POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master's Degree Thesis

Detecting Echo Chambers in social media; a graph-based approach

Supervisors

Prof. Aristides GIONIS

Dr. Stefan NEUMANN

Prof. Aris ANAGNOSTOPOULOS

Prof. Giacomo COMO

Candidate

Francesco ZAPPIA

July 2021

Abstract

Social media are becoming more and more popular and are used to discuss a wide range of topics. On these platforms we are often experiencing polarization between the users, producing a clear separation between groups with different opinions. Echo Chambers are closely related to this phenomenon: an Echo Chamber is a group of users with the same beliefs that reinforce their ideas.

The growing complexity and quantity of online interactions requires us to find new techniques for detecting Echo Chambers. In this work we propose the Echo Chamber Problem (ECP) and the Densest Echo Chamber Problem (D-ECP), new formulations that take into account the concepts of *content* (the piece of information that is discussed) and *thread* (the "locality" discussing the content) in finding polarization.

Our idea is that Echo Chambers correspond to groups of users discussing a content which is *controversial*, i.e. globally triggers many hostile interactions, with no *controversy*, i.e. with mainly friendly interactions inside the Echo Chamber.

We will show that the problems we propose are hard to approximate within any non-trivial factor and propose Mixed Integer Programming (MIP) models and heuristics for solving them. Finally, we will focus on one of these methods and show that it is able to find Echo Chambers in synthetic data but has some limitations when applied to real-world data.

Keywords

Polarization, Echo Chambers, Social Networks, Signed Graphs, Contents

Acknowledgements

I would like to express my deep and sincere gratitude to the people that helped and guided me through these months of research: my supervisor, Dr. Stefan Neumann, my examiner, Professor Aristides Gionis, and Professor Aris Anagnostopoulos, who joined us in this adventure. Their friendship and support during this period have been invaluable.

I am also grateful to my family and friends for supporting me through all these years.

The computations were enabled by resources provided by the Swedish National Infrastructure for Computing (SNIC) at the High Performance Computing Center North (HPC2N) partially funded by the Swedish Research Council through grant agreement no. 2018-05973.

This work has been developed as part of the SoBigData++ and REBOUND projects.

Table of Contents

List of Tables

List of Figures

Acronyms

LP
: Linear Programming

MIP
: Mixed Integer Programming

MILP
: Mixed Integer Linear Programming

DCS
: Densest Common Subgraph

Bff
: Best Friends Forever

O²Bff
: On-Off Bff

INC$_O$
: Incremental overlap

ECP
: Echo Chamber Problem

D-ECP
: Densest Echo Chamber Problem

PA
: Pair Aggregated

TPA
Thread Pair Aggregated

Chapter 1

Introduction

Social networks are nowadays widely used by people, allowing users to discuss the most different topics and interact with each other. At the same time in these platforms we are observing an increasing polarization between the users. This inspired several studies which have been conducted about the topic [1, 2, 3, 4], the most recent ones focusing on COVID-19 [5, 6, 7, 8] and vaccination [9].

Polarization is the social phenomenon according to which people tend do separate in opposing communities with few people remaining neutral [2]. A close phenomenon is that of the Echo Chambers, groups in which people that have the same opinions enforce their respective ideas [1], a concept very similar to the definition of polarization as given in [10]: "group polarization arises when members of a deliberating group move toward a more extreme point in whatever direction is indicated by the members' predeliberation tendency. '[L]ike polarized molecules, group members become even more aligned in the direction they were already tending"[11].

In this research we aim at finding a method for detecting Echo Chambers, by analyzing data retrieved from social medias like Twitter and Reddit: we define the Echo Chamber Problem (ECP) and the Densest Echo Chamber Problem (D-ECP) and propose techniques for solving and approximating them.

We introduce these new approaches for finding Echo Chambers due to growing complexity and quantity of online interactions; the last year of social distancing forced many people to stay at home and consequently we can expect that the time they spent on social medias increased from the past, thus producing a denser network of interactions.

Our two problems are defined on a signed graph which distinguishes between *friendly* and *hostile* interactions between the users. Differently from previous studies of polarization on signed graph [12], we define and incorporate in our problems the ideas of *contents* (the piece of information which is discussed) and *threads* (the "locality" discussing a content).

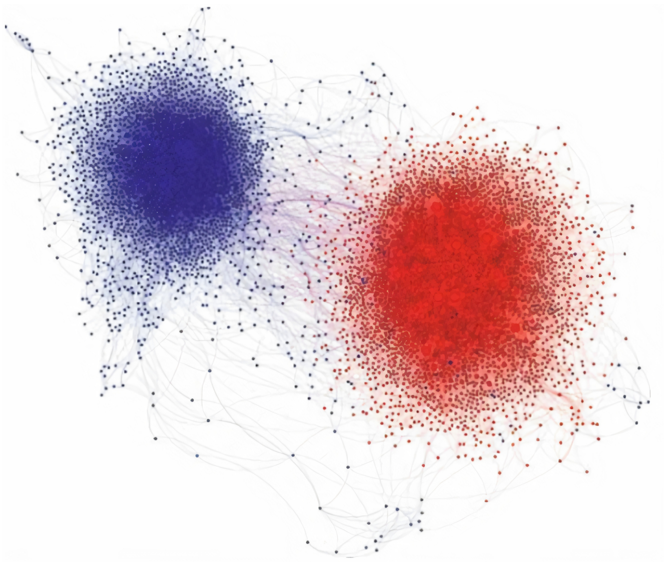


Figure 1.1: The retweet network of posts regarding US during 2010 midterm elections. Red and blue nodes are associated with conservative and progressive users, respectively. The picture was taken from [14].

Our idea is that Echo Chambers correspond to subgraphs discussing one or more controversial topics (which trigger many negative reactions in the network when seen as a whole) with few or no negative interactions: users in this bubble agree with each other, thus reinforcing their initial positions.

1.1 Background

A *graph* $G = (V, E)$ is a collection of *vertices* or *nodes* V and *edges* or *links* $E \subseteq V \times V$ between the nodes, representing relationships between entities. Graphs are very useful in representing many interesting concepts from social sciences, biology, physics, chemistry and geography (for example, we can see in Figure 1.1 that they can be used to represent the users' retweets during US 2010 midterm elections)[13, 14].

Different Types of Graphs In its simplest form graph are *undirected* and *unweighted*. In an *undirected* graph relationships are bi-directional, while in directed graph the order of nodes in a link reflects the direction (i.e. an edge e_{ij} is different from an edge e_{ji}). A weighted graph associates a weight ω_e to each edge $e \in E$ [14, 15].

Sometimes edges are allowed to be either positive or negative: these networks are

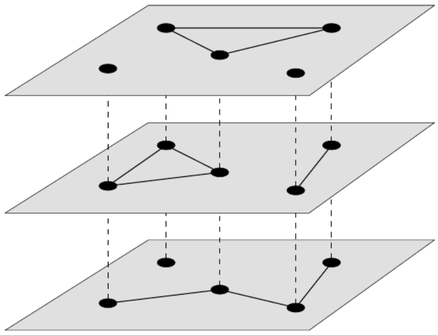


Figure 1.2: An example of *multiplex graph* with three layers. Picture taken from [13].

usually called *signed graphs*. An acquaintance network, for example, can be modeled through a signed graph, with negative and positive edges denoting animosity and friendship, respectively [13].

In the rest of the document we will abuse notation and refer to vertices both as $v_i \in V$ and $i \in V$; similarly we will refer to edges both as $e_{ij} \in E$ and as $ij \in E$.

Representing relationships between entities which span over more than two dimensions requires the definition of an ever more complex structure, the *multiplex graph*. A *multiplex graph* is a set of graphs (also referred to as *layers*) $G = \{G_i = (V, E_i)\}_i$ over the same set of vertices V, each G_i having its own set of edges $E_i \subseteq V \times V$. An example can be seen in Figure 1.2. Multiplex graphs can be used, for example, to model temporal networks, where each *layer* corresponds to a snapshot of the relationship at a certain point in time [13].

1.2 Problem

In this section, after defining the graph on which the research is carried out, we give a formal definition of the problems that we study in the thesis.

