUSTERING, that can be applied in a distributed setting to obtain a scalable approach. Namely, given a node u, they want to output a globally consistent cluster index $\mathcal{C}(u)$ while making at most t queries to a sign oracle. Here t is a parameter that depends on the quality of clustering produced but not on the size of the graph. And because this procedure is local to each node, it can be run in parallel. Finally, one can get a full clustering by computing $\mathcal{C}(u)$ for all the nodes in the graph. Despite the problem being apparently more challenging, they obtain approximation factors that are close to the best known (which in this model would make $\Omega(n^2)$ total queries). More precisely, they use two techniques. The first one is inspired by KWIKCLUSTER. It starts by finding a good set of pivots, seeing the problem as finding a maximal independent set on a sampled part of the positive graph. Then, for a given node, it finds the closest such pivot or creates a singleton. Given a quality parameter $\epsilon \in (0,1)$, it yields a $4 \cdot OPT + \epsilon n^2$ approximation of MINDISAGREE requiring $O(\frac{n}{\epsilon})$ time and queries [BGK13, Theorem 3.3]. Roughly stated, the second technique relies on an existing low-rank approximation of the adjacency matrix, that partition the graph into similar sized classes such that edges between those classes behave as in a random graph. This initial partition is "coarsened" into a good clustering by considering all possible ways of assigning classes to clusters. It gives an $OPT + \epsilon n^2$ additive approximation for MINDISAGREE that runs in time $n \cdot poly(1/\epsilon) + 2^{poly(1/\epsilon)}$ [BGK13, Corollary 3.7].

As mentioned earlier, not having to set the number of clusters is an attractive feature of the CORRELATION CLUSTERING problem, but in some cases we may want to use prior knowledge. The problem was studied by Giotis *et al.* [GG06] on general graphs and we compile their results in Table 3.6. On complete unweighted graphs and with K being the number of clusters, they provide PTASs for MAXAGREE$[K]$ running in $nk^{O(\epsilon^{-3}\log(\frac{K}{\epsilon}))}$ time and for MINDISAGREE$[K]$ running in $n^{O(\epsilon^{-2}9^K)}\log(n)$ time. The latter was improved by Karpinski *et al.* [KS09], with a PTAS running in $n^2 2^{O(\epsilon^{-3}K^6 \log d)}$ and that can handle weighted graphs. For $K=2$, MINDISAGREE$[2]$ admits a faster local search method with a factor 2 approximation [CSW08a]. For complete weighted graph, Bonizzoni *et al.* [Bon+08a] provide a PTAS for MAXAGREE$[K]$ under the condition that the ratio between the largest and smallest weights is bounded by a constant.

Table 3.6 – Approximation results for CORRELATION CLUSTERING on general graphs with K clusters

K	2	≥ 3
MAXAGREE	0.878 (improved to 0.884 by [MS09])	0.7666 [Swa04] (and it can be proved that this creates at most 6 clusters)
MINDISAGREE	$O(\sqrt{\log n})$ as it reduces to Min 2CNF Deletion for which Agarwal *et al.* [Aga+05] give such an approximation	this can be reduced from K-coloring, which for any $\epsilon > 0$ is NP-COMPLETE to approximate within $n^{1-\epsilon}$ [Zuc07]

3.2.2.3 Heuristics

While all the methods we discuss so far are either exact or come with some approximation guarantees, practitioners have also develop approaches that are designed to efficiently reach a solution that is satisfactory enough for the application at hand.

Greedy methods Several of such methods fall into the greedy framework. For instance, while studying small scale signed social networks, Doreian *et al.* [DM96] describe the following procedure. Start with a random initial k clustering of the graphs and for T steps, sample randomly P neighboring partitions, compute their number of disagreements and move the one with the least disagreements. Neighboring partitions are obtained either by moving one node from cluster to another or by exchanging a pair of nodes between two clusters. The overall complexity is $O(nTP)$. This is quite similar to the *Best One Element Move* described in [GMT07], except that there the exchange operation is replaced by moving one node to its own singleton cluster. The *Cluster Affinity Search Technique* algorithm [BSY99] instead grows clusters one by one by maintaining for the current cluster the affinity of all nodes, which is the sum of the weights between that node and the nodes in the cluster. Nodes above a certain threshold are added to the current cluster and nodes below the threshold are removed, with the affinity to the current cluster being recomputed after each addition or deletion.

After defining the net weight of an edge to be $w^{\pm}_{uv} = w^{+}_{uv} - w^{-}_{uv}$, Elsner *et al.* [ES09] describe three folklore heuristics that start with empty clusters and add node one by one: "The BEST algorithm adds each node u to the cluster with the strongest $w^{\pm}$ connecting to u, or to a new singleton if none of the $w^{\pm}$ are positive. The FIRST algorithm adds each node u to the cluster containing the most recently considered node v with $w^{\pm}_{uv} > 0$. The VOTE algorithm adds each node to the cluster that minimizes the CORRELATION CLUSTERING objective, i.e. to the cluster maximizing the total net weight or to a singleton if no total is positive." Empirically, VOTE turns out to be the best. Among other related heuristics, Lingas *et al.* [LPS14] describe the random maximum merging algorithm, that starts with singleton clusters and keep merging two clusters chosen at random among those whose merge would result in the maximum improvement of the score function. This runs in $O(n^2 \log n)$, and empirically results in fewer disagreements than KWIKCLUSTER. Building upon their previous GRASP work [Dru+13], an iterated local search (ILS) heuristic is presented in [Lev+15; Lev+17]. Each iteration of this algorithm starts by greedily building a clustering in a fashion similar to VOTE, albeit with more randomness in the node ordering. This clustering is locally improved by moving blocks of $r \in \{1, 2\}$ nodes from one cluster to another as long as the number of disagreements decreases, a phase called neighborhood descent. The algorithm then enters an inner loop where the current clustering is perturbed by t random one-node-moves and updated if a subsequent neighborhood descent can improve it compared with before the perturbation. The authors note that both the outer loop and the neighborhood descent can be run in parallel, which allow them to process a 10 000 nodes graph on 10 cores in around 700 seconds. Bastos *et al.* [Bas+16] show how to use ILS to generate initial solutions of the Cluster Editing problem that are then fed to an integer program. Instead of moving nodes, Wang *et al.* [WL13] starts from the observation that in a balanced graph, G^{+} is a disjoint union of cliques in which

all the nodes of a given cluster share the shame neighbors. Their algorithm is initialized with the set E_s^+ of edges where $w_{uv}^+ > w_{uv}^-$, and repeatedly samples an edge (u, v) from E_s^+ before trying to make the neighborhoods of u and v coincide by adding or removing edges in E_s^+. Yet another idea is to modify the canonical LP to replace the binary variable x_{uv} by a k clusters indicator matrix $L \in \mathbb{R}^{k \times n}$ where L_{iu} is 1 is $u \in C_i$ and -1 otherwise [Wan+13]. Indeed, $x_{u\cdot} \equiv L_{\mathcal{C}(u)\cdot}$, where $\mathcal{C}$ is the cluster assignment. By relaxing the integer constraints on L, it is possible to do alternate optimization on L and $\mathcal{C}$.

Physics inspired energy methods Here we describe a physics particle model that can readily be adapted to model CORRELATION CLUSTERING and provide additional heuristic methods. The Potts model [Wu82] describes a general model of *spins* organized in a lattice and being in one of k possible states. A spin u interacts with each of its neighbors v through a coupling J_{uv}. The energy of this system, called the Hamiltonian, is defined by $\mathcal{H}(\boldsymbol{\sigma}) = -\sum_{u,v} J_{uv}\delta(\sigma_u, \sigma_v)$ where $\boldsymbol{\sigma}$ is the spin configuration (that is, $\sigma_u \in \{1, \dots, k\}\, \forall u$) and $\delta(\sigma_u, \sigma_v)$ is equal to one if u and v are in the same state and zero otherwise. It is a general principle of physics that isolated systems tend to minimize their energy, which in this case amounts to finding a spin configuration minimizing the Hamiltonian. Viewing the spins as nodes of a graph, the couplings as the graph weighted edges and the k possible states as clusters, it is quite natural to formulate the clustering problem as a Hamiltonian minimization problem [RB06].

Letting A be the matrix such that $A_{uv} = w_{uv}^+ - w_{uv}^-$ on a general directed graph, Traag *et al.* [TB09] reward positive and absent negative links within cluster and penalize negative and absent positive links across clusters to come up with the following Hamiltonian: $\mathcal{H}(\boldsymbol{\sigma}) = -\sum_{u,v} \left(A_{uv} - (\gamma^+ p_{uv}^+ - \gamma^- p_{uv}^-)\right) \delta(\sigma_u, \sigma_v)$ where γ^+ and γ^- are user parameters and $p_{uv}^{\pm}$ are null model probabilities, which are equal to $p_{uv}^{\pm} = \frac{|E^{\pm}|}{|V|(|V|-1)}$, or $p_{uv}^{\pm} = \frac{d_{\text{out}}^{\pm}(u) d_{\text{in}}^{\pm}(v)}{|E^{\pm}|}$ in the degree corrected model. They note that setting $\gamma^+ = 0 = \gamma^-$ make minimizing the Hamiltonian equivalent to the MINDISAGREE problem, which they do by simulated annealing [KGV83]. The idea is to start from a random partition, and jump to another partition by moving a single node from one cluster to another. Such moves are made with a probability proportional to how each move reduces the Hamiltonian and as the procedure goes on, large jumps are made less and less probable by reducing a parameter called the temperature. One advantage of this energy formulation is that it requires only one variable per node instead of one variable per edge as in the case of the linear program.

In the case of 2-CORRELATION CLUSTERING, Facchetti *et al.* [FIA11] rewrite the Hamiltonian as $\mathcal{H}(\boldsymbol{\sigma}) = -\frac{1}{2}\boldsymbol{\sigma}^T A \boldsymbol{\sigma} = -\frac{1}{2}\mathbf{1}^T T_\sigma A T_\sigma \mathbf{1}$. There $T_\sigma = diag(\boldsymbol{\sigma})$ (where $\boldsymbol{\sigma} \in \{0, 1\}^n$) is the outcome of a local search algorithm such that $A_\sigma = T_\sigma A T_\sigma$ has the same number of disagreements as A but the smallest number of negative edges. This is called a gauge transformation in the spin glass literature and the benefit of that heuristic is that it scales gracefully to large graphs. Bagon *et al.* [BG11] also write the MAXAGREE objective as a Potts model, and show that it can be interpreted as the log posterior of a partition matrix under a simple generative model and as a pair-wise conditional random field energy without unary terms. This allows them to adapt existing discrete energy optimization algorithms in order to cope with the following three challenges of the CORRELATION CLUSTERING energy: "(i) the energy is non sub-modular, (ii) the number of clusters is not known in advance, and (iii) there is no unary term". Doing so, they are able to handle large problems with more than 100K nodes. Also adopting an energy minimization approach, Kappes *et al.* [Kap+16] assign a probability to each cut of a signed graph proportional to the exponential of the number of disagreements of that cut. They also develop efficient cut sampling methods. Several of these methods have recently been evaluated empirically by Levinkov *et al.* [LKA17].

3.2.2.4 Active and online settings

Except for the local algorithms of Bonchi *et al.* [BGK13], all works presented so far considered the batch case of CORRELATION CLUSTERING, where the whole graph and all the signs are available at all time and with no cost. This might not always be the case, for instance if the graph is too large to fit in memory or if the edge labels are given by an expensive external procedure.

Case in point, in the context of entity resolution, Kanani *et al.* [KMP07] consider the CORRELATION CLUSTERING problem where, by querying an oracle, one can either reduce the uncertainty about one edge weight or add an extra node with edges connecting it to existing nodes. However, this requires web queries that are resource bounded and thus yields an active setting learning problem, where one has to choose the most informative queries given a budget. A formal definition is given in [KM07] and in practice, after finding an initially good partition, they select in each cluster a node to be its centroid, and query the edges connecting the centroid as ordered by an entropy-based criterion. In the case the oracle answering the queries is consistent, there exists an information theoretic bound on the number of queries needed to recover the clusters, and an algorithm matching that bound up to a $O(\log n)$ factor [MS17]. Ailon *et al.* [ABE14, Section 5] present another active algorithm that solves MINDISAGREE$[k]$ with a linear number of queries but an exponential running time in n. In a noisy setting, Mitzenmacher *et al.* [MT16] assume there is a planted k-partition and that we have access to an oracle that for $u, v \in V^2$ returns $\mathcal{C}(u) - \mathcal{C}(v) \mod k$ with probability $1-p$ and a noisy $\mathcal{C}(u) - \mathcal{C}(v) \pm 1 \mod k$ otherwise. For $k = 2$, this is exactly 2-CORRELATION CLUSTERING, whereas for $k > 2$, the oracle gives more information than simply the sign of the path from u to v. Whenever $p < 1/2$, they show that $O(n^{1+\frac{1}{\log\log n}} \log n)$ queries are enough to recover the planted partition in polynomial time and with high probability. Those queries are actually random (i.e. non-adaptive), and the clusters are found by looking at almost edge-disjoint path between all pairs of nodes.

In the online setting, Mathieu *et al.* [MSS10] give a greedy algorithm that upon node arrival creates a singleton cluster and then merge all pairs of clusters for which it increases the total number of agreements. For MINDISAGREE, this is $O(n)$-competitive algorithm and they show such ratio is optimal by exhibiting an instance[9] on which any strategy ends up with $n-k$ disagreements whereas optimal cost is k. On the MAXAGREE side, this greedy strategy result in a $1/2$-competitive algorithm. If it is randomly mixed with a DENSE variation, it increases to $1/2 + \eta$, still far from the demonstrated 0.834 upper bound.

This state of the art depicts a nuanced landscape for the CORRELATION CLUSTERING problem. On one hand, minimizing the disagreements is difficult, although it is the most natural way given our bias. On the other hand, it can be solved with approximation guarantees, at least in theory. Indeed, the best methods require solving linear programs with m variables and up to $O(n^3)$ constraints, which results in a high complexity of $O(m^{4.5})$ [MG07, Section 7.2]. Fortunately, we can obtain reasonable solutions in practice, thanks to efficient heuristics (using the output of KWIKCLUSTER as initialization for instance). Furthermore, instances obeying our bias can be solved more easily and almost exactly. We next present settings where instances are indeed close to our bias. Before that, we remark that while KWIKCLUSTER is often used on general graphs in practice (by assuming the missing edges are negative), doing so does not preserve its approximation guarantee.

[9] Namely, two positive cliques A and B joined by positive edges except between $a \in A$ and $\{b_1, \ldots, b_k\} \in B$. Those nodes are given first and thus form a cluster, which yields at least one disagreement every time one the $n-(k+1)$ remaining nodes is added.

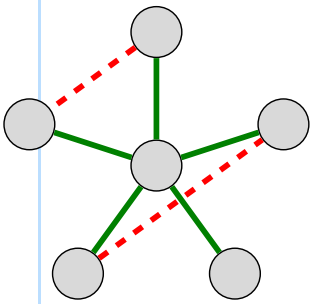


Figure 3.5 – A positive star with few negative edges

Consider the graph G showed in Figure 3.5. Assume there are n nodes connected positively to the center of the star, and $k < n - 1$ negative edges between the peripheral nodes. We define G' as the complete graph obtained from G by setting all missing edges to be negative. On G', the optimal solution is to have a cluster with the center of the star and one peripheral node, while others peripheral nodes are put in singleton clusters. This creates $n - 1$ positive disagreements. When running KWIKCLUSTER on G', if the first pivot is a peripheral node, this is the solution we obtain. Otherwise, with probability $1/n$, the center of the star is chosen as pivot, creating a single big cluster that incurs $(1/2)(n-1)(n-2)$ negative disagreements. In expectation, KWIKCLUSTER thus achieves

$$\frac{1}{n}\left(\frac{(n-1)(n-2)}{2} + (n-1)^2\right) = \frac{3n-4}{2n}(n-1) \sim \frac{3}{2}n$$

disagreements, which is roughly a $3/2$-approximation as n grows larger.

Now let us get back to the original G. Here the optimal solution is to have a single big cluster, which incurs k negative disagreements. With the same reasoning as before, the expected number of disagreements of KWIKCLUSTER is now $\frac{1}{n}\left(k + (n-1)^2\right)$. If for instance $k = \lfloor\sqrt{n}\rfloor$ (as in Figure 3.5), then the previous expression is equivalent to n as n tends to infinity. KWIKCLUSTER is thus $\sqrt{n}$ worst in expectation than the optimal when run on this general graph.

3.2.3 Beyond worst case instances

We now look at two settings where our bias is likely to be present. In the first one, we assume the input signs are randomly perturbed of an initial assignment derived from a consistent clustering. In the second one, we do not assume anything about such a consistent clustering but expect instead that the optimal solution is "clear" enough that it does not change when weights are multiplicatively perturbed.

CORRELATION CLUSTERING under noise As we have seen, assuming the Unique Games Conjecture, the k-MINIMUM MULTICUT problem, and therefore the CORRELATION CLUSTERING problem on a general graph, cannot be approximated to within a constant factor in the worst case. But maybe we can do better in the average case, which motivates the study of semi-random model, where real graphs are seen as being obtained from the controlled perturbation of a perfectly clusterable graph. In the simplest case, each edge sign is independently flipped with probability $p \in [0, 1/2)$. This situation on complete graphs was considered in [BBC02, Section 6], showing a simple algorithm that with high probability makes $\tilde{O}(n^{\frac{3}{2}})$ mistakes, and in [BSY99, Theorem 2.6], with an algorithm recovering with high probability the planted partition of an unweighted graph in $O(n^2(\log n)^c)$, where c depends on the size of the smallest cluster and on the noise probability.

Joachims *et al.* [JH05] analyze a more refined weighted model, where weights are generated by a probability distribution whose mean on true positive edges is larger than $\mu^+ > 0$ and whose mean on true negative edges is smaller than $\mu^- < 0$. They give a finite-sample bound on the number of nodes misclustered w.r.t the planted partition as a function on the probability distribution parameters. Indeed, as pointed out by Ailon *et al.* [AL09], uniform and independent noise is not a good model of real situations, where the input of CORRELATION CLUSTERING, i.e. the similarity between nodes, is often the result of a preprocessing, which may present strong correlations. Therefore, instead of measuring the quality of a solution against the input (i.e. the similarity information), they argue it is more sensible to measure it against the (unknown) true optimal clustering that gave rise to the input, and show that KWIKCLUSTER allows this, thanks to a new analysis. Mathieu *et al.* [MS10] also consider an adversarial model, where all edges are flipped with probability p

but the adversary then decides whether to reveal the true sign or the flipped sign. On complete unweighted graphs, they find a solution of MINDISAGREE at most $1 + O(n^{-1/6})$ times the optimal whenever $p \leq 1/2 - n^{-1/3}$ by rounding the usual SDP solution. If, in addition, there is no adversary, $p \leq 1/3$ and each planted cluster has at least $\Theta(\sqrt{n})$ nodes, then the planted partition can be recovered exactly. Makarychev *et al.* [MMV15] study the same adversarial model but on general weighted graphs, giving a PTAS for MINDISAGREE when $p \leq 1/4$. Under additional assumptions on the density of edges, they present another algorithm that finds the ground truth clustering with an arbitrarily small classification error.

CORRELATION CLUSTERING under stability assumption Whereas clustering objective functions are NP-HARD to optimize, we expect meaningful instances in which we are interested to have additional structure which allows for guaranteed polynomial time algorithms. We refer the reader to a critical overview [Ben15] of some such notions of structure proposed recently. Informally, a general idea is that the clustering should not change (or at least very little) if the data are slightly perturbed. For instance, a weighted graph is α-*stable* (with $\alpha > 1$) for some partition objective if its optimal partition remains the same whenever every weight w_i is multiplied by a factor c_i between 1 and α. Another notion is the (c, ϵ)-approximation stability. Formally, a dataset X is (c, ϵ)-stable if, for an objective function Φ, any clustering whose cost is within a factor c of the optimal cost OPT_Φ is ϵ-close to the optimal clustering (as measured by the fraction of points on which the two clusterings disagree). When the objective is MINDISAGREE on complete graphs, Balcan *et al.* [BB09] show that for $(1 + \alpha, \epsilon)$-instances, the approximation algorithms we describe in Section 3.2.2.2 find a solution that is $(49/\alpha + 1)\epsilon$ close to the optimal clustering.

It is also possible to study the stability of a CORRELATION CLUSTERING instance over a general graph w.r.t edge weights via its canonical MINDISAGREE LP. Nowozin *et al.* [NJ09] present a method to determine to which extent the weights of a CORRELATION CLUSTERING instance can be perturbed before the optimal solution changes.

3.2.4 Variants and extensions

So far we focused on CORRELATION CLUSTERING in its rawest form, that is solving the MINDISAGREE and MAXAGREE objectives in the case where the general binary-labeled graph is known in advance. We will now see first some special cases, namely when the weights obey the triangle inequality (to solve CONSENSUS CLUSTERING) or when the graph is bipartite and then move to variants. We will consider more general objectives, when the edge labels are categorical instead of binary, when nodes can belong simultaneously to several clusters or when we optimize local objectives per nodes instead of global ones. Finally we will also look at clustering in signed graphs in general, using spectral methods or heuristics from the community detection literature.

CONSENSUS CLUSTERING In CONSENSUS CLUSTERING, the goal is to output a clustering which best summarizes (or agrees with) the several given input clusterings of the same set of objects. Motivations include robustness —by using an ensemble of clusterings from diverse methods— and privacy —if the clusterings were computed by different parties each considering only a subset of the objects attributes. We can build the complete graph of these objects, with weights set to the fraction of clusterings that place two objects in different clusters, thus representing a kind of distance in the space of clusterings. As first show by Filkov *et al.* [FS03] , finding the optimal clustering is therefore an instance of CORRELATION CLUSTERING where the weights obey the triangular inequality. Gionis *et al.* [GMT07] give a deterministic 3-approximation using the REGION-GROWING method. Later Bonizzoni *et al.* [Bon+08b] show that the minimization version is APX-HARD, even when the input is made of only three clusterings and give a com-

binatorial $4/5$-approximation for the maximization problem. Experimental evaluations are conducted by Bertolacci *et al.* [BW07] and Filkov *et al.* [FG08]. The former describe a scalable approach that first samples a small portion of the data, runs a (potentially computationally expensive) approximation algorithm and finally augment the resulting partition by adding to it the unsampled nodes one by one. Experiments confirm that the running time is greatly improved compared with the linear program methods while the resulting objective value is essentially the same. Note, however, that LP methods can be applied in practice thanks to some tricks [DSW10]. If we have k input clusterings $\mathcal{C}^1, \ldots, \mathcal{C}^k$ and we parameterized the problem by t, which is the sum over the input clusterings of the number of pairs of objects that are clustered differently by a solution $\mathcal{C}^\star$ and $\mathcal{C}^i$, then there is a polynomial algorithm running in $O(4.24^{t/k} \cdot t/k^3 + kn^2)$ [Dör+14].

Bipartite CORRELATION CLUSTERING Bipartite graphs are an interesting special case for CORRELATION CLUSTERING, as they often appear in the context of recommendation systems, where users rate products positively or negatively, although in this setting we cannot expect to have complete bipartite graphs in practice. The first results was given by Amit [Ami04], who obtains an 11-approximation for MINDISAGREE by adapting the REGIONGROWING method. The KWIKCLUSTER is adapted to the bipartite case by Ailon *et al.* [Ail+12], who prove it results in a randomized 4-approximation (and provide a matching deterministic approximation by rounding a LP). By using their idea of rounding the results of the LP differently for positive, negative and in that case same-side absent edges, Chawla *et al.* [Cha+15] bring down this approximation factor to 3, even for $K \geq 2$-partite graphs. Through formulating the MAXAGREE$[k]$ problem as a bilinear maximization problem and computing a low-rank approximation of the graph biadjacency matrix, Asteris *et al.* [Ast+16] obtain a efficient PTAS, that is a $(1-\delta)$ approximation running in time exponential in k and δ^{-1} but linear in n. By an appropriate choice of k, it is possible to use this PTAS to solve the general MAXAGREE problem. Beyond those results on CORRELATION CLUSTERING, we further discuss bipartite signed graphs in Section 5.2.1.

Categorical edge labelling In the so called CHROMATIC-CORRELATION CLUSTERING setting, "positive" edges are now associated with one of L possible colors and the goal is to form clusters mostly made up of edges with the one same color. Namely, a disagreement is now a negative edge between clusters or a within-cluster edge whose color differs from the majority color of that cluster. This is motivated by edge-labeled graphs in social networks, biology and citation networks and we will further discuss such applications in Chapter 4. As a generalisation of CORRELATION CLUSTERING, it is NP-COMPLETE but Bonchi *et al.* [Bon+12] present a modification of the KWIKCLUSTER algorithm that pick edges instead of nodes as pivots, and grow clusters by adding monochromatic triangles. This gives an approximation factor of six times the maximum degree of the graph. They also present a method when the number k of clusters is fixed beforehand, starting with an initial partition and then alternating between finding the majority color of clusters and finding better clusters. An improved heuristic algorithm is given in [GB16]. Unfortunately, the maximum degree of a graph can be as large as n. However, Anava *et al.* [AAG15] present constant factor approximations. Namely, they show that the problem can be reduced to classical CORRELATION CLUSTERING by setting all edges incident to a node u to negative if they are not of the majority color of u. They then apply the regular KWIKCLUSTER and show this gives an 11-approximation to the original problem. Furthermore, they also write a linear program and round it using the REGIONGROWING method to obtain an approximation factor of 4. Bonchi *et al.* [Bon+15] extend this line of work to the case were a single edge can carry a *set* of labels and adapt their randomized algorithm so that the approximation factor is multiplied by the size of the input label set.

Overlapping CORRELATION CLUSTERING While in CORRELATION CLUSTERING, each node is assigned to a single cluster, in other settings we may want to relax this constraint. Given a complete weighted graph, Bonchi *et al.* [BGU12] want to output a clustering $\mathcal{C}$ that minimizes the following cost:

$$\sum_{(u,v)\in E} |H(\mathcal{C}(u), \mathcal{C}(v)) - w_{uv}|$$

where H is a similarity function between two sets of labels, chosen in their paper to be the Jaccard similarity or a $0/1$ indicator of nonempty intersection. These problems are showed to be NP-COMPLETE, and approximated by a local search algorithm, iteratively optimizing the assignment of one node while all others are fixed. As one of the demonstration of their theoretical work, Johansson *et al.* [Joh+15] show a faster solution based on a weighted extension of the Lovász's theta function, the corresponding geometric embedding of graphs and a solver derived from one-class SVM, while Andrade *et al.* [And+14] propose a genetic algorithm to solve this problem. Finally, Rebagliati *et al.* [RRP13] also deals with overlapping clustering by relaxing the problem to a stochastic setting and using "the Baum-Eagon inequality, which provides an effective iterative scheme for maximizing polynomial functions in probability domains".

Local CORRELATION CLUSTERING In classical CORRELATION CLUSTERING, all nodes have an identical role, in the sense that they contribute equally to the final objective in terms of (dis)agreements. Here, we instead look at approaches where we either add a local penalty to each in order to better control their behavior, or where we altogether modify the objective to focus on (dis)agreements at specific nodes.

Puleo *et al.* [PM15] adapt the linear program of [CGW03] and its REGIONGROWING method to the case where all clusters have to contain less than K nodes, by assigning to each node u a penalty μ_u. If u is placed in a cluster C_i, the original MINDISAGREE objective is penalized by an extra $\mu_u \left(|C_i| - (K+1)\right)$. By varying μ_v between 0 and 1 and because the positive weights are assumed to smaller than 1, this cluster size constraint can be made hard or soft. They also handle more general weights, since they allow w^-_{uv} to be as large as τ for $\tau \in [1, \infty)$ while still guaranteeing a $5 - 1/\tau$-approximation on complete graphs, and adapt KWIKCLUSTER to unweighted graphs with the hard cluster size constraint, obtaining a randomized 7-approximation. These soft constraints are for instance used in a biological application where nodes are genes and where singleton and giant clusters are uninformative [Hou+16]

Puleo *et al.* [PM16] also modify the MINDISAGREE objective to make it more general. Based on the classic CORRELATION CLUSTERING linear program, they define a "*fractional clustering* of G as a vector x indexed by V such that $x_{uv} \in [0,1]$ for all $uv \in \binom{V}{2}$ and such that $x_{vz} \leq x_{vw} + x_{wz}$ for all distinct $v, w, z \in V$". They also define "the error vector $err(x)$ of x, as a real vector indexed by V whose coordinates are"

$$\mathrm{err}(x)_u = \sum_{v \in \mathcal{N}^+(u)} x_{uv} + \sum_{v \in \mathcal{N}^-(u)} (1 - x_{uv})$$

Given a function $f : \mathbb{R}^n_{\geq 0} \to \mathbb{R}$ verifying two elementary conditions, the problem is then to find a fractional clustering x minimizing $f(err(x))$. The classical CORRELATION CLUSTERING corresponds to setting $f(x) = \frac{1}{2}\ell^1(x)$ whereas the authors here are interested in Minimax CORRELATION CLUSTERING that arises by setting $f(x) = \ell^\infty(x)$. Minimizing the maximum number of disagreements incurred by a single node is motivated by the example of recommendation systems: if errors correspond to unsatisfying recommendations, we do not want a single user to suffer many of them. Minimax CORRELATION CLUSTERING is NP-COMPLETE on both complete graphs and complete bipartite graphs but by modifying the REGIONGROWING method, the authors respectively a 48 and 10 approximation, the latter for the one-sided error (that only counts disagreements for the nodes in one of the two clusters). The idea is to chose pivots not randomly but by maximizing a given criteria and to grow balls with a radius α computed numerically to optimize the approximation factor. Interestingly, and in contrast with the classic CORRELATION CLUSTERING situation, minimax MAXAGREE is not easier than minimax MINDISAGREE and seems not to have a constant factor approximation, even on complete graphs. Furthermore, these algorithms are deterministic, as opposed to many CORRELATION CLUSTERING approximations, since bounds on expected disagreements of an edge does not translate easily on their maximum. Charikar *et al.* [CGS17] improve these two factors to 7, using a simpler version of the algorithm of Puleo *et al.* [PM16]. Namely, find the ball of radius $1/7$ with the largest number node and create a cluster from its center with a radius of $3/7$. They also show that on general weighted graphs, the LP has a large integrality gap of $n/2$. Yet they combine it with a combinatorial approach to reach a $O(\sqrt{n})$ approximation. Finally, they consider the complementary problem of maximizing the minimum number of agreements at a single node, and provide a $\frac{1}{2+\epsilon}$ approximation.

Spectral Clustering A classic method for clustering graphs is to leverage their spectral properties. Namely, if A is the adjacency matrix of G and D its degree diagonal matrix (that is $D_{u,u} = \deg(u)$), the *Laplacian* of G, defined by $L_G = D - A$, is a symmetric positive semidefinite matrix. As such, it has n real non-negative eigenvalues, and its spectrum provides additional information on the connectivity of G. For instance, 0 is always the smallest eigenvalue and its multiplicity is equal to number of connected components of G, while —if G is connected— the second eigenvalue is the algebraic connectivity of G, whose magnitude is an indication of how well connected is the graph. This matrix is typically used for clustering by computing its first k eigenvectors, which embed the n nodes of G in $\mathbb{R}^k$, where there are then clustered with the k-means algorithm. This can be seen as a relaxation of the discrete RATIOCUT objective, which asks for the partition $\{C_1, \dots, C_k\}$ minimizing $\frac{1}{2}\sum_{i=1}^{k} \frac{\mathrm{cut}(C_i, \bar{C}_i)}{|C_i|}$, where $\bar{C}_i$ is the complement of C_i in V and $\mathrm{cut}(B, C) = \sum_{u \in B, v \in C} w_{uv}$ is the total weight of the edges between B and C [vLux07]. By considering the symmetric normalized Laplacian $L_{sym} = D^{1/2} L D^{1/2}$, it is also possible to approximate the normalized cut objective (NCUT), where $|C_i|$ is replaced by $vol(C_i) = \sum_{u \in C_i} \deg(u)$. We will now see how these kinds of approaches can be extended to signed graphs, noting first that they require to fix the number of clusters beforehand and are looking for clusters balanced in size, which makes the problem related but not equivalent to CORRELATION CLUSTERING.

The first line of research consider only MINDISAGREE[2]. For instance, Coleman *et al.* [CSW08b] show that both normalized cut and MINDISAGREE[2] objectives can be written as a SDP (or equivalently as eigenvalue problems) and thus combined, the intuition being that we look for NCUT solutions whose number of disagreements is not too much more than the approximate optimal.

Letting $\mathrm{NCUT}(C, \bar{C}) = \frac{\mathrm{cut}(C, \bar{C})}{\mathrm{bal}(C)}$ with $\mathrm{bal}(C) = 2\frac{vol(C)vol(\bar{C})}{vol(V)}$, Rangapuram *et al.* [RH12] define a new objective:

$$\hat{F}_\gamma(C) = \frac{\mathrm{cut}(C, \bar{C}) + \gamma\left(\hat{M}(C) + \hat{N}(C)\right)}{\mathrm{bal}(C)}$$

where $\gamma \in \mathbb{R}_+$ is a parameter, while $\hat{M}(C)$ and $\hat{N}(C)$ are respectively the number of positive and negative disagreements of the $(C, \bar{C})$ clustering. They show how to optimize a tight continuous relaxation of $\hat{F}_\gamma$ as the non-negative ratio of a difference of convex function and a convex function.

On the other hand, one can also adapt these two cut objectives to directly include negative edges. Kunegis *et al.* [Kun+10] define the signed Laplacian as $\bar{L} = \bar{D} - A$, where $\bar{D}$ is the signed degree matrix such that $\bar{D}_{uu} = \sum_{v \in \mathcal{N}(u)} |A_{uv}|$, as well as a signed variant of the symmetric normalized Laplacian $\bar{L}_{sym} = \bar{D}^{1/2} \bar{L} \bar{D}^{1/2}$. They show that the signed Laplacian is positive semidefinite, and even positive-definite as soon as the graph is unbalanced (that is contains a cycle with an odd number of negative edges). From positive and negative cuts defined as $\mathrm{cut}^+(B, C) = \sum_{u \in B, v \in C} w^+_{uv}$ and $\mathrm{cut}^-(B, C) = \sum_{u \in B, v \in C} w^-_{uv}$, a natural signed cut is $\mathrm{scut}(B, C) = 2\mathrm{cut}^+(B, C) + \mathrm{cut}^-(B, B) + \mathrm{cut}^-(C, C)$ which can then be used to defined signed RATIOCUT and NCUT. Arguing that those definitions force negatively linked nodes to be symmetric around the origin, do not take into account the balance of negative edges in each cluster and are difficult to extend to more than two clusters, Zheng *et al.* [ZS15] instead propose two new normalized cuts:

$$SNScut(C_1, \dots, C_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}^+(C_i, \bar{C}_i) - \mathrm{cut}^-(C_i, \bar{C}_i)}{vol(C_i)}$$

$$BNScut(C_1, \dots, C_k) = \sum_{i=1}^{k} \frac{\mathrm{cut}^+(C_i, \bar{C}_i) - \mathrm{cut}^-(C_i, \bar{C}_i) + vol^-(C_i)}{vol(C_i)}$$

Noting that if $x_i \in \mathbb{R}^n$ is the vector indicator of cluster C_i (that is the u^{th} entry of x_i is 1 is u belongs to C_i and 0 otherwise), $x_i^T \bar{L} x_i = 2\mathrm{cut}^-(C_i, C_i) + \mathrm{cut}^-(C_i, \bar{C}_i) + \mathrm{cut}^+(C_i, \bar{C}_i)$,

Sedoc *et al.* [Sed+17] introduce the following cut objective:

$$sNcut(C_1, \dots, C_k) = \sum_{i=1}^{k} \frac{2\text{cut}^-(C_i, C_i) + \text{cut}^-(C_i, \bar{C}_i) + \text{cut}^+(C_i, \bar{C}_i)}{vol(C_i)}$$

Additional cut formulations for k-clusters are also presented in [CWD12], although Knyazev [Kny17] argue that using the non signed Laplacian and considering negative eigenvalues might be just as effective, citing for instance numerical instability of signed Laplacian.

Finally, Mercado *et al.* [MTH16] show that the Laplacians defined so far can be seen as arithmetic means of the Laplacian L^+ of the positive subgraph $G^+ = (V, E^+)$ and the signless Laplacian Q^- of the negative subgraph $G^- = (V, E^-)$, where $Q^- = D^- + A^-$. They suggest instead to use a geometric mean, defined for two positive matrices A and B as $A\#B = A^{1/2} \left(A^{-1/2} B A^{-1/2}\right)^{1/2} A^{1/2}$. This suggestion is based on the fact that if u is a common eigenvector of both A and B with eigenvalue λ and μ respectively, then u is an eigenvector of $A + B$ with eigenvalue $\lambda + \mu$ and an eigenvector of $A\#B$ with eigenvalue $\sqrt{\lambda\mu}$. Therefore, the k smallest eigenvalues of the geometric mean Laplacian will be influenced by both smallest eigenvalues of L^+ (corresponding to assortative clusters in G^+) and of Q^- (corresponding to disassortative clusters in G^-), while this is not the case for the arithmetic mean of Laplacians.

Community detection The clustering problem is often named community detection in the context of social networks, and several methods developed by practitioners have been extended to signed graphs. While they do not necessarily considered the CORRELATION CLUSTERING objective, and especially not its optimum, we still give a brief overview of them, as they tend to have been more tested experimentally. For instance, to find the cluster of node u, Yang *et al.* [YCL07] use a random walk approach on the positive subgraph G^+ to compute the probability of each node to reach u in T steps, sort the nodes accordingly and then find a threshold based on the number of disagreements. The one node move local heuristic that we described in the Physics-inspired paragraph on page 62 can also be formulated as genetic algorithms that simultaneously try to minimize the number of disagreements and maximize a signed variant of the modularity [LLL13; AP13]. Anchuri *et al.* [AM12] also consider these two objectives by seeing them as eigenvalue problems and devise an iterative splitting procedure. The overlapping community detection variant is considered by Chen *et al.* [Che+14], who used a signed probabilistic mixture model. Namely, an edge selects a pair of cluster r, s with probability ω_{rs} (where $r = s$ if the edge is positive and $r \neq s$ otherwise) and chooses its endpoints u and v with probability θ_{ru} and θ_{sv}. θ_{ru} is therefore the soft membership of node u to cluster r, and those parameters are estimated using the expectation-maximization algorithm. The same model is extended to directed graphs by Jiang [Jia15], who strangely enough names it stochastic blockmodel, although the focus is still on edge and not nodes. In a similar spirit to MAXAGREE$[k]$, Chu *et al.* [Chu+16] focus on finding k subgraphs dense in positive edges and densely connected by negative edges to each other. They dub such subgraph *Oppositive Cohesive Groups*, or more vividly *Gangs in War*, and after formulating the problem as a constrained quadratic optimization, they propose a faster iterative local search heuristic. When $k = 2$, these subgraphs are called antagonistic communities and a specific data mining approach was proposed by Gao *et al.* [Gao+16].