1.2.1 The Interaction Graph

The *interaction graph* G is the graph we utilize to encode the information regarding the interactions between the users.

Definition 1.2.1. *A multiplex graph $G = \{G_k = (V, E_k)\}_k$ is called an interaction graph if each G_k is a directed, weighted and signed multigraph with weights in $[-1, +1]$. We will often refer to the layers G_k as threads and therefore also denote them as T_k, i.e. we set $T_k = G_k$.*

In this graph each user is associated to a vertex $v \in V$. For this reason we will sometimes refer to vertices as users in the rest of the document.

Each edge of the graph corresponds to an interaction: edge e_{ij} going from v_i to v_j represents user i replying to user j. Also, the corresponding weight w_e encodes the *sentiment* of the interaction: negative and positive values are associated with hostile and friendly interactions, respectively, with smaller values of w_e being associated to more hostile interactions.

Let a content C be any kind of resource that triggers a discussion in one or more threads T, where a thread can be any social media post sharing the content C. The set of threads associated to C is denoted as $\mathcal{T}_C$. A content is usually represented by a newspaper article and it is identified by its URL, e.g.

`https://www.nytimes.com/2021/03/04/us/richard-barnett-pelosi-tantrum.html`

A corresponding thread then may be, for example, the one generated by a user posting and commenting the same URL on its Twitter account (see Figure 1.3), thus generating a discussion.

In our *interaction graph* each layer is associated to a thread T whose edges are the interactions happening in it. Note that since it is a multigraph, each of the layers can contain more than one edge between two users, as each pair of users can reply to each other more than one time.

We will also use $\mathcal{C}$ for denoting the set of contents.

An example of an *interaction graph* can be seen in Figure 1.4.

1.2.2 The Problem Definition

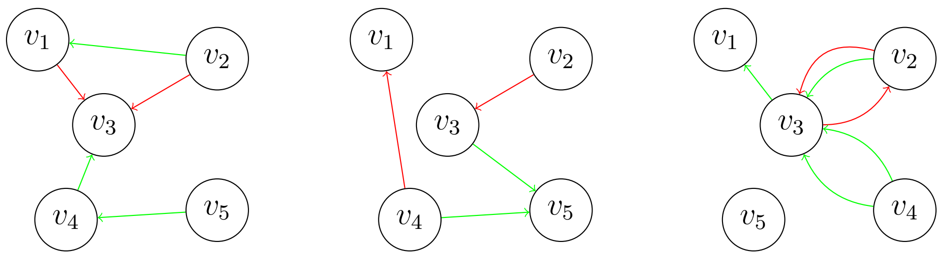
The main goal of the research is finding echo chambers in social medias, more specifically on the *interaction graph* as defined in Subsection 1.2.1.

Our definition is based on the idea that echo chambers can be identified by looking at contents which are highly debated (we will call this type of content *controversial*) but which are discussed with little or no animosity in some subgraphs. These subgraphs are the *Echo Chambers*.

Given an *interaction graph* $G = \{G_k = (V, E_k)\}_k$ on some contents $\mathcal{C}$ and threads, let E_k^+ and E_k^- be the set of positive and negative edges associated to thread T_k, respectively. We define $\eta(T_k)$ to be the ratio between the number of negative edges and the total number of edges in the layer associated to thread T_k, i.e.

$$\eta(T_k) = \frac{|E_k^-|}{|E_k^-| + |E_k^+|}.$$

Figure 1.3: A thread associated to the mentioned New York Times article.



(a) $T_1 \in \mathcal{T}_{C_1}$ **(b)** $T_2 \in \mathcal{T}_{C_1}$ **(c)** $T_3 \in \mathcal{T}_{C_2}$

Figure 1.4: An example of an *interaction graph*, green and red edges representing positive and negative interactions with weights -1 and $+1$, respectively. It contains three threads (T_1, T_2 and T_3) and two contents (C_1 and C_2), the first two layers each being associated to a thread of content C_1, the last layer to a thread of content C_2.

Now, for a content C, let $E_l^+ = \bigcup_{T_k \in \mathcal{T}_C} E_k^+$ and $E_l^+ = \bigcup_{T_k \in \mathcal{T}_C} E_k^+$. Similarly to $\eta(T_k)$, $\eta(C)$ is defined as the fraction of negative edges associated to content C, i.e.

$$\eta(C_l) = \frac{|E_l^-|}{|E_l^-| + |E_l^+|}.$$

Definition 1.2.2 (Controversial thread)**.** *Let $\alpha \in [0,1]$. A thread (or content) is* controversial *if $\eta(T) > \alpha$ (or, similarly, $\eta(C) > \alpha$). Conversely, a thread (or content) is* non-controversial *if $\eta(T) \leq \alpha$ ($\eta(C) \leq \alpha$).*

Intuitively, *controversial* threads contain many negative interactions. We denote as $\hat{\mathcal{C}} \subseteq \mathcal{C}$ the set of *controversial* contents.

Echo Chambers correspond to *non-controversial* subgraphs (i.e. with few negative edges) discussing a *controversial* content.

More formally, for a set of vertices $U \subseteq V$, let $T[U]$ be the subgraph induced in the layer associated to thread T; let $|T^+[U]|$ and $|T^-[U]|$ be its number of positive and negative edges, respectively.

We define $\mathcal{S}_C(U)$ as the set of *non-controversial* threads induced by U, for *controversial* contents $C \in \hat{\mathcal{C}}$, i.e.

$$\mathcal{S}_C(U) = \{T[U] \ s.t. \ T[U] \ non\text{-}controversial, T \in \mathcal{T}_C, U \subseteq V\}. \tag{1.1}$$

Thus, $\mathcal{S}_C(U)$ will contain threads which are *locally* non-controversial but it is defined only for contents that are *globally* controversial. In the rest of the document we will refer to elements of $\mathcal{S}_C(U)$ both as $T \in \mathcal{S}_C(U)$ and $T[U] \in \mathcal{S}_C(U)$.

We now define the Echo Chamber Score of a set of vertices U.

Definition 1.2.3 (Echo Chamber Score)**.** *Let $U \subseteq V$ be a subset of vertices. Its* Echo Chamber Score $\xi(U)$ *is*

$$\xi(U) = \sum_{\mathcal{C} \in \hat{\mathcal{C}}} \sum_{T[U] \in \mathcal{S}_C(U)} (|T^+[U]| - |T^-[U]|). \tag{1.2}$$

We can now define the Echo Chamber Problem (ECP).

Problem 1.2.1 (Echo Chamber Problem (ECP))**.** *Given an* interaction graph G *and $\alpha \in [0,1]$ find a set of vertices $U \subseteq V$ maximizing the Echo Chamber Score* (1.2)*.*

We will denote with $\hat{U}$ the set of users maximizing (1.2) and with $\xi(G)$ its corresponding score, i.e.

$$\hat{U} \coloneqq \arg\max_{U \subseteq V} \xi(U), \qquad \xi(G) \coloneqq \xi(\hat{U}).$$

1.2.3 The Densest Echo Chamber Problem

The ECP does not take into account the number of users producing a certain score; this means that the set U may involve also very sparse subgraphs, depending on the structure of the graph G.

For this reason it is interesting also to study another variant of the ECP, the Densest Echo Chamber Problem (D-ECP), which we now define.

Definition 1.2.4 (Densest Echo Chamber Score). *Let $U \subseteq V$ be a subset of vertices. Its* Densest Echo Chamber Score $\psi(U)$ *is*

$$\psi(U) = \sum_{\mathcal{C} \in \hat{\mathcal{C}}} \sum_{T[U] \in \mathcal{S}_C(U)} \frac{(|T^+[U]| - |T^-[U]|)}{|U|}. \tag{1.3}$$

Similarly to the ECP we can now define the corresponding problem

Problem 1.2.2 (Densest Echo Chamber Problem (D-ECP)). *Given an* interaction graph *G and $\alpha \in [0, 1]$ find a set of vertices $U \subseteq V$ maximizing the Densest Echo Chamber Score* (1.3)*.*

Note that $\psi(U) = \xi(U)/|U|$. The solutions to this problem are, in a certain sense, a *stronger* concept of Echo Chambers: we look for a group of vertices U whose score $\xi(U)$ is high when compared to $|U|$, i.e. a smaller subgraph with a large $\xi(U)$ will be preferred over a much bigger and sparser subgraph, even if the latter achieves an higher Echo Chamber Score.

1.3 Goals and Results

This work addresses the following research questions:

1. How can we solve the Echo Chamber Problem and the Densest Echo Chamber Problem?

2. Are they solvable or approximable in polynomial time?

3. Are these definitions capable of finding echo chambers in real world data?

We answer Questions 1 and 2 showing that these problems are not approximable in polynomial time within some non-trivial factor (see Chapter 3). In Chapter 4, we present different methods for solving and approximating them. In Chapter 5, we will validate one of the presented approximation algorithms over synthetic and real-world data, and see that, while in the first case it is able to reconstruct subgraphs whose vertices have many positive edges, in the second one it fails to recognize communities.

1.4 Structure of the thesis

The thesis is structured as follows:

1. Chapter 2 presents previous works and concepts needed for the development of the methods presented in the following chapters.
2. Chapter 3 provides proofs regarding the approximability of ECP and D-ECP.
3. Chapter 4 defines methods for solving and approximating the ECP and D-ECP problems.
4. Chapter 5 focuses on analyzing the data, how it is retrieved and preprocessed, and discussing the results obtained by applying the introduced methods.
5. Chapter 6 presents the positive effects and the drawbacks of the results, as well as possible future developments and improvements.

1.5 About the Thesis

The Python code used to obtain the results presented in the rest of the document is available at the following URL

`https://github.com/morpheusthewhite/master-thesis`

Chapter 2

Background

This chapter provides the background knowledge relevant for the thesis work. It will present concepts of Computational Complexity (Section 2.1) and Linear Programming (Section 2.2) as well as graph density problems (Section 2.3) which are significant in the following used methodologies.

2.1 Computational Complexity and Approximability

Complexity Theory deals with the study of the intrinsic complexity of computational problems. It also elaborates on the relationships between the complexity of different problems, for example proving that two problems are computationally equivalent [16], through a notion called *reduction*.

2.1.1 Optimization Problems and $\mathcal{NPO}$

Optimization problems are defined from a problem instance x, a set of feasible solutions S and a cost function that takes as input the problem instance x and a feasible solution $s \in S$, denoted as $\mathrm{cost}_O(x, s)$. Given a minimization (maximization) problem the optimal solution is defined as the s minimizing (maximizing) the value of $\mathrm{cost}_O(x, s)$, and we denote this value by $\mathrm{opt}_O(x)$[17].

$\mathcal{NPO}$ is then the set of optimization problems with the following properties:

- instances x can be recognized in polynomial time;
- $\mathrm{cost}_O(x, s)$ can be computed in polynomial time for $s \in S$;

- it takes polynomial time to decide if solution r of the instance x is feasible, i.e. whether $r \in S$;
- for every instance of the problem x and feasible solution for that problem $s \in S$ there is a polynomial q $s.t.$ $|s| \leq q(|x|)$ (i.e. the size of every solution is bounded by a polynomial in x).

If $\mathcal{P} \neq \mathcal{NP}$ for many optimization problems there is no algorithm for finding the optimal solution in polynomial time [17]. This is again a fundamental limitation about what we can compute which then requires the definition of some alternative approaches, like *approximation algorithms* which in polynomial time compute a solution which lies in a given factor from the optimal one [18].

Approximation. A is an r-approximation algorithm for an $\mathcal{NPO}$ minimization problem O if, for every instance x of O it holds that

$$cost_O(x, A(x)) \leq r \cdot opt_O(x)$$

(or, respectively, $\text{cost}_O(x, A(x)) \leq 1/r \cdot \text{opt}_O(x)$ for maximization problems), $A(x)$ being the optimal solution found by the approximation algorithm [17].

2.1.2 Approximation Preserving Reductions

If $\mathcal{P} \neq \mathcal{NP}$ the approximability of problems varies widely: while for some of them there exist constant factor approximations, for some others even a remotely approximate solution cannot be found [19] (some examples are listed in Table 2.1).

Approximation preserving reductions are a fundamental notion for proving a partial order among optimization problems [19]. Given a function f mapping instances of A to B and a function g mapping solutions of B to solutions of A, an approximation preserving reduction must have the following properties (when reducing from a problem A to a problem B) [21]:

- any instance x of A should be mapped to an instance $x' = f(x)$ of B in polynomial time,
- any solution $y' \in \text{sol}(f(x))$ of B should be associated to a corresponding solution $y = g(x, y') \in \text{sol}(x)$ of A in polynomial time.

The process is illustrated in Figure 2.1.

There are at least nine different kinds of approximation preserving reductions [21](Figure 2.2) but we will focus only on one type.

Table 2.1: Examples of known inapproximability results, assuming $\mathcal{P} \neq \mathcal{NP}$ [20]

Problem	Description	Inapproximability
MaxClique	Biggest complete subgraph	$\lvert V\rvert^{1-\epsilon}, \epsilon > 0$
MaximumIndipendentSet	Biggest set of not connected nodes	$\lvert V\rvert^{1-\epsilon}, \epsilon > 0$
MaxCut	Partition of nodes in two sets V_1 and V_2 minimizing the number of edges between the 2 sets	1.0624
MaximumSetPacking	Given a collection of finite sets C, finding the biggest collection $C' \subseteq C$ of disjoint sets	$\lvert C\rvert^{1-\epsilon}, \epsilon > 0$

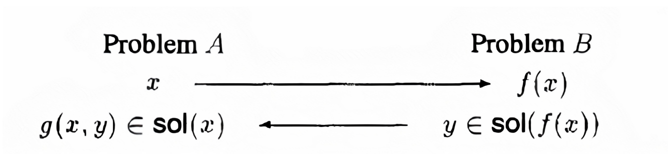


Figure 2.1: The reduction scheme [22].

S Reductions

An *S reduction* from problem A to problem B has the following properties [22]:

- for any instance x of problem A it holds that $\mathrm{opt}_A(x) = \mathrm{opt}_B(f(x))$,
- for any instance x of A and solution y' of B, $\mathrm{cost}_A(x, g(x, y')) = \mathrm{cost}_B(f(x), y')$.

S reductions are the strongest type of *approximation preserving reductions* and imply all the others [22].

2.2 Linear and Mixed Integer Programming

Linear Programming (LP) is a widely used optimization technique and one of the most effective; the term refers to problems in which both the constraints and *objective function* are linear [23, 24, 25, 26]. LPs are solvable in polynomial time [27, 28].

2.2.1 The Structure of LPs

In a LP problem we are given a vector $\mathbf{c} = (c_1, \ldots, c_n)$ and we want to maximize (or minimize) a linear function over the variables $\mathbf{x} = (x_1, \ldots, x_n)$ with the coefficients of the vector $\mathbf{c}$, i.e.

$$\mathbf{cx} = \sum_{i=1}^{n} c_i x_i$$

(known as the *objective function*) while satisfying some linear constraints over the variables [29, 24]:

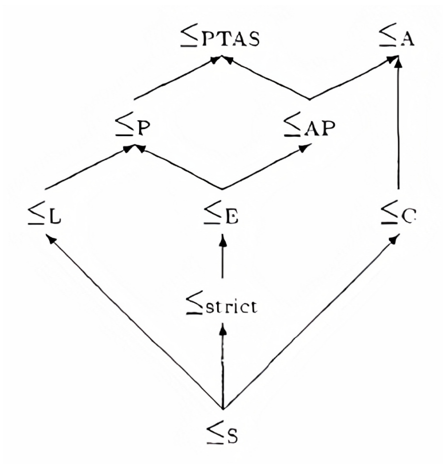


Figure 2.2: Taxonomy of approximation preserving reductions [22].

$$a_1x_1 + \cdots + a_nx_n \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} b$$

In general it is possible to formulate any LP problem as follows (called *standard form*) [24]

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} c_i x_i && (2.1)\\
\text{subject to} \quad & \sum_{i=1}^{n} a_{1i} x_i \leq b_1 && (2.2)\\
& \vdots && (2.3)\\
& \sum_{i=1}^{n} a_{mi} x_i \leq b_m && (2.4)\\
& x_i \geq 0, \quad i = 1, \ldots, n && (2.5)
\end{aligned}$$

The x_i are known also as *decision variables*; a choice of $\mathbf{x}$ is called *solution* and *feasible solution* if it satisfies the constraints, *optimum* if it is *feasible* and maximizes the *objective function* [24].