Let us review this material about CORRELATION CLUSTERING in the light of our thesis objective: efficiently and accurately characterize edges in complex networks, or rather signed graphs in this chapter. The results on the hardness of CORRELATION CLUSTERING, and the fact that the best approximations rely on a linear program with a large number of constraints seems to run counter to such efficiency and accuracy requirements. To avoid this pitfall, we presented many existing heuristics. More importantly, recall that our learning bias is that nodes are assigned to K groups, and that signs are consistent with those groups. Informally, the closer an input signed graph is to this ideal situation, the more regular it is with respect to this bias, and thus the easier the EDGE SIGN PREDICTION problem is. Accordingly, the works described in Section 3.2.3 point out that such non worst case instances are indeed where we expect clustering algorithms to be able to identify those K groups with little to no error. In the next section, we present an algorithm relying

on a similar intuition, in an active setting and where $K = 2$.

3.3 Low stretch trees and spanners

We now show how to apply the learning bias of Section 3.1 on undirected graphs. Furthermore, we assume that the strong balance holds, meaning that there are only two groups according to Theorem 4. In other words, the labelling of E is consistent with a two-clustering of V. Namely, V can be partitioned in two clusters such that edges within each cluster are positive and edges across clusters are negative. In that case, the following *multiplicative rule* holds: for any nodes u, v in V, and any path p between u and v in G, the sign $y_{u,v}$ is equal to the product of the signs along p. Hereafter, we call this product the parity of p, and denote it by $\pi(p)$. While it is a simple and convenient hypothesis, this is too strong of a requirement to be satisfied in practice. Therefore, we relax it by assuming that, starting from a consistent labeling Y, we can only observe a randomly perturbed version Y' of Y. Specifically, given a constant $q \in [0, 1/2)$, every sign of Y is flipped with a probability smaller than q. We denote by $E_{\text{flip}} \subset E$ the set of edges whose sign has been flipped.

In this section, we are interested in active learning algorithms that first query a subset E_{train} of the edges, observe the signs in E_{train} and use them to predict the remaining signs. More precisely, we focus on an algorithm that queries a spanning tree T of G and predicts the sign of an edge $(u, v) \in E_{\text{test}} = E \setminus E_T$ as the parity of $\text{path}^T(u, v)$. Intuitively, since each sign has been potentially flipped, the longer the path in T, the more likely its parity will be not be equal to the true sign $y_{u,v}$. Therefore we would like each such path to be as short as possible. Formally, the number of mistakes of such an algorithm is upper bounded by [Ces+12b, Equation (3)]

$$|E_{\text{flip}}| + \sum_{(u,v)\in E_{\text{test}}} \sum_{e\in E} \mathbb{I}\left\{e \in \text{path}^T(u, v)\right\} \mathbb{I}\left\{e \in E_{\text{flip}}\right\}$$

which in expectation is equal to:

$$q\left(|E| + \sum_{(u,v)\in E_{\text{test}}} |\,\text{path}^T(u, v)|\right) \tag{3.7}$$

In the following, we describe a way to build spanning trees tailored for this situation. More precisely, we implement and analyze a suggestion made to us by Vitale [Vit14].

3.3.1 GALAXY TREE: a spanning tree designed for sign prediction

To achieve our objective of building a spanning tree that minimizes the distances between all connected pair of nodes in the original graph, we rely on a particular subgraph structure, namely the star. We therefore introduce two algorithmic primitives. EXTRACT-STARS, which partition a graph G into a set of disjoint stars. And COLLAPSE-STARS, which selects edges from E to assemble these stars into a new, smaller graph. Given a graph topology $G_0 = (V_0, E_0)$ and assuming for simplicity that G_0 consists of a single connected component,[10] the GALAXY TREE algorithm repeatedly applies these two primitives to produce a sequence of graphs $\{G_t\}_{t=0}^{K}$ of decreasing size, until G_K is made of a single node. All the edges selected while reaching this stage then form the spanning tree we were looking for. This can be seen as a simple instantiation of a general procedure described in [Alo+95, Section 5.2].

In the following, we provide a more precise description of our two primitives and analyze their complexity. Then we state formally the complete GALAXY TREE algorithm, prove its termination and correctness, and show a detailed example of

[10] For we can otherwise run our algorithm in parallel on each connected components of G_0.

its execution. Finally, we provide a conjecture on the number of iterations needed to finish.

EXTRACT-STARS EXTRACT-STARS takes as input a graph $G_t = (V_t, E_t)$. While the nodeset V_t is not exhausted, it repeatedly samples a node c_i, creates a star S_i^t with c_i at its center and the neighbors of c_i on the periphery, removes all the nodes of S_i^t from V_t and all the edges incident to S_i^t from E_t, and finally decrements accordingly the degree of the 2-hop neighbors of c_i (see Figure 3.6 for a visual representation of this notation). Upon completion, it returns a list of stars, the set of all the edges within a star, and a map (or associative array) that associates each node of V_t to the index of the unique star it belongs to. According to the definition of Mehlhorn *et al.* [MS08], an associative array is an abstract data type composed of a collection of (key, value) pairs, such that each possible key appears at most once in the collection. It efficiently supports the addition, removal and modification of a pair, as well as the lookup of a value associated with a particular key.

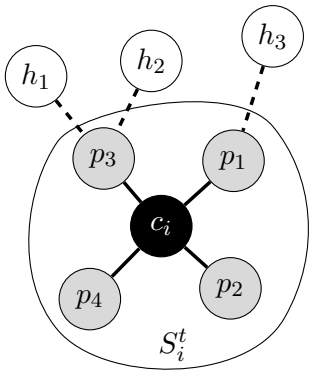


Figure 3.6 – A sample star created during the t^{th} collapse level. The black node is the center c_i of the star S_i^t, which is also made of the four light gray peripheral nodes as well as the solid edges. The 2-hops neighbors of c_i are the white nodes h_1 to h_3, whose degree will decrease once S_i^t is removed from G_t.

As showed in the following pseudo code[11], we sample centers by choosing the node with the current highest degree, with ties broken arbitrarily.[12] This is achieved efficiently by maintaining a max-priority queue Q, initially containing all the nodes of V_t. The priority of a node is its current degree and we equip Q with two standard operations described by Cormen *et al.* [Cor+09, section 6.5]: EXTRACT-MAX(Q) removes and returns the node of Q with the largest degree and DECREASE-KEY(Q, u, Δ) decrements the degree of the node u by an amount Δ. We also assume that G is the adjacency list of the graph, so that $G[u]$ is the set of neighbors of u, i.e. $G[u] \equiv \mathcal{N}(u)$. Finally $membership$ is the map storing for each node the index of the star it belongs to. This map is updated by the STAR function, which creates a star given a center c, a list $periphery$ of peripheral nodes, and a star index i. After creating the i^{th} star, for every node u belonging to that star, the STAR function sets $membership[u] = i$.

```
1:  function EXTRACT-STARS(G_t = (V_t, E_t))
2:      Let Q be the max-priority queue described above
3:      Let remaining be a set of nodes, initially containing all the nodes in V_t
4:      Let membership be an empty map
5:      stars ← []
6:      inner_edges ← ∅
7:      while Q is not empty do
8:          c ← EXTRACT-MAX(Q)
9:          if c not in remaining then
10:             continue        ▷ c is part of an existing star so there is nothing to do
11:         periphery ← G_t[c] ⋂ remaining
12:         stars ← stars ⋃ {STAR(c, periphery, |stars|)}
13:         inner_edges ← inner_edges ⋃ {(c, p) : p ∈ periphery}
14:         remaining ← remaining \ {{c_i} ∪ periphery}
15:         for p in periphery do
```

[11] Note that for clarity, we removed some bookkeeping code in all listings, mainly the part related to maintaining mapping between nodes at different collapse level. However, the full python implementation is available at https://github.com/daureg/magnet/blob/master/veverica/new_galaxy.py#L27.

[12] We also consider more involved heuristics but, in the interest of simplicity, they are presented later on page 78.

16: **for** h in $G_t[p] \bigcap remaining$ **do**
17: DECREASE-KEY(Q, h, 1)
18: **return** $stars$, $inner\_edges$, $membership$

Proposition 1. *For any connected graph $G = (V, E)$,* EXTRACT-STARS(G) *terminates in $O(|E|)$ time.*

Proof. EXTRACT-STARS terminates because at each iteration of the while loop line 7, we remove one node from Q and never add any. Let us now analyze its complexity. We first build a priority queue of all the nodes according to their degree (line 2), which requires $|V|$ insertions into Q. Then we execute $|V|$ iterations of the while loop. However, the main idea here is that we process each node and each edge exactly once. We first find the center c of the next star by extracting the maximum of the queue (line 8) and testing if c is still part of the graph, which happens $|V|$ times in total. Then we build the corresponding star (line 11–14). In total, we test the membership of $|V|$ nodes in line 11, STAR updates the $membership$ map $|V|$ times in line 12, $inner\_edges$ consists of $|E|$ edges at most in line 13 and $remaining$ is only updated $|V|$ times in line 14. Finally, we decrease the priority (i.e. the degree) of all nodes adjacent to the new star (line 15–17). Each decrement is supported by a single edge, thus DECREASE-KEYS is called at most $|E|$ times. Since all queue operations require constant time when using a Strict Fibonacci Heap [BLT12], the complexity is $O(|E| + |V|)$, which is also $O(|E|)$. □

COLLAPSE-STARS The second primitive, COLLAPSE-STARS takes as input the edges E_t of the current graph, along with the $membership$ result of EXTRACT-STARS and an $eccentricity$ array. It builds a new graph G_{t+1} where each star s_u from EXTRACT-STARS becomes a node and where there is a link between two nodes s_u and s_v if the nodes in V_t making up s_u and s_v are connected in E_t. The $eccentricity$ array is needed because when connecting two stars, we would prefer to join their center rather than two of their peripheral points. Therefore, we maintain an eccentricity count for all of the nodes of the original G_0, which is incremented by 1 each time a node is chosen to be on the periphery of a star. In other words, the eccentricity of an original node quantifies to which extent it has been pushed to the border of the galaxy.

COLLAPSE-STARS not only returns the new graph G_{t+1} but also a map $cross\_edges$. This map is used to keep track of which edge in E_t connects any pair of node in V_{t+1} (as illustrated in Figure 3.7). It associates to any edge in E_{t+1} a set of edges from E_t. This set can have an arbitrary size during the execution of COLLAPSE-STARS, yet it is guaranteed to contain a single edge upon termination. Although this does not appear in the following pseudo code, in practice we ensure that (s_u, s_v) and (s_v, s_u) refer to the same edge in $cross\_edges$.

1: **function** COLLAPSE-STARS(E_t, $membership$, $eccentricity$)
2: Let G_{t+1} be an empty graph
3: Let $cross\_edges$ be the map described above
4: **for every** edge (u, v) in E_t **do**
5: $s_u \leftarrow membership[u]$
6: $s_v \leftarrow membership[v]$
7: **if** u and v are not in the same star (i.e. $s_u \neq s_v$) **then**
8: $cross\_edges[(s_u, s_v)] \leftarrow cross\_edges[(s_u, s_v)] \bigcup \{(u, v)\}$
9: **for every** pair of will-be connected stars (s_u, s_v) in $cross\_edges$ **do**
10: $possible\_underlying\_edges \leftarrow cross\_edges[(s_u, s_v)]$
11:

$$(u_0, v_0) \leftarrow \underset{(u,v) \in possible\_underlying\_edges}{\arg\min} eccentricity[u] + eccentricity[v]$$

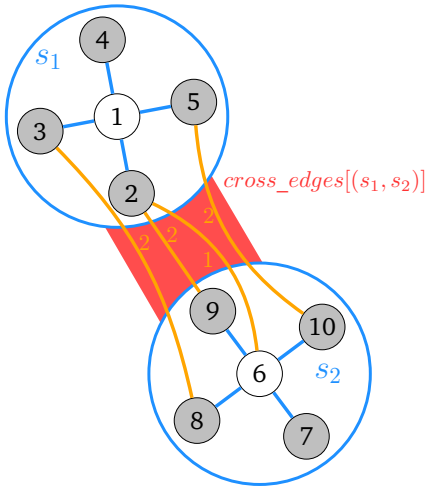


Figure 3.7 – In this graph, there are four possible edges between the two stars s_1 and s_2, and all are part of $cross\_edges[(s_1, s_2)]$, here represented in light red. Those edges are labeled with the sum of eccentricity of their underlying endpoints. White node have an eccentricity of 0, and gray node an eccentricity of 1. The edge with the minimal eccentricity is linking a peripheral node of s_1 directly to the center of s_2.

12: $cross\_edges[(s_u, s_v)] \leftarrow \{(u_0, v_0)\}$
13: $E_{t+1} \leftarrow E_{t+1} \bigcup \{(s_u, s_v)\}$
14: **return** G_{t+1}, $cross\_edges$

Proposition 2. *For any graph $G = (V, E)$,* COLLAPSE-STARS$(E, membership, eccentricity)$ *terminates in $O(|E|)$ time.*

Proof. The analysis of COLLAPSE-STARS is rather straightforward because the function only executes two loops over E. During the for loop of line 4, it only performs constant time operations on maps. Likewise, in the for loop of line 9, the most expensive operation is finding the minimum in line 11. Computing the eccentricity of an edge is done in constant time and only once for each edge of E. As a consequence, the total time of COLLAPSE-STARS is indeed $O(|E|)$. □

Putting the pieces together EXTRACT-STARS and COLLAPSE-STARS are truly the core the of GALAXY TREE algorithm but to obtain our final spanning tree, we need additional work, namely updating the eccentricity of the nodes of V_0 and keeping track of each edge within and between stars along every collapse steps. Despite our earlier promise, this entails showing some bookkeeping code, because it has an influence on the runtime of GALAXY TREE. Namely in the listing of Algorithm 1 on the next page, we use the three following maps:

name	keys set at t	values set at t	comment
$star\_membership$	nodes in V_t	nodes in V_{t+1}	this is the map returned by EXTRACT-STARS(G_t)
$full\_membership$	nodes in V_0	nodes in V_{t+1}	at $t = 0$, this is equals to $star\_membership$ and then it gets updated at every iteration to maintain its original keys set.
$original\_node$	nodes in V_t	sets of V_0 nodes	This can be seen as the reverse of $full\_membership$ and it is the one needed to update the eccentricity of the original nodes.

As described in Algorithm 1, at every collapse level, we first extract stars from the current graph (line 5), then update the eccentricity and node mappings (lines 7–8) and finally collapse the graph (line 9). We perform these operations until there

Algorithm 1 GALAXY TREE($G_0 = (V_0, E_0)$)

1: Let $eccentricity$ be an array of size $|V_0|$ initially all set to 0
2: $G_t \leftarrow G_0$
3: $inner\_edges\_seq$, $outer\_edges\_seq \leftarrow [], []$
4: **repeat**
5: $stars$, $inner\_edges$, $star\_membership \leftarrow$ EXTRACT-STARS(G_t)
6: Add $inner\_edges$ to the list $inner\_edges\_seq$
7: UPDATE-ECCENTRICITY($stars$, $eccentricity$, $original\_node$)
8: $full\_membership$, $original\_node \leftarrow$ UPDATE-NODES-MAPPING($full\_membership$, $star\_membership$)
9: G_{t+1},$outer\_edges \leftarrow$ COLLAPSE-STARS($E_t \setminus inner\_edges$,$star\_membership$, $eccentricity$)
10: Add $outer\_edges$ to the list $outer\_edges\_seq$
11: $G_t \leftarrow G_{t+1}$
12: **until** $|outer\_edges| > 0$
13: **return** ASSEMBLE-SPANNING-TREE($inner\_edges\_seq$, $outer\_edges\_seq$)

```python
def update_eccentricity(stars, eccentricity, original_node):
    for star in stars:
        for top_p in star.points:
            for real in original_node[top_p]:
                eccentricity[real] += 1
def update_nodes_mapping(star_membership, new_sm, first_iter):
    """After a collapse step, update the mapping between node
    indices in the new graph and original ones."""
    if first_iter:
        star_membership = new_sm
    else:
        new_mm = {}
        for orig_nodes, previous_star in star_membership.items():
            new_mm[orig_nodes] = new_sm[previous_star]
        star_membership = new_mm
    original_basis = defaultdict(set)
    for orig_id, star_id in star_membership.items():
        original_basis[star_id].add(orig_id)
    return star_membership, original_basis
def assemble_spanning_tree(stars_edges, interstellar_edges):
    final_edges = []
    data = zip(reversed(stars_edges), reversed(interstellar_edges[:-1]))
    for current_level_edges, translation_to_lower_level in data:
        final_edges.extend((e for one_star_edges in current_level_edges
                            for e in one_star_edges))
        for i, e in enumerate(final_edges):
            final_edges[i] = translation_to_lower_level[e]
    return final_edges+[e for star in stars_edges[0] for e in star]
```

are no edge connecting stars anymore. At this point, we revisit every outer edge to build the spanning tree (line 13).

Proposition 3. *For any connected graph G_0, GALAXY TREE(G_0) terminates after $K \leq |V_0|$ iterations. Furthermore, it runs in $O(K|E_0|)$ time and returns a spanning tree of G_0.*

We will need the following lemma.

Lemma 4. *If G_0 is connected, then any subsequent graph G_t, $t \leq K$ is also connected.*

Proof. Suppose not and assume, to the contrary, that there exists at least one (or more) disconnected graph in $\{G_t\}_{t=1}^{K}$ and let t_0 be the smallest index such that G_{t_0} is disconnected. Then there exist two nodes s_u and s_v in V_{t_0} with no path between them in E_{t_0}. Letting $\mathcal{U}, \mathcal{V} \subset V_{t_0-1}$ be respectively the nodes forming stars u and v, this implies there is no path between nodes in $\mathcal{U}$ and nodes in $\mathcal{V}$. However, G_{t_0-1} is connected by hypothesis, which leads to a contradiction. □

Proof of Proposition 3. Let us first show that GALAXY TREE terminates in less than $|V_0|$ iterations. This follows from the fact that every time we collapse the graph

G_t, we strictly reduce the number of nodes. Indeed, according to Lemma 4, G_t is connected so at least two nodes of V_t are joined by an edge and will form a star, i.e. a single node. We can thus claim that $|V_{t+1}| < |V_t|$. Note also that for all t, $|V_t| > 0$ and that when $|V_t| = 1$, EXTRACT-STARS creates a singleton and COLLAPSE-STARS does not return any outer edges. At this point, GALAXY TREE finishes, proving that the number of iterations K satisfies $K \leq |V_0|$.

Then we analyze the time complexity. We already know that during the t^{th} iteration, EXTRACT-STARS and COLLAPSE-STARS take time $O(|E_t|)$. We only provide the actual python code instead of pseudo code for the remaining functions since they do not present any relevant algorithmic aspect. However, we can say that UPDATE-ECCENTRICITY and[13] UPDATE-NODES-MAPPING take $O(|V_0|)$ time, since they go through every node of the original graph. Because G_0 is connected, $|V_0| \leq |E_0|$, and since for all t, $|E_t| \leq |E_0|$, we have that the GALAXY TREE inner loop takes $O(K|E_0|)$ time. Finally, since ASSEMBLE-SPANNING-TREE goes again through every edge visited at every collapse step, it also takes $O(K|E_0|)$ time, which is thus the overall complexity of the GALAXY TREE algorithm.

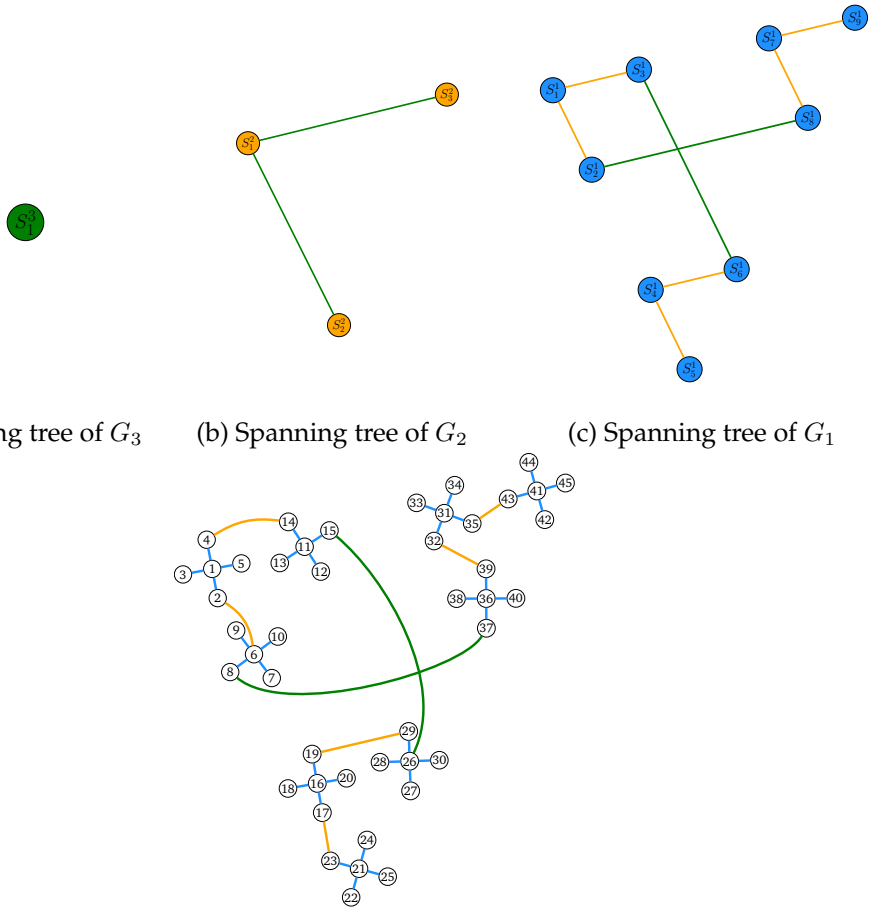
Finally, we prove that GALAXY TREE indeed returns a spanning tree of G_0. Specifically, we show that by starting from G_K and going in reverse order, we build a spanning tree T_t of every G_t, including eventually G_0. This process is illustrated in Figure 3.8 and works as follow: when moving from G_{t+1} to G_t, we expand each node of G_{t+1} into a star of nodes of V_t, and update all the outer edges we met so far to have their endpoints in V_t. These outer edges are exactly T_{t+1} and by translating them into E_t and adding the inner star edges of G_t, we build T_t. G_K consists of a single node with no edges so we have trivially that $T_K = G_K$. Then, assume we have a spanning tree T_{t+1} of G_{t+1}, so that all nodes in V_{t+1} are connected without cycle. Expanding each node of V_{t+1} into a star of V_t nodes ensures that the spanning property is preserved since each node of V_t is covered by construction. Those stars are also without cycle by construction. The second step, translating the existing edges of T_{t+1}, only connects nodes belonging to different stars and because those edges were cycle free in G_{t+1}, this remains the case in G_t. Therefore we have build a spanning tree T_t of G_t. By repeating this procedure, the GALAXY TREE algorithm eventually builds a spanning tree of G_0. □

The bound of Proposition 3 amounts to $O(|V_0||E_0|)$, which we believe is overly pessimistic. Indeed, we expect the number of nodes to decrease by more than one at each step, potentially leading to only $O(\log |V_0|)$ iterations.

Example of GALAXY TREE We illustrate the operation of the GALAXY TREE algorithm on a small example. Let us start with the initial graph G_0 depicted in Figure 3.9 on page 77 and initialize the eccentricity of all nodes to 0. When running EXTRACT-STARS, we see that the maximum degree is 4, achieved at nodes $\{1, 6, 11, 16, 21, 26, 31, 36, 41\}$. For the sake of simplicity, assume nodes are picked according to their index. First, node 1 forms the star S_1^1 with peripheral nodes 2, 3, 4 and 5. This increments the eccentricity of those peripheral nodes by 1. Then node 6 forms its star S_2^1 with 7, 8, 9 and 10. The process continues until node 41 is chosen to be the center of star S_9^1, at which point the max-priority queue has been exhausted and EXTRACT-STARS finishes.

We then call COLLAPSE-STARS. This will connect all possible pairs of stars. For instance, the edge between nodes 19 and 29 leads to the edge between S_4^1 and S_6^1. This is actually the only possible edge between S_4^1 and S_6^1. Consider on the other hand the case of edges $(2, 6)$ and $(2, 9)$. They both connect S_1^1 and S_2^1. Yet at this point of the algorithm, the eccentricity of node 2 is 1, the eccentricity of node 6 is 0 and the eccentricity of node 9 is 1. The edge $(2, 6)$ has therefore the

[13] We can get rid of this $O(|V_0|)$ at each iteration by ignoring eccentricity and taking a random edge between star, although it is not clear at the moment how it would affect the quality of the resulting tree.



(a) Spanning tree of G_3 (b) Spanning tree of G_2 (c) Spanning tree of G_1

(d) Spanning tree of G_0

Figure 3.8 – Unfolding stars to recover spanning trees

smallest total eccentricity and is chosen to connect S_1^1 and S_2^1. The full result of the COLLAPSE-STARS procedure is G_1, which can be seen on Figure 3.10a.

We now run EXTRACT-STARS on G_1. Because all nodes have degree 2, they could all be chosen to be the center of a star. We again assume they are picked according to their index and therefore we choose S_1^1 to be the center of the star S_1^2 with peripheral nodes S_2^1 and S_3^1. The original nodes belonging to those peripheral stars (nodes 4 to 15) have their eccentricity incremented by 1. The next node with highest degree in G_1 is now S_4^1, which forms a star with S_5^1 and S_6^1. This choice means that nodes 21 through 30 have their eccentricity incremented by 1. Finally, S_7^1 forms the last star with S_8^1 and S_9^1. Then COLLAPSE-STARS connects the resulting three stars, and this time there is only a single choice between each pair of stars, leading to the graph G_2 showed in Figure 3.10b

The action of EXTRACT-STARS on G_2 is quite simple, because there is only one star that can be created, so let say we choose S_1^2 as its center, with S_2^2 and S_3^2 as peripheral nodes. This increases the eccentricity of their underlying G_0 nodes by 1 (namely nodes 16 to 45). Because there is only one star S_1^3 left, COLLAPSE-STARS returns G_3 showed in Figure 3.10c and an empty list of outer edges, meaning that the inner loop of GALAXY TREE is finished and we can go through every edges we chose between stars at every level to recover the final spanning tree, a process illustrated in Figure 3.8 on the current page. For completeness, we can also look at the edges which are not part of the spanning tree, that is the test edges, and compute how long are the paths joining their endpoints. Indeed, this is the second quantity appearing in the mistakes bound of equation (3.7). As shown in Table 3.7, this average length is 7.

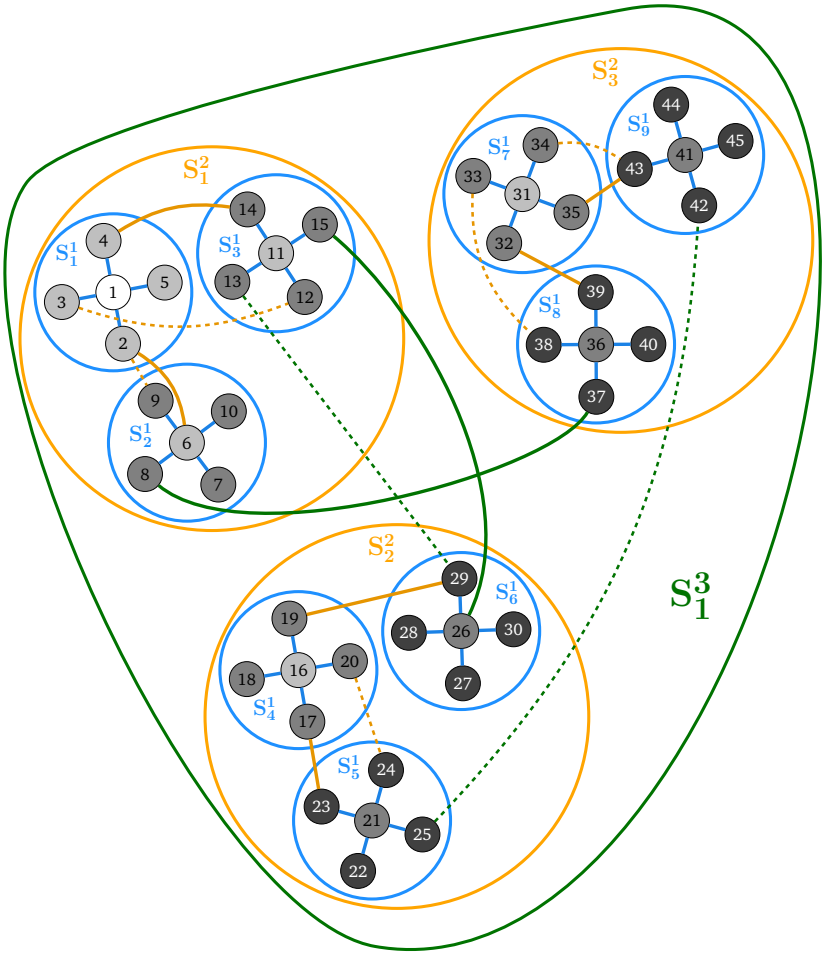
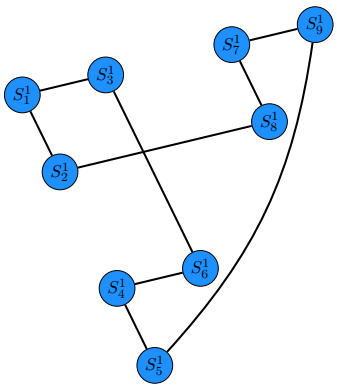
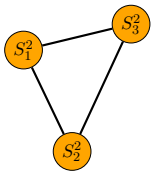


Figure 3.9 – The execution of the GALAXY TREE algorithm. The original graph is made of the solid and dashed edges connecting the nodes labeled by their index. Edges forming the final spanning tree are solid while the others are dashed. Their colors indicate at which iteration they were chosen to be inside a star. The four shades of gray, from white to dark gray denote increasing node eccentricity (as computed at the end of the algorithm). The i^{th} star created during the j^{th} iteration of the algorithm is denoted S_i^j. Refer to the main text for the complete description of the execution.



(a) Resulting graph after the first iteration



(b) Resulting graph after the second iteration

(c) Resulting graph after the third iteration

Figure 3.10 – The other iterations of GALAXY TREE

Table 3.7 – Length of the paths not in the resulting spanning tree.

test edge	path in the tree	length
$2, 9$	2–6–9	2
$3, 12$	3–1–4–14–11–12	6
$13, 29$	13–11–15–26–29	4
$20, 24$	20–16–17–23–21–24	5
$25, 42$	25–21–23–17–16–19–29–26–15–11–14–4–1–2–6–8–37–36–39–32–31–35–43–41–45	24
$33, 38$	33–31–32–39–36–38	5
$34, 43$	34–31–35–43	3

Discussion on the number of iterations needed A crucial quantity of the GALAXY TREE algorithm, both in terms of complexity and resulting stretch, is the number of collapses K needed before termination. While this is still elusive to express in the general case, let us first look at some simple cases. For instance, a very sparse example of graph is the line graph. While it is already a tree, let us look how the GALAXY TREE algorithm operates on it. Say we have n nodes and m edges in that line. Here, a star is made at most of three consecutive nodes. In the worst case, centers will be chosen such that we have a succession of three and two nodes stars, as in Figure 3.11. This will results in $2n/5$ stars organized in a line, which is less than half of n. Thus we can conclude that in this case, GALAXY TREE will finish after $O(\log n)$ iterations. Note that a barbell graph (two cliques connected by a line) would requires a number of iterations logarithmic in the length of that central line, despite having many more nodes and edges.

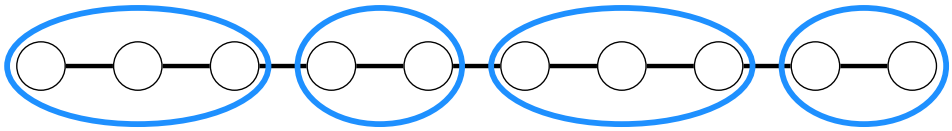


Figure 3.11 – A line graph with stars in blue.

In fact, it is helpful to look at this question from the other direction. Namely, we can wonder what is the minimum number $n(K)$ of nodes necessary for a graph to requires at least K collapses before being reduced to a single node. Checking by hand, we have that $n(1) = 2$, $n(2) = 4$ and $n(3) = 8$. Furthermore, this number is achieved by a line or circle consisting of exactly 2^K nodes. By induction, we can even show that on any line or circle graphs with 2^n nodes, the GALAXY TREE algorithm will terminate after n collapses. Assuming we could prove that $n(K) = 2^K$, given a graph G with $n = 2^{\log_2 n}$ nodes, by definition of $n(K) = 2^K$, it would require less than $\lceil \log_2 n \rceil = O(\log n)$ collapses before being reduced to a single node. However, we were not able to rule out the possibility that there is a graph more parsimonious than a line still able to survive the same number of collapses. Therefore, we cannot conclude that $n(K) = 2^K$ and we leave this statement as a conjecture.

Variants of EXTRACT-STARS The execution of EXTRACT-STARS is mainly deterministic, except for the fact the ties between nodes with the same highest degree are broken arbitrarily. This allows for an efficient implementation, and simplifies the analysis of the resulting sequence of stars and therefore the induced spanning tree. However, it can be detrimental in an adversarial context, where we could end up with a tree forcing a lot of mistakes. We add an element of randomization to EXTRACT-STARS by letting it use two optional arguments, a *threshold function* τ or a *degree function* $\widetilde{d}$. Such functions modify the center sampling process in the following way:

- if $n_{t,i}$ is the number of node remaining in V_t before choosing the i^{th} center, choose a node uniformly at random among those with a degree larger than $\tau(n_{t,i})$. The idea is to choose among a small set of high degree nodes, for instance by letting $\tau(n) = \sqrt{n}$. Note however we cannot guarantee there will always be nodes with degree above the threshold, in which case we default on the highest degree node
- if $\deg_i(u)$ is the degree of node u before choosing the i^{th} center, choose node proportionally to $\widetilde{d}(\deg_i(u))$. Again, the degree function is designed so that it favors the selection of high degree nodes. For instance, one could use $\widetilde{d}(\deg_i(u)) = \deg_i(u)^2$.

These two variants are more time consuming because they require additional bookkeeping. Therefore, we don't provide a full complexity analysis and only briefly sketch their implementations here.[14] For the threshold function, we maintain two queues, $high$ and low, containing nodes whose degree is respectively above and below the current threshold. We select a node uniformly at random in $high$, remove the corresponding star from G_t, recompute the new threshold and if necessary, move nodes which fell under the threshold from $high$ to low and those who climb above the threshold from low to $high$. For the degree function, we can draw any node as center proportionally to its weight (where the weight of node u is defined as $d_f(\deg(u))$). Yet we cannot use the standard method of computing the cumulative sum of weights since some of them change at each iteration. Therefore, we construct a binary tree whose leaves are the nodes of V_t and where each tree nodes maintain the sum of weights in its left and right subtrees. To sample, we draw a random number r between 0 and the total weight of the tree and go down from the root to the leaf spanning the weight interval containing r. When degrees are updated (or graph node removed), we update the weights along a path from the corresponding leaves to the root of the tree.

A variant of the GALAXY TREE algorithm as a whole we did not explore so much in practice is its ability to produce spanners by stopping early. Basically if we stop at iteration t, we output the graph G_t (which in general is not a tree) with its edges unfolded to lie in E_0. This corresponds to a trade off between having more edges than $|V_0| - 1$ but potentially making shorter connection. This would also be interesting for the EDGE SIGN PREDICTION problem. Assuming the treewidth of G_t is decreasing with t, we could try to show there is "reasonable" number of paths between u and v, compute their parity and take a majority vote on the results.

3.3.2 Related work

Looking for a subgraph H of G that best preserves the distance in G while being sparse is an old problem, driven originally by network design in fields such as transportation [Qua60] and electrical circuits [Kni60]. The way we define "preserving the distance", and the exact form of H give rise to several problems, which we summarize later in Table 3.9. We first give some definitions, then cover the most relevant problems in details, and finally give some pointers for the others problems.

Let the distance between u and v in G be

$$d_G(u,v) = \sum_{e \in \text{path}^G(u,v)} \ell(e)\,,$$

where $\ell(e)$ is the *length* of the edge e and $\text{path}^G(u,v)$ is the shortest path between u and v in G. In the following, we consider only the uniform case, in which the length of an edge is equal to its weight. The stretch of an edge (u,v) in H is defined as

$$\text{stretch}(u,v) = \frac{d_H(u,v)}{d_G(u,v)}.$$

[14] Although they are available online at `https://github.com/daureg/magnet/blob/master/veverica/`{ThresholdSampler.py, NodeSampler.py}.

We may then want to minimize the stretch of:
1) some pairs of nodes. That is, given L and R in V, minimize $\sum_{u\in L, v\in R} \text{stretch}(u,v)$
2) all pairs of nodes corresponding to edges of G, i.e. minimize $\sum_{(u,v)\in E} \text{stretch}(u,v)$
3) all pairs of nodes, i.e. minimize $\sum_{(u,v)\in V^2} \text{stretch}(u,v)$

Note that for unweighted graphs, the second problem reduces to minimizing

$$\text{stretch}(H) = \sum_{(u,v)\in E} |\,\text{path}^H(u,v)|. \tag{3.8}$$

If furthermore H is tree, this is equivalent to minimize the second term of equation (3.7). Therefore we focus mainly of that definition of stretch, and consider the other two only briefly.

The second point affecting the problem is the structure of H. The only requirements are that it must be spanning all the nodes involved in the computation of the chosen stretch, and that $\forall (u,v) \in E,\ d_H(u,v) \geq d_G(u,v)$. Beside that, H can be a tree of G, a general subgraph of G or even a subset of V^2 (i.e. containing edges not in E). We focus mainly on the first two cases, since they are covered by the GALAXY TREE algorithm.

Trees One early mention of seeking a low-stretch spanning tree is given by Hu [Hu74], albeit in more general form:

Problem 1 (Optimal Communication Spanning Tree)**.** *Given a set of nodes $V = \{v_1, \ldots, v_n\}$, a set of distances d_{ij} and a set of requirements r_{ij} between v_i and v_j, find a spanning tree connecting these n nodes such that the total cost of communication of the spanning tree is a minimum among all spanning trees. The cost of communication for a pair of nodes is $r_{i,j}$ multiplied by the sum of the distances of arcs which form the unique path connecting v_i and v_j in the spanning tree. The cost of a spanning tree is the sum of costs over all pairs of nodes.*

For a weighted graph $G = (V, E, w)$, by letting $d_{ij} = w_{ij}$ and $r_{i,j} = \mathbb{I}\{(i,j) \in E\}$, finding an Optimal Communication Spanning Tree thus amounts to finding a low-stretch spanning tree. Table 3.8 present a list of works where the stretch was improved.

We start with the seminal paper of Alon *et al.* [Alo+95]. It touches on many topics, and frame the problem in a game theoretic way but here we only focus on two of their results: a lower bound of $\Omega(\log n)$ for the average stretch of any tree and their construction of a tree with $\exp O(\sqrt{\log n \log\log n})$ average stretch in time $O(m^2)$. The lower bound follows from an existing result in extremal graph theory [Bol04, pages 107–109]: there is a positive constant a such that for all $n \in \mathbb{N}$, one can construct a graph G with n vertices and $2n$ edges such that every cycle G has a length of at least $a \log n$. Now consider any spanning tree T of G. While all the $n-1$ edges of T have a stretch of 1, the $n+1$ remaining ones form a cycle in T hence in G as well and thus incur a stretch of at least $a \log n$. This shows that the average stretch is at least $\frac{1}{2} a \log n$.