2.2.2 Solving an LP Problem

An option for solving an LP is called the *simplex method* which has two different phases.

Starting from the *standard form* (2.1) *slack* variables $x_{n+1}, \ldots, x_{n+m}$ are introduced, allowing to express the problem as follows [24, 23]:

$$\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n} c_i x_i && (2.6)\\
\text{subject to} \quad & x_{n+1} = b_1 - \sum_{i=1}^{n} a_{1i} x_i && (2.7)\\
& \vdots && (2.8)\\
& x_{n+m} = b_m - \sum_{i=1}^{n} a_{mi} x_i && (2.9)\\
& x_i \geq 0, \qquad i = 1, \ldots, n+m && (2.10)
\end{aligned}$$

The first phase involves finding a feasible solution for the problem. More specifically, we look for m variables, called *basic variables*, whose value we choose in order to satisfy the m equality constraints (while the remaining variables, the *nonbasic* ones, are set to 0); if no such feasible solution exists then the problem is *unfeasible*. Let $\mathcal{B}$ be the set of *basic variables*, $\mathcal{N}$ the set of *nonbasic variables* [24, 29] and $\bar{\zeta}$ the value of the objective function associated to this feasible solution. Then the problem can be reformulated as follows:

$$
\begin{aligned}
\text{maximize} \quad & \bar{\zeta} + \sum_{j \in \mathcal{N}}^{n} c_j x_j && && (2.11)\\
\text{subject to} \quad & x_i = \bar{b}_i - \sum_{j \in \mathcal{N}}^{n} \bar{a}_{ij} x_j && i \in \mathcal{B} && (2.12)\\
& x_i \geq 0, && i = 1, \ldots, n+m && (2.13)
\end{aligned}
$$

The second phase of the *simplex method* aims at improving the current solution: if $c_j \geq 0$ for all $j \in \mathcal{N}$, then the value of the objective function cannot be increased and we found an optimum. If, instead, there is at least one $c_j > 0$ then we can increase the value of ζ by increasing x_j; now there are two different cases [24]:

- As x_j increases there is at least a variable $\tilde{x}_j$ whose value needs to decrease to satisfy equality constraints. The first of these variables $\tilde{x}_j$ reaching 0 moves from $\mathcal{B}$ to $\mathcal{N}$, while x_j moves from $\mathcal{N}$ to $\mathcal{B}$. The problem is reformulated again as in (2.11) and the process is repeated [24].

- If no such $\tilde{x}_j$ variable exists then the value of x_j can be increased indefinitely and the problem is said to be *unbounded*, i.e. it can achieve any arbitrarily large value.

The process is illustrated with an example in Figure 2.3.

2.2.3 Mixed Integer Programming

Many problems involve not only continuous variables but also variables that take binary or integer values: these are known as Mixed Integer Programming (MIP) problems. Furthermore, some of these problems are linear in the constraints and in the objective function and are known as Mixed Integer Linear Programming (MILP) problems [23, 31].

A generic MILP can be expressed as follows [32]:

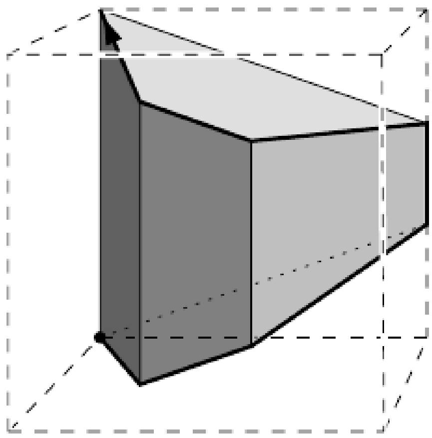


Figure 2.3: An example of the progress of the *simplex method*: the process moves along the vertices of the polygon defined by the constraints while improving the value of the solution. Picture taken from [30].

$$
\begin{aligned}
\text{maximize} \quad & \sum_{i=1}^{n_1} c_i x_i + \sum_{i=1}^{n_2} h_i y_i \\
\text{subject to} \quad & \sum_{i=1}^{n_1} a_{1i} x_i + \sum_{i=1}^{n_2} g_{1i} y_i \le b_1 \\
& \vdots \\
& \sum_{i=1}^{n_1} a_{mi} x_i + \sum_{i=1}^{n_2} g_{1i} y_i \le b_m \\
& x_i \ge 0, \qquad i = 1, \dots, n_1 \\
& y_i \ge 0, \qquad i = 1, \dots, n_2 \text{ integral}
\end{aligned}
\tag{2.14}
$$

For convenience, we will refer to MILP problems as MIP in the rest of the document.

The *relaxation* of a MIP problem is defined as the same problem where the integrality constraints have been removed [23].

Solving a MIP is a difficult task in general, differently from the LP problems. It has been shown also that MIP is $\mathcal{NP}$-**Hard** [33, 34, 35]. This is why the *relaxation* is often considered for getting an approximation of the exact solution and it can be solved in polynomial time [32].

2.2.4 Solving a MIP

One approach that has been proven successful for solving MIP is the Branch-and-Bound, which is guaranteed to find an optimal solution [32, 23].

Given a problem P, the process starts by solving the *relaxation* of P and finding its optimal solution $(\tilde{x}, \tilde{y})$. Let S and $\tilde{S}$ be the set of feasible solutions for the original problem and its relaxation, respectively. By definition, we have that $S \subseteq \tilde{S}$. Therefore, [23]

- If the relaxation problem is not feasible so will be the original problem.
- If $\tilde{y}$ has only integer values then we found the optimal solution for the original problem.
- If, instead, $\tilde{y}$ contains some fractional values, we start by initializing the value of the best solution so far, ζ, with $-\infty$. Then we choose one of the fractional variables that are required to be integral in the original problem, say y_j with value f, and create two subproblems, respectively adding the constraint $y_j \le \lfloor f \rfloor$ and $y_j \ge \lceil f \rceil$. This step is called *branching*. We now consider the solution of each subproblem (x_j, y_j) with value of the objective function z_j [23, 32].

- If either of the subproblems is not feasible or its value z_j is lower than the best one found so far then it does not need to be considered further. This is called *pruning*.
- If y_j are all integer values then $\zeta = z_j$.
- Otherwise, we subdivide again in two subproblems as above.

When there are no remaining subproblems to consider then Branch-and-Bound terminates [23].

2.3 Density in Graphs

2.3.1 The Densest Subgraph Problem

Finding dense subgraphs is a problem which has received a lot of attention and different definitions of density have been used [36, 37, 38, 39].

We will refer to the definition in [36] and present some of its results which are used and important for the development of the methods in the following chapters.

Let $G = (V, E)$ be an undirected graph, let $S \subseteq V$ a subset of the nodes, and let $E(S)$ denote the edges of G induced by S, i.e.

$$E(S) = \{e_{ij} \in E \; s.t. \; v_i \in S \wedge \; v_j \in S\}.$$

The *density* $f(S)$ is defined as

$$f(S) = \frac{|E(S)|}{|S|}. \tag{2.15}$$

According to this definition it is easy to see that $2 \cdot f(S)$ is the average degree of the subgraph induced by S.

The density of the graph $f(G)$ is then defined as

$$f(G) = \max_{S \subseteq V} f(S). \tag{2.16}$$

The problem of computing $f(G)$ is known as the *Densest Subgraph Problem* [36].