They construct a low stretch spanning tree in a bottom up manner like the GALAXY TREE algorithm. First, they extend the definition of stretch to multigraph [Alo+95, Section 4] and then describe a procedure to transform in linear time any multigraph G with n nodes to a multigraph G' on the same nodeset with at most $n(n+1)$ edges such the average stretch of G' is at most twice that of G [Alo+95, Lemma 5.2]. The next ingredient is an algorithm to build a low diameter decomposition of a multigraph G, parametrized by a number $x(n)$ depending of n. It works by repeatedly selecting an arbitrary node and growing a ball around it until the number of edges leaving the ball is at most a fraction $1/x(n)$ of the number of edges with both endpoints in the ball. The key property of this decomposition is that it yields a partition of G in clusters such that the radius of each cluster is small (namely at most $O(x(n)\log n)$) and there is most a fraction $1/x(n)$ of edges between

clusters. Finally, the iterative procedure is a follows: once a partition has been built, we compute a shortest path spanning tree in each cluster that are then collapsed into super nodes to form the next graph G' and the process repeats. Another difference from GALAXY TREE, besides the partition procedure, is that G' is a multigraph, taking into account the number of edges joining cluster, while COLLAPSE-STARS picks only the most direct one.

Another interesting idea from this paper is to consider a distribution over trees instead of a single instance, especially when one is concerned about the maximum stretch instead of the average one. For instance, on a cycle with n nodes, a tree is obtained by removing one edge, and that edge incurs a stretch of $n-1$. The uniform distribution over such trees has a maximum stretch of $2\left(1-\frac{1}{n}\right)$ [Kar89].

Table 3.8 – Reproduction of Table 1 from [AN12], showing the evolution of the best asymptotic average stretch over time.

work	average stretch	time
[Alo+95]	$\exp(O(\sqrt{\log n \log\log n}))$	$O(m^2)$
[Elk+05]	$O((\log n)^2 \log\log n)$	$O(m \log^2 n)$
[ABN08]	$O(\log n (\log\log n)^3)$	$O(m \log^2 n)$
[KMP11]	$O(\log n (\log\log n)^3)$	$O(m \log n \log\log n)$
[AN12]	$O(\log n \log\log n)$	$O(m \log n \log\log n)$

The idea of recursively partitioning the graph and construction a low-stretch spanning tree in each part is common to all the papers of Table 3.8. Elkin *et al.* [Elk+05] devise a (δ, ϵ)-star decomposition such that all the stars have comparably low radius. It was modified in [ABN08] to improve the stretch. Then Koutis *et al.* [KMP11] improve the runtime by rounding the edge weights to the closest power of 2 and using a modified implementation of the Dijkstra's algorithm in the case of at most k distinct edge weights [Orl+10]. Finally, Abraham *et al.* [AN12] describe an even more complex but tighter petal decomposition.

Spanners As we mentioned, by stopping the GALAXY TREE algorithm before it finishes, we obtain a set of edges spanning the graphs that is not a tree. Such structure are called *spanner*. More precisely, the subgraph H is said to be an t-spanner of G if, for a parameter $t \geq 1$, and for every pair $u, v \in V$ of vertices, it holds that $d_H(u,v) \leq t \cdot d_G(u,v)$. The problem was introduced by Peleg *et al.* [PS89] and Peleg *et al.* [PU89] and has been extensively studied since then, for it has many applications in network design. It was also showed to be NP-HARD to approximate [EP07]. The most simple construction is a greedy algorithm [Alt+93] that works similarly to the minimum spanning tree construction. Starting from an empty subgraph H, it goes through every edge (u,v) of G sorted by weight and check if there is a path between u and v in H of length at most t. If this is the case, the edge (u,v) is dropped, otherwise it is inserted in H. This results in a $(2t-1)$-spanner with $O(n^{1+1/t})$ edges, which is an optimal trade-off between those two quantifies. Furthermore on weighted graphs, the greedy spanner total weight is essentially optimal [FS16]. However, the best implementation of it, using a dynamic data structure [RZ04] is not scalable for it runs in $O(tn^{2+1/t})$ and cannot easily be parallelized. Parallelization therefore requires other kind of approaches [Pet08; Mil+15]. Recently, Elkin *et al.* [EN17] showed how to obtain, for any $\epsilon > 0$, a $(2t-1)$-spanner with $O(n^{1+1/k}/\epsilon)$ edges in t rounds, with probability at least $1-\epsilon$.

Other problems Finding low stretch trees and spanners with respect to the existing edges is the most relevant problem when addressing the EDGE SIGN PREDICTION problem. For the sake of completeness, we nonetheless give an overview of some related problems.

For instance, Johnson *et al.* [JLK78] define the following problem, where the stretch is defined over all possible pairs of nodes[15]:

Problem 2 (Network Design Problem). *Given an undirected integer-weighted graph $G = (V, E, w)$, a budget $B \in \mathbb{N}$ and a criterion threshold $C \in \mathbb{N}$, does there exist a spanning subgraph $G' = (V, E')$ of G with weight $w(E') \leq B$ and criterion value $F(G') \leq C$, where the criterion function $F(G')$ denotes the sum of the weights of the shortest paths in G' between all vertex pairs?*

They prove that finding such a subgraph is NP-COMPLETE, by exhibiting a reduction from the KNAPSACK problem. They also prove that the less general problem of finding a spanning tree on an unweighted graph, that is

Problem 3 (Simple Network Design Problem). *Problem 2 with w being the equal to 1 for all edges in E and $B = |V| - 1$.*

is also NP-COMPLETE by reduction from EXACT 3-COVER. However, it has recently been show that this Simple Network Design problem can be approximated to a constant factor 6 [Che+10]. Moreover, even when the graph is weighted, Abraham *et al.* [ABN07] achieve a universal constant bound for any weighted graph.

Another problem appear when the low-stretch structure H can include edges not in G (as long as the distances in H remain larger than the distances in G). This is captured by the following problem [Sco69]:

Problem 4 (Optimal Network Problem). *Given a set V of n vertices, find a set of spanning edges $E \subset V^2$ that minimizes the sum of the length of the shortest paths between all vertex pairs while the total length of the resulting network does not exceed some upper bound $B \in \mathbb{N}$.*

This can be seen as a special case of Problem 2 with G being the unweighted n-complete graph. Scott [Sco69] proposes a backtracking solution and two local search approximate algorithms. Some early branch and bound heuristic solutions to Problem 4 are surveyed in [Min89, Section 2.3.2] although they do not come with asymptotic guarantee on the stretch. Furthermore, Wong [Won80] proves that for any $\epsilon \in (0, 1)$, finding a $|V|^{1-\epsilon}$ approximation is NP-COMPLETE. However, if we consider the average stretches over a distribution of trees, then this approximation factor can be reduced to $\Theta(\log n)$ [FRT03].

Finally, the stretch can also be computed for a subset of the edges. This is useful in cases where we have prior information on the importance of individual nodes or edges. For instance, Abraham *et al.* [Abr+17] show that for every t, any n-nodes graph $G = (V, E)$ has a subset S of size at least $n^{1-1/k}$, and a spanning tree that has stretch $O(k \log \log n)$ between any node in S and any node in V. Likewise, Gupta *et al.* [Gup+17] describe how to maintain a light subgraph H that minimizes the distance between pairs of source and sink that are given in an online fashion.

As shown by Table 3.9, those problems defined in the seventies are still being discussed nowadays in top tier conferences, proving their relevance and impact beyond the EDGE SIGN PREDICTION problem.

3.3.3 Empirical evaluation

In this section, we provide empirical evidences of the properties of GALAXY TREE over several classes of graph, and compare it with a BREADTH FIRST TREE baseline. Namely, we consider three kinds of graph topology (with both synthetic and real world instances that carry signs on their edges) and evaluate (i) what average stretch is reached by various trees and (ii) how accurate is the sign prediction.

[15] We adapt their notations to match ours

Table 3.9 – A summary of the lowest stretches achievable for various problems.

kind of stretch	only existing edges		extra edges allowed
	tree	not tree	
some pairs	$O(k \log\log n)$ [Abr+17]	[Gup+17, Section 4]	—
all existing pairs	$O\left(\log n(\log\log n)\right)$ [AN12]	$(2t-1)$-spanner, $O(n^{1+1/t})$ edges [Alt+93]	$\Theta(\log n)$ in expectation [FRT03]
all possible pairs	6 for unweighted graphs [Che+10] and $O(1)$ in general [ABN07]	—	no need for extra edges in that case

3.3.3.1 Graph topology

The three kinds of topology we consider are:

GRID which are 2D lattices, where each node has four neighbors except on the boundary. The synthetic ones are square, while the "real world" ones represents the four neighbors pixel connectivity of the pictures showed in Figure 3.12 on page 87.

PREFERENTIAL ATTACHMENT which are built synthetically according to the model of Barabási *et al.* [BA99]. While this does not follow the more rigorous specification of Bollobás *et al.* [BR04], informally, we start with a line graph of m nodes and add node one by one until the graph consists of n nodes. Each time a new node is added, it is connected to m of the existing nodes with a probability proportional to their degree. Here we choose $m = 3.13$, that is when adding a new node, we pick 3 or 4 existing neighbors such the initial expected number of neighbors for each new nodes is 3.13. Such graphs are quite sparse and have short diameter, thus providing a crude but reasonable approximation of online social networks. Therefore, the real world instances of the PREFERENTIAL ATTACHMENT model are WIKIPEDIA, SLASHDOT and EPINION networks from Chapter 2 along with GOOGLE+. The last one is constructed from ego networks of GOOGLE+[16] by keeping the largest connected component of users whose gender is known. Basic statistics of those real PREFERENTIAL ATTACHMENT graphs are presented in (Table 3.10).

TRIANGLE which consists of a Delaunay triangulation of a set of points randomly located in a 2D space.[17]

Table 3.10 – Dataset description

	$\lvert V\rvert$	$\lvert E\rvert$	fraction of + edges	$\frac{2\lvert E\rvert}{\lvert V\rvert\cdot(\lvert V\rvert-1)}$
WIKIPEDIA	7 065	99 936	78.5%	$4.00 \cdot 10^3$
GOOGLE+	74 917	10 130 461	67.6%	$3.61 \cdot 10^3$
SLASHDOT	82 052	498 527	76.4%	$1.48 \cdot 10^4$
EPINION	119 070	701 569	83.2%	$9.90 \cdot 10^5$

[16] Available at http://snap.stanford.edu/data/egonets-Gplus.html
[17] As implemented by the graph-tool library (https://graph-tool.skewed.de)

3.3.3.2 Stretch

The first property of Galaxy trees we wish to evaluate is their stretch, which depends only of graph topology. Recall that following equation Equation 3.8, we define the average test edge stretch as $\frac{1}{|E_{\text{test}}|}\sum_{(u,v)\in E_{\text{test}}}|\text{path}^T_{u,v}|$, where $|\text{path}^T_{u,v}|$ is the unique path between u and v in T.

As we consider unweighted graphs, we compare GALAXY TREE with a natural baseline, namely a spanning tree rooted at the highest degree node and obtained through a breadth first visit of the graph. This involves randomness in the order in which nodes are visited. Likewise in GALAXY TREE, the choice of the edge linking two stars is not always unique, meaning that we have to break ties at random. Therefore, for each graph, we repeat the tree construction 12 times and present the average result, noting that the variance (showed as error bar in Figure 3.13) is small.

On PREFERENTIAL ATTACHMENT and TRIANGLE, we see in Figure 3.13 on page 88 that both trees exhibits logarithmic stretch, although with a larger constant for GALAXY TREE. Note that this is also the case for others low stretch tree methods [PKK14, Section 5.3.1]. On GRID however, GALAXY TREE preserves this logarithmic stretch growth while this is visually no longer the case for BREADTH FIRST TREE. In that case, we cannot expect a better stretch than $\frac{\log n}{2048}$ according to [Alo+95, Theorem 6.6].

3.3.3.3 Sign prediction

The second design goal of Galaxy trees is to accurately predict the sign of edges in E_{test}. Except for the three real datasets that already include signs[18], all the other are constructed, meaning we have to set sign on their edges in the first place. This is done by partitioning the nodes into two clusters. For GOOGLE+ we use node gender, for pictures we use node color (black or white), and for all others, we propagate labels 0 and 1 from randomly selected high degree nodes. Once each node belongs to one of the two clusters, we set the sign of an edge between two nodes to be $+$ if they are in the same cluster and $-$ otherwise. Predicting using path parity will thus gives perfect result. To test performance in real situations, we then add noise, that is we select a fraction of edges uniformly at random and flip their sign.

Like in Section 2.5, we evaluate the performance of our prediction using the Matthews Correlation Coefficient (MCC), defined in equation Equation 2.9 on page 26. As showed in Figure 3.14 on page 89, when the noise level is low, GALAXY TREE performs better than BREADTH FIRST TREE. As the noise level gets higher, they have similar performance. Note also than in Figure 3.14c, GALAXY TREE is less sensible to the size of the graph.

To further assess the quality of our trees, we plug them in them into an existing heuristic method to predict edge sign: *A sym exp* [KLB09]. It computes the exponential of the adjacency matrix after it has been reduce to z dimension. This allows to count the sign of all paths between two pairs of nodes with decreasing weight depending of their length. To simulate an active learning setting, we reveal only a subset of edge in A. This subset can be: i) the edges forming a BREADTH FIRST TREE, ii) the edges forming a GALAXY TREE iii) $|V|-1$ edges chosen uniformly at random. We set the parameter z equal to 15 because i) it is one of the best in [KLB09, Fig. 11] and ii) it performs well on real datasets in [Ces+12a, Fig.3].

As the *A sym exp* has a $O(n^3)$ complexity and uses quite some memory at prediction time, the larger graphs used previously are not all included. The conclusion of Figure 3.15 on page 89 is that except on social networks, it is better to use spanning trees than random edges. Specifically, GALAXY TREE on GRID and BREADTH FIRST TREE elsewhere.

[18] We nonetheless perform some preprocessing in order to make them undirected and to remove the small proportion of conflicting edges (e.g. positive from u to v but negative from v to u).

3.4 Conclusions

3.4.1 Summary

In this chapter, we expanded our efforts on characterizing edges in complex networks by addressing the EDGE SIGN PREDICTION problem in undirected signed graphs. We first demonstrated, conceptually and practically, that the methods we designed in Chapter 2 are not suitable to this new setting. Drawing on the well established theory of social balance, we therefore came up with a different learning bias. More precisely, we posit that nodes latently belong to K groups. Then that the edge signs are initially set to be positive if both endpoints belong to the same group and negative otherwise. And that finally we observe such assignment after it had been perturbed uniformly at random.

We connected twice this bias for the EDGE SIGN PREDICTION problem with the CORRELATION CLUSTERING problem. First, solving CORRELATION CLUSTERING on a training set would provide us with the most reasonable partition of the nodes into K groups, allowing us to predict the test signs accordingly. Second, the optimal value of the CORRELATION CLUSTERING objective is a measure of the difficulty of the EDGE SIGN PREDICTION problem [Ces+12b]. We thus presented a detailed overview of the CORRELATION CLUSTERING problem and its proposed solution. Our goal was to point out that despite its hardness in the worst case, there exist efficient heuristics in practice, as well as methods with formal guarantees in more favorable cases. We furthermore noted that such cases are those when the sign assignment is close to follow our bias.

Such general CORRELATION CLUSTERING approaches would typically be used in a batch setting. In the last part of this chapter, we instead shifted our attention to the active setting, in the special case where $K = 2$. This allowed us to exploit our bias in a more specific way. Namely, we described an implementation of the GALAXY TREE spanning data structure [Vit14]. The idea is to query the sign of edges that connect the endpoints of every other edge by the shortest possible path. Our experiments showed that this GALAXY TREE construction used in active EDGE SIGN PREDICTION is competitive for some classes of graphs.

3.4.2 Future work

Beyond those preliminary experiments, we could push further our methods along several directions:

1) The basic analysis of Proposition 3 shows that on any graph with n nodes, the GALAXY TREE will perform at most n collapses before terminating. However, as we mentioned on page 76 at the end of Section 3.3.1, we believe that this maximum number of collapses might be as low as $O(\log n)$. It turns out we were not able to prove this statement. This is unfortunate, for besides giving us guarantees on the algorithm runtime, this would also inform us about the stretch of the resulting tree. Indeed, let us consider the reverse direction in which the GALAXY TREE algorithm operates. That is, instead of a sequence of collapses transforming the original graph into a single node, we look at a sequence of expansions turning this single node into the original graph. Informally, the length of the longest path can only increase by a factor of five as each pair of nodes in G_{t+1} expands into two connected stars in G_t.[19] Whereas the behavior of the average stretch is less straightforward, it is reasonable to expect that the smaller the number of expansions, the lower the stretch.

[19] More precisely, consider the edge (u, v) in G_{t+1}. The two nodes are expanded into the two stars s_u and s_v in G_t, respectively with peripheral nodes $\{u_1, \dots, u_k\}$ and $\{v_1, \dots, v_\ell\}$ and center c_u and c_v. In the worst case, there is now a path of length 5 in G_t, e.g. $(u_1, c_u, u_k, v_\ell, c_v, c_1)$.

2) As mentioned in Section 2.4, EDGE SIGN PREDICTION could be extended to weighted signed graphs. This is also relevant in undirected signed graphs. For instance, the signs might have been generated by thresholding an expensive symmetric similarity function. In that case, we might want to avoid extra evaluation of that function by predicting the signed weight of some new edges. A natural way to modify the GALAXY TREE algorithm is the following. When choosing the center of stars, pick the node with the current highest weighted degree. When choosing an edge to connect two centers, pick among the edges with the lowest endpoints eccentricity the one with the lowest weight. What is less clear is to which extent such changes would affect the analysis.

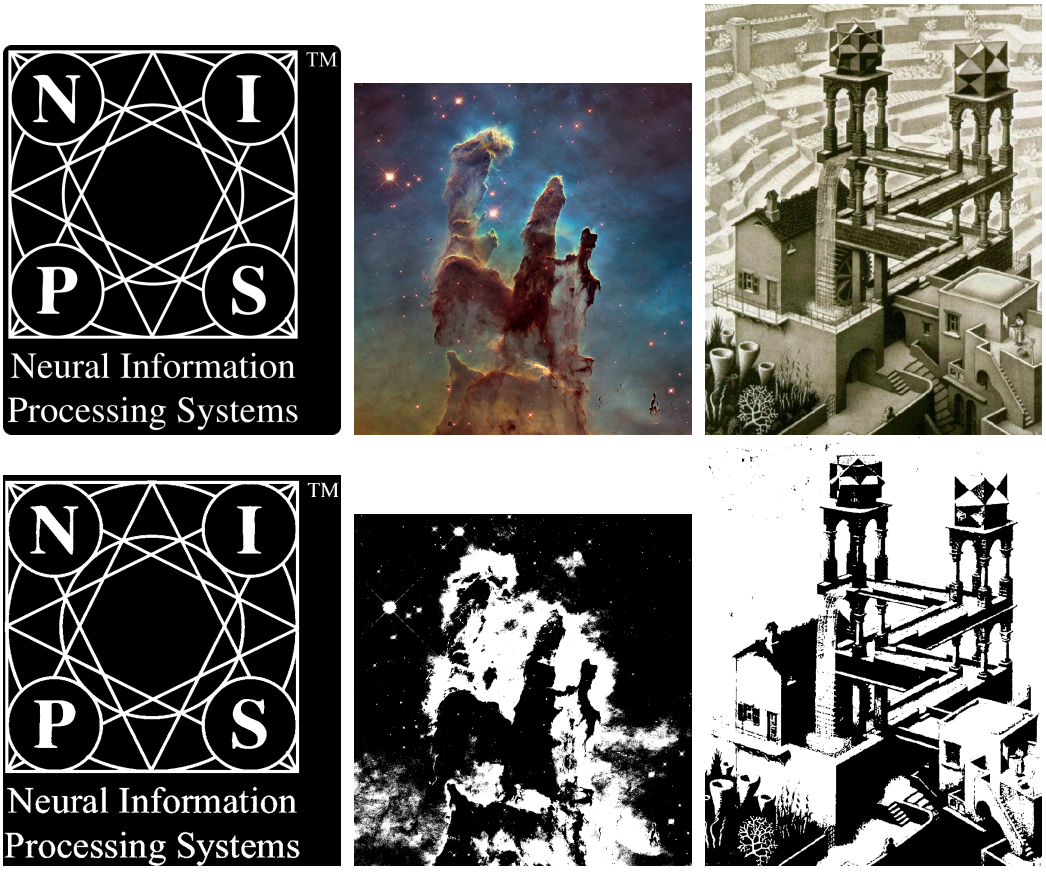
3) We wish we had more time to further explore CORRELATION CLUSTERING under stability assumptions, especially as a way to handle the case where $K > 2$. Recall that a clustering instance, made of a weighted graph and a partition objective, is α-*stable* (with $\alpha > 1$) if its optimal partition remains the same whenever every weight w_i is multiplied by a factor c_i between 1 and α. Furthermore, we say that an algorithm is *robust* if, given a clustering instance $\mathcal{I}$, it yields one of the two following outcomes in polynomial time:

   (i) if $\mathcal{I}$ is α-stable, returns the optimal solution of $\mathcal{I}$;
   (ii) if $\mathcal{I}$ is not α-stable, either returns the optimal solution of $\mathcal{I}$ or reports that $\mathcal{I}$ is not α-stable.

   This is a handy property because, in general, we cannot practically check the stability of an instance. Indeed, since we do not know the optimal solution, we cannot tell whether it changes or not under perturbations.

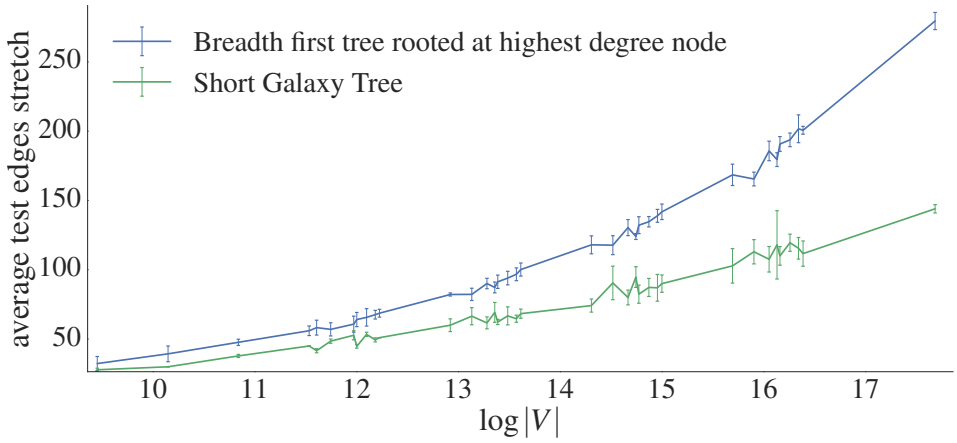
   As an example, let us consider the k-MINIMUM MULTIWAY CUT problem. Given a weighted undirected graph G and a set of k terminal nodes, the goal is to partition G into k clusters, each containing exactly one terminal, such that the total weight of edges across clusters is minimized. This bears some superficial similarity with k-MINIMUM MULTICUT. Recall from Section 3.2.2.2 that k-MINIMUM MULTICUT requires us to find the lightest set of edges to cut in order to separate k pairs of nodes. Recall further that k-MINIMUM MULTICUT is equivalent to CORRELATION CLUSTERING. As for k-MINIMUM MULTIWAY CUT, we can cast it as a CORRELATION CLUSTERING problem [BBC04, Theorem 24]. Interestingly, there exists a robust algorithm for $2 - 2/k$-stable instances of k-MINIMUM MULTIWAY CUT [AMM17]. Under the assumption that graphs obeying our learning bias are α-stable, for some α depending of the graph irregularity, having such a stable algorithm for k-MINIMUM MULTICUT would be very valuable. However, this might prove challenging. Indeed, k-MINIMUM MULTICUT is harder than k-MINIMUM MULTIWAY CUT, as illustrated by their respective approximation factor, $O(\log n)$ and less than 1.3 [BSW17].

(a) monastery (b) world (c) poster

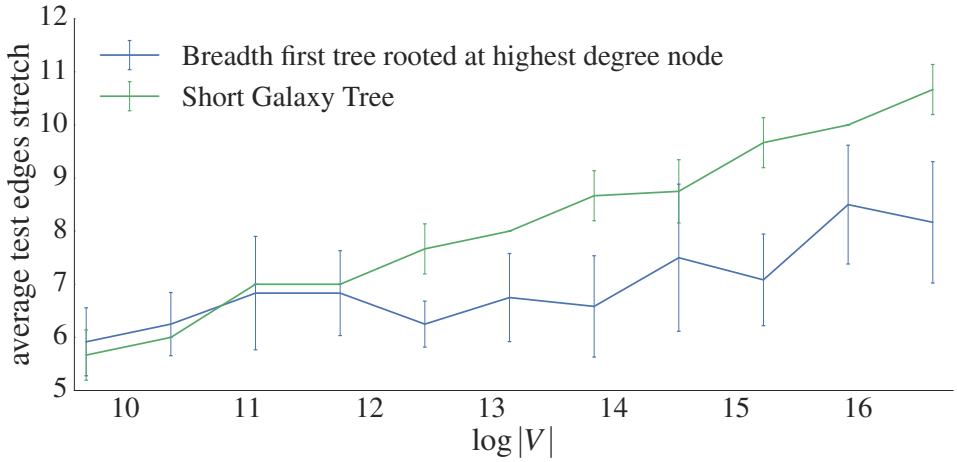


(d) logo (e) space (f) waterfall

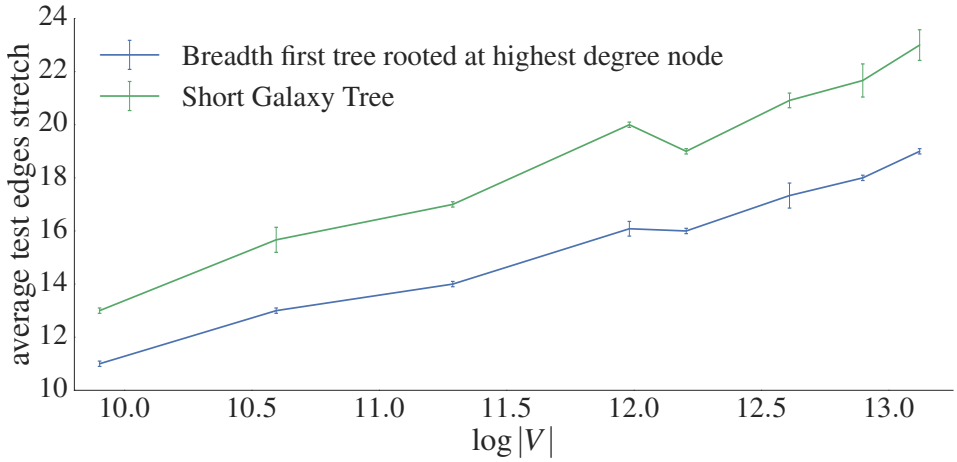
Figure 3.12 – Real world pictures and their binarized version



(a) GRID

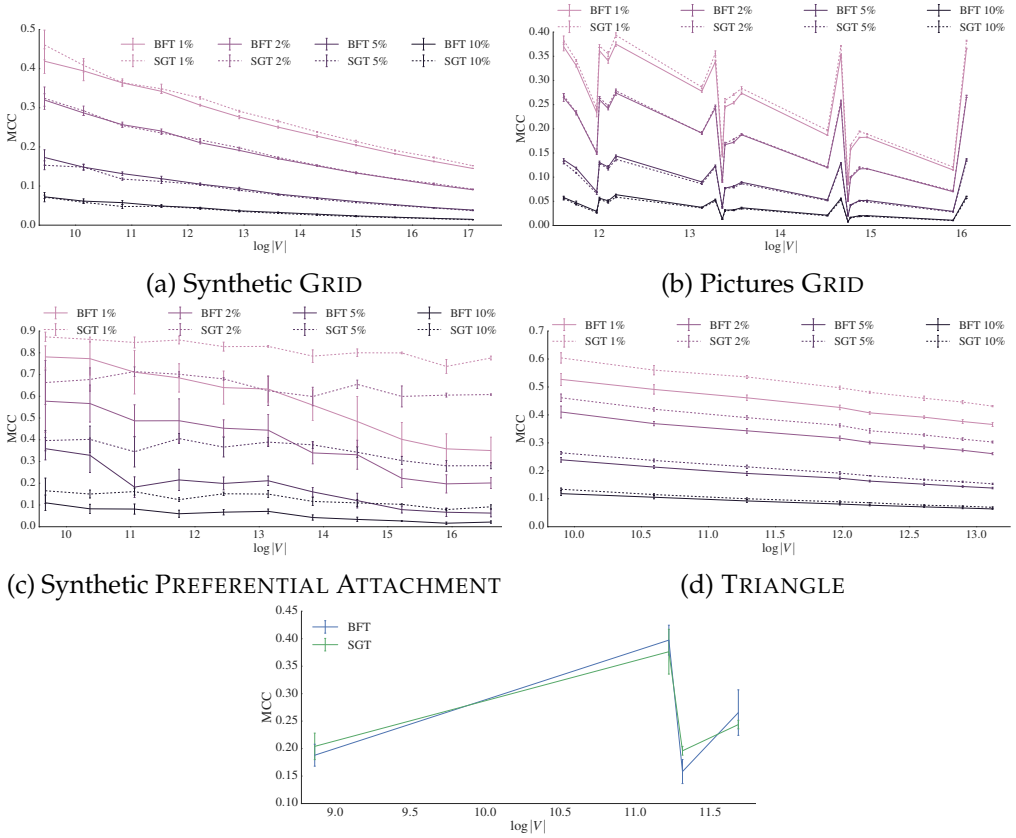


(b) PREFERENTIAL ATTACHMENT



(c) TRIANGLE

Figure 3.13 – Stretch over graphs of increasing size



(a) Synthetic GRID

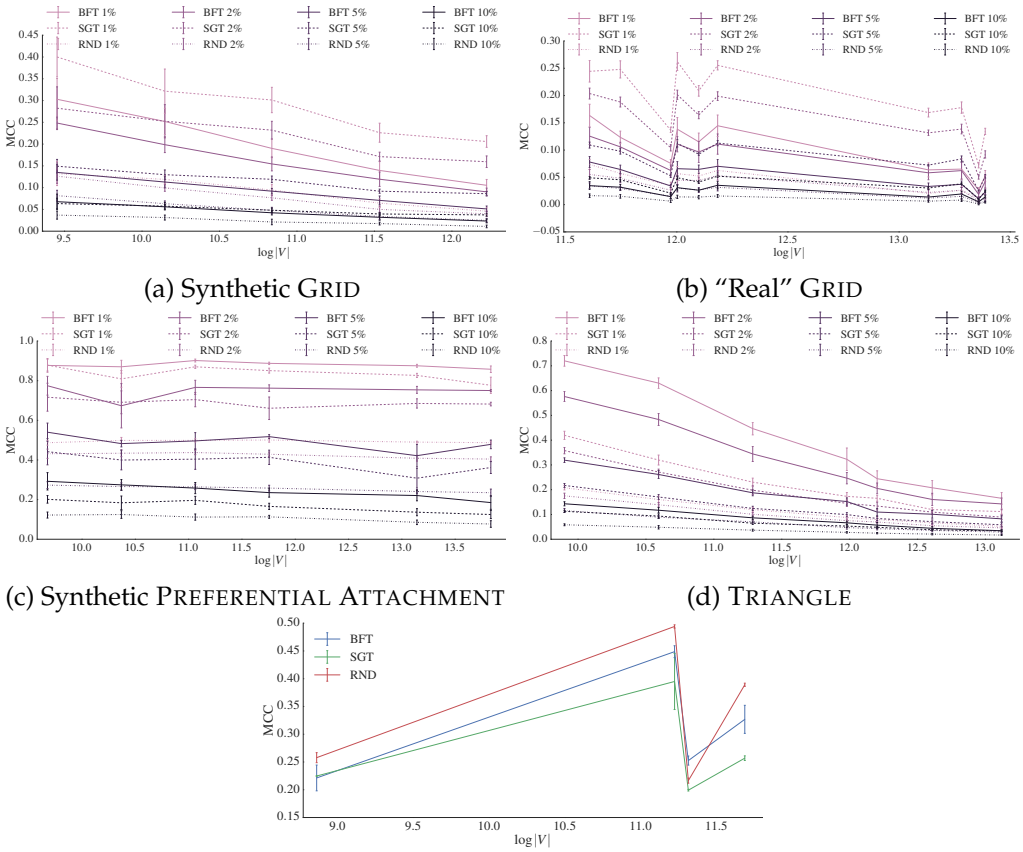
(b) Pictures GRID

(c) Synthetic PREFERENTIAL ATTACHMENT

(d) TRIANGLE

(e) Real world network

Figure 3.14 – MCC over various graphs



(a) Synthetic GRID

(b) "Real" GRID

(c) Synthetic PREFERENTIAL ATTACHMENT

(d) TRIANGLE

(e) Real world network

Figure 3.15 – *A sym exp* over various graphs

Chapter 4

Edge clustering in node attributed graphs

In this chapter, we extend the problem of predicting edge types beyond the binary case. Furthermore, we assume that we are provided with node attributes in addition to the mere graph topology we used in the previous chapters. Our motivation is twofold: intrinsic and extrinsic. First, we want an interpretable model that, given two connected nodes, can explain why they are linked. This leads us to use a linear model, for it is both easily amenable to interpretation and computationally efficient. Second, edge-labeled graphs are susceptible to further studies by existing networks analysis methods, as we will describe shortly. Thus, our method is a useful way of enriching data to extract new insights from it.

As demonstrated by our prior extensive coverage, predicting edge sign is an interesting and challenging problem in its own right. Yet it is natural to extend and generalize it. First, there are other edge binary classification problems that arise in graphs. For instance, predicting whether a link in a social network is strong or weak [De +14; RTG17]; predicting whether two authors in a citation network are advisor-advisee or regular coauthors [Wan+10]; predicting whether two persons in a company network have a relationship of manager-subordinate or not [DNG07]; predicting whether two proteins interact physically or genetically; or whether two neurons have a chemical or ionic link between them [NL15].

Second, and more generally, we are also interested in multiclass problems, within the context of multilayer graphs [Kiv+14; Boc+14]. While Kivela *et al.* [Kiv+14] define an elaborate and highly expressive notion of multilayer graph, in this chapter we shall restrict ourselves to the following[1]:

Definition 4.0.1 (k-multilayer graph)**.** *A k-multilayer graph G is a pair $(V, \{E_1, \dots, E_k\})$, where V is a set of n nodes and each $E_i \subset V \times V$ is a set of edges, for $i \in [\![k]\!]$. We call the subgraph $G_i = (V, E_i)$ the i^{th} layer of G and it will be convenient to think of G as the superposition of its k layers, so that by a slight abuse of notation we can write $G = \bigcup_{i=1}^{k} G_i$.*

We naturally identify each layer with a type of relation, so that our classification problem is to decide to which layer a given edge belongs. Furthermore, we make the simplifying assumption that two nodes can be connected in at most one layer (i.e. $\forall u, v \in V,\ (u, v) \in E_i \Rightarrow \forall j \neq i,\ (u, v) \notin E_j$), meaning that the resulting superposition of all layers does not turn into a multigraph. As mentioned in the introduction, such multigraphs can be used to model different relation types in networks related to social interactions [ST10; KMK11], bibliographical citation [Cai+05; DKK11], economic exchange [Kal+10; BFG10; CMM15] and biology [Li+11; Kéf+15; MB16]. This additional information on the nature of the relationships allow traditional graph tasks to be performed more precisely and with higher granularity. Consider for instance node ranking [Cos+13]; node clustering [Bot+15, Section 2; Kiv+14, Section 4.5.1] and the study of information spreading [Sal+15]. However, to produce

[1] Which roughly corresponds to the *single aspect*, *node-aligned* multilayer graph of [Kiv+14].

such finer results, the methods we just referenced require labeled edges that in many cases are not available, thus justifying the usefulness of our work. Indeed, in Table 4.1, we give some examples of networks whose edges are currently unlabeled and what type of relations they could possibly denotes. In other words, given a small list of possible reasons in terms of attributes and two nodes, why are they connected?

Table 4.1 – Real attributed graphs that we have collected or are easily accessible. Unfortunately, none of them comes with known edge types, which implies the last column is merely speculation.

Source	Node	Edge	Node profiles	Possible edge meanings
DBLP	author	citation	keyword count of their papers	scientific topic
Github	developer	following	popularity, number of lines written in various programming languages	common hobby, involvement in the same open source project, same workplace, famous role model
Amazon	video game products	"frequently bought together"	price, genre, time of release, popularity and evaluation	part of a series, cheap bundle, same genre and level of satisfaction
Wikipedia	article	internal link	category, length, edit history, bag of words	generalization, specialization, organisation
Flickr	user	following	activity statistics, groups membership, count of photo tags	geography, same tags distribution, famous role model,
Foursquare	venue	"frequently visited together"	location, popularity, category, the time distribution at which they are visited	closeness, part of a "series" (e.g. movie theater $\rightarrow$ restaurant $\rightarrow$ bar $\rightarrow$ club)
IMDB	movie	sharing actors	genre, release date and popularity	part of a series, common theme, same time period and genre
BlaBlaCar	driver or passenger	shared ride	experience, location, preferences, rating	geography, professional, leisure

Besides predicting potentially more than two classes, the other difference with the setting of the previous chapters is that we now assume that nodes are associated with attributes, which we call *profiles*. One could argue this makes our framework less general, for attributes are not always available, or may come with missing values. Another recent finding is that, despite a natural assumption, the nodes profiles in attributed graphs are not necessarily correlated with known ground truth communities [HDF14], casting further doubt on the usefulness of that extra information. This raises the question of why we cannot simply extend the troll/trust method of counting the type of edges incident to every node and learn a model from such features. A similar tension between relying solely on always available albeit sparse topological information or leveraging possibly richer attributes information also exists in link prediction [MBC16]. However, the presence of profiles is motivated by the last fundamental difference with the previous problems. In this chapter, we do not have access to any direct supervision in the form of labels. Rather than a strict classification problem, we are thus faced with unsupervised clustering, guided by the information contained in node profiles. Note that contrary to the relation between CORRELATION CLUSTERING and EDGE SIGN PREDICTION, this time the objects we seek to cluster are edges and not nodes. Furthermore, we stress that the profiles crucially increase the interpretability of our models.

It would be tempting to assume that nodes are connected because they share the same attributes. Indeed, a common learning bias in graphs is that nodes are connected together because they are similar, or as often said proverbially: "birds of a feather flock together". Formally this is called the homophily principle [MSC01] and has been consistently verified, both in offline [HL78; Mar88] and online social networks [The09b; BAX12]. While our every day life experience makes homophily

not surprising in social networks, the same assortative patterns [New02] are present in other kinds of networks [MF09]. Here though, we assume that links cannot only be explained by a global homophily along all attributes but rather by partial homophily, partial heterophily or even both. An example of this combination is the balanced news diet problem [Gar+17], where we try to connect users of a social network to prevent them from being locked into topical echo chambers. In that situation, a user A might be more likely to listen to the opposite point of view of user B about gun control if they both share common demographics and interests. This suggests, as we make it formal in the next section, that node attributes can include polarized opinions, i.e. both positive and negative numerical values.