There are different techniques for solving it: a solution based on parametric maximum flow has been proposed in [40]; Charikar in [36] proposed an alternative solution based on the following Linear Programming model (a more in-depth discussion about LP can be found in Section 2.2)

$$\begin{aligned}
\text{maximize} \quad & \sum_{ij \in E} x_{ij} && (2.17)\\
\text{subject to} \quad & x_{ij} \le y_i \quad \forall ij \in E && (2.18)\\
& x_{ij} \le y_j \quad \forall ij \in E && (2.19)\\
& \sum_{i \in V} y_i \le 1 && (2.20)\\
& y_i \ge 0 \qquad \forall i \in V && (2.21)\\
& x_{ij} \ge 0 \quad \forall ij \in E && (2.22)\\
& && (2.23)
\end{aligned}$$

Intuitively, the problem associates non-zero y_i to vertices in $S \subseteq V$ and non-zero x_{ij} to edges induced by S. The concept of "density" is introduced by (2.20), which distributes a fixed quantity (one in this case) to the vertices, such that the value y_i of each vertex generally decreases (and consequently also that of the x_{ij}) as the number of non-zero y_i increases.

Let $S(r) \coloneqq \{v_i : y_i \ge r\}$ and $E(r) \coloneqq \{e_{ij} : x_{ij} \ge r\}$. It is easy to see that, given the model as defined above, $E(r)$ is the set of edges induced by the vertices in $S(r)$.

The set of vertices S maximizing the density $f(S)$ can then be reconstructed from the results of the LP by finding the density of $S(r)$ for all choices of $r = y_i,\ i \in V$ [36].

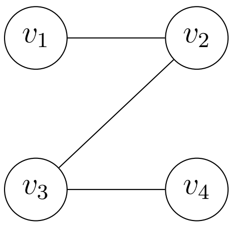
An approximate algorithm for the Densest Subgraph Problem. In [36], Charikar also defines a greedy approach for solving the Densest Subgraph problem which gives a 2-approximation for $f(G)$.

The algorithm starts by defining a set of vertices S which is initialized with V and, through the iterations, it removes from S the vertex v_i which has the lowest degree in the subgraph induced by S, until S is empty. Then it returns the set S which, during the process, was associated with the highest density $f(S)$.

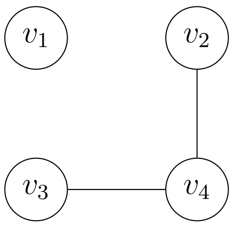
2.3.2 The Densest Common Subgraph Problem

The Densest Common Subgraph (DCS) Problem was initially introduced by Jethava and Beerenwinke in [41] and later studied in [42], [43] and [44] that also introduced new variants.

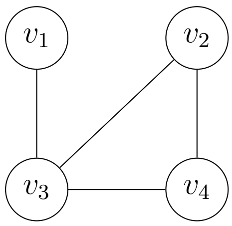
Let $\mathcal{G} = (G_1, G_2, \ldots, G_T)$ be a sequence of graphs on the same set of vertices V (an example is shown in Figure 2.4), $S \subseteq V$ a subset of the nodes, $G_i[S]$ the subgraph induced by S in G_i, $\deg_{G_i[S]}(v_j)$ the degree of $v_j \in S$ in $G_i[S]$ and min-deg$(G_i[S])$ the minimum induced degree, i.e. min-deg$(G_i[S]) \coloneqq \min_{v_j \in S} \deg_{G_i[S]}(v_j)$.



(a) G_1



(b) G_2



(c) G_3

Figure 2.4: An example of a graph sequence $\mathcal{G} = (G_1, G_2, G_3)$ over four vertices.

Solving the Densest Common Subgraph Problem means finding a subset of the vertices $S \subseteq V$ that maximizes some aggregate density function over the graph sequence. More formally, let g be a function that calculates the density of a set of nodes S in a undirected graph G, i.e.

$$g : S \times G \to \mathbb{R}.$$

Also, let h be a function aggregating the value of the density of each snapshot of the graph sequence, i.e.

$$h : g(S, G_1) \times g(S, G_2) \times \cdots \times g(S, G_T) \to \mathbb{R}.$$

Then, the DCS problem consists in finding S maximizing the following quantity:

$$f(S, \mathcal{G}) = h(\{g(S, G_1),\ \ldots,\ g(S, G_T)\}).$$

We call f the *density aggregation function*. According to the definition of f there are different variants of the problem

- DCS-MM maximizes the Minimum of the Minimum degrees along the graph sequence, i.e.

$$g = \text{min-deg}(G_i[S]), \qquad f = \min_{i \in [T]} \text{min-deg}(G_i[S]). \tag{2.24}$$

  A non-trivial solution means finding a set of nodes which are linked in all the graphs $G_i \in \mathcal{G}$. In [44] it is shown a simple greedy approach for finding the solution in polynomial time.

- DCS-MA uses the following functions

$$g = \frac{\sum_{v_j \in S} \deg_{G_i[S]}(v_j)}{|S|}, \qquad f = \min_{i \in [T]} \frac{\sum_{v_j \in S} \deg_{G_i[S]}(v_j)}{|S|}. \tag{2.25}$$

  This means finding a set of vertices S which are dense in the sense that they have a non-trivial average degree in all the graphs of the sequence.

  Charikar, Naamad and Yu in [43] and Semertzidis, Pitoura, Terzi, Tsaparas in [44] provide some approximation algorithms with guaranteed bounds; in [43] also they prove its inapproximability to within a $\mathcal{O}(2^{\log^{1-\epsilon} n})$ factor unless $\mathcal{NP} \subseteq \mathbf{DTIME}(n^{\text{poly} \log n})$, $\epsilon > 0$ [1].

[1] They show this results through a reduction from MinRep, which has been shown to have the mentioned inapproximability [43, 45].
$\mathbf{DTIME}(f(n))$ refers to the class of problems that have time complexity $f(n)$ [16].

- DCS-AM, whose density aggregation function is

$$f = \sum_{i \in [T]} \text{min-deg}(G_i[S]), \tag{2.26}$$

  while g is the same as in (2.24). This choice will push the algorithms to find a set of vertices which have degree greater than 0 in some of the subgraphs they induce in the graph sequence.

  Charikar, Naamad and Yu proved in [43] that DCS-AM is inapproximable within factor $n^{1-\epsilon}$ unless $\mathcal{P} = \mathcal{NP}$, $\epsilon > 0$ [2]. For fixed T, they also provide a fixed parameter polynomial time algorithm which can be used for solving this problem exactly, as well as a $(1+\epsilon)$-approximation algorithm.

- DCS-AA maximizing

$$f = \sum_{i \in [T]} \frac{\sum_{v_j \in S} \deg_{G_i[S]}(v_j)}{|S|} \tag{2.27}$$

  which puts fewer restrictions than the previous variants on the solutions, requiring only a high average degree on the union of the graphs. Note that g is the same as in (2.25).

  This problem can be solved optimally in polynomial time as it can be reduced to the classical Densest Subgraph problem (Subsection 2.3.1) [44]; similarly the approximation algorithm in Subsection 2.3.1 provides a 2-approximation for the optimal solution.

  More specifically, solving DCS-AA is the equivalent of solving the Densest Common Subgraph (DCS) on the average graph $\hat{H}_{\mathcal{G}}$, which is defined as a weighted graph whose weight of each edge is the fraction of graphs in the sequence $\mathcal{G}$ where the edge is present [44].

2.3.3 The O^{2}Bff Problem

The DCS is also known as the Best Friends Forever (Bff) Problem as defined by Semertzidis, Pitoura, Terzi, Tsaparas [44]; in the same paper also another class of similar problem is defined, the On-Off Bff Problem.

Let $\mathcal{G} = (G_1, G_2, \ldots, G_T)$ be a sequence of graphs on the same vertex set V. The On-Off Bff (O^{2}Bff) is defined as the set of vertices $S \subseteq V$ and the set of k graphs $\mathcal{L}_k \subseteq \mathcal{G}$ that maximize some density aggregation function $f(S, \mathcal{L}_k)$.

[2] By reducing from the MaximumIndipendentSet problem.

As this problem relies again on a function f which aggregates the density across the graphs G_i, similarly to the DCS Problem four variants can be defined, using the same functions mentioned in Subsection 2.3.2.

We will focus and present only an algorithm for approximating O^2Bff-AM; the other algorithms can be found in [44].

Let us first define Score$_a$, a procedure that removes the node with the lowest degree in a graph while properly updating the degree of the other nodes. When called for the first time, this function initializes $\hat{H}_{\mathcal{G}}$, $\hat{E}$ and $\mathcal{F}[d]$, as described in Algorithm 2.1, that are updated during the subsequent calls to the function.