After having motivated our problem of predicting multiple edge types in attributed graphs, we carry on by stating more formally the EDGE ATTRIBUTED CLUSTERING problem in Section 4.1. Namely, we define a similarity between nodes profiles and seek a small number of explanations (corresponding to edge types) achieving good scores as prescribed by a linear model. After explaining how this problem relates to the EDGE SIGN PREDICTION problem on 2-clusters signed graphs on page 97, we introduce in Section 4.2 several methods to solve it in practice. This ranges from tailoring the k-MEANS algorithm to our goodness measure (Section 4.2.1), to a convex relaxation of our objective function (Section 4.2.2) all the way through a richer and higher capacity matrix formulation allowing overlapping clustering of edges (Section 4.2.3). Because finding publicly available real data with both node attributes and edge types proved difficult, we describe a synthetic model in Section 4.3, on which we perform extensive experiments. We then survey in Section 4.4 works that are related to the EDGE ATTRIBUTED CLUSTERING, while noting that very few address it directly. Finally, we suggest in Section 4.5 several directions in which our model could be extended to handle more general classes of graphs.

4.1 Attributed graphs and problem definition

We start by introducing some notations and terminology about edge types in node-attributed graphs. Then we present two options to model the interactions between nodes and the role of attributes in "explaining" edges (Section 4.1.1). Choosing one of these two options leads us to formulate the EDGE ATTRIBUTED CLUSTERING problem in Section 4.1.2. We then describe additional constraints allowing us to take into account the topology of the graph. Finally, we elaborate on the relation between EDGE ATTRIBUTED CLUSTERING and EDGE SIGN PREDICTION in signed graphs.

4.1.1 Setting and modelling

Like we wrote in the introduction, to classify edges, we make use of both the topology of the graph and the attributes of its nodes. Namely, we are given an undirected, unweighted graph $G = (V, E)$. We also have side information in addition to the graph topology. It takes the form, for every node u, of a d-dimensional feature vector $x_u \in \mathbb{R}^d$ that we call the *profile* of u. Because we want to model heterophily, we more explicitly have that $x_u \in [-1, 1]^d$. To justify this statement, let us take the example of encoding the gender of a user in a social network, assuming there is only two values possible: male and female. Traditionally, this attribute would be set to 1 if the user is female and 0 otherwise. Instead, we use $+1$ and -1, to clearly emphasize the difference between the two values. The same is valid for continuous variables. For instance, an attribute could range from -1 to $+1$ to represent the ideological stance of a user on the conservative/liberal scale [BMA15].[2] This applies

[2]Note of course that in both cases, negative values do not represent a judgement but are merely a reversible mathematical convenience.

even to non diverging positive quantities such as age, which can be projected to $[-1, 1]$ using pre-processing like standardization or z-score.

Stacking all these node profiles in a $|V| \times d$ matrix X, our input so far is (G, X). As it is standard in classification problem, we are also given the number of edge types k.[3] Moreover, recall that we do not merely want to classify edges into type, but also provide an explanation of why two nodes are connected. This requires the introduction of three further concepts: 1) how to represent a connection (i.e. an edge), 2) how to represent an edge's explanation, and 3) how to evaluate which explanation is the best. We now present our modelling choice:

1) We naturally describe an edge (u, v) through the profiles of its endpoints. More precisely, we combine x_u and x_v by an operator s defined as:

$$s(u, v) = x_u \circ x_v = s_{uv} \in \mathbb{R}^d \,,$$

where $\circ$ stands for the Hadamard (or component wise) product. As showed in Table 4.2, for each dimension $i \in [\![d]\!]$, the vector s_{uv} holds a value indicating whether the profiles of u and v agree, disagree or are indifferent along that dimension.

Table 4.2 – We show the effect of the Hadamard product in combining information from two users u and v. Instead of actual numbers, in this table we use $+$ to denote a large positive value, $-$ to denote a large negative value, 0 to denote a small absolute value and $\star$ to denote an arbitrary value.

$x_{u;i}$	$x_{v;i}$	$s_{uv;i}$
$+$	$+$	$+$
$-$	$-$	$+$
$+$	$-$	$-$
0	$\star$	0

2) The k explanations take the form of bounded norm vectors $\mathcal{D}_k = \{w_1, w_2, \ldots, w_k\} \subset \mathbb{B}^d$, where $\mathbb{B}^d$ is the ℓ_2-unit ball of $\mathbb{R}^d$. We call such a vector w_ℓ, for $\ell \in [\![k]\!]$, a *direction*. As we shall explain more formally later on, those directions are exactly the parameters learned while solving our problem.

3) Given an edge (u, v), all the k directions $\{w_1, w_2, \ldots, w_k\}$ provide a possible explanation of why u and v are connected. We score these explanations by assigning them a *goodness* of explanation[4] defined as:

$$g(s_{uv}, w_\ell) = {s_{uv}}^T w_\ell \,.$$

As the w notation hints, a direction can be interpreted as a set of weights. These weights indicates to which extent each attribute contributes to the connection, according to that direction. For a given set of directions $\mathcal{D}_k$, the classification of edges is naturally performed by the following operator $\mathcal{E}$, returning the index of the direction with the maximal goodness:

$$\begin{array}{rccl} \mathcal{E}: & E \times \left(\mathbb{R}^d \times \ldots \times \mathbb{R}^d\right) & \longrightarrow & [\![k]\!] \\ & (u, v), \mathcal{D}_k & \longmapsto & \arg\max_{\ell \in [\![k]\!]} g(s_{uv}, w_\ell) \end{array}$$

Because it will be clear from context, we lighten the notation and simply write $\mathcal{E}(u, v)$ instead of $\mathcal{E}\left((u, v), \mathcal{D}_k\right)$.

This choice of s and g enjoy two interesting properties. First, it encompasses all cases of homophily, indifference and heterophily. For instance, given $x_u = \begin{pmatrix} 1 & 0 & -1 \end{pmatrix}^T$ and $x_v = \begin{pmatrix} -1 & 1 & -1 \end{pmatrix}^T$, the direction $w = \begin{pmatrix} -1/\sqrt{2} & 0 & 1/\sqrt{2} \end{pmatrix}^T \in \mathbb{B}^d$

[3] Because our problem is essentially unsupervised, assuming that we are given k is a simplification, on which we elaborate on page 96.

[4] Whereas the name is inspired by the concept of *goodness of fit*, there is no formal relation.

achieves the maximal goodness. In doing so, it highlights that u and v disagree on the first component (i.e. heterophily), agree on the last component (i.e. homophily) and that the middle component is irrelevant (i.e. indifference). This also shows that, second, given the linear nature of the goodness g once s_{uv} has been computed, it provides interpretable explanation of why two nodes are connected.

Those two properties of s and g should not be taken for granted. Indeed, a natural choice for the operator s is the difference between the profiles, that is $s_{uv} = x_u - x_v$. In that spirit, if u and v are close along a direction w_ℓ, we expect ${x_u}^T w_\ell$ to be close to ${x_v}^T w_\ell$, as measured by $|{x_u}^T w_\ell - {x_v}^T w_\ell|$. And because it is easier to deal with smooth functions, we define the goodness to be $g(s_{uv}, w_\ell) = -\left({s_{uv}}^T w_\ell\right)^2$, where the minus preserve the semantic than the "better" the direction w_ℓ, the larger the goodness. However, with this formulation, there is no way to set $w_{\ell;i}$ in order to "highlight" the fact that $x_{u;i}$ and $x_{v;i}$ are widely different. Furthermore, the square in the goodness expression loses the linearity.

We conclude with a last comment on the norm of the directions. While being exactly of norm one is not primordial, the role of bounding the norm of the directions (i.e. having $\sum_{i=1}^{d} w_{\ell;i}^2 \leq B^2$) is to tie the d dimensions together. Without such a constraint, and with our choice of s and g, we could simply study each dimension separately, whereas this bound adds a coupling across different attributes, since increasing the weight of an attribute mechanically affects the weights of the other attributes. For consistency, we also consider node profiles to be normalized and having unit norm. A further practical justification is that if nodes are users of a social network and the profiles measure their activity across several domains, this allows to compare users with various level of total activity.

4.1.2 Learning problem and additional constraints

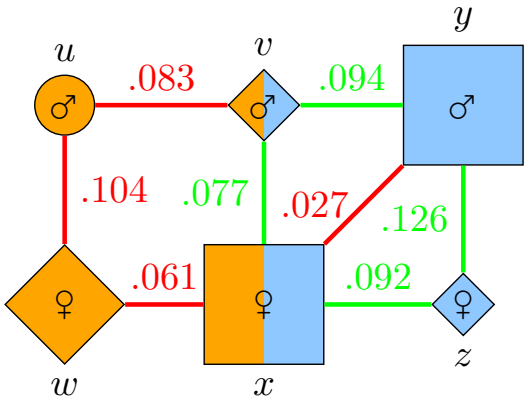
Problem definition With working definitions of s and g at hand, we now formally write down the problem of finding the directions maximizing the goodness of the graph.

Problem 5 (EDGE ATTRIBUTED CLUSTERING)**.** *Given a graph $G = (V, E)$, node profiles $X \in [-1, 1]^{n \times d}$ and an integer $k \in \mathbb{N}$, find a set of k norm-bounded directions $\mathcal{D}_k = \{w_1, w_2, \ldots, w_k\} \in \mathbb{B}^d$ and associate to every edge of E the direction with the maximal goodness. Formally, solve*

$$\underset{\mathcal{D}_k = \{w_1, \ldots, w_k\} \subset \mathbb{B}^d}{\arg\max} \sum_{u,v \in E} g(s_{uv}, w_{\mathcal{E}(u,v)}) \tag{4.1}$$

where $s_{uv} = x_u \circ x_v$, $g(s_{uv}, w_\ell) = {s_{uv}}^T w_\ell$ and $\mathcal{E}(u, v) = \arg\max_{\ell \in [\![k]\!]} g(s_{uv}, w_\ell)$.

Example In Figure 4.1, we present a small instance of the EDGE ATTRIBUTED CLUSTERING problem, where we are given the graph shown in Figure 4.1a, whose node attributes are listed in Table 4.1b. The first two attributes are categorical, denoting the company employing each user, either G or F. Instead of a classic binary one-hot encoding, we additionally use -1 to denote that a user had been employed by a given company. Gender is encoded as presented earlier, and we also apply a pre processing to transform age between -1 and 1. Finally, opinion represents the view of former F employees about a societal issue, with 0 denoting a neutral or unknown position. In Table 4.1b, we also show two directions that explain the connections in the graph G. The first covers being of the same age and same companies, especially at G, while the second directions covers former employees of F having different age and opinion. Not only are the results interpretable as claimed before, but we can also assess the strength of each explanation, using their goodness. For instance, we see that the edge between x and y has a low score, because although those two users have the same age and used to work at F, not both of them works at G.



(a) The graph G. Larger nodes represent old users while smaller ones are younger. Their gender is inscribed in the node and their shape represent their opinion: rectangle for -1, circle for 0 and diamond for $+1$. Finally, orange stands for being employed at company G and faded blue for having been employed at company F. As for the edges, the red ones are labeled by the first direction and the green ones by the second direction, and we label them with the associated (normalized) goodness.

	G	F	gender	age	opinion
u	$+1$	0	-1	-0.8	0
v	$+1$	-1	-1	-0.7	$+1$
w	$+1$	0	$+1$	0.6	$+1$
x	$+1$	-1	$+1$	0.9	-1
y	0	-1	-1	0.7	-1
z	0	-1	$+1$	-0.6	$+1$
w_1	$+3$	$+1$	0	$+1$	0
w_2	0	$+4$	0	-2	-4

(b) The numerical attributes of the six nodes. All are categorical encoded with a ternary scheme, except for age, which has been transformed to fit in $[-1, 1]$. They are followed by the (non normalized) two directions w_1 and w_2.

Figure 4.1 – A small instance of EDGE ATTRIBUTED CLUSTERING and an handcrafted solution, albeit non-optimal.

Choice of k For clarity, the previous example of Figure 4.1 has only a handful of nodes and dimensions, and can therefore reasonably be explained by two directions. But in larger graphs, the choice of k is a legitimate question. At one extreme, picking $k = |E|$ renders the topology of the graph irrelevant, as we can simply exploit the trivial solution of having a single direction $w_{uv} = \frac{s_{uv}}{||s_{uv}||}$ for every edge (u, v). More generally, as k increases, we expect the value

$$\max_{\mathcal{D}_k} \sum_{u,v \in E} \max_{\ell \in [\![k]\!]} g(s_{uv}, w_\ell)$$

to increase as well, at the cost of interpretability. There could be a principled approach to finding the "best" k, based on a information theoretical measure of the complexity of $\mathcal{D}_k$ and the minimum description length principle [Grü05]. However, in the interest of simplicity, in the following we focus on the formulation stated in Problem 5, where k is given. In practice, we further constrain k to be small (say less than 10). Within the multilayer framework, k could also be seen as the number of underlying layers. To retain a fair level of interpretability, we deem appropriate to have k not too large.

Topological constraints In addition to assuming that the input graph is the superposition of a small number of layers, another way to leverage the graph topology is to define local constraints at the node level. Indeed, one of our motivation is that it is not enough to look at edges in isolation, but rather we expect the connections of one node to influence the connections of its neighbors. We now present two of these local, node-level constraints.

First, if a graph was to perfectly follow our model, every edge (u, v) would be maximally explained by one of the k directions. This would happen if the profiles of u and v were collinear with this direction, up to sign reversal. When considering several edges, we could imagine that the nodes profiles result from the following process. Two nodes u and v establish a connection by first picking a direction $w_\ell \in \mathcal{D}_k$ and add to their base vector b_u and $b_v \in \mathbb{R}^d$ a fraction $a_{uv} > 0$ of w_ℓ. We

therefore subtract the following term

$$\mathcal{L}_{\text{node}} = \sum_{u \in V} \left|\left| x_u - b_u - \sum_{v \in \mathcal{N}(u)} a_{uv} w_{\mathcal{E}(u,v)} \right|\right|_2^2 \tag{4.2}$$

from (4.1). The cost we pay is introducing the extra parameters $\{a_{uv}\}_{u,v \in E}$ and $\{b_u\}_{u \in V}$ to be learned, but we will see in the next section how to handle that in practice. On the other hand, it has two advantages. First, this is another way, besides goodness, in which the (given) profiles provide prior information about the (unknown) directions, and thus restrain the search space. Second, this makes the directions dependent of the topology of the graph, by coupling them with the neighborhood of each node.

Second, we also make the assumption that each node u is only involved in $k_{\text{local}} < k$ different directions. The intuition is that when d is large enough, users have to focus their energy and can only express their interest in a few number of dimensions. In other words, all the edges incident to u can only be associated with one of k_{local} directions. For some values of k_{local}, this is a NP-COMPLETE graph colouring problem and thus cannot necessarily be enforced exactly. Still, it can be seen as a learning bias and will be used to generate synthetic data. Furthermore, we can also express it as an optimization constraint. Specifically, suppose that for an edge (u, v) associated with the direction $w_i \in \mathcal{D}_k$, we have a k-dimensional vector y_{uv} equal to e_i (that is all zero except for the i^{th} component set to one). Let us stack all such vectors into a $|E| \times k$ matrix Y, and let $C \in \mathbb{R}^{|V| \times |E|}$ be the incidence matrix of G[5], with c_u being its u^{th} row. Then $c_u^T Y$ is a k-dimensional vector that counts how many times each direction is incident to node u. We thus want the ℓ_0 norm of $c_u^T Y$ (i.e. the number of its non-zero component) to be upper bounded by k_{local}. This is not a convex constraint but we can relax it, for such sparsity inducing problems have been well studied [Bac+12a]. Namely, we let y_{uv} be a softmax membership k-dimensional vector, that is we set, $\forall i \in [\![k]\!]$,

$$y_{uv;i} = \frac{\exp(\beta g(s_{uv}, w_i))}{\sum_{j=1}^k \exp(\beta g(s_{uv}, w_j))},$$

with $\beta > 0$ a large positive constant. And we replace the ℓ_0 norm by a ℓ_1 norm (or even a k-support norm [AFS12]), yielding the following $\mathcal{L}_{\text{local}}$ term to be minimized:

$$\mathcal{L}_{\text{local}} = \sum_{u \in V} \left|\left| \sum_{v \in \mathcal{N}(u)} y_{uv} \right|\right|_1 = \sum_{u \in V} \left(\sum_{i=1}^k \sum_{v \in \mathcal{N}(u)} \frac{\exp(\beta g(s_{uv}, w_i))}{\sum_{j=1}^k \exp(\beta g(s_{uv}, w_j))} \right) \tag{4.3}$$

It is reasonable to further expect that two neighboring nodes share one common direction among their k_{local} ones. One way to ensure this condition is to imagine that the set of allowed k_{local} directions changes smoothly over the graph. This property depends of the topology, as we can see by considering two extreme cases: a line and a complete graph. On the line, once a node has picked two allowed directions from its two neighbors, the remaining ones can be chosen arbitrarily. On the complete graph on the other hand, this choice is much more constrained.

Connection with signed graphs The learning bias of Section 3.1.3 implicitly defined an idealized EDGE SIGN PREDICTION problem as:

Problem 6 (idealized EDGE SIGN PREDICTION)**.** *Given n nodes belonging latently to K groups, predict whether two nodes u and v belong to the same group (that is (u, v) is positive) or not (that is (u, v) is negative).*

[5]The incidence matrix of a graph G, usually denoted by B, is the $|V| \times |E|$ matrix such that $B_{i,j} = 1$ if the node v_i and edge e_j are incident and 0 otherwise.

In Section 3.3 we studied in particular the important special case where $K = 2$. Now consider an instance of Problem 5 where $k = 2$, and where the profiles and edges are based on group membership. We now check that the optimal solution of such an instance is consistent with the solution of this idealized EDGE SIGN PREDICTION problem on the same graph.

Let us first define this instance formally. Given a graph $G = (V, E)$, call the $K = 2$ groups of Problem 6 L and R. We then define the nodes profiles X as follow. We first draw uniformly at random a vector p from $\{-1, 1\}^d$, then let the profiles in L be $\frac{p}{\sqrt{d}}$ and the profiles in R be $-\frac{p}{\sqrt{d}}$. We call this instance $\mathcal{I} = (G, X, k = 2)$. With this choice, we have

$$s_{uv} = \begin{cases} \frac{\mathbf{1}}{d} = s_{\text{inner}} & \text{if } u \text{ and } v \text{ are in the same group,} \\ -\frac{\mathbf{1}}{d} = s_{\text{outer}} & \text{if } u \text{ and } v \text{ are in different groups.} \end{cases}$$

Let us denote by m_i the number of edges that are internal to L and R, and by $m_o = |E| - m_i$ the number of edges between L and R.

Proposition 4. *Let $\mathcal{D}_2 = \{w_1^\star, w_2^\star\}$ be an optimal solution of Problem 5 on the instance $\mathcal{I}$. Then $\mathcal{D}_2$ provides the following solution to Problem 6: each every edge explained by $w_1^\star$ is predicted positive, and every edge explained by $w_2^\star$ is predicted negative.*

Proof. Take any $w_1, w_2 \in \boldsymbol{S}^d \times \boldsymbol{S}^d$ such that $w_1 \neq w_2$. Assume without loss of generality that $w_1 = \arg\max_{w \in \{w_1, w_2\}} {s_{\text{inner}}}^T w$. This implies that:

$${s_{\text{inner}}}^T w_1 = \frac{1}{d} \sum_{i=1}^{d} w_{1;i} > \frac{1}{d} \sum_{i=1}^{d} w_{2;i} = {s_{\text{inner}}}^T w_2 \,.$$

In turn, we have that:

$${s_{\text{outer}}}^T w_2 = -\frac{1}{d} \sum_{i=1}^{d} w_{2;i} > -\frac{1}{d} \sum_{i=1}^{d} w_{1;i} = {s_{\text{outer}}}^T w_1 \,.$$

We can thus rewrite the maximization objective (4.1) of Problem 5 as:

$$\begin{aligned} &\arg\max_{w_1, w_2 \in \boldsymbol{S}^d} \quad \sum_{u,v \in E} \max_{w \in \{w_1, w_2\}} {s_{uv}}^T w = \\ &\arg\max_{w_1, w_2 \in \boldsymbol{S}^d} \quad \sum_{\substack{u,v \in E \\ s_{uv} = s_{\text{inner}}}} {s_{\text{inner}}}^T w_1 + \sum_{\substack{u,v \in E \\ s_{uv} = s_{\text{outer}}}} {s_{\text{outer}}}^T w_2 = \\ &\arg\max_{w_1, w_2 \in \boldsymbol{S}^d} \quad m_i \frac{\mathbf{1}^T}{d} w_1 + m_o \frac{-\mathbf{1}^T}{d} w_2 \end{aligned} \tag{4.4}$$

One can check using Lagrange multipliers that for any $c \in \mathbb{R}^n$, $\max_{w \in \boldsymbol{S}^d} c^T w = \frac{c}{\|c\|}$. The solution of (4.4) is therefore:

$$w_1^\star = \frac{\mathbf{1}}{\sqrt{d}} \qquad \text{and} \qquad w_2^\star = \frac{-\mathbf{1}}{\sqrt{d}}$$

As claimed, the edges associated with $w_1^\star$ (respectively $w_2^\star$) are the edges that are to be predicted positive (respectively negative) in Problem 6. □

This is nothing more than a sanity check, as the profiles explicitly encode the membership of each node to one of the two groups. Indeed, the proof makes no use of the dimension d of the profiles, so that $d = 1$ would be enough. Slightly more interesting is the case were those profiles are set up as before, but perturbed uniformly at random. Namely, the sign of each coordinate is changed with probability $p < 1/2$. One can then check that

$$\mathbb{E}\left[s_{\text{inner}}\right] = \frac{(1-2p)^2}{d} \mathbf{1} = -\mathbb{E}\left[s_{\text{inner}}\right]$$

and that furthermore, the variance of each component is $\frac{8(p-p^2)-16(p-p^2)^2}{d^2}$. While the previous $\{w_1^\star, w_2^\star\}$ solution is now only optimal in expectation, we see that the variance decrease quadratically with the dimension of the profiles. In practice we thus expect this solution to be very close to the optimal. Intuitively this can be seen as taking the majority of d votes on whether each node belongs to one group or the other.

> It is natural to ask how this result can be extended to more than $K = 2$ groups. The most natural idea is to use one-hot encoding of the group membership as profiles, with additional negative values as in the example of Figure 4.1. Namely, set the profiles of the i^{th} group as follow: all component are equal to $-b$ except for the i^{th} one, which is equal to a, where a and b are real constants such that $a^2 + (K-1)b^2 = 1$. However, in that case, we check numerically that $w_+ = \frac{\mathbf{1}}{\sqrt{d}}$ and $w_- = \frac{-\mathbf{1}}{\sqrt{d}}$ is the not the optimal solution, and that a solution achieving a larger objective function value does not cluster the edges between inner and outer groups.

4.2 Proposed approaches

We now introduce the five methods solving Problem 5 that we will experimentally compare in the next section. Recall that given a graph topology (V, E) and node profiles X, we first build the edges representation $\mathcal{S} = \{s_{uv}\}_{(u,v)\in E}$ and then look for k bounded directions that maximize the total goodness of the graph, possibly under topological constraints. We start with a straightforward baseline, which simply performs a

1) k-MEANS clustering of $\mathcal{S}$,

and propose a post-processing to improve it, plugging our custom goodness metric into the existing

2) LLOYD algorithm for clustering.

Because these two methods consider each edge in isolation with no regard for the topology, we then describe a convex relaxation of the objective in equation (4.1), which is further

3) COMBINED with the node constraint of equation (4.2).

Finally, we depart slightly from finding k directions by considering a low-rank matrix formulation, where each edge is assigned a linear combination of a small number of base directions, paving the way to overlapping edge clustering. More precisely, we give a convex formulation of that objective that can be solved by a

4) FRANK–WOLFE algorithm,

as well as by an alternating optimization method, where we make the low-rank factorization

5) EXPLICIT.

4.2.1 k-MEANS baseline and improvement

A simple and natural baseline is to cluster the set of all similarity edge vectors $\{s_{uv} : (u, v) \in E\}$ using the k-means algorithm, where k is the number of directions set in Problem 5. Formally, given the graph $G = (V, E)$ and the node profiles X as input, we order the edges from e_1 to e_m. Then we build the matrix $S \in \mathbb{R}^{|E|\times d}$, whose i^{th} row is the similarity $s_{uv} = x_u \circ x_v$ between the profiles of the endpoints of $e_i = (u, v)$. Running k-means on the rows of S will thus partition E into k sets of edges, call them $E_1, \ldots, E_k$. A drawback of that simplicity is that it does not solve Problem 5. Indeed, the partition of E does not provide the set of directions $\mathcal{D}_k = \{w_1, w_2, \ldots, w_k\} \in \boldsymbol{S}^d$ we are looking for. A natural way of obtaining directions from that partition is to set w_ℓ to be the normalized cluster center of E_ℓ. We refer to this method as k-MEANS, and assuming it requires T iterations to converge, it has a linear complexity of $O(Td|E|)$.

However, this does not guarantee that we have assigned the best direction to

every edge. We therefore propose the following heuristic, inspired by the Lloyd algorithm for k-means, but where the directions in $\mathcal{D}_k$ play the role of centroids. Namely, we alternate between two steps:

(i) create a new partition $\{E'_1, \dots, E'_k\}$ by assigning every edge (u, v) to the direction w maximizing its goodness ${s_{uv}}^T w$, that is

$$E'_\ell = \left\{ (u,v) \in E \;:\; w_\ell = \operatorname*{arg\,max}_{w_i \in \mathcal{D}_k} {s_{uv}}^T w_i \right\}$$

(ii) update the directions to make them optimal with respect to the edges currently assigned to them. Specifically, set $\mathcal{D}_k = \left\{ w_\ell = \frac{s_\ell}{\|s_\ell\|} \right\}_{\ell=1}^k$, where $s_\ell = \sum_{u,v \in E'_\ell} s_{uv}$ is the sum of all the similarity vectors of the edge in E'_ℓ,

until the edge assignment is stable or we reach a maximum number of iterations. By analogy with the k-means algorithm, obtaining convergence guarantees in the general case would be surprising, but in practice we expect that only a small number T of iterations would be needed. We call this method LLOYD and note it also has a linear complexity of $O(Td|E|)$.

In both cases, those methods do not take the topology of the graph into account, motivating the following approaches.

4.2.2 Convex relaxation

By unrolling the operator $\mathcal{E}$ in (4.1), we can rewrite our optimization objective as

$$\operatorname*{arg\,max}_{\mathcal{D}_k = \{w_1, \dots, w_k\} \subset \mathbb{B}^d} \sum_{u,v \in E} \max_{\ell \in [\![k]\!]} g(s_{uv}, w_\ell) \tag{4.5}$$

The difficulty in optimizing directly (4.5) stems from the inner $\max$ operator, which we replace here by a convex relaxation. Indeed, one can check that given a set of numbers $S = \{a_1, a_2, \dots, a_{|S|}\}$ and a real number $\beta > 0$, we have

$$\max_{a_i \in S} a_i = \lim_{\beta \to +\infty} \frac{1}{\beta} \log \left(\sum_{i=1}^{|S|} \exp\left(\beta a_i\right) \right)$$

Using this fact, and dividing the goodness term and the $\mathcal{L}_{\text{node}}$ term of (4.2) respectively by the number of edges and nodes so that they have comparable magnitude, we now have the following objective:

$$\operatorname*{arg\,max}_{\substack{w_1, \dots, w_k \in \boldsymbol{S}^d, \\ a_{uv} \in \mathbb{R},\, b_u \in \mathbb{R}^d}} \frac{1}{|E|\beta} \sum_{u,v \in E} \log\Big(\sum_{\ell=1}^k \exp\left(\beta {s_{uv}}^T w_\ell\right) \Big) - \frac{\mu}{|V|} \sum_{u \in V} \left\| x_u - b_u - \sum_{v \in \mathcal{N}(u)} a_{uv} w_{\mathcal{E}(u,v)} \right\|^2, \tag{4.6}$$

where $\mu > 0$ is a trade-off parameter.

Note that in the second term, $w_{\mathcal{E}(u,v)} = \operatorname{arg\,max}_{w_\ell \in \mathcal{D}_k} {s_{uv}}^T w_\ell$. However, the $\operatorname{arg\,max}$ function is again not convex and we thus use the same relaxation as before. Namely, this relaxation $\mathcal{A}$ takes as input a k-tuple of directions and return their sum weighted by the softmax function as follow:

$$\begin{array}{rccl} \mathcal{A}: & \mathbb{R}^d \times \ldots \times \mathbb{R}^d & \longrightarrow & \mathbb{R}^d \\ & (w_1, \dots, w_k) & \longmapsto & \sum_{i=1}^k \frac{\exp(\beta {s_{uv}}^T w_i)}{\sum_{j=1}^k \exp(\beta {s_{uv}}^T w_j)} w_i \end{array}$$

This function has an explicit derivative with respect to any w_i, but it is rather costly to compute and in practice we found automatic differentiation [MDA15] to be more efficient.

We make two further simplifications. First, notice that the set $\boldsymbol{S}^d \times \ldots \times \boldsymbol{S}^d$ is not convex. To ease the optimization procedure, we thus only impose the directions

w_ℓ to lie within the unit ball and expect them to have norm close to one in order to maximize the dot product in the ${s_{uv}}^T w_\ell$ term[6]. Second, while (4.6) requires to optimize $kd + |E| + |V|d$ parameters, we fix $a_{uv} = \deg(u)$ and $b_u = \mathbf{0}$ for all edges and nodes in order to be independent of the size of the graph. The final problem solved by our COMBINED method using (full) projected gradient descent is therefore:

$$\underset{w_1,\dots,w_k \in \mathbb{B}^d}{\arg\max} \frac{1}{|E|\beta} \sum_{u,v \in E} \log\Big(\sum_{\ell=1}^{k} \exp\left(\beta {s_{uv}}^T w_\ell\right)\Big) - \frac{\mu}{|V|} \sum_{u \in V} \left|\left| x_u - \frac{1}{\deg(u)} \sum_{v \in \mathcal{N}(u)} \mathcal{A}(w_1, \dots, w_k) \right|\right|^2 \tag{4.7}$$

4.2.3 Matrix optimization

Besides relaxing the max operator from (4.1), another solution to remove it would be to increase k to be equal to $|E|$. As we already mentioned, this makes the problem trivial. To avoid this, we add the constraint that all those $|E|$ directions are linear combinations of a small number k of basis directions. In order to formulate this more clearly, we rewrite the previous objectives with matrices. We first show this can be expressed in a form that is amenable to the Frank–Wolfe algorithm. Then we take the alternative road of explicit low-rank matrix factorization, and we conclude by commenting on the respective merits and disadvantage of all these approaches.

4.2.3.1 FRANK–WOLFE method

Seeking one direction for every edge, our goal is thus to maximize $\sum_{u,v \in E} {s_{uv}}^T w_{uv}$. This objective can be written in matricial form as $\operatorname{tr}(SW) = \langle S^T, W \rangle_{\mathrm{F}}$, where $S \in \mathbb{R}^{|E| \times d}$ is the s_{uv} vectors for all edges stacked vertically and $W \in \mathbb{R}^{d \times |E|}$ is the matrix of all directions stacked horizontally and $\langle A, B \rangle_{\mathrm{F}} = \operatorname{tr}\left(A^T B\right)$ is the Frobenius inner product of two real matrices. More specifically, because of the unit norm constraint on directions, we look for W in the set $\mathbb{M}^{d \times |E|}$ of $d \times |E|$ matrices with unit ℓ_2 norm columns. To avoid the trivial solution of letting each direction w_{uv} merely be $\frac{s_{uv}}{\|s_{uv}\|}$, we further constraint W to be of low rank. That is, what we want to optimize is

$$\min_{W \in \mathbb{M}^{d \times |E|}} -\langle S^T, W \rangle_{\mathrm{F}} + \operatorname{rank}(W)\,.$$

However, while the Frobenius inner product is linear in W, the rank operator is highly irregular and optimizing it directly is NP-COMPLETE. Therefore, we relax the rank term by the nuclear norm of W, defined as $\|W\|_* = \sum_{i=1}^{\min(d,|E|)} \sigma_i(W)$, where $\sigma_i(W)$ is the i^{th} largest singular value of W. The intuition is that the rank of W is equal to the number of non-zero singular value, while the nuclear norm is essentially an ℓ_1 norm on the spectrum of W. Therefore, as minimizing the ℓ_1 enforces sparsity, minimizing the nuclear norm tends to make the rank smaller by ensuring the presence of zeros in the singular values of W. [7] This results in the following optimization problem:

$$\min_{W \in \mathbb{M}^{d \times |E|}} -\langle S^T, W \rangle_{\mathrm{F}} + \lambda \|W\|_*\,,$$

where λ is a regularization parameter. Since $\|\cdot\|_*$ is a norm, this objective function is convex (although the domain over which we optimize is not), but it does not have a gradient, forcing us to rely on potentially costly proximal gradient descent methods [PB14]. Instead, we consider the equivalent problem obtained by replacing

[6] Indeed, since the first term is convex, its maximum can only be reach at the boundary of the feasible domain [Roc70, Theorem 32.1]. However, we are also subtracting another convex term and the resulting function is therefore more complicated.

[7] As mentioned in Section 2.4 about [Wan+17a], a tighter convex relaxation of the low-rank constraint is the max norm, but we do not pursue that venue further.

the regularization parameter with an upper bound δ on the nuclear norm of W, yielding:

$$\min_{\substack{W \in \mathbb{M}^{d \times |E|} \\ \|W\|_* \leq \delta}} -\langle S^T, W \rangle_{\mathrm{F}} .$$

The remaining issue is that $W \in \mathbb{M}^{d \times |E|}$ is not a convex constraint. As in the previous formulation, we thus relax it by simply imposing that the ℓ_2 norm of the columns of W are not too large. In fact, we already have a bound on these column norms, because of the nuclear norm constraint.

Proposition 5. *For any $W \in \mathbb{R}^{d \times |E|}$ such that $\|W\|_* \leq \delta$, the i^{th} column $W_{:,i}$ of W satisfies $\|W_{:,i}\|_2 \leq \delta\sqrt{d}$.*

Proof. Let W be a matrix in $\mathbb{R}^{d \times m}$ with $d < m$ such that $\|W\|_* \leq \delta$. Furthermore, let its singular value decomposition be such that we can write $W = \sum_{l=1}^{d} \sigma_l u_l v_l^T$, where u_l and v_l are d unit-norm vectors, of dimension d and m respectively. We can thus express the general term $W_{k,i}$ as $\sum_{l=1}^{d} \sigma_l v_{l,i} u_{l,k}$. Because v_l is unit norm, $v_{l,i} \leq 1$ and

$$W_{k,i}^2 \leq \left(\sum_{l=1}^{d} \sigma_l u_{l,k} \right)^2 \leq \sum_{l=1}^{d} \sigma_l^2 \sum_{l=1}^{d} u_{l,k}^2$$

using the Cauchy–Schwartz inequality. The norm of the i^{th} column of W then satisfies:

$$\|W_{:,i}\|_2 = \left(\sum_{k=1}^{m} W_{k,i}^2 \right)^{\frac{1}{2}} \leq \left(\sum_{k=1}^{m} \left(\sum_{l=1}^{d} \sigma_l^2 \sum_{l=1}^{d} u_{l,k}^2 \right) \right)^{\frac{1}{2}} = \left(\left(\sum_{l=1}^{d} \sigma_l^2 \right) \left(\sum_{l=1}^{d} \sum_{k=1}^{m} u_{l,k}^2 \right) \right)^{\frac{1}{2}}$$

From $\sum_{l=1}^{d} |\sigma_l| \leq \delta$, we have that $\sum_{l=1}^{d} \sigma_l^2 \leq \delta^2$. Combined with u_l being unit norm, we conclude that $\|W_{:,i}\|_2 \leq \delta\sqrt{d}$. □

However, setting $\delta = 1/\sqrt{d}$ is too stringent in high dimension, and because this inequality is rather loose, it will likely results in very small column norms. Instead of relying solely on the nuclear norm constraint, we therefore add an additional regularization term equal to $\mu \sum_{i=1}^{|E|} \|W_{:,i}\|_2^2 = \mu \langle W, W \rangle_{\mathrm{F}}$. Our final convex formulation is then:

$$\min_{\substack{W \in \mathbb{R}^{d \times |E|} \\ \|W\|_* \leq \delta}} \langle \mu W - S^T, W \rangle_{\mathrm{F}} . \tag{4.8}$$

Because (4.8) is of the form $\min_{x \in \mathcal{X}} f(x)$ where f and $\mathcal{X}$ are convex, we can solve it using the Frank–Wolfe algorithm [FW56; Jag13], which enjoys properties like a $O(\frac{1}{t})$ convergence rate, a low computational cost by avoiding projection step and the sparsity of its solution. It is an iterative procedure that maintains an estimate $x^{(t)}$ of the solution and works succinctly as follow: it linearizes the objective f at the current position $x^{(t)}$ by computing $\nabla f(x^{(t)})$, it finds a minimum $s^{(t)}$ of that linearization within the domain $\mathcal{X}$ by solving $\arg\min_{s \in \mathcal{X}} \langle s, \nabla f(x^{(t)}) \rangle$ and it moves toward that minimizer by a step γ, setting $x^{(t+1)} = (1 - \gamma) x^{(t)} + \gamma s^{(t)}$.[8] In the case of (4.8), the linear minimization step amounts to $\arg\min_{\|W\|_* \leq \delta} 2\mu W - S^T = \arg\max_{\|W\|_* \leq \delta} S^T - 2\mu W$. Moreover, the δ-ball of the nuclear norm is the convex hull of the matrix of the form $\delta u v^T$ with $\|u\|_2 = 1 = \|v\|_2$ [Jag13]. By letting $u_1^{(t)}$ and $v_1^{(t)}$ be respectively the left and right vectors associated with the largest singular value of $S^T - 2\mu W^{(t)}$, the minimizer $s^{(t)}$ we are looking for is therefore $\delta u_1^{(t)} v_1^{(t)^T}$, and we have $W^{(t+1)} = (1 - \gamma) W^{(t)} + \frac{t}{2+t} \delta u_1^{(t)} v_1^{(t)^T}$. We perform a small number T of such iterations, and these two eigenvectors are computed in a time linear in the

[8] γ is either found by a line search or, in our case, set to $2/2+t$.

number of non-zero entries of $S^T - 2\mu W^{(t)}$ [KW92]. Thus the overall complexity of this method, which we call FRANK–WOLFE, is $O(Td|E|)$.

As said at the beginning of this section, the matrix formulation is not exactly solving Problem 5. Therefore, for evaluation purposes, here we describe two ways to extract k directions and an edge assignment from the matrix W obtained through this FRANK–WOLFE method. The first is rather straightforward. It performs a k-means clustering of the columns of W and use it as assignment, while the directions are the normalized cluster centers. The second involves an extra optimization step. Specifically, letting r be the rank of W, we first compute a reduced SVD such that $W = U\Sigma V^T$, and then let $P = U \in \mathbb{M}^{d\times r}$ and $Q^T = \Sigma V^T \in \mathbb{R}^{r\times|E|}$. We then note that for any invertible matrix $R \in \mathbb{R}^{r\times r}$, we have $W = PR^{-1}RQ^T$. Thus we look for a matrix R that makes the columns of RQ^T as close as possible of having unit ℓ_2-norm, that is $R = \min_{R\in\mathbb{R}^{r\times r}} \sum_{i=1}^{E} \left| \|RQ^T_{:,i}\|_2^2 - 1 \right|$. Finally, as before, we cluster the columns of RQ^T using k-means, and use the columns of PR^{-1} as basis directions, noting they are not necessarily unit norm anymore.