Note that we refer to the degree d of a vertex v_i in a weighted graph as the sum of the weights of the edges of the node, i.e. $d_{v_i} = \sum_{(v_i,v_j)\in E} w_{ij}$.

Algorithm 2.1: The Score$_a$ algorithm

Result: The vertex with the lowest degree is removed and returned

$\hat{H}_{\mathcal{G}} \leftarrow$ average graph of $\mathcal{G}$;

$\hat{E} \leftarrow$ set of edges of $\hat{H}_{\mathcal{G}}$;

$\mathcal{F}[d] \leftarrow$ set of nodes with degree d in $\hat{H}_{\mathcal{G}}$;

function ScoreAndUpdate($\mathcal{G}$) {

  $score_a \leftarrow$ smallest d s.t. $\mathcal{F}[d] \neq \emptyset$;

  $u \leftarrow$ a node with degree d ;

  remove u from $\mathcal{F}[d]$;

  foreach $(u, v) \in \hat{E}$ **do**

    remove v from $\mathcal{F}[d_v]$;

    remove (u, v) from $\hat{E}$ and update d_v ;

    add v to $\mathcal{F}[d_v]$;

  end

  $V = V \setminus \{u\}$;

  return u ;

}

Let us also define FindBff$_a$, a greedy approach for finding the subgraph with the highest density f (which corresponds to (2.26) in our case, but it can be replaced by any of the other density aggregation functions). This function repeatedly calls ScoreAndUpdate to remove the node with the lowest degree and efficiently update the graph. When the graph is empty it returns the subset of nodes that obtained the highest density score f (Algorithm 2.2).

Semertzidis, Pitoura, Terzi, Tsaparas in [44] present two different approaches for approximating O^2Bff-AM: an *iterative* one which starts with a set $\mathcal{L}_k \in \mathcal{G}$ of

Algorithm 2.2: The $\textsc{FindBff}_a$ algorithm

Result: A subset of nodes $S \subseteq V$

$S_0 = V$;

for $i \in \{1, \ldots, |V|\}$ **do**

  $v_i = \textsc{ScoreAndUpdate}(\mathcal{G}[S_i])$;

  $S_i = S_{i-1} \setminus \{v_i\}$;

end

return $\text{argmax}_{i \in \{1,\ldots,|V|\}} f(S_i, \mathcal{G})$

k graphs and improves it to increase the score, and an *incremental* one in which the set of k graphs $\mathcal{L}_k$ is selected along the k iterations, starting with a pair and adding snapshots $G \in \mathcal{G}$ one by one.

Furthermore, they identify two possible approaches for each of them. We will focus on the Incremental overlap (INC_O).

The algorithm starts by solving the DCS problem on each of the graphs G_i in $\mathcal{G}$ and finding the corresponding set of vertices S_i of the solution. Then $\mathcal{L}_2$ is chosen as the pair of graphs which have the most similar set of vertices in the respective solutions, where the similarity is measured through the Jaccard coefficient.

The Jaccard coefficient is measure of similarity between two sets. More specifically, given two sets C_1 and C_2, the Jaccard coefficient is equal to [46]

$$Jaccard(C_1, C_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}.$$

The DCS problem is now solved on $\mathcal{L}_2$ to obtain a set of vertices S_C which is compared against the other S_i previously computed solutions to find the most similar one, as before, and the process continues until $\mathcal{L}_k$ is constructed (Algorithm 2.3). Finally, the $\textsc{FindBff}_a$ is called on $\mathcal{L}_k$ and the resulting set of vertices is returned along with $\mathcal{L}_k$.

Algorithm 2.3: The INC_O algorithm for approximating $\mathrm{O}^2\textsc{Bff}$-AM

```
Result: A subset of nodes S ⊆ V and of graphs L_k ⊆ G
for i ∈ {1, ..., |G|} do
    S_i = FindBff_a({G_i}) ;
end
L_2 = arg max_{G_i,G_j ∈ G} Jaccard(S_i, S_j) ;
for i ∈ {3, ..., k} do
    S_C = FindBff_a(L_{i-1}) ;
    G_m = arg max_{G_j ∈ G, G_j ∉ L_{i-1}} Jaccard(S_C, S_j) ;
    L_i = L_{i-1} ∪ {G_m} ;
end
S = FindBff_a(L_k) ;
return S, L_k ;
```

Chapter 3

Problem Complexity and Approximability

We will now prove the inapproximability of the ECP and D-ECP within some nontrivial factor.

3.1 Hardness of ECP

Theorem 3.1.1. *The Echo Chamber Problem (ECP) has no $n^{1-\epsilon}$-approximation algorithm for any $\epsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$.*

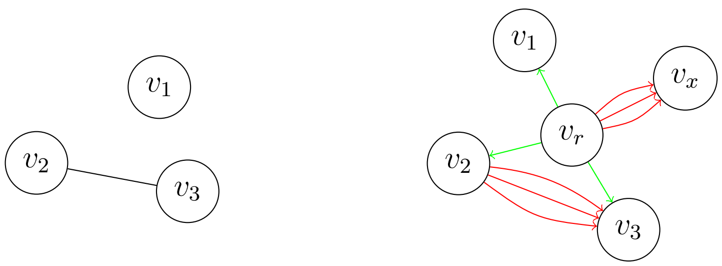
Proof. We show this by presenting a direct reduction from Maximum Independent Set (MIS), which is known having the mentioned hardness factor (Table 2.1).

Let $G_1 = (V_1, E_1)$ be an undirected and unweighted graph for which we want to solve MIS.

We show how to construct an interaction graph G_2 as instance for ECP with parameter α. Let $\lambda > \frac{\alpha}{1-\alpha}$, $\lambda \in \mathbb{N}$ and $n_1 := |V_1|$. G_2 is constructed as follows:

- for each vertex $v_i \in V_1$ we add a vertex in G_2,
- for each edge $e_{ij} \in E_1$ we add λn_1 negative edges between v_i and v_j,
- we add a vertex v_r and a positive edge between v_r and any other vertex $v_i \in V_2$ that we already inserted in G_2,
- we add a vertex v_x and λn_1 negative edges between v_x and v_r.

Furthermore, all the edges in G_2 are associated to the same content C and the same thread $T \in \mathcal{T}_C$. Thus, our ECP instance only contains a single thread and a single content. An illustration of the reduction can be found in Figure 3.1.



(a) G_1, undirected graph

(b) G_2, directed signed graph, for $\lambda = 1$

Figure 3.1: Example construction of the interaction graph G_2 from G_1, for $\alpha = \frac{1}{3}$.

Claim 3.1.2. *Content C is* controversial*, i.e. $\eta(C) > \alpha$.*

Proof. Let m_2^- and m_2^+ be the number of negative and positive edges in G_2, respectively.

By construction in G_2 there is exactly one positive edge between v_r and each vertex v_i from G_1, i.e. $m_2^+ = n_1$. Also, $m_2^- \geq \lambda n_1$, since G_2 contains at least the λn_1 negative edges between v_r and v_x. Consequently, given that for any $a, b, c \in \mathbb{R}^+$ it holds that $\frac{a+b}{a+b+c} \geq \frac{a}{a+c}$, we have

$$\eta(C) = \frac{m_2^-}{m_2^- + m_2^+} \geq \frac{\lambda n_1}{\lambda n_1 + n_1} = \frac{\lambda}{\lambda + 1} > \alpha. \tag{3.1}$$

□

Thus, the content C is *controversial*. Since our instance only contains a single content, this reduces the ECP on G_2 to the maximization of

$$\xi(U) = \sum_{T \in \mathcal{S}_C(U)} (|T[U]^+| - |T[U]^-|). \tag{3.2}$$

Claim 3.1.3. *Let* OPT(ECP) *and* OPT(MIS) *be the maximum Echo Chamber score on G_2 and the size of the MIS on G_1, respectively. We have that*

$$\text{OPT(ECP)} = \text{OPT(MIS)} \tag{3.3}$$

Proof. Let $I \subseteq V_1$ be an independent set of G_1 of size $|I| > 1$. Consider the associated solution in G_2 in which $U = I \cup \{v_r\}$. By construction, $T[U]$ only contains $|I|$ positive edges, so $T[U] \in \mathcal{S}_C(U)$ and also

$$\text{OPT(ECP)} \geq \xi(U) = |T^+[U]| = |I| \implies \text{OPT(ECP)} \geq \text{OPT(MIS)}. \tag{3.4}$$

Now let $S \subseteq V_2$ be a solution of the ECP on G_2, and suppose $\xi(S) > 0$. We will have that $v_r \in S$ and that $v_x \notin S$. Let $J \coloneqq S \setminus \{v_r\}$.