4.2.3.2 EXPLICIT low rank factorization

We can avoid the rank regularization term altogether by making the low-rank decomposition of W explicit, that by writing $W = PQ^T$ and optimizing over both P and Q. We thus call this method EXPLICIT. However, this comes at cost of convexity. Indeed, given a rank $k \leq \min(d, |E|)$, the problem becomes, assuming A and B are fixed:

$$\min_{P\in\mathbb{M}^{d\times k}, Q\in\mathbb{M}^{|E|\times k}} \quad \tilde{h}(P,Q) = -\langle S^T, PQ^T\rangle_{\mathrm{F}} + \mu\|X - B - C\left(A \circ QP^T\right)\|_F^2 \quad (4.9)$$

The second term corresponds to the node loss term of equation (4.2), expressed in matricial form as $\|X - B - C\left(A \circ W^T\right)\|_F^2$, where $B \in \mathbb{R}^{|V|\times d}$ is the node bias vectors b_u stacked vertically, $C \in \mathbb{R}^{|V|\times|E|}$ is the incidence matrix of G, $A \in \mathbb{R}^{|E|\times d}$ is the matrix whose each row is the corresponding $a_{uv}\mathbf{1}$ vector and $\|\cdot\|_F$ denotes the Frobenius norm. As previously, we set $B = 0$ and the rows of A to be one over the degree of the node at the origin of the corresponding edge.

While (4.9) is not jointly convex in both P and Q, it is convex in one of them when the other is fixed. Therefore it can be solved via alternating optimization over P and Q [BH02] using standard (projected) gradient descent algorithms, with the following partial derivatives:

$$\frac{\partial\tilde{h}}{\partial P} = S^TQ + 2\mu A^T \circ (((X - B)^T - A^T \circ (PQ^T)C^T)C)Q$$
$$\frac{\partial\tilde{h}}{\partial Q} = SP + 2\mu A \circ (C^T(X - B - C(A \circ (QP^T))))P.$$

Because the problem is not jointly convex, we are less concerned with the domains of the variables not being convex, as we anyway have no guarantee to find a global minimum. We also do not add a regularization term of the norm of the columns of PQ^T. Indeed, since $P \in \mathbb{M}^{d\times k}$ and $Q \in \mathbb{M}^{|E|\times k}$, each column of the resulting PQ^T is a linear combination of k unit norm vectors of P with weights whose square sum to 1 and thus their norm is bounded by $\sqrt{k}$. As opposed to all the previous methods, computing the full gradient is not linear in the number of edges. Indeed, computing the product S^TQ requires $O(|E|^\omega)$ arithmetic operations, where $\omega \geq 2$ is the optimal exponent of matrix multiplication algorithm. This could be mitigated by using stochastic gradient descent. Finally, to obtain a solution of the EDGE ATTRIBUTED CLUSTERING Problem 5, we take the directions to be the columns of the resulting P, and the assignment to be a k-means clustering of the rows of Q.

Discussion While the EXPLICIT formulation is not convex, in practice we initialize it with the solution from COMBINED and it thus converge after few iterations. Namely, we set $P_0 \in \mathbb{M}^{d\times k}$ to be the final directions found by COMBINED and the rows of $Q_0 \in \mathbb{M}^{|E|\times k}$ to be equal to the corresponding $e_{\mathcal{E}(u,v)}$. Furthermore, in the projection step of the Q variable, we actually ensure that all the entries of Q are positive, which makes them easier to interpret as weights for the basis directions of P. This is something that we cannot guarantee on the decomposition of the matrix W obtained by the FRANK–WOLFE method. On the other hand, if we run the FRANK–WOLFE algorithm for more than k iterations, the rank of W can be higher than k, which might allow a finer clustering of the edges.

Compared with the COMBINED approach of Section 4.2.2, these two matrix formulations require learning more parameters. Indeed, it increases from dk to $d|E|$, as it each edge has a different direction, albeit made up of a small number of basis directions. In the case of EXPLICIT, we could furthermore impose a sparsity constraint on the columns of Q to reduces the number of coefficient to learn. Yet this additional complexity has the advantage of allowing overlapping clustering of the edges. This is useful in cases where two nodes have a relationship that is best described by a combination of the k base directions. Moreover, in the case of FRANK–WOLFE, this additional flexibility comes with no increase in computational cost.

4.3 Synthetic experiments

In the previous section, we introduced five methods to address Problem 5, which we now compare in our experiments. Specifically, this includes:

1. the k-MEANS baseline,
2. our LLOYD-like iterative refinement,
3. the COMBINED optimization of the two terms in (4.7),
4. the FRANK–WOLFE convex optimization of (4.8) and
5. the EXPLICIT explicit low-rank factorization, optimizing (4.9).

They all take as input a node-attributed graph, that is a pair $G = (V, E)$ and X. For evaluation purposes, we extract from their output a set of k directions and an assignment of every edge to one of these directions. We try those five methods on data generated synthetically to fit our model described previously in Section 4.1.1, and study how they perform under various conditions.

4.3.1 Data generation

While there exists several methods to generate attributed graph with community structure [YML13; Xu+14b; Kat+16], here we present the one we devised for our initial experiments, as it is specifically tailored to the model introduced in Section 4.1.1. It takes the following parameters as input:

- the number of nodes n,
- the dimension of their profiles d,
- the total number of directions k,
- the maximum number k_{local} of directions incident to each node, and
- an integer n_o controlling to which extent two distinct directions explain common dimensions.

It then returns:

- a graph topology $G = (V, E)$,
- a profile matrix $X \in \mathbb{R}^{n\times d}$,
- a set of k directions $\mathcal{D}_k \subset \mathbb{B}^d$, and
- an assignment from every edge (u, v) to a direction index $y_{uv} \in [\![k]\!]$ such that in most cases, $\mathcal{E}(u, v) = y_{uv}$. In other words, the direction assigned to (u, v) is

indeed the one achieving maximum goodness. We will explain later why this is not always the case.

Our methods essentially proceeds as follow:

1) We create an appropriate number of small Erdős-Rényi subgraphs and assign to each node a set of k_{local} directions such that most pairs of adjacent nodes have at least one directions in common (as we motivated after having introduced equation (4.3) on page 97).
2) We connect the blocks by several pairs of nodes having one direction in common, and pick for each edge (u, v) a direction index $y_{uv} \in [\![k]\!]$ among the ones shared by its endpoint.
3) We draw the k directions at random, ensuring they respect the parameter n_o.
4) Finally, we optimize the profiles so as to simultaneously maximize the edges goodness, minimize the term $\mathcal{L}_{\text{node}}$ of equation (4.2) and enforces as much as possible that for every edge $(u, v) \in E$, $\mathcal{E}(u, v) = y_{uv}$.

Steps 1 and 2 generate a graph topology (V, E), while steps 3 and 4 generates directions and profiles $(\mathcal{D}_{\|}, X)$. Note however that step 3 is independent of the steps 1 and 2, while step 4 is dependent on all the previous steps. We now describe each of these steps in more details.

1) We first create small Erdős-Rényi subgraphs [ER59; Gil59] that we call *blocks*. Then we assign to each node a k_{local}-tuple of directions such that two adjacent nodes have *at least* one direction in common. For convenience, let us first identify directions with colors and thus call such a k_{local}-tuple of directions a *palette*. For a given palette p, we call adjacent palettes, denoted $\text{adj}(p)$, all the palettes different from p[9] but sharing one color with p. Finally, we say that an edge u, v is *colorless* if the palette of u and v have no color in common. Because in the general case, it is not always possible to assign a palette to every node such that there is no colorless edge,[10] we now describe a simple heuristic. It performs a breath first visit of a subgraph, starting from a random node. Upon visiting an uncolored node u, it builds a set P of palette respecting the colouring constraint. Namely, for every colored neighbor v of u, it retrieves the adjacen palettes of v's palette and builds their intersection:

$$P = \bigcap_{v \in \mathcal{N}(u)} \text{adj}(p_v)\,.$$

Then it selects uniformly at random a palette from P, or an arbitrary palette if P is empty: see Figure 4.2 for a small illustration. Once all nodes have been colored, we count the number of edges that are colorless. We repeat this procedure, keeping track of the palettes assignment minimizing the number of colorless edges.

2) Once we colored the nodes of every block, we look at pair of blocks, build a list of all edges between blocks that are not colorless and sample from it to connect blocks. The last step is to a assign a color y_{uv} to each edge (u, v) from the shared color of its endpoint (or an arbitrary color for colorless edges). We note that this is reminiscent of the stochastic block model, although the probabilities of edges between blocks are not uniform, for they depend of the palette assigned to every node.

[9] We ruled out p being adjacent to itself to avoid the trivial solution of all nodes being assigned the same palette.

[10] For instance, consider a 4-clique with $k = 4$ and $k_{\text{local}} = 2$. Without loss of generality, say we assign the palette $(1, 2)$ to node 1. The adjacent palettes are then $(1, 3)$, $(1, 4)$, $(2, 3)$ and $(2, 4)$. If we assign those starting by 1 (respectively 2) to the second and third node, then the fourth palette has to contain a 1 (respectively 2), which is not possible (because two connected nodes cannot have the same palette). On the other hand, if we assign $(1, 3)$ to the second, the third node can only have $(2, 3)$, meaning the fourth node has again no palette available. The same situation arises with $(1, 4)$ and $(2, 4)$ respectively.

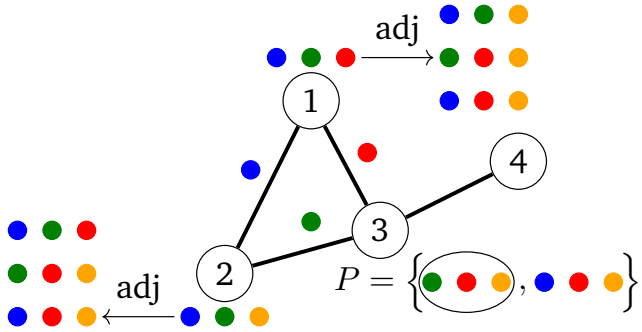


Figure 4.2 – A small example of the node palette assignment, with $k = 4$ colors (blue, green, red and orange) and palettes of size $k_{\text{local}} = 3$. We assume nodes 1 and 2 have already been visited, and got assigned the palette (blue, green, red) and (blue, green, orange) respectively. Moreover, we are currently visiting node 3 while node 4 is yet uncolored. In that case, there are two possible palettes for node 3. If we select (green, red, orange), a possible color assignment for the edges $(1, 2)$, $(1, 3)$ and $(2, 3)$ is respectively blue, red, and green.

3) We then generate k sparse, unit-norm directions $\mathcal{D}_k = \{w_1, \ldots, w_k\}$, independently of the graph. An underlying assumption of our model is that each direction should provide an explanation of why two nodes are connected that is markedly different from the explanation of another direction. A way to achieve that is to create directions with only a few non-zeros components, and ensure that the non-zero components of one direction do not appear at the same position as the non-zero components of another direction. Therefore, we control through n_o the number of dimensions where more than one non-zero component exists across the k directions. That is, when $n_o = 0$, each direction has exactly k/d non-zero components that are disjoint.[11] On the other hand, if for instance $n_o = 5$, then 5 of the d dimensions will have two directions with non-zero components on it. Those non-zero components are drawn uniformly at random from $[-1, 1]$ and each direction is then normalized. For instance, with $d = 6$, $k = 3$ and $n_o = 2$, we could generate the following directions, where the blue non-zero components are "overlapping" with the black ones:

$$\begin{matrix} w_1 \\ \begin{pmatrix} 0.729 \\ 0 \\ 0 \\ 0 \\ 0.450 \\ -0.516 \end{pmatrix} \end{matrix} \quad \begin{matrix} w_2 \\ \begin{pmatrix} 0 \\ 0 \\ -0.483 \\ -0.533 \\ 0.694 \\ 0 \end{pmatrix} \end{matrix} \quad \begin{matrix} w_3 \\ \begin{pmatrix} 0 \\ -0.639 \\ -0.769 \\ 0 \\ 0 \\ 0 \end{pmatrix} \end{matrix}$$

4) Returning to directions instead of colors, we have a direction $w_{y_{uv}} \in \mathcal{D}_k$ assigned to every edge (u, v). For easy of notation, in that paragraph, we will let $w_{uv} = w_{y_{uv}}$. The final step in generating our node-attributed graph is to find a set of user profiles $X = \{x_u\}_{u \in V}$ that maximizes the edge goodness and minimizes the node loss term, defined respectively as:

$$\mathcal{L}_{\text{edge}} = \sum_{u,v \in E} {s_{uv}}^T w_{uv} \quad \text{and}$$

$$\mathcal{L}_{\text{node}} = \sum_{u \in V} \left\| x_u - b_u - \sum_{v \in \mathcal{N}(u)} a_{uv} w_{uv} \right\|^2 ,$$

[11] We choose d to be a multiple of k.

where for simplicity, we fix as before $b_u = \mathbf{0}$ and $a_{uv} = \frac{1}{\deg(u)}$. Furthermore, to ensure that the direction assigned to any edge (u, v) achieves a higher goodness than the other directions (that is $\forall w_\ell \in \mathcal{D}_k \setminus \{w_{uv}\}, g(s_{uv}, w_{uv}) \geq g(s_{uv}, w_\ell)$), we also minimize a cross-entropy loss commonly used in non-binary classification problem [Bis06, Section 4.3.4]. Specifically, denoting by $p_{uv,\ell} = \frac{\exp(g(s_{uv}, w_{uv}))}{\sum_{\ell=1}^{k} \exp(g(s_{uv}, w_\ell))}$ the softmax "probability" of s_{uv} being explained by direction ℓ, the cross-entropy loss is defined by:

$$\mathcal{L}_{\text{cross-entropy}} = -\sum_{u,v \in E} \sum_{l=1}^{k} \mathbb{I}\{w_{uv} = w_\ell\} \, p_{uv,\ell} \,. \tag{4.10}$$

In practice, we first minimize

$$\lambda \mathcal{L}_{\text{cross-entropy}} - \mathcal{L}_{\text{edge}} \tag{4.11}$$

with respect to X using the Adam algorithm [KB15] and automatic differentiation [MDA15][12], projecting the current iterate back to the set of matrix with unit L_2 norm columns $\mathbb{M}^{n \times d}$ at each step.

Finally, to minimize $\mathcal{L}_{\text{node}}$, we iterate over the nodes and take some gradient steps to minimize $\left|\left| x_u - \frac{1}{\deg(u)} \sum_{v \in \mathcal{N}_u} w_{uv} \right|\right|^2$, but only to the extent that w_{uv} remains the direction with the highest goodness on the edge $(u, v) \in E$.

4.3.2 Results

In this section, we start by describing how we generate test instances of Problem 5 for the five methods we want to compare. Then we present two evaluation measures: how well those methods classify edges, and how well they recover known directions. We use the first measure to comment on the quality of the generated instances. Finally we present the results of the five methods in Table 4.4 and Table 4.5.

Our test instances are generated as follow. We first choose $n = 500$, and generate once a graph topology with directions assigned to all edges according to steps 1 and 2. This results in a graph G with roughly $1\,350$ edges. We then generate the directions $\mathcal{D}_k$ according to step 3 and the following choice of parameters: $d = 35$, $k = 7$, $k_{\text{local}} = 3$ and $n_o = 0$. Finally, we create the profiles X as described in step 4. To account for the randomness of this generation process, we repeat our measurement over 200 such generations of $\mathcal{D}_k$ and X. On the other hand, we verify experimentally that generating various graphs while keeping $\mathcal{D}_k$ fixed gives the same results, hence we will not report them in the main text.

In the rest of this section, we will refer to the set of parameters $d = 35$, $k = 7$, $k_{\text{local}} = 3$ and $n_o = 0$ as the *default* configuration. We also try other configurations to study the parameter sensibility of our methods. Specifically, we experiment with less directions ($k = 5$) or more directions ($k = 9, d = 36$)[13], with two higher levels of overlap between the directions ($n_o = 6$ and $n_o = 12$), with more directions per node ($k_{\text{local}} = 4$) and in larger dimension ($d = 77$).

Let us now describe how we evaluate the results of our methods on such instances. As said at the beginning of this section, we extract from the output of each method a set $\widehat{\mathcal{D}}_k = \{\widehat{w}_1, \ldots, \widehat{w}_k\}$ of k directions, and assign a direction to an edge (u, v) as $\mathcal{E}(u, v) = \arg\max_{\ell \in [\![k]\!]} g(s_{uv}, \widehat{w}_\ell)$. Recall that during the second of the step of the instance generation, we assigned a direction index y_{uv} to every edge. However, we cannot simply evaluate our predictions by simply comparing $\mathcal{E}(u, v)$ and y_{uv}. Indeed, here is no guarantee that the $\widehat{\mathcal{D}}_k$ directions are recovered in the same order as they were generated. In case of strong disagreement between the predictions

[12] Both being implemented by the Pytorch package: http://pytorch.org/.

[13] We add one dimension to ensure that d remains a multiple of k.

and ground truth labels, it might additionally be difficult to find a permutation to conciliate them. Therefore, we see the problem as a clustering one with known ground truth, and turn to the Adjusted Mutual Information (AMI) [VEB09]. It is an information theoretic measure that enjoys several properties: it is invariant to permutations of the labels, invariant to the shape of clusters and it is bounded between 0 and 1, where an AMI of 1 indicates perfect correlation with the ground truth labels and 0 indicates perfect independence with the ground truth labels.

Besides the AMI score, we also measure how far the recovered directions $\widehat{\mathcal{D}}_k$ are from the actual ones, generated initially. More precisely, we associate each vector $w_\ell \in \mathcal{D}_k$ with the closest one in $\widehat{\mathcal{D}}_k$ and report the average ℓ_2 distance between the two elements of these k pairs, that is

$$d(\mathcal{D}_k, \widehat{\mathcal{D}}_k) = \frac{1}{k} \sum_{\ell \in [\![k]\!]} \min_{w'_i \in \widehat{\mathcal{D}}_k} \|w_\ell - w'_i\|_2$$

Before moving to the results, let us first note that, in the default configuration, we do not always generate user profiles leading to a perfect assignment of edge according to their goodness. This can be seen in Table 4.3. In general, this is not always possible. Consider a k-regular subgraph, where the k edges incident to every node are each assigned a different directions. By symmetry, it is impossible to find node profiles that would achieve maximum goodness for all the edges. This is partly what motivated the k_{local} local constraint. Even so, and more pragmatically, whereas those AMI scores could be made higher by increasing the coefficient λ in (4.11), this would imply a lower edge goodness. With our choice of λ, only a few percent of the edges have a mismatch between the directions we assigned them and the one with the largest goodness. Note also that depending of the parameters of the generation, the AMI is not the same. Therefore, in the following, we divide all the AMI scores by this natural score, in order to obtain a "standardized" measure.

Table 4.3 – The degree to which the edge assignment of step 2 agree with the directions and profiles of steps 3 and 4. This is measured by the AMI between the assignment $\{y_{uv}\}_{uv \in E}$ and what maximal goodness would assign, i.e. $\{\mathcal{E}(u,v)\}_{uv \in E}$. We report the average AMI and standard deviation over 200 generations of $\mathcal{D}_k$, for different configurations.

$\mathcal{D}_k$ parameters	default	$k = 5$	$k = 9, d = 36$	$n_o = 6$	$n_o = 12$	$k_{\text{local}} = 4$	$d = 77$
AMI	$.922 \pm .03$	$.925 \pm .03$	$.905 \pm .02$	$.880 \pm .04$	$.874 \pm .05$	$.893 \pm .03$	$.973 \pm .01$

As showed in Table 4.4, COMBINED is always the best method and, excluding EXPLICIT for now, the LLOYD heuristic is second except in one case, while k-MEANS is third. Note that although the differences are not large in absolute value, they are generally statistically significant. Coming back to EXPLICIT, not only does it have the same performance as COMBINED, but we verified that it actually returns the same predictions, up to a few edges that are classified differently. This can be explained because they optimize the same objective and EXPLICIT is initialized with the solution from COMBINED. However, we also verified that both edge score and node loss are respectively higher and lower at the end of the optimization. Finally, the results of FRANK–WOLFE are much worse than all the others, despite the optimization leading to edge scores comparable to the other methods. This suggests that the subsequent clustering is not adequate. Regarding the generative parameters, and compared with the default configuration, the problem is easier when the k directions are spread in a larger number of dimensions, and more difficult when each nodes is involved in four directions instead of three. On the other hand, whereas we expect the performance to decrease with increase in the overall number of directions or their overlap, there is not consistent evidence of that.

Table 4.4 – Standardized AMI of 5 methods, when generating directions with 7 configurations. For each configuration, we generate directions 200 times, and report the mean standardized AMI along with the standard deviation. Among the first three methods (that do not use matrix formulation), we highlight the best one in **bold brown** and the second best one in *italic red*. When the difference between a score and the next best one is statistically significant (i.e. when we can confidently reject the hypothesis that the two distributions have the same mean), we furthermore show in parenthesis the supporting p-value of a paired Student's t-test.

$\mathcal{D}_k$ parameters	k-MEANS	LLOYD	COMBINED	FRANK–WOLFE	EXPLICIT
default	$.818 \pm .06$	*$.873 \pm .05$* ($1.25{\cdot}10^{-63}$)	**$.893 \pm .04$** ($5.68{\cdot}10^{-33}$)	$.381 \pm .05$	$.893 \pm .04$
$k = 5$	$.836 \pm .08$	*$.838 \pm .07$*	**$.875 \pm .06$** ($2.17{\cdot}10^{-58}$)	$.213 \pm .11$	$.875 \pm .06$
$k = 9, d = 36$	$.803 \pm .04$	*$.881 \pm .04$* ($2.66{\cdot}10^{-94}$)	**$.894 \pm .04$** ($8.98{\cdot}10^{-17}$)	$.421 \pm .04$	$.894 \pm .04$
$n_o = 6$	$.813 \pm .07$	*$.824 \pm .06$* ($7.57{\cdot}10^{-6}$)	**$.856 \pm .06$** ($2.99{\cdot}10^{-57}$)	$.378 \pm .05$	$.855 \pm .06$
$n_o = 12$	*$.827 \pm .07$*	$.823 \pm .06$	**$.852 \pm .06$** ($1.90{\cdot}10^{-25}$)	$.370 \pm .06$	$.851 \pm .06$
$k_{\text{local}} = 4$	$.772 \pm .07$	*$.814 \pm .07$* ($6.02{\cdot}10^{-42}$)	**$.853 \pm .06$** ($2.13{\cdot}10^{-47}$)	$.320 \pm .06$	$.853 \pm .06$
$d = 77$	$.905 \pm .05$	*$.933 \pm .04$* ($1.32{\cdot}10^{-31}$)	**$.941 \pm .03$** ($1.77{\cdot}10^{-22}$)	$.222 \pm .10$	$.931 \pm .04$

Similar conclusions carry out when we evaluate methods according to their ability of recovering the original directions. Keeping in mind that in the case of perfect recovery, $d(\mathcal{D}_k, \widehat{\mathcal{D}}_k)$ would be equal to zero, while the distance between two d-dimensional unit vectors is $\sqrt{2}$ in expectation[14], we see in Table 4.5 that no method gets very close to the original directions. COMBINED is again the closer overall, but now k-MEANS and LLOYD are alternating at the second place. The fact $\mathcal{D}_k$ cannot be completely recover is not surprising, for in absence of prior information the problem is under-constrained. Indeed, while COMBINED and EXPLICIT deliver almost the exact same prediction, their directions are clearly different.

Table 4.5 – Same as Table 4.4, but reporting $d(\mathcal{D}_k, \widehat{\mathcal{D}}_k)$, which should be as close as possible to 0.

$\mathcal{D}_k$ parameters	k-MEANS	LLOYD	COMBINED	FRANK–WOLFE	EXPLICIT
default	$.533 \pm .08$	*$.528 \pm .09$* ($9.61{\cdot}10^{-4}$)	**$.516 \pm .06$** ($4.95{\cdot}10^{-5}$)	$.877 \pm .05$	$.564 \pm .06$
$k = 5$	*$.581 \pm .08$* ($4.78{\cdot}10^{-30}$)	$.606 \pm .09$	**$.560 \pm .05$** ($8.07{\cdot}10^{-24}$)	$1.007 \pm .05$	$.633 \pm .06$
$k = 9, d = 36$	$.541 \pm .07$	*$.521 \pm .07$* ($5.08{\cdot}10^{-23}$)	**$.520 \pm .05$**	$.811 \pm .06$	$.551 \pm .06$
$n_o = 6$	*$.554 \pm .08$* ($2.74{\cdot}10^{-52}$)	$.589 \pm .08$	**$.538 \pm .05$** ($2.38{\cdot}10^{-12}$)	$.908 \pm .05$	$.596 \pm .07$
$n_o = 12$	*$.554 \pm .09$* ($1.50{\cdot}10^{-58}$)	$.595 \pm .08$	**$.545 \pm .06$** ($4.08{\cdot}10^{-5}$)	$.935 \pm .05$	$.604 \pm .08$
$k_{\text{local}} = 4$	$.565 \pm .08$	*$.564 \pm .09$*	**$.546 \pm .05$** ($6.30{\cdot}10^{-8}$)	$.928 \pm .06$	$.588 \pm .06$
$d = 77$	*$.571 \pm .07$* ($2.02{\cdot}10^{-89}$)	$.602 \pm .07$	**$.567 \pm .05$**	$1.033 \pm .06$	$.641 \pm .05$

4.4 Related work

Whereas we believe we are the first to study Problem 5, the prevalence of multilayer graphs in real world applications, as well as the wealth of information provided by nodes attributes has attracted a lot of attention. We start by presenting methods to classify edges, noting they either require supervision or do not consider node attributes. We then discuss works predicting relations in knowledge graphs, and more generally embedding edges in low dimensional spaces. Another approach we highlight includes combining topology and subspace clustering. We also note how ideas such as heterophily, learning k ways to weight edge's endpoint similarity and distinguishing a superposition of layers have been used in different contexts. Finally, we conclude by the more distant but very rich topic of generative model for attributed graphs and how it enables overlapping node clustering.

[14] One can derive this from the fact after normalization, vectors whose coordinates are drawn from a standard normal distribution are uniformly distributed on the unit sphere [Mul59].

Edge classification To the best of our knowledge, there are not many works that directly attempt to classify edges in attributed networks. A first natural idea is actually to rely on the numerous existing node clustering algorithms, by considering the line graph $L(G)$ of G. $L(G)$ is a graph whose each of the $|E|$ nodes corresponds to an edge of G, and where two nodes of $L(G)$ are connected whenever their corresponding edges share a common node in G. Evans *et al.* [EL09] apply a modularity algorithm to $L(G)$ is order to partition the edges of G but, contrary to us, their final goal is to discover an overlapping clustering of the nodes of G, and they rely solely on the graph topology. In their review of mining social networks, Tang *et al.* [TL15, Section 2.3] present several methods to solve what they call *inferring social ties*. The first one is based on a graphical model where each node is an edge. The label of an edge is influenced by the attributes of its endpoints, the correlation between the type of neighboring edges and some global constraints. These three influences are modeled by pairwise factors. The parameters of the model are learned by maximizing the likelihood of the observed labels through gradient descent. Those parameters are then use to predict the remaining labels that further maximize the total likelihood [TZT11]. An extension of this graphical model is to actively query the most informative edges, in the case where it is costly to ask social network users to label their relationships [Zhu+12]. Another way to leverage additional supervision is to use transfer learning [Tan+16a]. Namely, given two distinct and partially labeled graphs, where the fraction of labelled edges is much higher for the first graph than the second one, the goal is to fully label the edges of the second graph, exploiting the rich information of the first one. This requires the attributes to be comparable between the two graphs, and such attributes are therefore computed based on topological features. Finally, a last direct approach to labelling edges is presented by Aggarwal *et al.* [AHZ16]. They consider that the input graph has three types of edges labeled by -1, labeled by $+1$ or unlabeled. In each of the three induced subgraphs, they define a similarity between two nodes u and v as the Jaccard coefficient of the two sets of edges incident to u and v. A weighted combination of these three coefficients is the final similarity between u and v. Then, $S_k(u)$ are the top-k most similar nodes to u, and the label of the edge (u, v) is chosen as the majority label in $(S_k(u) \times S_k(v)) \bigcap E$. All these works assign a label to the edges of the input graph, but they either require supervision or they do not use attributes, and in both cases they provide little interpretability over their results.

Prediction in knowledge graphs Besides social networks, the task of predicting the semantic of an edge has also been addressed in the context of knowledge graphs. In such graphs, nodes are abstract concepts and concrete entities from the real world, and edges are directed predicates representing facts connecting two entities [Nic+16]. A typical example is an edge $(h, \ell, t) = (\text{Donald J. Trump}, \text{President of}, \text{USA})$[15] where the (h, ℓ, t) notation stands respectively for head, type of relation and tail. This illustrates some differences with our setting. First, there might be a very large number of types of edge in knowledge graphs, since they cover a domain as large as possible. Second, this coverage makes it difficult to describe directly entities by a consistent set of d attributes, and one usually has to rely on low dimensional embeddings. [Bor+13] is primarily concerned with learning such embeddings in $\mathbb{R}^k$ for both nodes and relations, that are denoted by the boldface letters $\boldsymbol{h}$, $\boldsymbol{\ell}$, $\boldsymbol{t}$. Similarly to word embedding, the intuition is that relations can be modeled by translations. Thus, the existence of an (h, l, t) edge offers evidence that $\boldsymbol{h} + \boldsymbol{\ell} \approx \boldsymbol{t}$. Given a training set of existing triplets S and a dissimilarity measure D, the authors learn an embedding in a fashion reminiscent of metric learning [BHS15]. Namely, they create negative examples by corrupting at random training examples and then minimize with a margin the positive part of the difference between

[15] At least at the time of writing

dissimilarity of positive and negative examples. More generally, we refer the interested reader to a recent survey on embedding knowledge graphs [Wan+17b], which among other ideas present the interesting notion of composing relations along paths between two entities. We note that, in contrast with us, those methods require some form of supervision when it comes to predict edge types. One way to avoid supervision is to rely on a large amount of unstructured text, extract entities from this corpus, map entities and possible relations to a knowledge base and finally embed them in a low dimensional space using text features such as POS tags [Ren+17].

Edge embedding A related idea is embedding edges in general graphs, not necessarily in the context of knowledge graphs. This is the counterpart of node embedding techniques we mentioned in Section 2.4. The most straightforward way is to keep track of edges instead of nodes when performing random walks, learn an embedding from this corpus using word2vec-like methods and then cluster these representations using k-means [Li+17b]. We cannot directly compare this with our work since there is no attribute and thus the clustering is not interpretable (indeed the goal of [Li+17b] is eventually to find an overlapping clustering of the nodes, not of the edges). More closely related to our work, Ahmed *et al.* [Ahm+17] seek to assign a role to the edges of a graph. First they learn edge features as follow: starting from topological features, they use combination operators (like max, min, mean, sum, product and so on) to iteratively learn higher order features while pruning those that are correlated in order to avoid a combinatorial explosion. Once they obtain such a d-dimensional feature vector for each edge, they stack them in a matrix $S \in \mathbb{R}^{|E| \times d}$ and look for $U \in \mathbb{R}^{|E| \times r}$ and $V \in \mathbb{R}^{r \times d}$ such that the distance between S and UV^T is minimized. V is a description of r roles in terms of features while U is the mixed membership of each edge. Finally the authors choose the number of roles r using the Minimum Description Length principle. Their approach bears some similarities with our EXPLICIT method, but have the additional properties of scaling linearly with the number of edges and being partially parallelizable. However, the process in which edge feature are learned does not lend itself easily to interpretation.

Subspace clustering Another direction is to see the problem as clustering the similarity vectors s_{uv}, like our approaches in Section 4.2.1. Because of our bias that nodes are connected through partial homophily and heterophily, this can more precisely be casted as a subspace clustering problem [Vid11]. Namely, we are given a set of m points in $\mathbb{R}^d$, with the assumption they belong to the union of affine subspaces of unknown dimension. The goal is to recover the number K of such subspaces, their dimension, their parameters and the assignment of the m point to those subspaces. If K is known, one can use iterative, k-mean like methods (reminiscent of our LLOYD method). Otherwise, it also possible to factorize the matrix of all the data points and interpret it as a similarity matrix, to use iterative statistical approaches such as random sample consensus or to build a similarity graph and apply spectral clustering algorithms to it. Finally, it is also possible to leverage sparsity assumptions and express each point as a sparse linear combination of other points, and use these coefficients as weights of a similarity graph, which is then clustered into subspaces by a spectral method [EV13]. While subspace clustering comes with information theoretic guarantees of its ability to retrieve optimal subspaces, when formulated directly over the $\{s_{uv}\}_{(u,v)\in E}$, it makes no use of graph topology, as we consider each edge independently of all others. As an example of applying related ideas to graph data, Huang *et al.* [HCY15] build a d-dimensional grid of the attributes space and use subspace clustering to find cells that have low entropy and high connectivity, before merging such cells into clusters of nodes.

Further similar ideas Taskar *et al.* [Tas+04] represent a node-attributed graph with partially labelled edges as a probabilistic graphical model (called Relational Markov Model) and learns its parameters from the data using gradient descent. It is a flexible modeling approach that let the designer express complex graph patterns. Using a slightly less principled approach, Stattner *et al.* [SC12] first cluster nodes based solely on their attributes and then count the links between such groups to keep only the maximally frequent ones, arguing this reveals the sets of attributes that support the connection between two nodes. In the same data mining vein, and related to our departure from global homophily, Liang *et al.* [LWZ16] look for link patterns whose support and confidence are not only above some given thresholds, but that also diverge from homophily. Indeed, the authors explicitly exclude attributes taking the same value within such link patterns. The objective of Zhang *et al.* [ZLZ16] is eventually to hard cluster the nodes into communities but their work shares similarities with ours. Namely, they tweak the traditional modularity objective (maximizing the density of intra community edges) by weighting the edges with the similarity of their endpoints, and simultaneously learn a weight vector w_ℓ for a community C_ℓ. Formally they defined the R criterion as :

$$R(\mathcal{C}, \beta; \alpha, \mu) = \sum_{\ell=1}^{k} \frac{1}{|\mathcal{C}_\ell|^\alpha} \sum_{u,v \in \mathcal{C}_\ell} A_{uv} \left(\mu - e^{-g(s(u,v),w_\ell)} \right)$$

where A is the adjacency matrix, α penalizes unbalanced communities, μ is a trade off parameter between information from edges and node features, and g corresponds to our goodness function while s is an arbitrary similarity function between nodes. They optimize it by alternately optimizing over the labels with fixed parameters and over the parameters with fixed labels, using block coordinate descent. Finally, we mention a more quantitative work. Abraham *et al.* [Abr+15] assume that there are K social categories modelled by K Euclidean spaces $\mathcal{D}_i$. Nodes of the graph have an associated point in each of these spaces. The key assumption is that space have small local correlation: informally, the intersection of any two small balls from two distinct spaces is small. These K spaces give rise to small world networks $\mathcal{G}_i$, where the edge probability is proportional to $\mathcal{D}_i(u,v)^{-d}$ and we observe the real network $\mathcal{G} = \bigcup_i \mathcal{G}_i$. From $\mathcal{G}$, the proposed algorithm recovers in $O(n \operatorname{polylog} n)$ time a bounded approximation $\mathcal{D}'_i$ of all $\mathcal{D}_i$, that is there exists positive constants σ, δ and Δ such that $\sigma\mathcal{D}_i(u,v) \leq \mathcal{D}'_i(u,v) \leq \delta\mathcal{D}_i(u,v) + \Delta$.

Generative models and node clustering Although there has not been so many works on clustering edges in node attributed networks, there is a wealth of papers on clustering nodes in such networks. Since this is not exactly our topic, we only mention a selection of the most relevant to our objective (especially the overlapping ones), and refer the interested reader to the survey of Bothorel *et al.* [Bot+15][16]. We also reiterate the warning given in the introduction of this chapter. Namely, in large graphs, node attributes are not necessarily aligned with annotated communities. Such attributes thus need to be used *in addition* to topological information. This is further covered in [FH16, Section 3.4].

An early attempt is the SA-Cluster algorithm of Zhou *et al.* [ZCY09] that partitions the nodes of a graph based on a distance combining nodes structural and attributes similarities, along with its faster incremental version [ZCY10]. Nodes in the same cluster are well connected and have a set of similar attributes. The balance between these two objectives was further studied in a setting where it is tunable by the user of the algorithm [Bar+17]. In a more "data mining fashion", the same problem can also be worded as finding subgraphs induced by a set of attributes that are more dense than what one would expect in a null model [SMZ12].

[16] Especially the section 2 dealing with edge labeled graphs.

Instead of detecting all such subgraphs or communities, one can adopt a query-based approach. Namely, given a query node u and a set S of attributes, find all the subgraphs containing u that are both tightly connected and share enough common attributes [Fan+16]. In that spirit of focusing on a given node, Leskovec *et al.* [LM12] introduce a generative model for ego networks in social networks where the neighbors of a node u can belong to k categories (such as family, colleagues, school friends). Using our notations, those k categories (or circle) $\{C_i\}_{i=1}^k$ of a node u are defined by a vector w_i and a weight α_i. The probability $p(e)$ of an edge $e = (u, v)$ favors the presence of edges with high goodness within circles, as expressed by

$$p((u,v) = e \in E) \propto \exp\left(\sum_{C_i \supseteq u,v} g(s(u,v), w_i) - \sum_{C_i \not\supseteq u,v} \alpha_i g(s(u,v), w_i)\right), \quad (4.12)$$

where s is an arbitrary similarity function between two node profiles. The unsupervised problem of maximizing the likelihood of observing the input graph according to this probability distribution is solved by alternating two steps: assigning nodes to circles and optimizing the parameters $\{w_i, \alpha_i\}_{i=1}^k$ given a circle assignment. Similar to us, the vector w_i explain why some nodes belong to the i^{th} circle. However, it is not obvious how to transfer this knowledge to the edges themselves.

In fact, this generative model approach has proved very fruitful when it comes to community detection in node attributed graphs [Xu+14a; ZDB17; YML13; Kat+16; WF16; NC16]. The general idea is the following. First, design a model to generate some of the following aspects: the nodes attributes, the topology of the graph and the community membership of the nodes. Then, infer the parameters that maximize the likelihood of observing the current graph. Finally, extract from these parameters community membership for every nodes. We give a very succinct description of the generative models of some representative recent works in Table 4.6. Seen at the light of our Problem 5, given the overlapping membership obtained by inference of a generative model, and an edge (u, v), one could look at the highest shared community coefficient between u and v and use that to explain the edge (among the $\frac{K(K+1)}{2}$ possible edge types induced by the K communities and their $\frac{K(K-1)}{2}$ pairs). However, this is clearly an ad-hod post processing, as indeed these works are concerned with nodes and not edges clustering.

We also point to the work of De Bacco *et al.* [De +17], which despite not considering node attributes, present a model with interesting applications. Specifically, the nodes have mixed membership to K overlapping groups and each of the L layers is generated by a specific $K \times K$ block matrix, taking into account the group membership, like in a degree corrected SBM. Unlike us, this allows for multigraphs, although the authors assume for simplicity that this does not happen. Once its parameters are found, this model can then be used to predict the existence of extra edges in each layer, which can be seen as edge type prediction. Furthermore, measuring the extent to which one layer helps predict links in another layer provides a way to measure the relationships between the layers, from redundancy to complete independence, allowing information compression.

4.5 Open directions

We could extend our approach to more general types of graphs and tasks:

- First, as we mentioned earlier, **networks are dynamic** and evolve over time, with nodes and edges constantly appearing and disappearing [AS14]. Slicing time into successive periods is one way to create edge types, one per period. For instance, Mucha *et al.* [Muc+10] study the 110 congresses of the US senate between the years 1789 and 2008, where 1 884 individual Senators are linked by their voting similarity. They show how it improves the detection of relevant

Table 4.6 – We summarize how each model generates: 1) the membership of a node u to a community $c \in [\![k]\!]$, 2) the attributes of u knowing its community membership and 3) the edges between nodes.

ref	community	attributes	links
[Xu+14a]	single community drawn from a multinomial	drawn from a distribution parametrized by the node community	drawn from a distribution parametrized by the pair of endpoints communities
[YML13]	intensity of membership $F_{uc} \in [0, \infty)$ are given	logistic function of $F_{u1}, \dots, F_{uC}$	sharing more communities makes link more likely: $P_{uv} = 1 - \exp(-\sum_c F_{uc}F_{vc})$
[Kat+16]	multinomial of dimension k	drawn from one normal distribution per community	Stochastic Block Model (SBM), where blocks are identified with communities
[ZDB17]	gamma distribution, whose parameters depends of the nodes attribute	given binary attributes	SBM, with the block matrix drawn from a hierarchical relational gamma process
[LM12; ML14]	overlapping circle defined by θ_k, but not generated	given	higher probability of appearing within common circle, as given in (4.12)
[WF16]	logistic function of the attributes	given	SBM
[NC16]	one multinomial for each discrete value of the single attribute	given	degree corrected SBM

communities. Beyond predicting the relative order of links, it would also be interesting to include time into node similarity itself. For instance, while the photo similarity between two Flickr users rises before they connect, it later decreases if they have the same level of popularity, as both try to differentiate themselves [ZW13]. More generally, by defining our problem on a static snapshot of a graph, our model is agnostic to the question of whether edges were created because of the similarity/dissimilarity of the linked nodes' attributes or because attributes started to change after the edge creation. This is traditionally framed as dichotomy between homophily and contagion, and is an active topic of research in social networks analysis [AKM08; ST11] as noted by Golder *et al.* [GM14]: "Homophily refers to a variety of selection mechanisms by which a social tie is more likely between individuals with similar attributes and environmental exposures [MSC01]. Contagion refers to influence mechanisms (e.g. imitation or peer pressure) by which traits diffuse along network edges. Homophily and contagion offer competing explanations for network autocorrelation, which refers to the greater similarity in the attributes of closely connected nodes."

- Another rich class of graphs where predicting edge types would be a challenging and rewarding task is **multigraph**, which allow several edges between two nodes [Ber+11]. It can also be seen as the flattening of several graphs sharing the same nodeset, but where the provenance of each edge would have been lost and need to be recovered. With our original formulation of goodness as ${s_{uv}}^T w_{uv}$, only one direction can achieve the higher score (discarding the rare case of tie). One alternative we proposed was to let each w_{uv} be a linear combination of a small number of base directions. Symmetrically, another idea could be to have more than one way to compute the similarity s_{uv} between two nodes.
- Generalizing the balance theory from signed graphs, we could imagine that in triangles or short cycles, only certain combinations of directions are valid (or at

least desirable). Interestingly, this would likely require different optimization algorithms, probably non convex. However, while this is a general concept, its exact implementation might be application-dependant. Furthermore, in our current setting, we do not know in advance the semantic of each direction, nor do we have labeled edges. Therefore, it is not clear how those constraints would be specified, or whether they can be learned, if applicable. A limited solution exists in the context of heterogeneous information networks, where in addition to having several types of edges, there are also several types of nodes. Since not all types of node can be connected with any types of edge, the possible connections are themselves represented as a meta graph called the network schema [Shi+17, Definition 3]. This approach would need to be extended to deal with higher order patterns, for instance in a manner reminiscent of composition of paths in multilayer graphs [Sun+11; Wan+17b, Section 4.2].

- Among tasks that could benefit from labeling the edge of complex networks, we mention two. The first is **graph summarization** where the idea is to transmit a fraction of a large graph plus additional information that can be used to reconstruct the original graph. Namely, our goal here is to send only a few node profiles along with the edge directions, enabling the missing profiles to be reconstructed from the known goodness values. Note that we spare information regarding the profiles but still transmit the full topology. Therefore, this differs from lossless schemes [Chi+09; Bol+11] that leverage typical edge locality patterns to encode edges using as few bits as possible, or from lossy graph sketching methods [AGM12] discussed in Section 3.3.2, which preserve spectral and topological properties of the original graph. The second is **node embedding**, where every node u is represented by a vector of size k containing the relative proportion of each direction with the edges incident to u.

Chapter 5

Conclusion

5.1 Contributions

The objective of this thesis was to provide "efficient and accurate methods to predict edge type in complex networks". By complex networks, we mean graphs where edges can have one among k semantics (or types) and where nodes are connected by both partial homophily and heterophily.

- In Chapter 2 [Le +17], we first focused on directed signed graphs, as they model social interactions in several domains, such as e-commerce and collaborations. In such graphs, edges have two possible semantics, positive or negative. Our goal was to solve a supervised binary classification problem by predicting the sign of test edges. For that, we first introduced a simple sign generative model. It endows nodes with two parameters –trollness and unpleasantness–, governing their incoming and outgoing behavior. This might seem like a simplistic way of expressing the notion of partial homophily and heterophily. However, we presented two methods to approximate this Bayes predictor, suggesting a trade off between theoretical guarantees and input requirement. We showed experimentally that those methods are competitive with the state of the art, while being more scalable. Motivated by the regularity measure offered by trollness and unpleasantness, and the inability of existing methods to cope with new nodes joining the graph, we furthermore developed the first online algorithm for the EDGE SIGN PREDICTION problem.
- We started Chapter 3 by showing that the methods of the previous chapter are not straightforward to extend to undirected signed graphs. Therefore, we changed our learning bias from a generative model to a clustering assumption inspired by social balance theories. Namely, we posit that the nodes are assigned to one of K groups, and that the sign of the edge (u, v) is positive if the nodes u and v belong to the same group and negative otherwise. We then presented two approaches to leverage this bias. First, we drew a connection between the EDGE SIGN PREDICTION problem and the CORRELATION CLUSTERING problem. While CORRELATION CLUSTERING is hard to approximate in general, we suggested, based on existing works, that the more an instance complies with our bias, the better CORRELATION CLUSTERING methods could recover the K underlying groups and thus provide accurate prediction. Second, we addressed the problem in an active setting, where $K = 2$. For that, we implemented a promising spanning tree algorithm [Vit14], that queries some edges in order to connect the remaining edges by paths as short as possible. Experimentally, we showed that on both synthetic and real graphs, our method is competitive in terms of sign prediction with a strong natural baseline.
- Finally, in Chapter 4, we considered the case where there are more than two types of edge. This means that instead of signed graphs, our input shifted to

multilayer graphs, which are also ubiquitous in describing real world phenomena. Moreover, we modified the learning setting, so that no edge label are available, but instead nodes carry a profile vector. However, our goal remained to predict edge type, in what could then be described as a clustering problem with side information and interpretability requirements. In order to do so, we first defined a similarity between nodes, which accommodates both heterophily and homophily. We then formulated the problem as simultaneously finding k special vectors called directions, and assigning one of them to every edge in order to maximize a scalar *goodness* function. After introducing baselines oblivious of the graph topology, we added constraints resulting in one vectorial and one matricial optimization objectives. Finally, we showed on synthetic data how those methods can recover the planted ground truth.

5.2 Future work

In this section, we revisit the problems summarized above by showing how they could be applied in a new context or enriched by existing node embedding methods.

5.2.1 Reciprocal recommendation

We first present a task tightly related to our edge characterization problem, namely the reciprocal recommendation problem [Piz+13]. As we shall see, it indeed incorporates aspects from all the problems we discussed previously, along with new constraints. Broadly speaking, given two sets of users L and R, we want to recommend users in R to users in L, and vice versa. These recommendations are based on mutual interest, but in most cases there is an asymmetry between the two groups. To better illustrate this general definition, we now give some concrete examples.

- One recent application that quickly rose to prominence is online dating [Krz+15; Xia+16; AB16]. In the most common case, users are divided between men and women, and we want to recommend users of the opposite gender in order to create couples. A well known example of such a system is Tinder [Tys+16].
- In another domain, we can also match job seekers and employers [Sit+12; Hon+13; KA15].
- As a specific case in point, Zhang *et al.* [Zha+16] propose a method to match prospective PhD students and their future supervisors.
- In large organizations or associations, having a mentor is usually beneficial for both employees and employers [All+04] and finding mentee/mentor pairs can be cast as a reciprocal recommendation problem [SWG12].
- After moving to their new jobs, people also need a place to live. Again, we can match tenants and landlords, or people to share a flat.
- Finally, in a more critical setting, finding relevant matches between patients and organ donors can improve the success of transplants [Yoo+17].

Reciprocal recommendations present specific challenges [And15]. Among them are asymmetry and volume. Asymmetry refers to the imbalance between the two groups, in terms of size and bargaining power. For instance, depending of the economic situation and the field, there might be a lot of job seekers for a few open positions, in which case employers can be very selective. Likewise, it is generally observed that dating websites have a higher proportion of men, meaning that women can afford to choose among numerous suitors [Tys+16]. By volume, we mean that users on both side can only handle a few recommendations at the same time. This is sharp contrast with traditional user-item recommendation systems,

where recommending simultaneously the same item (say a movie) to thousands of users is not an issue. On the other hand, we have to keep users engaged and prevent them from being idle. This is done by ensuring they are regularly the subject or object of some recommendations.

As the name implies, reciprocal recommendation has often be seen as a special case of recommender system [And15]. Besides this paradigm, we could also cast it as an adaptive, active, cost sensitive EDGE SIGN PREDICTION problem on bipartite directed signed graphs with side information. Let us decompose this last sentence, starting with the input graph. It is naturally bipartite, because of the presence of two groups in which there is no internal recommendation. Furthermore, every time one user expresses interest or disinterest in another user, this creates a directed signed edge. Such a graph is thus a special case of the Directed Signed Social Networks studied in Chapter 2. Each nodes carries two kind of (side) information. The first is explicitly provided by the user, in order to arise interest in users from the other group. The second is implicitly inferred from the user activity. In particular, the interactions with other nodes allow us to build an implicit preference model for every user. Combining these two sources of information yield a profile of each user, like in Chapter 4. Making recommendations in such a graph therefore boils down to finding non existing positive edges. However, to cope with the two challenges presented in the previous paragraph, we follow the setting of Chapter 3 and use an active approach. More precisely, respecting the volume constraints would require an adaptive algorithm, which acquires the most useful information without over- or under-loading any nodes. As for the group asymmetry, this could be addressed by using a cost sensitive classification error, so that the error rate at a given node is inversely proportional to its importance in the network.

As in any worthy machine learning endeavor, reliably evaluating and comparing approaches require large amount of real data. This is particularly true in such an applied setting, where the most realistic evaluation is to perform A/B testing on a live system with enough traffic to deliver statistical significance. This explains why most previous studies rely on the cooperation of industrial partners [And15; KA15; Krz+15; Xia+15; AB16]. Thus, this is also something we would like to do in the future.

5.2.2 Representation learning

A fundamental question underlying many machine learning approaches is the representation of data. When extracting information from the real world, we are usually given some freedom in how to present it to learning algorithms. Ideally, we would like to use the representation providing the best performance in the learning task. One solution is to ask expert domains to craft relevant features based on their prior knowledge. While this has achieved successes, it is not scalable to the present day situation, where the deluge of available data has prompted the use of machine learning in every corner of the society. An alluring alternative to expert knowledge is to learn the data representation itself, guided by the task.

As an illustration in the context of this thesis, we assumed that data was organized as a graph, where nodes were possibly associated with a feature vector. However, there might an different representation of those nodes that would better suit our needs. Indeed, we mentioned already in Section 2.4 and Section 4.4 that a recent and promising trend regarding learning in graph is node embedding. In addition to the references cited earlier, this also witnessed by recent works on embedding attributed nodes, either in signed graphs [CLL17; Wan+17c; Wan+18] or in general graphs [Lia+17; HLH17; Li+17a]. At first sight, those methods seem radically different from traditional graph methods. However, just like word embedding methods have been linked to matrix factorization [LG14; LC14; Aro+16], so have been popular and successful node embedding methods [Qiu+18]. These methods are the simple ones that do not consider node attributes and multiple types of edges,

such as DeepWalk [PAS14], LINE [Tan+15] and node2vec [GL16]. For instance, Qiu *et al.* [Qiu+18] show that the node embedding of DeepWalk on the weighted graph G are approximately the left singular vectors of the following matrix:

$$\log\left(vol(G)\left(\frac{1}{T}\sum_{r=1}^{T}(D^{-1}A)^r\right)D^{-1}\right) - \log b \tag{5.1}$$

where $vol(G)$ is the sum of all weighted degrees, T is the context window size, D is the diagonal matrix of degrees, A is the adjacency matrix of the graph and r is the number of negative samples in skip-gram. There are several ways such findings could be used in our context:

- If the edges of the input graph are fully labeled into k types, then we have k matrices D_k and A_k, by considering the subgraphs induced by each edge type. If we simply decompose the k matrices constructed according to equation (5.1), we would have k embedding for each node. Instead, it seems natural to stack those matrices together to form two third order tensors $\mathcal{D} \in \mathbb{R}^{n\times n\times k}$ and $\mathcal{A} \in \mathbb{R}^{n\times n\times k}$. One could then use one tensor decomposition methods [KB09; Sid+17] to extract node embedding taking into account the multiplicity of edge types.

- If the edges of the input graph are partially labeled, then predicting edge type can be seen as a tensor completion problem. Like in the matrix case, we seek a low rank tensor that coincides with the observed entries of the input data. Furthermore, when we have access to node profiles, we can use methods tailored to leverage such side information [Son+17, Section 4].

- If the input graph has no edge labels at all but we are given node profiles, we can use existing attributed embedding approaches [Lia+17; HLH17; Li+17a] to generate extra information about each node, before applying the methods presented in Chapter 4 on these extended profiles. We note however that doing so might reduce the interpretability of the final result.

Bibliography

[ABN08] I. Abraham, Y. Bartal, and O. Neiman, "Nearly tight low stretch spanning trees", in *49th Annual IEEE Symposium on Foundations of Computer Science*, 2008, pp. 781–790. DOI: 10.1109/FOCS.2008.62 (cit. on p. 81).

[ABN07] I. Abraham, Y. Bartal, and O. Neiman, "Embedding metrics into ultrametrics and graphs into spanning trees with constant average distortion", in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2007, pp. 502–511 (cit. on pp. 82, 83).

[Abr+17] I. Abraham, S. Chechik, M. Elkin, A. Filtser, and O. Neiman, *Ramsey spanning trees and their applications*, 2017. arXiv: arXiv:1707.08769 (cit. on pp. 82, 83).

[Abr+15] I. Abraham, S. Chechik, D. Kempe, and A. Slivkins, "Low-distortion inference of latent similarities from a multiplex social network", *SIAM Journal on Computing*, vol. 44, no. 3, pp. 617–668, 2015. DOI: 10.1137/130949191. arXiv: 1202.0922 (cit. on pp. 10, 112).

[AN12] I. Abraham and O. Neiman, "Using petal-decompositions to build a low stretch spanning tree", in *Proceedings of the 44th symposium on Theory of Computing - STOC '12*, 2012, p. 395. DOI: 10.1145/2213977.2214015 (cit. on pp. 10, 81, 83).

[Abu17] F. N. Abu-Khzam, "On the complexity of multi-parameterized cluster editing", *Journal of Discrete Algorithms*, 2017. DOI: 10.1016/j.jda.2017.07.003. arXiv: 1511.09360 (cit. on p. 55).

[Aga+05] A. Agarwal, M. Charikar, K. Makarychev, and Y. Makarychev, "O(sqrt log n) approximation algorithms for min uncut, min 2cnf deletion, and directed cut problems", in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, 2005, pp. 573–581. DOI: 10.1145/1060590.1060675 (cit. on p. 61).

[AHZ16] C. Aggarwal, G. He, and P. Zhao, "Edge classification in networks", in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 1038–1049. DOI: 10.1109/ICDE.2016.7498311 (cit. on pp. 10, 110).

[AS14] C. Aggarwal and K. Subbian, "Evolutionary network analysis: a survey", *ACM Computer Survey*, vol. 47, no. 1, 10:1–10:36, 2014. DOI: 10.1145/2601412 (cit. on pp. 3, 113).

[Ahm+17] N. K. Ahmed, R. A. Rossi, T. L. Willke, and R. Zhou, "Edge role discovery via higher-order structures", in *21st Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*. 2017, pp. 291–303. DOI: 10.1007/978-3-319-57454-7_23. eprint: http://ryanrossi.com/pubs/ahmed-et-al-pakdd17-preprint.pdf (cit. on pp. 10, 111).

[AGM12] K. J. Ahn, S. Guha, and A. McGregor, "Graph sketches: sparsification, spanners, and subgraphs", in *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, 2012, pp. 5–14. DOI: 10.1145/2213556.2213560 (cit. on p. 115).

[Ahn+15] K. Ahn, G. Cormode, S. Guha, A. McGregor, and A. Wirth, "Correlation clustering in data streams", in *Proceedings of The 32nd International Conference on Machine Learning*, 2015, pp. 2237–2246 (cit. on p. 60).

[Ail+12] N. Ailon, N. Avigdor-Elgrabli, E. Liberty, and A. van Zuylen, "Improved approximation algorithms for bipartite correlation clustering", *SIAM Journal on Computing*, vol. 41, no. 5, pp. 1110–1121, 2012. DOI: 10.1137/110848712 (cit. on p. 66).

[ABE14] N. Ailon, R. Begleiter, and E. Ezra, "Active learning using smooth relative regret approximations with applications", *Journal of Machine Learning Research*, vol. 15, pp. 885–920, 2014 (cit. on p. 63).

[ACN05] N. Ailon, M. Charikar, and A. Newman, "Aggregating inconsistent information: ranking and clustering", in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, 2005, pp. 684–693. DOI: 10.1145/1060590.1060692 (cit. on p. 58).

[AL09] N. Ailon and E. Liberty, "Correlation clustering revisited: the "true" cost of error minimization problems", in *Automata, Languages and Programming: 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12*. 2009, pp. 24–36. DOI: 10.1007/978-3-642-02927-1_4 (cit. on pp. 9, 64).

[AB16] A. Alanazi and M. Bain, "A scalable people-to-people hybrid reciprocal recommender using hidden markov models", in *2nd International Workshop on Machine Learning Methods for Recommender Systems*, 2016 (cit. on pp. 118, 119).

[All+04] T. D. Allen, L. T. Eby, M. L. Poteet, E. Lentz, and L. Lima, "Career benefits associated with mentoring for proteges: a meta-analysis", *Journal of Applied Psychology*, vol. 89, no. 1, pp. 127–136, 2004. DOI: 10.1037/0021-9010.89.1.127 (cit. on p. 118).

[Alo+95] N. Alon, R. Karp, D. Peleg, and D. West, "A graph-theoretic game and its application to the k-server problem", *SIAM Journal on Computing*, vol. 24, no. 1, pp. 78–100, 1995. DOI: 10.1137/S0097539792224474 (cit. on pp. 70, 80, 81, 84).

[Alt+93] I. Althöfer, G. Das, D. Dobkin, D. Joseph, and J. Soares, "On sparse spanners of weighted graphs", *Discrete & Computational Geometry*, vol. 9, no. 1, pp. 81–100, 1993. DOI: 10.1007/BF02189308 (cit. on pp. 81, 83).

[AP13] A. Amelio and C. Pizzuti, "Community mining in signed networks", in *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining - ASONAM '13*, 2013, pp. 95–99. DOI: 10.1145/2492517.2492641 (cit. on p. 69).

[Ami04] N. Amit, "The bicluster graph editing problem", M.Sc. Thesis, Tel Aviv University, 2004 (cit. on p. 66).

[AKM08] A. Anagnostopoulos, R. Kumar, and M. Mahdian, "Influence and correlation in social networks", in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008, pp. 7–15. DOI: 10.1145/1401890.1401897 (cit. on p. 114).

[AAG15] Y. Anava, N. Avigdor-Elgrabli, and I. Gamzu, "Improved theoretical and practical guarantees for chromatic correlation clustering", in *Proceedings of the 24th International Conference on World Wide Web - WWW '15*, 2015, pp. 55–65. DOI: 10.1145/2736277.2741629 (cit. on p. 66).

[AM12] P. Anchuri and M. Magdon-Ismail, "Communities and balance in signed networks: a spectral approach", in *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2012, pp. 235–242. DOI: 10.1109/ASONAM.2012.48 (cit. on p. 69).

[And+14] C. E. Andrade, M. G. Resende, H. J. Karloff, and F. K. Miyazawa, "Evolutionary algorithms for overlapping correlation clustering", in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014, pp. 405–412. DOI: 10.1145/2576768.2598284 (cit. on p. 67).

[And+12] B. Andres *et al.*, "Globally optimal closed-surface segmentation for connectomics", in *Computer Vision – ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part III*. 2012, pp. 778–791. DOI: 10.1007/978-3-642-33712-3_56 (cit. on pp. 4, 54).

[And15] E. Andrews, "Recommender systems for online dating", Master Thesis, University of Helsinki, 2015 (cit. on pp. 118, 119).

[AMM17] H. Angelidakis, K. Makarychev, and Y. Makarychev, "Algorithms for stable and perturbation-resilient problems", in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, 2017, pp. 438–451. DOI: 10.1145/3055399.3055487 (cit. on p. 86).

[ARS09] A. Arasu, C. Ré, and D. Suciu, "Large-scale deduplication with constraints using dedupalog", in *2009 IEEE 25th International Conference on Data Engineering*, 2009, pp. 952–963. DOI: 10.1109/ICDE.2009.43 (cit. on pp. 6, 54).

[AMW16] S. Aref, A. J. Mason, and M. C. Wilson, "An exact method for computing the frustration index in signed networks using binary programming", 2016. arXiv: 1611.09030 (cit. on p. 55).

[AFS12] A. Argyriou, R. Foygel, and N. Srebro, "Sparse prediction with the k-support norm", in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1457–1465 (cit. on p. 97).

[Aro+16] S. Arora, Y. Li, Y. Liang, T. Ma, and A. Risteski, "A latent variable model approach to pmi-based word embeddings", *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 385–399, 2016. arXiv: https://www.transacl.org/ojs/index.php/tacl/article/view/742 (cit. on p. 119).

[Ast+16] M. Asteris, A. Kyrillidis, D. Papailiopoulos, and A. G. Dimakis, "Bipartite correlation clustering – maximizing agreements", in *AISTATS*, 2016. arXiv: 1603.02782 (cit. on p. 66).

[ACG02] P. Auer, N. Cesa-Bianchi, and C. Gentile, "Adaptive and self-confident on-line learning algorithms", *Journal of Computer and System Sciences*, vol. 64, no. 1, pp. 48–75, 2002. DOI: 10.1006/jcss.2001.1795 (cit. on p. 34).

[Aus+99] G. Ausiello *et al.*, *Complexity and Approximation*. 1999. DOI: 10.1007/978-3-642-58412-1 (cit. on p. 55).

[Aus80] C. J. Auster, "Balance theory and other extra-balance properties: an application to fairy tales", *Psychological Reports*, vol. 47, no. 1, pp. 183–188, 1980. DOI: 10.2466/pr0.1980.47.1.183 (cit. on p. 48).

[Bac+12a] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Optimization with sparsity-inducing penalties", *Foundations and Trends® in Machine Learning*, vol. 4, no. 1, pp. 1–106, 2012. DOI: 10.1561/2200000015 (cit. on p. 97).

[Bac+12b] G. Bachi, M. Coscia, A. Monreale, and F. Giannotti, "Classifying trust/distrust relationships in online social networks", *2012 International Conference on Privacy, Security, Risk and Trust*, pp. 552–557, 2012. DOI: 10.1109/SocialCom-PASSAT.2012.115 (cit. on p. 23).

[BG11] S. Bagon and M. Galun, "Large scale correlation clustering optimization", 2011 (cit. on pp. 4, 54, 62).

[BMA15] E. Bakshy, S. Messing, and L. Adamic, "Exposure to ideologically diverse news and opinion on facebook", *Science*, 2015. DOI: 10.1126/science.aaa1160. eprint: http://science.sciencemag.org/content/early/2015/05/06/science.aaa1160.full.pdf (cit. on p. 93).

[BB09] M. Balcan and M. Braverman, "Finding low error clusterings", in *Proceedings of the 22nd Conference on Learning Theory*, (Jun. 18–21, 2009), 2009 (cit. on pp. 9, 65).

[Bal+00] P. Baldi, S. Brunak, Y. Chauvin, C. A. Andersen, and H. Nielsen, "Assessing the accuracy of prediction algorithms for classification: an overview.", *Bioinformatics (Oxford, England)*, vol. 16, no. 5, pp. 412–424, 2000. DOI: 10.1093/bioinformatics/16.5.412 (cit. on p. 26).

[BBC02] N. Bansal, A. Blum, and S. Chawla, "Correlation clustering", *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings.*, pp. 238–247, 2002. DOI: 10.1109/SFCS.2002.1181947 (cit. on pp. 9, 56, 57, 59, 64).

[BBC04] ——, "Correlation clustering", *Machine Learning*, vol. 56, no. 1-3, pp. 89–113, 2004. DOI: 10.1023/B:MACH.0000033116.57574.95 (cit. on p. 86).

[BA99] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks", *Science*, vol. 286, no. 5439, pp. 509–512, 1999. DOI: 10.1126/science.286.5439.509 (cit. on p. 83).

[BFG10] M. Barigozzi, G. Fagiolo, and D. Garlaschelli, "Multinetwork of international trade: a commodity-specific analysis", *Physical Review E*, vol. 81, p. 046 104, 4 2010. DOI: 10.1103/PhysRevE.81.046104 (cit. on pp. 7, 91).

[Bar+17] A. Baroni, A. Conte, M. Patrignani, and S. Ruggieri, "Efficiently clustering very large attributed graphs", in *ASONAM 2017*, 2017. DOI: 10.1145/3110025.3110030. arXiv: 1703.08590 (cit. on p. 112).

[Bas+16] L. Bastos *et al.*, "Efficient algorithms for cluster editing", *Journal of Combinatorial Optimization*, vol. 31, no. 1, pp. 347–371, 2016. DOI: 10.1007/s10878-014-9756-7 (cit. on p. 61).

[BHK15] T. Beier, F. A. Hamprecht, and J. H. Kappes, "Fusion moves for correlation clustering", in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3507–3516. DOI: 10.1109/CVPR.2015.7298973 (cit. on pp. 4, 54).

[BHS15] A. Bellet, A. Habrard, and M. Sebban, "Metric learning", *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 9, no. 1, pp. 1–151, 2015. DOI: 10.2200/S00626ED1V01Y201501AIM030 (cit. on p. 110).

[Ben15] S. Ben-David, *Computational feasibility of clustering under clusterability assumptions*, 2015. arXiv: arXiv:1501.00437 (cit. on pp. 9, 65).

[BSY99] A. Ben-Dor, R. Shamir, and Z. Yakhini, "Clustering gene expression patterns.", en, *Journal of computational biology*, vol. 6, no. 3-4, pp. 281–97, 1999. DOI: 10.1089/106652799318274 (cit. on pp. 5, 54, 56, 61, 64).

[BDL06] Y. Bengio, O. Delalleau, and N. Le Roux, "Label propagation and quadratic criterion", in *Semi-Supervised Learning*. 2006, pp. 193–216 (cit. on p. 13).

[BJ17] J. Berg and M. Järvisalo, "Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability", *Artificial Intelligence*, vol. 244, pp. 110–142, 2017. DOI: 10.1016/j.artint.2015.07.001 (cit. on p. 55).

[BGW17] A. Berger, A. Grigoriev, and A. Winokurow, "A ptas for the cluster editing problem on planar graphs", in *Approximation and Online Algorithms: 14th International Workshop, WAOA 2016, Aarhus, Denmark, August 25–26, 2016, Revised Selected Papers*. 2017, pp. 27–39. DOI: 10.1007/978-3-319-51741-4_3 (cit. on p. 55).

[Ber+11] M. Berlingerio, M. Coscia, F. Giannotti, A. Monreale, and D. Pedreschi, "Foundations of multidimensional network analysis", in *2011 International Conference on Advances in Social Networks Analysis and Mining*, 2011, pp. 485–489. DOI: 10.1109/ASONAM.2011.103 (cit. on p. 114).

[BW07] M. Bertolacci and A. Wirth, "Are approximation algorithms for consensus clustering worthwhile?", in *Proceedings of the 2007 SIAM International Conference on Data Mining*. 2007, ch. 41, pp. 437–442. DOI: 10.1137/1.9781611972771.41 (cit. on p. 66).

[Bey+10] A. Beyer, P. Thomason, X. Li, J. Scott, and J. Fisher, "Mechanistic insights into metabolic disturbance during type-2 diabetes and obesity using qualitative networks", in *Transactions on Computational Systems Biology XII: Special Issue on Modeling Methodologies*. 2010, pp. 146–162. DOI: 10.1007/978-3-642-11712-1_4 (cit. on p. 3).

[BH02] J. C. Bezdek and R. J. Hathaway, "Some notes on alternating optimization", in *Advances in Soft Computing — AFSS 2002*. 2002, pp. 288–300. DOI: 10.1007/3-540-45631-7_39 (cit. on p. 103).

[BLW86] N. Biggs, E. K. Lloyd, and R. J. Wilson, *Graph Theory, 1736-1936*. 1986, 252 pp. (cit. on p. 2).

[BAX12] H. Bisgin, N. Agarwal, and X. Xu, "A study of homophily on social media", *World Wide Web*, vol. 15, no. 2, pp. 213–232, 2012. DOI: 10.1007/s11280-011-0143-3 (cit. on p. 92).

[Bis06] C. Bishop, *Pattern Recognition and Machine Learning*. 2006 (cit. on p. 107).

[BC01] A. Blum and S. Chawla, "Learning from labeled and unlabeled data using graph mincuts", in *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001, pp. 19–26 (cit. on p. 13).

[Boc+14] S. Boccaletti *et al.*, "The structure and dynamics of multilayer networks", *Physics Reports*, vol. 544, no. 1, pp. 1–122, 2014. DOI: 10.1016/j.physrep.2014.07.001 (cit. on pp. 7, 91).

[Böc12] S. Böcker, "A golden ratio parameterized algorithm for cluster editing", *Journal of Discrete Algorithms*, vol. 16, pp. 79–89, 2012, Selected papers from the 22nd International Workshop on Combinatorial Algorithms (IWOCA 2011). DOI: 10.1016/j.jda.2012.04.005 (cit. on p. 55).

[BB13] S. Böcker and J. Baumbach, "Cluster editing", in *The Nature of Computation. Logic, Algorithms, Applications: 9th Conference on Computability in Europe, CiE 2013, Milan, Italy, July 1-5*. 2013, pp. 33–44. DOI: 10.1007/978-3-642-39053-1_5 (cit. on pp. 6, 55).

[Bol+11] P. Boldi, M. Rosa, M. Santini, and S. Vigna, "Layered label propagation: a multiresolution coordinate-free ordering for compressing social networks", in *Proceedings of the 20th International Conference on World Wide Web*, 2011, pp. 587–596. DOI: 10.1145/1963405.1963488 (cit. on p. 115).

[Bol04] B. Bollobas, *Extremal Graph Theory*. 2004 (cit. on p. 80).

[BR04] B. Bollobás and O. Riordan, "The diameter of a scale-free random graph", *Combinatorica*, vol. 24, no. 1, pp. 5–34, 2004. DOI: 10.1007/s00493-004-0002-2 (cit. on p. 83).

[BGK13] F. Bonchi, D. García-Soriano, and K. Kutzkov, "Local correlation clustering", 2013. arXiv: 1312.5105 (cit. on pp. 60, 63).

[BGL14] F. Bonchi, D. García-Soriano, and E. Liberty, "Correlation clustering: from theory to practice", in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014 (cit. on p. 52).

[Bon+15] F. Bonchi, A. Gionis, F. Gullo, C. E. Tsourakakis, and A. Ukkonen, "Chromatic correlation clustering", *ACM Trans. Knowl. Discov. Data*, vol. 9, no. 4, 34:1–34:24, 2015. DOI: 10.1145/2728170 (cit. on p. 66).

[Bon+12] F. Bonchi, A. Gionis, F. Gullo, and A. Ukkonen, "Chromatic correlation clustering", in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 1321–1329. DOI: 10.1145/2339530.2339735 (cit. on p. 66).

[BGU12] F. Bonchi, A. Gionis, and A. Ukkonen, "Overlapping correlation clustering", *Knowledge and Information Systems*, vol. 35, no. 1, pp. 1–32, 2012. DOI: 10.1007/s10115-012-0522-9 (cit. on p. 66).

[Bon+08a] P. Bonizzoni, G. D. Vedova, R. Dondi, and T. Jiang, "On the approximation of correlation clustering and consensus clustering", *Journal of Computer and System Sciences*, vol. 74, no. 5, pp. 671–696, 2008. DOI: 10.1016/j.jcss.2007.06.024 (cit. on p. 60).

[Bon+08b] ——, "On the approximation of correlation clustering and consensus clustering", *Journal of Computer and System Sciences*, vol. 74, no. 5, pp. 671–696, 2008. DOI: 10.1016/j.jcss.2007.06.024 (cit. on p. 65).

[Bor+13] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data", in *Advances in Neural Information Processing Systems 26*, 2013, pp. 2787–2795 (cit. on pp. 10, 110).

[Bot+15] C. Bothorel, J. D. Cruz, M. Magnani, and B. Micenková, "Clustering attributed graphs: models, measures and methods", *Network Science*, vol. 3, no. 3, pp. 408–444, 2015. DOI: 10.1017/nws.2015.9. arXiv: arXiv:1501.01676 (cit. on pp. 91, 112).

[BLT12] G. S. Brodal, G. Lagogiannis, and R. E. Tarjan, "Strict fibonacci heaps", in *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, 2012, pp. 1177–1184. DOI: 10.1145/2213977.2214082 (cit. on p. 72).

[BSW17] N. Buchbinder, R. Schwartz, and B. Weizman, "Simplex transformations and the multiway cut problem", in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2017, pp. 2400–2410. DOI: 10.1137/1.9781611974782.158 (cit. on p. 86).

[Bur10] C. J. C. Burges, "Dimension reduction: a guided tour", *Foundations and Trends in Machine Learning*, vol. 2, no. 4, pp. 275–365, 2010. DOI: 10.1561/2200000002 (cit. on p. 8).

[Cai+05] D. Cai, Z. Shao, X. He, X. Yan, and J. Han, "Community mining from multi-relational networks", in *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*. 2005, pp. 445–452. DOI: 10.1007/11564126_44 (cit. on pp. 7, 91).

[CZC17] H. Cai, V. W. Zheng, and K. C.-C. Chang. (2017). A comprehensive survey of graph embedding: problems, techniques and applications. under review of IEEE TKDE. arXiv: 1709.07604 (cit. on p. 22).

[CH56] D. Cartwright and F. Harary, "Structural balance: a generalization of heider's theory.", *Psychological Review*, vol. 63, no. 5, pp. 277–293, 1956. DOI: 10.1037/h0046049 (cit. on p. 21).

[Ces+12a] N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella, "A linear time active learning algorithm for link classification", in *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012, pp. 1–12 (cit. on pp. 25, 84).

[Ces+12b] ——, "A correlation clustering approach to link classification in signed networks", in *Proceedings of the 25th Annual Conference on Learning Theory*, vol. 23, 2012, pp. 1–20 (cit. on pp. 9, 10, 25, 53, 70, 85).

[Ces+13] ——, "Random spanning trees and the prediction of weighted graphs", *Journal of Machine Learning Research*, vol. 14, pp. 1251–1284, 2013 (cit. on pp. 13, 21).

[Ces+97] N. Cesa-Bianchi *et al.*, "How to use expert advice", *J. ACM*, vol. 44, no. 3, pp. 427–485, 1997. DOI: 10.1145/258128.258179 (cit. on p. 34).

[Cha+17] E. Chandrasekharan *et al.*, "You can't stay here: the efficacy of reddit's 2015 ban examined through hate speech", *Proc. ACM Hum.-Comput. Interact.*, vol. 1, no. 2, 31:22–31:22, 2017. DOI: 10.1145/3134666 (cit. on p. 12).

[CSZ06] O. Chapelle, B. Scholkopf, and A. Zien, *Semi-Supervised Learning*. 2006, 528 pp. DOI: 10.7551/mitpress/9780262033589.001.0001 (cit. on p. 3).

[CGS17] M. Charikar, N. Gupta, and R. Schwartz, "Local guarantees in graph cuts and clustering", in *19th Conference on Integer Programming and Combinatorial Optimization*, 2017. arXiv: 1704.00355 (cit. on p. 67).

[CGW03] M. Charikar, V. Guruswami, and A. Wirth, "Clustering with qualitative information", in *44th Annual IEEE Symposium on Foundations of Computer Science*, 2003, pp. 524–533. DOI: 10.1109/SFCS.2003.1238225 (cit. on pp. 9, 56–59, 67).

[CW04] M. Charikar and A. Wirth, "Maximizing quadratic programs: extending grothendieck's inequality", in *45th Annual IEEE Symposium on Foundations of Computer Science*, 2004, pp. 54–60. DOI: 10.1109/FOCS.2004.39 (cit. on p. 58).

[Cha+06] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar, "On the hardness of approximating multicut and sparsest-cut", *Computational complexity*, vol. 15, no. 2, pp. 94–114, 2006. DOI: 10.1007/s00037-006-0210-9 (cit. on p. 58).

[Cha+14] S. Chawla, K. Makarychev, T. Schramm, and G. Yaroslavtsev, *Near optimal lp rounding algorithm for correlation clustering on complete and complete k-partite graphs*, 2014. eprint: arXiv:1412.0681 (cit. on p. 59).

[Cha+15] ——, "Near optimal lp rounding algorithm for correlationclustering on complete and complete k-partite graphs", in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing - STOC '15*, 2015, pp. 219–228. DOI: 10.1145/2746539.2746604. arXiv: 1412.0681 (cit. on pp. 59, 66).

[CLC13] W. Chen, L. V. Lakshmanan, and C. Castillo, "Information and influence propagation in social networks", *Synthesis Lectures on Data Management*, vol. 5, no. 4, 2013. DOI: 10.2200/S00527ED1V01Y201308DTM037 (cit. on p. 3).

[Che+14] Y. Chen, X. L. Wang, B. Yuan, and B. Z. Tang, "Overlapping community detection in networks with positive and negative links", *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2014, no. 3, 2014. DOI: 10.1088/1742-5468/2014/03/P03021. arXiv: 1310.4023 (cit. on p. 69).

[CDL15] J. Cheng, C. Danescu-Niculescu-Mizil, and J. Leskovec, "Antisocial behavior in online discussion communities", in *International AAAI Conference on Web and Social Media*, 2015 (cit. on p. 24).