Next, we argue that J is an independent set for G_1. We prove this by contradiction. Suppose that two vertices v_i, $v_j \in J$ are linked in G_1. By construction there are at least λn_1 negative edges in $T[S]$, thus

$$\eta(T[S]) \geq \frac{\lambda n_1}{\lambda n_1 + |S - 1|} \geq \frac{\lambda n_1}{\lambda n_1 + n_1} = \frac{\lambda}{\lambda + 1} > \alpha. \tag{3.5}$$

This means that $T[S]$ is *controversial* and $T \notin \mathcal{S}_C(S)$; therefore, the sum in (3.2) resolves to zero, which is a *contradiction*.

Consequently, J contains vertices which are independent in G_1. Therefore, $T[S]$ contains only positive edges; more specifically,

$$\xi(S) = |T^+[S]| = |S| - 1 = |S \setminus \{v_r\}| = |J|. \tag{3.6}$$

Thus

$$\text{OPT(MIS)} \geq |J| \implies \text{OPT(MIS)} \geq \text{OPT(ECP)}. \tag{3.7}$$

So the optimal value of the constructed instance of ECP exactly equals that of the Maximum Independent Set instance.

□

So, if we were able to approximate ECP within $n^{1-\epsilon}$, $\epsilon > 0$ we would be also able to approximate MIS within the same factor, which is not possible unless $\mathcal{P} = \mathcal{NP}$, given also that our reduction takes polynomial time.

This means ECP has a hardness factor at least as large as that of MIS.

This concludes the proof of Theorem 3.1.1. □

3.2 Hardness of D-ECP

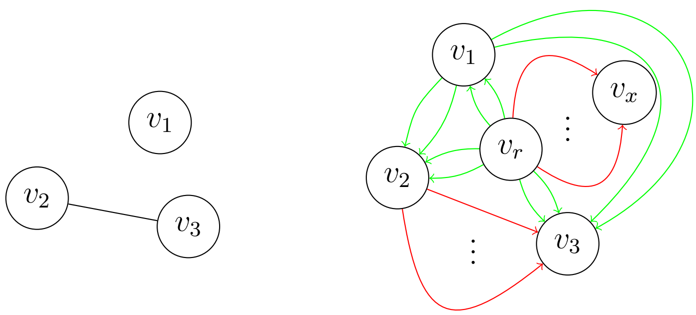
Theorem 3.2.1. *The Densest Echo Chamber Problem (D-ECP) has no $n^{1-\epsilon}$-approximation algorithm for any $\epsilon > 0$ unless $\mathcal{P} = \mathcal{NP}$.*

Proof. We again show this result by presenting a direct reduction from Maximum Independent Set. Differently from before, we will need to create an instance of the D-ECP with a positive clique over independent vertices of the original graph.

Let $G_1 = (V_1, E_1)$ be an undirected and unweighted graph for which we want to solve MIS.

We show how to construct an interaction graph G_2 as instance for D-ECP with parameter α. Let $\lambda > \frac{\alpha}{1-\alpha}$, $\lambda \in \mathbb{N}$ and $n_1 := |V_1|$. G_2 is constructed as follows:

- for each vertex $v_i \in V_1$ we add a vertex in G_2,
- for each edge $e_{ij} \in E_1$ we add $\lambda(n_1 + 1)^2$ negative edges between v_i and v_j,
- for each edge $e_{ij} \in V_1 \times V_1 \setminus E_1$ we add 2 positive edges between v_i and v_j,
- we add a vertex v_r and 2 positive edges between v_r and any other vertex $v_i \in V_2$ that we already inserted in G_2,
- we add a vertex v_x and λn_1^2 negative edges between v_x and v_r.



(a) G_1, undirected graph **(b)** G_2, directed signed graph

Figure 3.2: Example construction of the interaction graph G_2 from G_1.

Furthermore, all the edges in G_2 are associated to the same content C and the same thread $T \in \mathcal{T}_C$. Thus, our D-ECP instance only contains a single thread and a single content. An illustration of the reduction can be found in Figure 3.2.

Claim 3.2.2. *The content C is* controversial*, i.e. $\eta(C) > \alpha$.*

Proof. By construction G_2 will contain at most two positive edges between each pair of vertices from G_1 and v_r, i.e. $m_2^+ \leq n_1(n_1+1) < (n_1+1)^2$. Also, $m_2^- \geq \lambda(n_1+1)^2$ since G_2 contains at least the λn_1^2 negative edges we added between v_r and v_x. Thus, given that for any $a, b, c \in \mathbb{R}^+$ it holds that $\frac{a+b}{a+b+c} \geq \frac{a}{a+c}$, we have that

$$\eta(C) = \frac{m_2^-}{m_2^- + m_2^+} \geq \frac{\lambda(n_1+1)^2}{\lambda(n_1+1)^2 + (n_1+1)^2} = \frac{\lambda}{\lambda+1} > \alpha. \tag{3.8}$$

□

Thus, the content C is *controversial.* Since our instance contains a single content, this reduces the D-ECP on G_2 to the maximization of

$$\psi(U) = \sum_{T \in \mathcal{S}_C(U)} \frac{|T^+[U]| - |T^-[U]|}{|U|}. \tag{3.9}$$

Claim 3.2.3. *Let* OPT(ECP) *and* OPT(MIS) *be the maximum Echo Chamber score on G_2 and the size of the MIS on G_1, respectively. We have that*

$$\mathrm{OPT(D\text{-}ECP)} = \mathrm{OPT(MIS)}. \tag{3.10}$$

Proof. Let $I \subseteq V_1$ be an independent set of G_1 of size $n_I := |I| > 1$ (unless G_1 is a clique we can always trivially find an independent set of size two by choosing two vertices that are not connected by an edge). Consider the associated solution in G_2 in which $U = I \cup \{v_r\}$.

By construction, $T[U]$ only contains positive edges, more specifically:

- $2 \cdot n_I$ positive edges between v_r and $v_i \in I$,
- $n_I(n_I - 1)$ edges between vertices $v_i \in I$.

Thus $T[U] \in \mathcal{S}_C(U)$ and also

$$\psi(U) = \frac{|T^+[U]| - |T^-[U]|}{|U|} = \frac{2n_I + n_I(n_I-1)}{n_I+1} = \frac{n_I^2 + n_I}{n_I+1} = n_I. \tag{3.11}$$

Consequently,

$$\mathrm{OPT(D\text{-}ECP)} \geq \psi(U) = |I| \implies \mathrm{OPT(D\text{-}ECP)} \geq \mathrm{OPT(MIS)}. \tag{3.12}$$

Now let $S \subseteq V_2$ be a solution of the D-ECP on G_2, and suppose $\psi(S) > 0$ (we can always choose $S = \{v_r\} \cup \{v_i\}$ with v_i vertex from G_1, which will produce $\psi(S) = 1$). We will have that $v_r \in S$ and that $v_x \notin S$. Let $J \coloneqq S \setminus \{v_r\}$ be the corresponding solution for MIS.

Next, we argue that S is an independent set for G_1. We prove this by contradiction. Suppose that two vertices v_i, $v_j \in J$ are linked in G_1. By construction there are at least $\lambda(n_1+1)^2$ negative edges in $T[S]$, thus

$$\begin{aligned} \eta(T[S]) &= \frac{|T^-[S]|}{|T^-[S]| + |T^+[S]|} \\ &\geq \frac{\lambda(n_1+1)^2}{\lambda(n_1+1)^2 + n_j(n_j+1)} \\ &\geq \frac{\lambda(n_1+1)^2}{\lambda(n_1+1)^2 + (n_1+1)^2} \\ &= \frac{\lambda}{\lambda+1} \\ &> \alpha \end{aligned}$$

where $n_j \coloneqq |J|$.