[CLL17] K. Cheng, J. Li, and H. Liu, "Unsupervised feature selection in signed social networks", in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '17*, 2017, pp. 777–786. DOI: 10.1145/3097983.3098106 (cit. on p. 119).

[Che+10] V. Chepoi, F. F. Dragan, I. Newman, Y. Rabinovich, and Y. Vaxès, "Constant approximation algorithms for embedding graph metrics into trees and outerplanar graphs", in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX-RANDOM*. 2010, pp. 95–109. DOI: 10.1007/978-3-642-15369-3_8 (cit. on pp. 82, 83).

[Chi+07] C. Chiang, A. B. Kahng, S. Sinha, X. Xu, and A. Z. Zelikovsky, "Fast and efficient bright-field aapsm conflict detection and correction", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 1, pp. 115–126, 2007. DOI: 10.1109/TCAD.2006.882642 (cit. on p. 6).

[Chi+14] K.-y. Chiang, C.-j. Hsieh, N. Natarajan, I. S. Dhillon, and A. Tewari, "Prediction and clustering in signed networks: a local to global perspective", *Journal of Machine Learning Research*, vol. 15, pp. 1177–1213, 2014 (cit. on pp. 9, 22, 30, 56).

[CWD12] K.-Y. Chiang, J. J. Whang, and I. S. Dhillon, "Scalable clustering of signed networks using balance normalized cut", in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, 2012, pp. 615–624. DOI: 10.1145/2396761.2396841 (cit. on p. 69).

[CDK14] F. Chierichetti, N. Dalvi, and R. Kumar, "Correlation clustering in mapreduce", in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, 2014, pp. 641–650. DOI: 10.1145/2623330.2623743 (cit. on p. 59).

[Chi+09] F. Chierichetti *et al.*, "On compressing social networks", in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 219–228. DOI: 10.1145/1557019.1557049 (cit. on p. 115).

[Chu+16] L. Chu *et al.*, "Finding gangs in war from signed networks", in *KDD'16*, 2016, pp. 799–809. DOI: 10.1145/2939672.2939855 (cit. on p. 69).

[CMN08] A. Clauset, C. Moore, and M. E. J. Newman, "Hierarchical structure and the prediction of missing links in networks", *Nature*, vol. 453, no. 7191, pp. 98–101, 2008. DOI: 10.1038/nature06830 (cit. on p. 3).

[CSW08a] T. Coleman, J. Saunderson, and A. Wirth, "A local-search 2-approximation for 2-correlation-clustering", in *Algorithms - ESA 2008: 16th Annual European Symposium, Karlsruhe, Germany, September 15-17, 2008. Proceedings*. 2008, pp. 308–319. DOI: 10.1007/978-3-540-87744-8_26 (cit. on p. 60).

[CSW08b] ——, "Spectral clustering with inconsistent advice", in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 152–159. DOI: 10.1145/1390156.1390176 (cit. on p. 68).

[CO16] P. Compagnon and K. Olliver, "Graph embeddings for social network analysis, State of the art", Master's thesis, INSA Lyon, 2016 (cit. on p. 22).

[Cor+09] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd. 2009 (cit. on p. 71).

[Cos+13] M. Coscia, G. Rossetti, D. Pennacchioli, D. Ceccarelli, and F. Giannotti, ""you know because i know": a multidimensional network approach to human resources problem", in *2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2013)*, 2013, pp. 434–441. DOI: 10.1145/2492517.2492537 (cit. on p. 91).

[CMM15] S. J. Cranmer, E. J. Menninga, and P. J. Mucha, "Kantian fractionalization predicts the conflict propensity of the international system", *Proceedings of the National Academy of Sciences*, vol. 112, no. 38, pp. 11 812–11 816, 2015. DOI: 10.1073/pnas.1509423112. eprint: http://www.pnas.org/content/112/38/11812.full.pdf (cit. on pp. 7, 91).

[Cui+07] Q. Cui *et al.*, "A map of human cancer signaling", *Molecular Systems Biology*, vol. 3, no. 1, 2007. DOI: 10.1038/msb4100200. eprint: http://msb.embopress.org/content/3/1/152.full.pdf (cit. on p. 46).

[dFon+11] L. da Fontoura Costa *et al.*, "Analyzing and modeling real-world phenomena with complex networks: a survey of applications", *Advances in Physics*, vol. 60, no. 3, pp. 329–412, 2011. DOI: 10.1080/00018732.2011.572452 (cit. on p. 2).

[Dán14] Dániel Marx and Igor Razgon, "Fixed-parameter tractability of multicut parameterized by the size of the cutset", *SIAM Journal on Computing*, vol. 43, no. 2, pp. 355–388, 2014. DOI: 10.1137/110855247 (cit. on p. 56).

[DV15] S. Das and H. Vikalo, "Sdhap: haplotype assembly for diploids and polyploids via semi-definite programming.", *BMC genomics*, vol. 16, no. 1, p. 260, 2015. DOI: 10.1186/s12864-015-1408-5 (cit. on pp. 5, 54).

[Das+07] B. DasGupta, G. A. Enciso, E. Sontag, and Y. Zhang, "Algorithmic and complexity results for decompositions of biological networks into monotone subsystems", *Biosystems*, vol. 90, no. 1, pp. 161–178, 2007. DOI: 10.1016/j.biosystems.2006.08.001 (cit. on pp. 5, 54).

[DB13] J. Davies and F. Bacchus, "Exploiting the power of mip solvers in maxsat", in *16th International Conference on Theory and Applications of Satisfiability Testing, SAT 2013, Helsinki, Finland, July 8-12, 2013*. 2013, pp. 166–181. DOI: 10.1007/978-3-642-39071-5_13 (cit. on p. 56).

[Dav67] J. A. Davis, "Clustering and structural balance in graphs", *Human Relations*, vol. 20, no. 2, pp. 181–187, 1967. DOI: 10.1177/001872676702000206 (cit. on pp. 21, 49, 51).

[De +17] C. De Bacco, E. A. Power, D. B. Larremore, and C. Moore, "Community detection, link prediction, and layer interdependence in multilayer networks", *Physical Review E*, vol. 95, no. 4, p. 042 317, 2017. DOI: 10.1103/PhysRevE.95.042317. arXiv: 1701.01369 (cit. on p. 113).

[De +14] P. De Meo, E. Ferrara, G. Fiumara, and A. Provetti, "On facebook, most ties are weak", *Commun. ACM*, vol. 57, no. 11, pp. 78–84, 2014. DOI: 10.1145/2629438 (cit. on p. 91).

[dKD08] C. de Kerchove and P. V. Dooren, "The pagetrust algorithm: how to rank web pages when negative links are allowed?", in *Proceedings of the 2008 SIAM International Conference on Data Mining*. 2008, pp. 346–352. DOI: 10.1137/1.9781611972788.31 (cit. on p. 23).

[Dem+06] E. D. Demaine, D. Emanuel, A. Fiat, and N. Immorlica, "Correlation clustering in general weighted graphs", *Theoretical Computer Science*, vol. 361, no. 2-3, pp. 172–187, 2006. DOI: 10.1016/j.tcs.2006.05.008 (cit. on pp. 53, 57).

[DI03] E. D. Demaine and N. Immorlica, "Correlation clustering with partial information", in *6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2003, Princeton, USA*. 2003, pp. 1–13. DOI: 10.1007/978-3-540-45198-3_1 (cit. on p. 57).

[Dia15] N. Diakopoulos, "Picking the nyt picks: editorial criteria and automation in the curation of online news comments", *International Symposium on Online Journalism Journal*, vol. 5, no. 1, pp. 147–166, 2015 (cit. on p. 12).

[DNG07] C. P. Diehl, G. Namata, and L. Getoor, "Relationship identification for social network discovery", in *Proceedings of the 22nd National Conference on Artificial Intelligence*, 2007, pp. 546–552 (cit. on p. 91).

[DM96] P. Doreian and A. Mrvar, "A partitioning approach to structural balance", *Social Networks*, vol. 18, no. 2, pp. 149–168, 1996. DOI: 10.1016/0378-8733(95)00259-6 (cit. on pp. 56, 61).

[Dör+14] M. Dörnfelder, J. Guo, C. Komusiewicz, and M. Weller, "On the parameterized complexity of consensus clustering", *Theoretical Computer Science*, vol. 542, pp. 71–82, 2014. DOI: 10.1016/j.tcs.2014.05.002 (cit. on p. 66).

[DSW10] N. Downing, P. J. Stuckey, and A. Wirth, "Improved consensus clustering via linear programming", in *Proceedings of the Thirty-Third Australasian Conference on Computer Science - Volume 102*, 2010, pp. 61–70 (cit. on p. 66).

[Dru+13] L. Drummond, R. Figueiredo, Y. Frota, and M. Levorato, "Efficient solution of the correlation clustering problem: an application to structural balance", in *On the Move to Meaningful Internet Systems: OTM 2013, Graz, Austria, September 9 - 13*. 2013, pp. 674–683. DOI: 10.1007/978-3-642-41033-8_85 (cit. on p. 61).

[DP09] D. P. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomized Algorithms*. 2009. DOI: 10.1017/CBO9780511581274 (cit. on p. 37).

[Dun16] R. I. M. Dunbar, "Do online social media cut through the constraints that limit the size of offline social networks?", *Open Science*, vol. 3, no. 1, 2016. DOI: 10.1098/rsos.150292. eprint: http://rsos.royalsocietypublishing.org/content/3/1/150292.full.pdf (cit. on p. 19).

[DKK11] D. M. Dunlavy, T. G. Kolda, and W. P. Kegelmeyer, "Multilinear algebra for analyzing data with multiple linkages", in *Graph Algorithms in the Language of Linear Algebra*. 2011, ch. 7, pp. 85–114. DOI: 10.1137/1.9780898719918.ch7. eprint: http://www.cs.sandia.gov/~dmdunla/publications/DuKoKe10.pdf (cit. on pp. 7, 91).

[EK10] D. Easley and J. Kleinberg, "Chapter 5 positive and negative relationships", in *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*, 2010, ch. 5, pp. 119–152 (cit. on pp. 50, 51).

[Edm65] J. Edmonds, "Paths, trees, and flowers", *Canadian Journal of Mathematics*, vol. 17, pp. 449–467, 1965. DOI: 10.4153/cjm-1965-045-4 (cit. on p. 19).

[EV13] E. Elhamifar and R. Vidal, "Sparse subspace clustering: algorithm, theory, and applications", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013. DOI: 10.1109/TPAMI.2013.57 (cit. on p. 111).

[Elk+05] M. Elkin, Y. Emek, D. A. Spielman, and S.-H. Teng, "Lower-stretch spanning trees", in *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, 2005, pp. 494–503. DOI: 10.1145/1060590.1060665 (cit. on p. 81).

[EN17] M. Elkin and O. Neiman, "Efficient algorithms for constructing very sparse spanners and emulators", in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2017, pp. 652–669 (cit. on pp. 10, 81).

[EP07] M. Elkin and D. Peleg, "The hardness of approximating spanner problems", *Theory of Computing Systems*, vol. 41, no. 4, pp. 691–729, 2007. DOI: 10.1007/s00224-006-1266-2 (cit. on p. 81).

[ES09] M. Elsner and W. Schudy, "Bounding and comparing methods for correlation clustering beyond ilp", in *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, 2009, pp. 19–27 (cit. on pp. 5, 54, 61).

[EF03] D. Emanuel and A. Fiat, "Correlation clustering: minimizing disagreements on arbitrary weighted graphs", in *11th Annual European Symposium on Algorithms, ESA 2003, Budapest, Hungary*. 2003, pp. 208–220. DOI: 10.1007/978-3-540-39658-1_21 (cit. on pp. 56, 57).

[ER59] P. Erdős and A. Rényi, "On random graphs i", *Publicationes Mathematicae*, vol. 6, pp. 290–297, 1959 (cit. on p. 105).

[EB14] E. Estrada and M. Benzi, "Walk-based measure of balance in signed networks: detecting lack of balance in social networks", *Phys. Rev. E*, vol. 90, no. 4, p. 42 802, 2014. DOI: 10.1103/PhysRevE.90.042802 (cit. on p. 51).

[Eul41] L. Euler, "Solutio problematis ad geometriam situs pertinentis", *Commentarii academiae scientiarum Petropolitanae*, vol. 8, pp. 128–140, 1741 (cit. on p. 2).

[EL09] T. S. Evans and R. Lambiotte, "Line graphs, link partitions, and overlapping communities", *Phys. Rev. E*, vol. 80, p. 016 105, 1 2009. DOI: 10.1103/PhysRevE.80.016105. eprint: arXiv:0903.2181 (cit. on pp. 10, 110).

[FIA11] G. Facchetti, G. Iacono, and C. Altafini, "Computing global structural balance in large-scale signed social networks.", *Proceedings of the National Academy of Sciences of the United States of America*, vol. 108, no. 52, pp. 20 953–8, 2011. DOI: 10.1073/pnas.1109521108 (cit. on pp. 51, 62).

[FRT03] J. Fakcharoenphol, S. Rao, and K. Talwar, "A tight bound on approximating arbitrary metrics by tree metrics", in *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, 2003, pp. 448–455. DOI: 10.1145/780542.780608 (cit. on pp. 82, 83).

[Fan+16] Y. Fang, R. Cheng, S. Luo, and J. Hu, "Effective community search for large attributed graphs", *Proc. VLDB Endow.*, vol. 9, no. 12, pp. 1233–1244, 2016. DOI: 10.14778/2994509.2994538 (cit. on p. 113).

[FM13] R. Figueiredo and G. Moura, "Mixed integer programming formulations for clustering problems related to structural balance", *Social Networks*, vol. 35, no. 4, pp. 639–651, 2013. DOI: 10.1016/j.socnet.2013.09.002 (cit. on p. 55).

[FS03] V. Filkov and S. Skiena, "Integrating microarray data by consensus clustering", in *15th IEEE International Conference on Tools with Artificial Intelligence*, 2003, pp. 418–426. DOI: 10.1109/TAI.2003.1250220 (cit. on p. 65).

[FG08] V. Filkov and A. Goder, "Consensus clustering algorithms: comparison and refinement", in *Proceedings of the Tenth Workshop on Algorithm Engineering and Experiments (ALENEX)*. 2008, ch. 10, pp. 109–117. DOI: 10.1137/1.9781611972887.11 (cit. on p. 66).

[FS16] A. Filtser and S. Solomon, "The greedy spanner is existentially optimal", in *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, 2016, pp. 9–17. DOI: 10.1145/2933057.2933114 (cit. on p. 81).

[Fom+14] F. V. Fomin, S. Kratsch, M. Pilipczuk, M. Pilipczuk, and Y. Villanger, "Tight bounds for parameterized complexity of cluster editing with a small number of clusters", *Journal of Computer and System Sciences*, vol. 80, no. 7, pp. 1430–1447, 2014. DOI: 10.1016/j.jcss.2014.04.015. arXiv: 1112.4419 (cit. on p. 55).

[For10] S. Fortunato, "Community detection in graphs", *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010. DOI: 10.1016/j.physrep.2009.11.002 (cit. on p. 3).

[FH16] S. Fortunato and D. Hric, "Community detection in networks: a user guide", *Physics Reports*, vol. 659, pp. 1–44, 2016. DOI: 10.1016/j.physrep.2016.09.002. arXiv: 1608.00163 (cit. on p. 112).

[FW56] M. Frank and P. Wolfe, "An algorithm for quadratic programming", *Naval Research Logistics Quarterly*, vol. 3, no. 1-2, pp. 95–110, 1956. DOI: 10.1002/nav.3800030109 (cit. on p. 102).

[GZ07] J. V. Gael and X. Zhu, "Correlation clustering for crosslingual link detection", in *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, 2007, pp. 1744–1749 (cit. on pp. 6, 54).

[Gam+16] S. Gama-Castro *et al.*, "Regulondb version 9.0: high-level integration of gene regulation, coexpression, motif clustering and beyond", *Nucleic Acids Research*, vol. 44, no. D1, pp. D133–D143, 2016. DOI: 10.1093/nar/gkv1156 (cit. on p. 46).

[Gao+16] M. Gao, E.-P. Lim, D. Lo, and P. K. Prasetyo, "On detecting maximal quasi antagonistic communities in signed graphs", *Data Mining and Knowledge Discovery*, vol. 30, no. 1, pp. 99–146, 2016. DOI: 10.1007/s10618-015-0405-2 (cit. on p. 69).

[GVY93] N. Garg, V. V. Vazirani, and M. Yannakakis, "Approximate max-flow min-(multi)cut theorems and their applications", in *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, 1993, pp. 698–707. DOI: 10.1145/167088.167266 (cit. on p. 58).

[Gar+17] K. Garimella, A. Gionis, N. Parotsidis, and N. Tatti, "Balancing information exposure in social networks", in *Advances in Neural Information Processing Systems 30*, 2017. arXiv: arXiv:1709.01491 (cit. on pp. 4, 93).

[GL04] D. Garlaschelli and M. I. Loffredo, "Patterns of link reciprocity in directed networks", *Physical Review Letters*, vol. 93, p. 268 701, 26 2004. DOI: 10.1103/PhysRevLett.93.268701 (cit. on p. 26).

[GHP13] C. Gentile, M. Herbster, and S. Pasteris, "Online similarity prediction of networked data from known and unknown graphs", *JMLR Workshop and Conference Proceedings*, vol. 30, pp. 1–34, 2013. arXiv: 1302.7263 (cit. on p. 25).

[Gil59] E. Gilbert, "Random graphs", English, *Ann. Math. Stat.*, vol. 30, pp. 1141–1144, 1959. DOI: 10.1214/aoms/1177706098 (cit. on p. 105).

[GMT07] A. Gionis, H. Mannila, and P. Tsaparas, "Clustering aggregation", *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, 4–es, 2007. DOI: 10.1145/1217299.1217303 (cit. on pp. 5, 61, 65).

[GG06] I. Giotis and V. Guruswami, "Correlation clustering with a fixed number of clusters", *Theory of Computing*, vol. 2, no. 1, pp. 249–266, 2006. DOI: 10.4086/toc.2006.v002a013 (cit. on pp. 9, 59, 60).

[GW95] M. X. Goemans and D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming", *Journal of the ACM*, vol. 42, no. 6, pp. 1115–1145, 1995. DOI: 10.1145/227683.227684 (cit. on p. 55).

[GM14] S. A. Golder and M. W. Macy, "Digital footprints: opportunities and challenges for online social research", *Annual Review of Sociology*, vol. 40, no. 1, pp. 129–152, 2014. DOI: 10.1146/annurev-soc-071913-043145 (cit. on p. 114).

[Gor+17] G. Gori *et al.*, "Flowrep: descriptive curve networks for free-form design shapes", *ACM Transactions on Graphics*, vol. 36, no. 4, 59:1–59:14, 2017. DOI: 10.1145/3072959.3073639 (cit. on pp. 4, 54).

[GB16] J. Gothania and B. Buksh, "A fast chromatic correlation clustering algorithm", in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2016, pp. 1870–1874. DOI: 10.1109/ICACCI.2016.7732322 (cit. on p. 66).

[GL16] A. Grover and J. Leskovec, "Node2vec: scalable feature learning for networks", in *KDD'16*, 2016 (cit. on p. 120).

[Grü05] P. Grünwald, "A tutorial introduction to the minimum description length principle", *Advances in minimum description length: Theory and applications*, pp. 23–81, 2005. eprint: arXiv:math/0406077 (cit. on p. 96).

[Guh+04] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust", in *Proceedings of the 13th conference on World Wide Web - WWW '04*, ACM, 2004, p. 403. DOI: 10.1145/988672.988727 (cit. on p. 24).

[Gup+17] A. Gupta, R. Ravi, K. Talwar, and S. W. Umboh, "Last but not least: online spanners for buy-at-bulk", in *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*. 2017, pp. 589–599. DOI: 10.1137/1.9781611974782.38. arXiv: arXiv:1611.00052 (cit. on pp. 82, 83).

[HYL17] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs", in *Advances in Neural Information Processing Systems 30*, 2017. arXiv: arXiv:1706.02216 (cit. on p. 3).

[Har53] F. Harary, "On the notion of balance of a signed graph", *Michigan Math. J.*, vol. 2, no. 2, pp. 143–146, 1953. DOI: 10.1307/mmj/1028989917 (cit. on pp. 6, 50).

[HLW02] F. Harary, M. Lim, and D. C. Wunsch, "Signed graphs for portfolio analysis in risk management", *IMA Journal of Management Mathematics*, vol. 13, no. 3, pp. 201–210, 2002. DOI: 10.1093/imaman/13.3.201 (cit. on p. 6).

[Har10] C. Hardaker, "Trolling in asynchronous computer-mediated communication: from user discussions to academic definitions", *Journal of Politeness Research. Language, Behaviour, Culture*, vol. 6, no. 2, pp. 215–242, 2010. DOI: 10.1515/jplr.2010.011 (cit. on p. 12).

[HAR12] A. Hassan, A. Abu-Jbara, and D. Radev, "Extracting signed social networks from text", in *Workshop on Graph-based Methods for Natural Language Processing*, 2012, pp. 6–14 (cit. on p. 11).

[Has+09] O. Hassanzadeh, F. Chiang, H. C. Lee, and R. J. Miller, "Framework for evaluating clustering algorithms in duplicate detection", *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 1282–1293, 2009. DOI: 10.14778/1687627.1687771 (cit. on pp. 6, 54).

[Hei46] F. Heider, "Attitudes and cognitive organization", *The Journal of Psychology*, vol. 21, no. 1, pp. 107–112, 1946. DOI: 10.1080/00223980.1946.9917275 (cit. on p. 48).

[Hei58] ——, *The psychology of interpersonal relations*. 1958, p. 326. DOI: 10.1037/10628-000 (cit. on pp. 6, 21, 48).

[HLP09] M. Herbster, G. Lever, and M. Pontil, "Online prediction on large diameter graphs", in *Advances in Neural Information Processing Systems 21*, 2009, pp. 649–656 (cit. on p. 13).

[HPG15] M. Herbster, S. Pasteris, and S. Ghosh, "Online prediction at the limit of zero temperature", in *Advances in Neural Information Processing Systems 28*, 2015, pp. 2935–2943 (cit. on p. 13).

[HPV12] M. Herbster, S. Pasteris, and F. Vitale, "Online sum-product computation over trees", in *Advances in Neural Information Processing Systems 25*, 2012, pp. 2870–2878 (cit. on p. 13).

[HP07] M. Herbster and M. Pontil, "Prediction on a graph with a perceptron", in *Advances in Neural Information Processing Systems 19*, 2007, pp. 577–584 (cit. on p. 13).

[HL81] P. W. Holland and S. Leinhardt, "An exponential family of probability distributions for directed graphs", *Journal of the American Statistical Association*, vol. 76, no. 373, pp. 33–50, 1981. DOI: 10.1080/01621459.1981.10477598. eprint: http://www.tandfonline.com/doi/pdf/10.1080/01621459.1981.10477598 (cit. on p. 17).

[Hon+13] W. Hong, L. Li, T. Li, and W. Pan, "Ihr: an online recruiting system for xiamen talent service center", *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM*, pp. 1177–1185, 2013. DOI: 10.1145/2487575.2488199 (cit. on p. 118).

[Hou+16] J. P. Hou, A. Emad, G. J. Puleo, J. Ma, and O. Milenkovic, "A new correlation clustering method for cancer mutation analysis", *Bioinformatics*, vol. 32, no. 24, pp. 3717–3728, 2016. DOI: 10.1093/bioinformatics/btw546. arXiv: 1601.06476 (cit. on p. 67).

[HDF14] D. Hric, R. K. Darst, and S. Fortunato, "Community detection in networks: structural communities versus ground truth", *Phys. Rev. E*, vol. 90, p. 062 805, 6 2014. DOI: 10.1103/PhysRevE.90.062805 (cit. on p. 92).

[Hu74] T. C. Hu, "Optimum communication spanning trees", *SIAM Journal on Computing*, vol. 3, no. 3, pp. 188–195, 1974. DOI: 10.1137/0203015 (cit. on p. 80).

[HLH17] X. Huang, J. Li, and X. Hu, "Accelerated attributed network embedding", in *Proceedings of the 2017 SIAM International Conference on Data Mining*. 2017, pp. 633–641. DOI: 10.1137/1.9781611974973.71 (cit. on pp. 119, 120).

[HCY15] X. Huang, H. Cheng, and J. X. Yu, "Dense community detection in multi-valued attributed networks", *Information Sciences*, vol. 314, pp. 77–99, 2015. DOI: 10.1016/j.ins.2015.03.075 (cit. on p. 111).

[HBN10] F. Hüffner, N. Betzler, and R. Niedermeier, "Separator-based data reduction for signed graph balancing", *Journal of Combinatorial Optimization*, vol. 20, no. 4, pp. 335–360, 2010. DOI: 10.1007/s10878-009-9212-2 (cit. on p. 50).

[HL78] T. L. Huston and G. Levinger, "Interpersonal attraction and relationships", *Annual Review of Psychology*, vol. 29, no. 1, pp. 115–156, 1978. DOI: 10.1146/annurev.ps.29.020178.000555 (cit. on p. 92).

[IK04] T. Idé and H. Kashima, "Eigenspace-based anomaly detection in computer systems", in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 440–449. DOI: 10.1145/1014052.1014102 (cit. on p. 3).

[IIK16] V. Il'ev, S. Il'eva, and A. Kononov, "Short survey on graph correlation clustering with minimization criteria", in *Discrete Optimization and Operations Research: 9th International Conference, DOOR 2016, Vladivostok, Russia, September 19-23, 2016, Proceedings*. 2016, pp. 25–36. DOI: 10.1007/978-3-319-44914-2_3 (cit. on p. 52).

[IPR17] M. R. Islam, B. A. Prakash, and N. Ramakrishnan, "Signet: scalable embeddings for signed networks", 2017. arXiv: 1702.06819 (cit. on pp. 9, 22).

[Jag13] M. Jaggi, "Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization", in *Proceedings of the 30th International Conference on Machine Learning*, 2013 (cit. on p. 102).

[Jia15] J. Q. Jiang, "Stochastic block model and exploratory analysis in signed networks", *Physical Review E*, vol. 91, no. 6, 2015. DOI: 10.1103/PhysRevE.91.062805. arXiv: 1501.00594 (cit. on pp. 6, 54, 69).

[JH05] T. Joachims and J. Hopcroft, "Error bounds for correlation clustering", in *Proceedings of the 22nd international conference on Machine learning - ICML '05*, 2005, pp. 385–392. DOI: 10.1145/1102351.1102400 (cit. on p. 64).

[Joh+15] F. D. Johansson, A. Chattoraj, C. Bhattacharyya, and D. Dubhashi, "Weighted theta functions and embeddings with applications to max-cut, clustering and summarization", in *Advances in Neural Information Processing Systems 28*, 2015, pp. 1018–1026 (cit. on p. 67).

[JLK78] D. S. Johnson, J. K. Lenstra, and A. H. G. R. Kan, "The complexity of the network design problem", *Networks*, vol. 8, no. 4, pp. 279–285, 1978. DOI: 10.1002/net.3230080402 (cit. on p. 82).

[JG08] J. Juvonen and E. F. Gross, "Extending the school grounds?—bullying experiences in cyberspace", *Journal of School Health*, vol. 78, no. 9, pp. 496–505, 2008. DOI: 10.1111/j.1746-1561.2008.00335.x (cit. on p. 12).

[Kal+10] P. Kaluza, A. Kölzsch, M. T. Gastner, and B. Blasius, "The complex network of global cargo ship movements", *Journal of The Royal Society Interface*, vol. 7, no. 48, pp. 1093–1103, 2010. DOI: 10.1098/rsif.2009.0495. eprint: http://rsif.royalsocietypublishing.org/content/7/48/1093.full.pdf (cit. on pp. 7, 91).

[KM07] P. Kanani and A. McCallum, "Resource-bounded information gathering for correlation clustering", in *Learning Theory: 20th Annual Conference on Learning Theory, COLT 2007, San Diego, CA, USA; June 13-15*. 2007, pp. 625–627. DOI: 10.1007/978-3-540-72927-3_46 (cit. on p. 63).

[KMP07] P. Kanani, A. McCallum, and C. Pal, "Improving author coreference by resource-bounded information gathering from the web", in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, 2007, pp. 429–434 (cit. on p. 63).

[Kap+16] J. H. Kappes, P. Swoboda, B. Savchynskyy, T. Hazan, and C. Schnörr, "Multicuts and perturb & map for probabilistic graph clustering", *Journal of Mathematical Imaging and Vision*, vol. 56, no. 2, pp. 221–237, 2016. DOI: 10.1007/s10851-016-0659-3. arXiv: 1601.02088 (cit. on p. 62).

[Kar89] R. M. Karp, "A 2k-competitive algorithm for the circle", *Manuscript*, 1989 (cit. on p. 81).

[KS09] M. Karpinski and W. Schudy, "Linear time approximation schemes for the gale-berlekamp game and related minimization problems", in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, 2009, pp. 313–322. DOI: 10.1145/1536414.1536458 (cit. on p. 60).

[Kat+16] S. Kataoka, T. Kobayashi, M. Yasuda, and K. Tanaka, "Community detection algorithm combining stochastic block model and attribute data clustering", *Journal of the Physical Society of Japan*, vol. 85, no. 11, p. 114 802, 2016. DOI: 10.7566/JPSJ.85.114802. arXiv: 1608.00920 (cit. on pp. 104, 113, 114).

[KMK11] P. Kazienko, K. Musial, and T. Kajdanowicz, "Multidimensional social network in the social recommender system", *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 41, no. 4, pp. 746–759, 2011. DOI: 10.1109/TSMCA.2011.2132707 (cit. on pp. 7, 91).

[Kéf+15] S. Kéfi *et al.*, "Network structure beyond food webs: mapping non-trophic and trophic interactions on chilean rocky shores", *Ecology*, vol. 96, no. 1, pp. 291–303, 2015. DOI: 10.1890/13-1424.1 (cit. on pp. 8, 91).

[KKT15] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network", *Theory of Computing*, vol. 11, no. 4, pp. 105–147, 2015. DOI: 10.4086/toc.2015.v011a004 (cit. on p. 3).

[KT11] A.-M. Kermarrec and C. Thraves, "Can everybody sit closer to their friends than their enemies?", in *Proceedings of the 36th International Symposium on Mathematical Foundations of Computer Science*. 2011, pp. 388–399. DOI: 10.1007/978-3-642-22993-0_36. arXiv: arXiv:1405.5023 (cit. on p. 6).

[Kho02] S. Khot, "On the power of unique 2-prover 1-round games", in *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, 2002, pp. 767–775. DOI: 10.1145/509907.510017 (cit. on p. 57).

[KA15] B. Kille and F. Abel, "We know where you should work next summer: job recommendations", in *Proceedings of the 9th ACM Conference on Recommender Systems - RecSys '15*, 2015, pp. 230–235. DOI: 10.1145/2792838.2799496 (cit. on pp. 118, 119).

[Kim+11] S. Kim, S. Nowozin, P. Kohli, and C. D. Yoo, "Higher-order correlation clustering for image segmentation", in *Advances in Neural Information Processing Systems*, 2011, pp. 1530–1538 (cit. on pp. 4, 54).

[KB15] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization", in *3rd International Conference for Learning Representations*, 2015. arXiv: arXiv:1412.6980 (cit. on p. 107).

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing", *Science*, vol. 220, no. 4598, pp. 671–680, 1983. DOI: 10.1126/science.220.4598.671 (cit. on p. 62).

[Kiv+14] M. Kivela *et al.*, "Multilayer networks", en, *Journal of Complex Networks*, vol. 2, no. 3, pp. 203–271, 2014. DOI: 10.1093/comnet/cnu016 (cit. on pp. 7, 91).

[KvK09] S. Klamt and A. von Kamp, "Computing paths and cycles in biological interaction graphs", *BMC Bioinformatics*, vol. 10, no. 1, p. 181, 2009. DOI: 10.1186/1471-2105-10-181 (cit. on p. 46).

[Kni60] U. G. W. Knight, "Logical design of electrical networks", *Electrical Engineers, Journal of the Institution of*, vol. 6, no. 64, pp. 228–230, 1960. DOI: 10.1049/jiee-3.1960.0122 (cit. on p. 79).

[Kny17] A. V. Knyazev, "Signed laplacian for spectral clustering revisited", MERL - Mitsubishi Electric Research Laboratories, Tech. Rep. TR2017-001, 2017. arXiv: arXiv:1701.01394 (cit. on p. 69).

[KB09] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications", *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009. DOI: 10.1137/07070111X (cit. on p. 120).

[Kön36] D. König, *Theorie der endlichen und unendlichen Graphen*. 1936. DOI: 10.1007/978-1-4684-8971-2_2 (cit. on p. 50), trans. by R. McCoart as *Theory of finite and infinite graphs* (Birkhäuser Boston Inc., 1990).

[KMP11] I. Koutis, G. L. Miller, and R. Peng, "A nearly-$m \log n$ time solver for sdd linear systems", in *52nd Annual IEEE Symposium on Foundations of Computer Science*, 2011, pp. 590–598. DOI: 10.1109/FOCS.2011.85 (cit. on p. 81).

[KKZ09] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: a survey on subspace clustering, pattern-based clustering, and correlation clustering", *ACM Trans. Knowl. Discov. Data*, vol. 3, no. 1, 1:1–1:58, 2009. DOI: 10.1145/1497577.1497578 (cit. on p. 52).

[Krz+15] A. Krzywicki *et al.*, "Collaborative filtering for people-to-people recommendation in online dating: data analysis and user trial", *International Journal of Human-Computer Studies*, vol. 76, pp. 50–66, 2015. DOI: 10.1016/j.ijhcs.2014.12.003 (cit. on pp. 118, 119).

[KW92] J. Kuczyński and H. Woźniakowski, "Estimating the largest eigenvalue by the power and lanczos algorithms with a random start", *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 4, pp. 1094–1122, 1992. DOI: 10.1137/0613066 (cit. on p. 103).

[Kum16] S. Kumar, "Structure and dynamics of signed citation networks", in *Proceedings of the 25th World Wide Web conference*, 2016, pp. 63–64. DOI: 10.1145/2872518.2889391 (cit. on p. 26).

[Kum+16] S. Kumar, F. Spezzano, V. Subrahmanian, and C. Faloutsos, "Edge weight prediction in weighted signed networks", in *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 2016, pp. 221–230. DOI: 10.1109/ICDM.2016.0033 (cit. on p. 24).

[KLB09] J. Kunegis, A. Lommatzsch, and C. Bauckhage, "The slashdot zoo: mining a social network with negative edges", in *Proceedings of the 18th international conference on World wide web - WWW '09*, 2009, p. 741. DOI: 10.1145/1526709.1526809 (cit. on pp. 22, 84).

[Kun+10] J. Kunegis *et al.*, "Spectral analysis of signed graphs for clustering, prediction and visualization", in *Proceedings of the 2010 SIAM International Conference on Data Mining*. 2010, ch. 48, pp. 559–570. DOI: 10.1137/1.9781611972801.49 (cit. on pp. 6, 68).

[Le +17] G. Le Falher, N. Cesa-Bianchi, C. Gentile, and F. Vitale, "On the troll-trust model for edge sign prediction in social networks", in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, vol. 54, 2017, pp. 402–411 (cit. on pp. 10, 11, 117).

[LC14] R. Lebret and R. Collobert, "Word embeddings through hellinger pca", in *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, 2014, pp. 482–490 (cit. on p. 119).

[LHK10] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Predicting positive and negative links in online social networks", in *Proceedings of the 19th international conference on World wide web - WWW '10*, 2010, p. 641. DOI: 10.1145/1772690.1772756 (cit. on pp. 9, 23, 25, 30).

[LM12] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks", in *Advances in Neural Information Processing Systems 25*, 2012, pp. 539–547. arXiv: arXiv:1210.8182 (cit. on pp. 10, 113, 114).

[Lev02] R. L. Levien, "Attack resistant trust metrics", PhD thesis, University of California at Berkeley, 2002 (cit. on p. 25).

[LKA17] E. Levinkov, A. Kirillov, and B. Andres, "A comparative study of local search algorithms for correlation clustering", in *Proceedings of the 39th German Conference on Pattern Recognition*. 2017, pp. 103–114. DOI: 10.1007/978-3-319-66709-6_9 (cit. on p. 62).

[Lev+15] M. Levorato, L. Drummond, Y. Frota, and R. Figueiredo, "An ils algorithm to evaluate structural balance in signed social networks", in *Proceedings of the 30th Annual ACM Symposium on Applied Computing - SAC '15*, 2015, pp. 1117–1122. DOI: 10.1145/2695664.2695689 (cit. on p. 61).

[Lev+17] M. Levorato, R. Figueiredo, Y. Frota, and L. Drummond, "Evaluating balancing on social networks through the efficient solution of correlation clustering problems", *EURO Journal on Computational Optimization*, 2017. DOI: 10.1007/s13675-017-0082-6 (cit. on p. 61).

[LF17] M. Levorato and Y. Frota, "Brazilian congress structural balance analysis", *Journal of Interdisciplinary Methodologies and Issues in Sciences*, vol. 2, no. Graphs and social systems, 2017. DOI: 10.18713/JIMIS-280217-2-3 (cit. on pp. 6, 54).

[LG14] O. Levy and Y. Goldberg, "Neural word embedding as implicit matrix factorization", in *Advances in Neural Information Processing Systems 27*, 2014, pp. 2177–2185 (cit. on p. 119).

[Li+17a] J. Li *et al.*, "Attributed network embedding for learning in a dynamic environment", in *Proceedings of the 26th ACM International on Conference on Information and Knowledge Management*, 2017, pp. 387–396. DOI: 10.1145/3132847.3132919. arXiv: arXiv:1706.01860 (cit. on pp. 119, 120).

[Li+17b] S. Li, H. Zhang, D. Wu, C. Zhang, and D. Yuan, "Edge representation learning for community detection in large scale information networks", in *MATES workshop at VLDB 2017*, 2017 (cit. on p. 111).

[Li+11] W. Li *et al.*, "Integrative analysis of many weighted co-expression networks using tensor computation", *PLOS Computational Biology*, vol. 7, no. 6, pp. 1–13, 2011. DOI: 10.1371/journal.pcbi.1001106 (cit. on pp. 8, 91).

[LLL13] Y. Li, J. Liu, and C. Liu, "A comparative analysis of evolutionary and memetic algorithms for community detection from signed social networks", *Soft Computing*, vol. 18, no. 2, pp. 329–348, 2013. DOI: 10.1007/s00500-013-1060-4 (cit. on p. 69).

[LWZ16] H. Liang, K. Wang, and F. Zhu, "Mining social ties beyond homophily", in *IEEE 32nd International Conference on Data Engineering (ICDE)*, 2016, pp. 421–432. DOI: 10.1109/ICDE.2016.7498259 (cit. on p. 112).

[Lia+17] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed social network embedding", 2017. arXiv: 1705.04969 (cit. on pp. 119, 120).

[LPS14] A. Lingas, M. Persson, and D. Sledneu, "Iterative merging heuristics for correlation clustering", *International Journal of Metaheuristics*, vol. 3, no. 2, pp. 105–117, 2014. DOI: 10.1504/IJMHEUR.2014.063141 (cit. on p. 61).

[LW94] N. Littlestone and M. Warmuth, "The weighted majority algorithm", *Information and Computation*, vol. 108, no. 2, pp. 212–261, 1994. DOI: 10.1006/inco.1994.1009 (cit. on p. 34).

[Lit88] N. Littlestone, "Learning quickly when irrelevant attributes abound: a new linear-threshold algorithm", *Machine Learning*, vol. 2, no. 4, pp. 285–318, 1988. DOI: 10.1007/BF00116827 (cit. on p. 15).

[Maa+05] A. Ma'ayan *et al.*, "Formation of regulatory patterns during signal propagation in a mammalian cellular network", *Science*, vol. 309, no. 5737, pp. 1078–1083, 2005. DOI: 10.1126/science.1108876. eprint: http://science.sciencemag.org/content/309/5737/1078.full.pdf (cit. on p. 46).

[MDA15] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Autograd: effortless gradients in numpy", in *ICML 2015 AutoML Workshop*, 2015 (cit. on pp. 100, 107).

[MMP12] K. T. Macon, P. J. Mucha, and M. A. Porter, "Community structure in the united nations general assembly", *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 1, pp. 343–361, 2012. DOI: 10.1016/j.physa.2011.06.030 (cit. on pp. 6, 54).

[MMV15] K. Makarychev, Y. Makarychev, and A. Vijayaraghavan, "Correlation clustering with noisy partial information", in *Proceedings of The 28th Conference on Learning Theory*, 2015, pp. 1321–1342. arXiv: 1406.5667 (cit. on pp. 9, 65).

[MAC11] S. Maniu, T. Abdessalem, and B. Cautis, "Casting a web of trust over wikipedia: an interaction-based approach", in *Proceedings of the 20th International Conference Companion on World Wide Web*, 2011, pp. 87–88. DOI: 10.1145/1963192.1963237 (cit. on p. 25).

[MSL14] E. Manosevitch, N. Steinfeld, and A. Lev-On, "Promoting online deliberation quality: cognitive cues matter", *Information, Communication & Society*, vol. 17, no. 10, pp. 1177–1195, 2014. DOI: 10.1080/1369118X.2014.899610 (cit. on p. 12).

[Mar88] P. V. Marsden, "Homogeneity in confiding relations", *Social Networks*, vol. 10, no. 1, pp. 57–76, 1988. DOI: 10.1016/0378-8733(88)90010-X (cit. on p. 92).

[MBC16] V. Martínez, F. Berzal, and J.-C. Cubero, "A survey of link prediction in complex networks", *ACM Computer Survey*, vol. 49, no. 4, 69:1–69:33, 2016. DOI: 10.1145/3012704 (cit. on pp. 3, 33, 92).

[Mas+09] M. J. Mason, G. Fan, K. Plath, Q. Zhou, and S. Horvath, "Signed weighted gene co-expression network analysis of transcriptional regulation in murine embryonic stem cells.", *BMC genomics*, vol. 10, no. 1, p. 327, 2009. DOI: 10.1186/1471-2164-10-327 (cit. on p. 5).

[MSS10] C. Mathieu, O. Sankur, and W. Schudy, "Online correlation clustering", in *27th International Symposium on Theoretical Aspects of Computer Science - STACS 2010*, 2010, pp. 573–584 (cit. on p. 63).

[MS10] C. Mathieu and W. Schudy, "Correlation clustering with noisy input", *21st annual ACM-SIAM symposium on Discrete Algorithms*, pp. 712–728, 2010 (cit. on p. 64).

[MG07] J. Matoušek and B. Gärtner, *Understanding and Using Linear Programming*, 1st. 2007. DOI: 10.1007/978-3-540-30717-4 (cit. on p. 63).

[MS17] A. Mazumdar and B. Saha, *Query complexity of clustering with side information*, 2017. arXiv: arXiv:1706.07719 (cit. on p. 63), Short abstract: "Clustering with an oracle", in *54th Annual Allerton Conference on Communication, Control, and Computing*, Sep. 2016, pp. 738–739. DOI: 10.1109/ALLERTON.2016.7852305.

[ML14] J. Mcauley and J. Leskovec, "Discovering social circles in ego networks", *ACM Trans. Knowl. Discov. Data*, vol. 8, no. 1, 4:1–4:28, 2014. DOI: 10.1145/2556612 (cit. on p. 114).

[MW05] A. McCallum and B. Wellner, "Conditional models of identity uncertainty with application to noun coreference", in *Advances in Neural Information Processing Systems 17*, 2005, pp. 905–912 (cit. on pp. 5, 54).

[MSC01] M. McPherson, L. Smith-Lovin, and J. M. Cook, "Birds of a feather: homophily in social networks", *Annual Review of Sociology*, vol. 27, no. 1, pp. 415–444, 2001. DOI: 10.1146/annurev.soc.27.1.415 (cit. on pp. 92, 114).

[MS08] K. Mehlhorn and P. Sanders, "Hash tables and associative arrays", in *Algorithms and Data Structures: The Basic Toolbox*, 2008, ch. 4, pp. 81–98. DOI: 10.1007/978-3-540-77978-0 (cit. on p. 71).

[Men+15] I. Mendonca, R. Figueiredo, V. Labatut, and P. Michelon, "Relevance of negative links in graph partitioning: a case study using votes from the european parliament", in *2015 Second European Network Intelligence Conference*, 2015, pp. 122–129. DOI: 10.1109/ENIC.2015.25 (cit. on pp. 6, 54).

[MTH16] P. Mercado, F. Tudisco, and M. Hein, "Clustering signed networks with the geometric mean of laplacians", in *Advances in Neural Information Processing Systems 29*, 2016, pp. 4421–4429 (cit. on p. 69).

[Mik+13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality", in *Advances in Neural Information Processing Systems 26*, 2013, pp. 3111–3119 (cit. on p. 22).

[Mil+15] G. L. Miller, R. Peng, A. Vladu, and S. C. Xu, "Improved parallel algorithms for spanners and hopsets", in *27th ACM Symposium on Parallelism in Algorithms and Architectures*, 2015, pp. 192–201. DOI: 10.1145/2755573.2755574 (cit. on p. 81).

[Min89] M. Minoux, "Networks synthesis and optimum network design problems: models, solution methods and applications", *Networks*, vol. 19, no. 3, pp. 313–360, 1989. DOI: 10.1002/net.3230190305 (cit. on p. 82).

[MB11] A. Mishra and A. Bhattacharya, "Finding the bias and prestige of nodes in networks based on trust scores", in *Proceedings of the 20th International Conference on World Wide Web*, 2011, pp. 567–576. DOI: 10.1145/1963405.1963485 (cit. on pp. 23, 24).

[MS09] P. Mitra and M. Samal, "Approximation algorithm for correlation clustering", in *2009 1st International Conference on Networked Digital Technologies, NDT 2009*, 2009, pp. 140–145. DOI: 10.1109/NDT.2009.5272169 (cit. on p. 61).

[MT16] M. Mitzenmacher and C. E. Tsourakakis, "Predicting signed edges with o(nlog n) queries", 2016. arXiv: 1609.00750 (cit. on p. 63).

[MMB02] S. Mohammed, J. E. Mathieu, and A. L. 'Bart' Bartlett, "Technical-administrative task performance, leadership task performance, and contextual performance: considering the influence of team- and task-related composition variables", *Journal of Organizational Behavior*, vol. 23, no. 7, pp. 795–814, 2002. DOI: 10.1002/job.169 (cit. on p. 12).

[Muc+10] P. J. Mucha, T. Richardson, K. Macon, M. A. Porter, and J.-P. Onnela, "Community structure in time-dependent, multiscale, and multiplex networks", *Science*, vol. 328, no. 5980, pp. 876–878, 2010. DOI: 10.1126/science.1184819. eprint: http://science.sciencemag.org/content/328/5980/876.full.pdf (cit. on p. 113).

[MB16] S. F. Muldoon and D. S. Bassett, "Network and multilayer network approaches to understanding human brain dynamics", *Philosophy of Science*, vol. 83, no. 5, pp. 710–720, 2016. DOI: 10.1086/687857 (cit. on pp. 8, 91).

[Mul59] M. E. Muller, "A note on a method for generating points uniformly on n-dimensional spheres", *Commun. ACM*, vol. 2, no. 4, pp. 19–20, 1959. DOI: 10.1145/377939.377946 (cit. on p. 109).

[MF09] F. Murai and D. R. Figueiredo, "Assortative mixing in bittorrent-like networks", in *IEEE INFOCOM Workshops 2009*, 2009, pp. 1–2. DOI: 10.1109/INFCOMW.2009.5072115 (cit. on p. 93).

[New02] M. E. J. Newman, "Assortative mixing in networks", *Phys. Rev. Lett.*, vol. 89, p. 208 701, 20 2002. DOI: 10.1103/PhysRevLett.89.208701 (cit. on p. 93).

[NC16] M. E. J. Newman and A. Clauset, "Structure and inference in annotated networks", *Nature Communications*, vol. 7, p. 11 863, 2016. DOI: 10.1038/ncomms11863 (cit. on pp. 113, 114).

[Nic+16] M. Nickel, K. Murphy, V. Tresp, and E. Gabrilovich, "A review of relational machine learning for knowledge graphs", *Proceedings of the IEEE*, vol. 104, no. 1, pp. 11–33, 2016. DOI: 10.1109/JPROC.2015.2483592 (cit. on p. 110).

[NL15] V. Nicosia and V. Latora, "Measuring and modeling correlations in multiplex networks", *Phys. Rev. E*, vol. 92, p. 032 805, 3 2015. DOI: 10.1103/PhysRevE.92.032805 (cit. on p. 91).

[NJ09] S. Nowozin and S. Jegelka, "Solution stability in linear programming relaxations: graph partitioning and unsupervised learning", in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 769–776. DOI: 10.1145/1553374.1553473 (cit. on pp. 9, 65).

[OGP09] D. O. Olguín, P. A. Gloor, and A. S. Pentland, "Capturing individual and group behavior with wearable sensors", in *Proceedings of the AAAI Spring Symposium on Human Behavior Modeling*, 2009, pp. 68–74 (cit. on p. 12).

[Orl+10] J. B. Orlin, K. Madduri, K. Subramani, and M. Williamson, "A faster algorithm for the single source shortest path problem with few distinct positive lengths", *Journal of Discrete Algorithms*, vol. 8, no. 2, pp. 189–198, 2010, Selected papers from the 3rd Algorithms and Complexity in Durham Workshop ACiD 2007. DOI: http://dx.doi.org/10.1016/j.jda.2009.03.001 (cit. on p. 81).

[Pag+99] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing order to the web", Stanford InfoLab, Technical Report 1999-66, 1999 (cit. on p. 23).

[Pan+15] X. Pan *et al.*, "Parallel correlation clustering on big graphs", in *Advances in Neural Information Processing Systems 28*, 2015, pp. 82–90 (cit. on p. 59).

[Pap+14] A. Papaoikonomou, M. Kardara, K. Tserpes, and T. A. Varvarigou, "Predicting edge signs in social networks using frequent subgraph discovery", *IEEE Internet Computing*, vol. 18, no. 5, pp. 36–43, 2014. DOI: 10.1109/MIC.2014.82 (cit. on p. 23).

[PKV15] A. Papaoikonomou, M. Kardara, and T. Varvarigou, "Trust inference in online social networks", in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, 2015, pp. 600–604. DOI: 10.1145/2808797.2809418 (cit. on p. 24).

[PKK14] P. A. Papp, S. Kisfaludi-Bak, and Z. Király, "Low-stretch spanning trees", Bachelor thesis, Eötvös Loránd University, 2014 (cit. on p. 84).

[PB14] N. Parikh and S. Boyd, "Proximal algorithms", *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014. DOI: 10.1561/2400000003 (cit. on p. 101).

[PS89] D. Peleg and A. A. Schäffer, "Graph spanners", *Journal of Graph Theory*, vol. 13, no. 1, pp. 99–116, 1989. DOI: 10.1002/jgt.3190130114 (cit. on p. 81).

[PU89] D. Peleg and J. D. Ullman, "An optimal synchronizer for the hypercube", *SIAM Journal on Computing*, vol. 18, no. 4, pp. 740–747, 1989. DOI: 10.1137/0218050 (cit. on p. 81).

[PAS14] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: online learning of social representations", in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 701–710. DOI: 10.1145/2623330.2623732 (cit. on p. 120).

[Pet08] S. Pettie, "Distributed algorithms for ultrasparse spanners and linear size skeletons", in *27th ACM Symposium on Principles of Distributed Computing*, 2008, pp. 253–262. DOI: 10.1145/1400751.1400786 (cit. on p. 81).

[Pil+17] S. Pilosof, M. A. Porter, M. Pascual, and S. Kéfi, "The multilayer nature of ecological networks", *Nature Ecology & Evolution*, vol. 1, no. 4, p. 0101, 2017. DOI: 10.1038/s41559-017-0101 (cit. on p. 8).

[Piz+13] L. A. Pizzato *et al.*, "Recommending people to people: the nature of reciprocal recommenders with a case study in online dating", *User Modeling and User-Adapted Interaction*, vol. 23, no. 5, pp. 447–488, 2013. DOI: 10.1007/s11257-012-9125-0 (cit. on p. 118).

[Pra+13] B. A. Prakash, L. Adamic, T. Iwashyna, H. Tong, and C. Faloutsos, "Fractional immunization in networks", in *Proceedings of the 2013 SIAM International Conference on Data Mining*. 2013, pp. 659–667. DOI: 10.1137/1.9781611972832.73 (cit. on p. 3).

[PM15] G. J. Puleo and O. Milenkovic, "Correlation clustering with constrained cluster sizes and extended weights bounds", *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1857–1872, 2015. DOI: 10.1137/140994198. arXiv: 1411.0547 (cit. on p. 67).

[PM16] ——, "Correlation clustering and biclustering with locally bounded errors", in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 48, 2016, pp. 869–877 (cit. on p. 67).

[QBK06] Y. Qi, Z. Bar-Joseph, and J. Klein-Seetharaman, "Evaluation of different biological data and computational classification methods for use in protein interaction prediction", *Proteins: Structure, Function, and Bioinformatics*, vol. 63, no. 3, pp. 490–500, 2006. DOI: 10.1002/prot.20865 (cit. on p. 3).

[QA14] Y. Qian and S. Adali, "Foundations of trust and distrust in networks: extended structural balance theory", *ACM Trans. Web*, vol. 8, no. 3, 13:1–13:33, 2014. DOI: 10.1145/2628438 (cit. on p. 24).

[Qiu+18] J. Qiu *et al.*, "GNetwork Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec", in *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, 2018. arXiv: arXiv:1710.02971 (cit. on pp. 119, 120).

[Qua60] R. E. Quandt, "Models of transportation and optimal network construction", *Journal of Regional Science*, vol. 2, no. 1, pp. 27–45, 1960. DOI: 10.1111/j.1467-9787.1960.tb00833.x (cit. on p. 79).

[RH12] S. S. Rangapuram and M. Hein, "Constrained 1-spectral clustering", in *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, vol. 22, 2012, pp. 1143–1151 (cit. on p. 68).

[Rat+12] H. K. Rath, A. Chaturvedi, M. A. Rajan, and A. Simha, "Weighted signed graph (wsg) power-aware routing in distributed wireless networks", in *2012 IEEE Globecom Workshops*, 2012, pp. 475–480. DOI: 10.1109/GLOCOMW.2012.6477619 (cit. on p. 7).

[RRP13] N. Rebagliati, S. Rota Bulò, and M. Pelillo, "Correlation clustering with stochastic labellings", in *Similarity-Based Pattern Recognition: Second International Workshop, SIMBAD 2013, York, UK, July 3-5, 2013. Proceedings*. 2013, pp. 120–133. DOI: 10.1007/978-3-642-39140-8_8 (cit. on p. 67).

[RB06] J. Reichardt and S. Bornholdt, "Statistical mechanics of community detection", *Phys. Rev. E*, vol. 74, no. 1, p. 16 110, 2006. DOI: 10.1103/PhysRevE.74.016110 (cit. on p. 62).

[Ren+17] X. Ren *et al.*, "Cotype: joint extraction of typed entities and relations with knowledge bases", in *Proceedings of the 26th International Conference on World Wide Web - WWW '17*, 2017, pp. 1015–1024. DOI: 10.1145/3038912.3052708. arXiv: 1610.08763 (cit. on p. 111).

[Roc70] R. T. Rockafellar, *Convex Analysis*. 1970 (cit. on p. 101).

[RZ04] L. Roditty and U. Zwick, "On dynamic shortest paths problems", in *12th Annual European Symposium on Algorithms (ESA'04)*. 2004, pp. 580–591. DOI: 10.1007/978-3-540-30140-0_52 (cit. on p. 81).

[Rog03] E. . Rogers, *Diffusion of Innovations*, 5th edition. 2003, 576 pp. (cit. on p. 4).

[RB08] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure", *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1118–1123, 2008. DOI: 10.1073/pnas.0706851105. eprint: http://www.pnas.org/content/105/4/1118.full.pdf (cit. on p. 3).

[RTG17] P. Rozenshtein, N. Tatti, and A. Gionis, "Inferring the strength of social ties: a community-driven approach", in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1017–1025. DOI: 10.1145/3097983.3098199 (cit. on p. 91).

[Sal+15] M. Salehi *et al.*, "Spreading processes in multilayer networks", *IEEE Transactions on Network Science and Engineering*, vol. 2, no. 2, pp. 65–83, 2015. DOI: 10.1109/TNSE.2015.2425961 (cit. on p. 91).

[SC17] P. Sarkar and M. U. Chowdhury, "Secure iot using weighted signed graphs", in *Security and Privacy in Communication Networks: 12th International Conference, SecureComm 2016*. 2017, pp. 241–256. DOI: 10.1007/978-3-319-59608-2_13 (cit. on p. 7).

[Sco69] A. J. Scott, "The optimal network problem: some computational procedures", *Transportation Research*, vol. 3, no. 2, pp. 201–210, 1969. DOI: http://dx.doi.org/10.1016/0041-1647(69)90152-X (cit. on p. 82).

[Sed+17] J. Sedoc, J. Gallier, D. P. Foster, and L. H. Ungar, "Semantic word clusters using signed spectral clustering", in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, 2017, pp. 939–949. DOI: 10.18653/v1/P17-1087 (cit. on p. 69).

[SPU17] *Predicting Emotional Word Ratings using Distributional Representations and Signed Clustering*, 2017, pp. 564–571 (cit. on pp. 5, 54).

[SH10] P. Shachaf and N. Hara, "Beyond vandalism: wikipedia trolls", *Journal of Information Science*, vol. 36, no. 3, pp. 357–370, 2010. DOI: 10.1177/0165551510365390 (cit. on p. 12).

[SJ14] M. Shahriari and M. Jalili, "Ranking nodes in signed social networks", *Social Network Analysis and Mining*, vol. 4, no. 1, p. 172, 2014. DOI: 10.1007/s13278-014-0172-x (cit. on pp. 9, 24, 25, 30).

[ST11] C. R. Shalizi and A. C. Thomas, "Homophily and contagion are generically confounded in observational social network studies", *Sociological Methods & Research*, vol. 40, no. 2, pp. 211–239, 2011. DOI: 10.1177/0049124111404820. eprint: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3328971/ (cit. on p. 114).

[SST02] R. Shamir, R. Sharan, and D. Tsur, "Cluster graph modification problems", in *Graph-Theoretic Concepts in Computer Science*, vol. 2573, 2002, pp. 379–390. DOI: 10.1007/3-540-36379-3_33 (cit. on p. 55).

[SW14] Z. Shen and Q. Wang, "Entity resolution with weighted constraints", in *Advances in Databases and Information Systems: 18th East European Conference, ADBIS 2014, Ohrid, Macedonia, September 7-10*. 2014, pp. 308–322. DOI: 10.1007/978-3-319-10933-6_23 (cit. on pp. 6, 54).

[Shi+17] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, "A survey of heterogeneous information network analysis", *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2017. DOI: 10.1109/TKDE.2016.2598561 (cit. on p. 115).

[Sid+17] N. D. Sidiropoulos *et al.*, "Tensor decomposition for signal processing and machine learning", *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017. DOI: 10.1109/TSP.2017.2690524 (cit. on p. 120).

[SMZ12] A. Silva, W. Meira, and M. J. Zaki, "Mining attribute-structure correlated patterns in large attributed graphs", *Proceedings of the VLDB Endowment*, vol. 5, no. 5, pp. 466–477, 2012. DOI: 10.14778/2140436.2140443 (cit. on p. 112).

[SA17] R. Singh and B. Adhikari, "Measuring the balance of signed networks and its application to sign prediction", *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2017, no. 6, 2017. DOI: 10.1088/1742-5468/aa73ef (cit. on p. 51).

[Sit+12] Z. Siting, H. Wenxing, Z. Ning, and Y. Fan, "Job recommender systems: a survey", *2012 7th International Conference on Computer Science & Education (ICCSE)*, no. Iccse, pp. 920–924, 2012. DOI: 10.1109/ICCSE.2012.6295216 (cit. on p. 118).

[SSF13] R. Slonje, P. K. Smith, and A. Frisén, "The nature of cyberbullying, and strategies for prevention", *Computers in Human Behavior*, vol. 29, no. 1, pp. 26–32, 2013. DOI: 10.1016/j.chb.2012.05.024 (cit. on p. 12).

[SCC15] F. Solera, S. Calderara, and R. Cucchiara, "Learning to divide and conquer for online multi-target tracking", in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4373–4381. DOI: 10.1109/ICCV.2015.497 (cit. on pp. 5, 54).

[SM15] D. Song and D. A. Meyer, "Link sign prediction and ranking in signed directed social networks", *Social Network Analysis and Mining*, vol. 5, no. 1, p. 52, 2015. DOI: 10.1007/s13278-015-0288-7 (cit. on pp. 9, 23, 30).

[Son+17] Q. Song, H. Ge, J. Caverlee, and X. Hu, *Tensor completion algorithms in big data analytics*, 2017. arXiv: arXiv:1711.10105 (cit. on p. 120).

[SM03] V. Spirin and L. A. Mirny, "Protein complexes and functional modules in molecular networks", *Proceedings of the National Academy of Sciences*, vol. 100, no. 21, pp. 12 123–12 128, 2003. DOI: 10.1073/pnas.2032324100. eprint: http://www.pnas.org/content/100/21/12123.full.pdf (cit. on p. 3).

[Squ+13] T. Squartini, F. Picciolo, F. Ruzzenenti, and D. Garlaschelli, "Reciprocity of weighted networks", *Scientific Reports*, vol. 3, no. 2729, 2013. DOI: 10.1038/srep02729 (cit. on p. 26).

[SC12] E. Stattner and M. Collard, "Social-based conceptual links: conceptual analysis applied to social networks", in *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2012, pp. 25–29. DOI: 10.1109/ASONAM.2012.15 (cit. on p. 112).

[SWG12] I. Steinmacher, I. S. Wiese, and M. A. Gerosa, "Recommending mentors to software project newcomers", in *Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering*, 2012, pp. 63–67 (cit. on p. 118).

[ST14] 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. DOI: 10.2200/S00590ED1V01Y201408AIM029 (cit. on p. 3).

[Sul04] J. Suler, "The online disinhibition effect", *Cyberpsychology & Behavior*, vol. 7, no. 3, pp. 321–326, 2004. DOI: 10.1089/1094931041291295. eprint: https://www.ncbi.nlm.nih.gov/pubmed/15257832 (cit. on p. 12).

[Sun+11] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: meta path-based top-k similarity search in heterogeneous information networks", *PVLDB*, vol. 4, no. 11, pp. 992–1003, 2011 (cit. on p. 115).

[Swa04] C. Swamy, "Correlation clustering: maximizing agreements via semidefinite programming", in *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2004, pp. 526–527 (cit. on pp. 58, 59, 61).

[ST10] M. Szell and S. Thurner, "Measuring social dynamics in a massive multiplayer online game", *Social Networks*, vol. 32, no. 4, pp. 313–329, 2010. DOI: 10.1016/j.socnet.2010.06.001. arXiv: arXiv:0911.1084v1 (cit. on pp. 7, 91).

[TC09] P. P. Talukdar and K. Crammer, "New regularized algorithms for transductive learning", in *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*. 2009, pp. 442–457. DOI: 10.1007/978-3-642-04174-7_29 (cit. on p. 3).

[Tan+15] J. Tang *et al.*, "Line: large-scale information network embedding", in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 1067–1077. DOI: 10.1145/2736277.2741093 (cit. on pp. 3, 120).

[TL15] J. Tang and J. Li, "Semantic mining of social networks", *Synthesis Lectures on the Semantic Web: Theory and Technology*, vol. 5, no. 2, pp. 1–205, 2015. DOI: 10.2200/S00629ED1V01Y201502WBE011 (cit. on p. 110).

[Tan+16a] J. Tang, T. Lou, J. Kleinberg, and S. Wu, "Transfer learning to infer social ties across heterogeneous networks", *ACM Trans. Inf. Syst.*, vol. 34, no. 2, 7:1–7:43, 2016, conference version in WSDM'12. DOI: 10.1145/2746230 (cit. on p. 110).

[Tan+16b] J. Tang, Y. Chang, C. Aggarwal, and H. Liu, "A survey of signed network mining in social media", *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, p. 36, 2016. DOI: 10.1145/2956185. arXiv: 1511.07569 (cit. on pp. 4, 25).

[Tan+13] J. Tang, H. Gao, X. Hu, and H. Liu, "Exploiting homophily effect for trust prediction", in *Proceedings of the sixth ACM international conference on Web search and data mining - WSDM '13*, 2013, p. 53. DOI: 10.1145/2433396.2433405 (cit. on p. 24).

[TZT11] W. Tang, H. Zhuang, and J. Tang, "Learning to infer social ties in large networks", in *Machine Learning and Knowledge Discovery in Databases: European Conference, (ECML PKDD)*. 2011, pp. 381–397. DOI: 10.1007/978-3-642-23808-6_25 (cit. on pp. 10, 110).

[Tas+04] B. Taskar, M.-f. Wong, P. Abbeel, and D. Koller, "Link prediction in relational data", in *Advances in Neural Information Processing Systems 16*, 2004, pp. 659–666 (cit. on p. 112).

[The09a] M. Thelwall, "Homophily in myspace", *Journal of the American Society for Information Science and Technology*, vol. 60, no. 2, pp. 219–231, 2009. DOI: 10.1002/asi.20978 (cit. on p. 4).

[The09b] ——, "Homophily in myspace", *Journal of the American Society for Information Science and Technology*, vol. 60, no. 2, pp. 219–231, 2009. DOI: 10.1002/asi.20978 (cit. on p. 92).

[Tom16] M. Tomassen, *Exploring the black box of machine learning in human resource management: an hr perspective on the consequences for hr professionals*, Master Thesis, 2016 (cit. on p. 12).

[TB09] V. A. Traag and J. Bruggeman, "Community detection in networks with positive and negative links", *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, vol. 80, 2009. DOI: 10.1103/PhysRevE.80.036115 (cit. on pp. 6, 54, 62).

[TNV10] V. A. Traag, Y. E. Nesterov, and P. Van Dooren, "Exponential ranking: taking into account negative links", in *Social Informatics: Second International Conference*. 2010, pp. 192–202. DOI: 10.1007/978-3-642-16567-2_14 (cit. on pp. 9, 23).

[Tys+16] G. Tyson, V. C. Perta, H. Haddadi, and M. C. Seto, "A first look at user activity on tinder", in *International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2016, pp. 461–466. DOI: 10.1109/ASONAM.2016.7752275 (cit. on pp. 4, 118).

[vZW08] A. van Zuylen and D. P. Williamson, "Deterministic algorithms for rank aggregation and other ranking and clustering problems", in *Approximation and Online Algorithms: 5th International Workshop, WAOA 2007, Eilat, Israel*. 2008, pp. 260–273. DOI: 10.1007/978-3-540-77918-6_21 (cit. on p. 58).

[VWG17] N. Veldt, A. Wirth, and D. F. Gleich, "Correlation clustering with low-rank matrices", in *Proceedings of the 26th International Conference on World Wide Web - WWW '17*, 2017, pp. 1025–1034. DOI: 10.1145/3038912.3052586. arXiv: 1611.07305 (cit. on p. 56).

[Vid11] R. Vidal, "Subspace clustering", *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011. DOI: 10.1109/MSP.2010.939739 (cit. on p. 111).

[VEB09] N. X. Vinh, J. Epps, and J. Bailey, "Information theoretic measures for clusterings comparison: is a correction for chance necessary?", in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 1073–1080. DOI: 10.1145/1553374.1553511 (cit. on p. 108).

[Vit14] F. Vitale, private communication, 2014 (cit. on pp. 10, 70, 85, 117).

[Vit+11] F. Vitale, N. Cesa-Bianchi, C. Gentile, and G. Zappella, "See the tree through the lines: the shazoo algorithm", in *Advances in Neural Information Processing Systems 24*, 2011, pp. 1584–1592. arXiv: 1301.5160 (cit. on p. 13).

[vLux07] U. von Luxburg, "A tutorial on spectral clustering", *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007. DOI: 10.1007/s11222-007-9033-z (cit. on p. 68).

[Wan+10] C. Wang *et al.*, "Mining advisor-advisee relationships from research publication networks", in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2010, pp. 203–212. DOI: 10.1145/1835804.1835833 (cit. on pp. 7, 91).

[Wan+18] H. Wang *et al.*, "Shine: signed heterogeneous information network embedding for sentiment link prediction", in *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, 2018. DOI: 10.1145/3159652.3159666. arXiv: arXiv:1712.00732 (cit. on p. 119).

[Wan+17a] J. Wang, J. Shen, P. Li, and H. Xu, "Online matrix completion for signed link prediction", in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17*, 2017, pp. 475–484. DOI: 10.1145/3018661.3018681 (cit. on pp. 9, 22, 25, 30, 101).

[WL13] N. Wang and J. Li, "Restoring: a greedy heuristic approach based on neighborhood for correlation clustering", in *Advanced Data Mining and Applications: 9th International Conference, ADMA 2013, Hangzhou, China, December 14-16*. 2013, pp. 348–359. DOI: 10.1007/978-3-642-53914-5_30 (cit. on p. 61).

[Wan+17b] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: a survey of approaches and applications", *IEEE Transactions on Knowledge and Data Engineering*, vol. PP, no. 99, pp. 1–1, 2017. DOI: 10.1109/TKDE.2017.2754499 (cit. on pp. 111, 115).

[Wan+17c] S. Wang, C. Aggarwal, J. Tang, and H. Liu, "Attributed signed network embedding", in *Proceedings of the 26th ACM International on Conference on Information and Knowledge Management*, 2017. DOI: 10.1145/3132847.3132905 (cit. on pp. 24, 119).

[Wan+17d] S. Wang, J. Tang, C. Aggarwal, Y. Chang, and H. Liu, "Signed network embedding in social media", in *Proceedings of the 2017 SIAM International Conference on Data Mining*. 2017, pp. 327–335. DOI: 10.1137/1.9781611974973.37 (cit. on pp. 22, 24).

[Wan+13] Y. Wang, L. Xu, Y. Chen, and H. Wang, "A scalable approach for general correlation clustering", in *Advanced Data Mining and Applications: 9th International Conference, ADMA 2013, Hangzhou, China, December 14-16*. 2013, pp. 13–24. DOI: 10.1007/978-3-642-53917-6_2 (cit. on p. 62).

[WF16] H. Weng and Y. Feng, "Community detection with nodal information", 2016. arXiv: 1610.09735 (cit. on pp. 113, 114).

[WTM13] D. Wijaya, P. P. Talukdar, and T. Mitchell, "Pidgin: ontology alignment using web text as interlingua", in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, 2013, pp. 589–598. DOI: 10.1145/2505515.2505559 (cit. on p. 3).

[Wil17] William L. Hamilton and Rex Ying and Jure Leskovec, "Representation learning on graphs: methods and applications", *Bulletin of the Technical Committee on Data Engineering*, vol. 40, no. 4, pp. 52–74, 2017. arXiv: arXiv:1709.05584 (cit. on pp. 3, 22).

[Wir17] A. Wirth, "Correlation clustering", in *Encyclopedia of Machine Learning and Data Mining*. 2017, pp. 280–284. DOI: 10.1007/978-1-4899-7687-1_176 (cit. on p. 52).

[Won80] R. T. Wong, "Worst-case analysis of network design problem heuristics", *SIAM Journal on Algebraic Discrete Methods*, vol. 1, no. 1, pp. 51–63, 1980. DOI: 10.1137/0601008 (cit. on p. 82).

[Wu82] F. Y. Wu, "The potts model", *Rev. Mod. Phys.*, vol. 54, no. 1, pp. 235–268, 1982. DOI: 10.1103/RevModPhys.54.235 (cit. on p. 62).

[WAS16] Z. Wu, C. C. Aggarwal, and J. Sun, "The troll-trust model for ranking in signed networks", in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, 2016, pp. 447–456. DOI: 10.1145/2835776.2835816 (cit. on pp. 9, 24, 25, 30).

[Xia+15] P. Xia, B. Liu, Y. Sun, and C. Chen, "Reciprocal recommendation system for online dating", in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015 - ASONAM '15*, 2015, pp. 234–241. DOI: 10.1145/2808797.2809282. arXiv: 1501.06247 (cit. on p. 119).

[Xia+16] P. Xia, S. Zhai, B. Liu, Y. Sun, and C. Chen, "Design of reciprocal recommendation systems for online dating", *Social Network Analysis and Mining*, vol. 6, no. 1, p. 32, 2016. DOI: 10.1007/s13278-016-0340-2 (cit. on p. 118).

[Xu+14a] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "Gbagc", *ACM Transactions on Knowledge Discovery from Data*, vol. 9, no. 1, pp. 1–43, 2014. DOI: 10.1145/2629616 (cit. on pp. 10, 113, 114).

[Xu+14b] ——, "Gbagc: a general bayesian framework for attributed graph clustering", *ACM Trans. Knowl. Discov. Data*, vol. 9, no. 1, 5:1–5:43, 2014. DOI: 10.1145/2629616 (cit. on p. 104).

[YCL07] B. Yang, W. Cheung, and J. Liu, "Community mining from signed social networks", *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 10, pp. 1333–1348, 2007. DOI: 10.1109/TKDE.2007.1061 (cit. on p. 69).

[YML13] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes", in *2013 IEEE 13th International Conference on Data Mining*, 2013, pp. 1151–1156. DOI: 10.1109/ICDM.2013.167 (cit. on pp. 10, 104, 113, 114).

[Yan+12] S.-H. Yang, A. J. Smola, B. Long, H. Zha, and Y. Chang, "Friend or frenemy?", in *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '12*, 2012, p. 555. DOI: 10.1145/2348283.2348359 (cit. on p. 12).

[YH15] J. Yap and N. Harrigan, "Why does everybody hate me? balance, status, and homophily: the triumvirate of signed tie formation", *Social Networks*, vol. 40, pp. 103–122, 2015. DOI: 10.1016/j.socnet.2014.08.002 (cit. on p. 48).

[Ye+13] J. Ye, H. Cheng, Z. Zhu, and M. Chen, "Predicting positive and negative links in signed social networks by transfer learning", in *Proceedings of the 22Nd International Conference on World Wide Web*, 2013, pp. 1477–1488 (cit. on p. 24).

[Yoo+17] J. Yoon, A. M. Alaa, M. Cadeiras, and M. van der Schaar, "Personalized donor-recipient matching for organ transplantation", in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 2017, pp. 1647–1654 (cit. on p. 118).

[YWX17] S. Yuan, X. Wu, and Y. Xiang, "Sne: signed network embedding", in *Proceedings of Advances in Knowledge Discovery and Data Mining: 21st Pacific-Asia Conference*. 2017, pp. 183–195. DOI: 10.1007/978-3-319-57529-2_15. arXiv: arXiv:1703.04837 (cit. on pp. 9, 22).

[Yua+17] W. Yuan *et al.*, "Negative sign prediction for signed social networks", *Future Generation Computer Systems*, 2017. DOI: 10.1016/j.future.2017.08.037 (cit. on pp. 9, 23).

[Zas12] T. Zaslavsky, "A mathematical bibliography of signed and gain graphs and allied areas", en, *The Electronic Journal of Combinatorics*, vol. 1000, 2012 (cit. on p. 50).

[ZW13] X. Zeng and L. Wei, "Social ties and user content generation: evidence from flickr", *Information Systems Research*, vol. 24, no. 1, pp. 71–87, 2013. DOI: 10.1287/isre.1120.0464 (cit. on p. 114).

[ZYH14] C. Zhang, J. Yarkony, and F. A. Hamprecht, "Cell detection and segmentation using correlation clustering", in *Proceedings of the 17th International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2014, pp. 9–16. DOI: 10.1007/978-3-319-10404-1_2 (cit. on pp. 4, 54).

[Zha+16] M. Zhang, J. Ma, Z. Liu, J. Sun, and T. Silva, "A research analytics framework-supported recommendation approach for supervisor selection", *British Journal of Educational Technology*, vol. 47, no. 2, pp. 403–420, 2016. DOI: 10.1111/bjet.12244 (cit. on p. 118).

[ZLZ16] Y. Zhang, E. Levina, and J. Zhu, "Community detection in networks with node features", *Electron. J. Statist.*, vol. 10, no. 2, pp. 3153–3178, 2016. DOI: 10.1214/16-EJS1206 (cit. on pp. 10, 112).

[ZDB17] H. Zhao, L. Du, and W. Buntine, "Leveraging node attributes for incomplete relational data", in *ICML 2017*, 2017. arXiv: 1706.04289 (cit. on pp. 113, 114).

[ZS15] Q. Zheng and D. Skillicorn, "Spectral embedding of signed networks", in *Proceedings of the 2015 SIAM International Conference on Data Mining*. 2015, ch. 7, pp. 55–63. DOI: 10.1137/1.9781611974010.7 (cit. on pp. 22, 68).

[ZZW15] X. Zheng, D. Zeng, and F.-Y. Wang, "Social balance in signed networks", *Information Systems Frontiers*, vol. 17, no. 5, pp. 1077–1095, 2015. DOI: 10.1007/s10796-014-9483-8 (cit. on p. 48).

[ZCY09] Y. Zhou, H. Cheng, and J. X. Yu, "Graph clustering based on structural/attribute similarities", *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 718–729, 2009. DOI: 10.14778/1687627.1687709 (cit. on p. 112).

[ZCY10] ——, "Clustering large attributed graphs: an efficient incremental approach", in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2010, pp. 689–698. DOI: 10.1109/ICDM.2010.41 (cit. on p. 112).

[ZGL03] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions", *International Conference on Machine Learning - ICML 2003*, vol. 20, no. 2, p. 912, 2003 (cit. on pp. 13, 20, 21).

[Zhu+12] H. Zhuang *et al.*, "Actively learning to infer social ties", *Data Mining and Knowledge Discovery*, vol. 25, no. 2, pp. 270–297, 2012. DOI: 10.1007/s10618-012-0274-x (cit. on p. 110).

[ZA10] K. Zolfaghar and A. Aghaie, "Mining trust and distrust relationships in social web applications", in *Proceedings of the 2010 IEEE 6th International Conference on Intelligent Computer Communication and Processing*, 2010, pp. 73–80. DOI: 10.1109/ICCP.2010.5606460 (cit. on p. 23).

[Zuc07] D. Zuckerman, "Linear degree extractors and the inapproximability of max clique and chromatic number", *Theory of Computing*, vol. 3, no. 6, pp. 103–128, 2007. DOI: 10.4086/toc.2007.v003a006 (cit. on p. 61).