This means that $T[S]$ is *controversial* and $T \notin \mathcal{S}_C(S)$; therefore, the sum in (3.9) resolves to zero, which is a *contradiction*.

Consequently, J contains vertices which are independent in G_1. Therefore, $T[S]$ contains only positive edges. Similarly to (3.11),

$$\psi(S) = \frac{|T^+[S]|}{|S|} = |J|. \tag{3.13}$$

Thus,

$$\mathrm{OPT(MIS)} \geq |J| \implies \mathrm{OPT(MIS)} \geq \mathrm{OPT(D\text{-}ECP)} \tag{3.14}$$

□

So the optimal value of the constructed instance of D-ECP exactly equals that of the Maximum Independent Set instance. As motivated before (Section 3.1), it will have a hardness factor at least as large as that of MIS.

This concludes the proof of Theorem 3.2.1. □

Chapter 4

Solving the ECP and the D-ECP

We now present some techniques for calculating exactly and approximating both the ECP and the D-ECP (Subsection 1.2.2).

4.1 Exact Solutions

We start with our exact algorithms that are based on MIPs.

4.1.1 A MIP Model for the ECP

Let G be an *interaction graph* for contents $\mathcal{C}$ and threads $T \in \mathcal{T}_C,\ C \in \mathcal{C}$ for which we want to solve the ECP. Fix $\alpha \in [0,1]$. Let $\hat{\mathcal{C}} \subseteq \mathcal{C}$ be the set of controversial contents and E_k the set of all edges of thread T_k associated to a controversial content, i.e. $T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}}$; let also E_k^+ and E_k^- be the set of positive and negative edges in T_k, respectively.

The following MIP model is able to solve the ECP on G for values of $\alpha \leq 0.5$.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{T_k \in \mathcal{T}_C,\, C \in \hat{\mathcal{C}}} \Big(\sum_{ij \in E_k^+} x_{ij}^k - \sum_{ij \in E_k^-} x_{ij}^k \Big) && && (4.1)\\
\text{subject to} \quad & x_{ij}^k \leq y_i && \forall ij \in E_k && (4.2)\\
& x_{ij}^k \leq y_j && \forall ij \in E_k && (4.3)\\
& x_{ij}^k \leq z_k && \forall ij \in E_k && (4.4)\\
& x_{ij}^k \geq -2 + y_i + y_j + z_k && \forall ij \in E_k && (4.5)\\
& \sum_{ij \in E_k^-} x_{ij}^k - \alpha \sum_{ij \in E_k} x_{ij}^k \leq 0 && \forall T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} && (4.6)\\
& y_i \in \{0,1\} && \forall i \in V && (4.7)\\
& 0 \leq x_{ij}^k \leq 1 && \forall ij \in E_k && (4.8)\\
& 0 \leq z_k \leq 1 && \forall T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} && (4.9)
\end{aligned}
$$

The MIP model introduces variables x, y and z.

- y variables are associated to vertices (Equation 4.7). Intuitively $y_i = 1$ means that the vertex v_i is part of the set $U \subseteq V$ considered for the score.

- x variables are associated to edges (Equation 4.8). A value of $x_{ij}^k = 1$ should be interpreted as the fact that the edge $e_{ij} \in E_k$ is contributing to the score, i.e. $T_k \in \mathcal{S}_C(U)$.

- z variables are associated to threads (Equation 4.9). A value greater than 0 is generally associated to *non-controversial* threads and *controversial* threads have value $z_k = 0$.

We will now show that the ECP can be solved through MIP (4.1)-(4.9).

Theorem 4.1.1. *Let* $G = \{G_k = (V, E_k)\}_k$ *be an* Interaction Graph *and* $\alpha \in [0, 0.5]$. *Then*

$$\max_{U \subseteq V} \xi(U) = \text{OPT(MIP)}, \tag{4.10}$$

where OPT(MIP) *denotes the optimal solution to MIP* (4.1)-(4.9).

Proof. We will show the equality by first proving that $RHS \geq LHS$ and then that $LHS \geq RHS$.

Claim 4.1.2. *For any* $U \subseteq V$, *the MIP* (4.1)-(4.9) *achieves value at least* $\xi(U)$.

Proof. Let $E_k[U]$ the set of edges induced by U in thread T_k. We construct a MIP solution as follows:

$$y_i = \begin{cases} 1, & \text{if } v_i \in U, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall v_i \in V \tag{4.11}$$

$$z_k = \begin{cases} 1, & \text{if } T_k \in \mathcal{S}_C(U),\ C \in \hat{\mathcal{C}}, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall T_k \in \mathcal{T}_C, C \in \mathcal{C} \tag{4.12}$$

$$x_{ij}^k = \begin{cases} 1, & \text{if } e_{ij} \in E_k[U],\ T_k \in \mathcal{S}_C(U),\ C \in \hat{\mathcal{C}}, \\ 0, & \text{otherwise.} \end{cases} \qquad \forall e_{ij}^k \in E_k \tag{4.13}$$

To satisfy (4.2)-(4.5) we need that

$$x_{ij}^k = 1 \iff y_i = 1 \land y_j = 1 \land z_k = 1. \tag{4.14}$$

This is always true since we defined x_{ij}^k to be 1 only and if it is associated to an edge induced in a $T_k \in \mathcal{S}_C(U),\ C \in \hat{\mathcal{C}}$.

Let us now consider a thread $T_k \in \mathcal{S}_C(U)$. Then

$$\eta(T_k[U]) \leq \alpha \implies \frac{|E_k^-[U]|}{|E_k[U]|} \leq \alpha \implies |E_k^-[U]| - \alpha |E_k[U]| \leq 0. \tag{4.15}$$

so (4.6) is satisfied. It is easy to see that if $T_k \notin \mathcal{S}_C(U)$ then $x_{ij}^k = 0$ for all $ij \in E_k$ and the constraint is also satisfied.

Finally, any edge contributing to $\xi(U)$ will also equally contribute to the objective function. $\square$

Claim 4.1.3. *Given a feasible solution of MIP* (4.1)-(4.9) *with value v we can construct U s.t. $\xi(U) \geq v$.*

Proof. We define $U \coloneqq \{v_i \ s.t.\ y_i = 1\}$. Again, by (4.2)-(4.5) we have (4.14), so

$$x_{ij}^k = 1 \implies z_k = 1, \tag{4.16}$$

$$z_k = 1 \implies x_{i'j'}^k = 1\ \forall i'j' \in E_k[U], \tag{4.17}$$

meaning that if $z_k = 1$ then for all the edges $e_{ij} \in E_k$ induced by U we will have $x_{ij}^k = 1$ (i.e. they will contribute to the objective). Let us now consider T_k $s.t.$ $z_k = 1$. Because of (4.6) and (4.17) we have

$$|E_k^-[U]| - \alpha |E_k[U]| \leq 0 \implies \frac{|E_k^-[U]|}{|E_k[U]|} \leq \alpha. \tag{4.18}$$

i.e. $\eta(T_k[U]) \leq \alpha$. So $T_k \in \mathcal{S}_C(U),\ C \in \hat{\mathcal{C}}$, i.e. $z_k = 1 \implies T_k \in \mathcal{S}_C(U),\ C \in \hat{\mathcal{C}}$, thus $T_k[U]$ contributes to $\xi(U)$; more specifically any edge contributing to the objective function equally contributes to $\xi(U)$.

Now suppose there exists $T_k \in \mathcal{S}_C(U)$ *s.t.* $z_k = 0$. Then $x_{ij}^k = 0$ by (4.4). Since $\alpha \leq 0.5$,

$$\eta(T_k[U]) \leq \alpha \implies \frac{|E_k^-[U]|}{|E_k[U]|} \leq 0.5 \tag{4.19}$$

$$\implies |E_k^-[U]| \leq 0.5 \cdot (|E_k^+[U]| + |E_k^-[U]|) \tag{4.20}$$

$$\implies 0.5 \cdot |E_k^+[U]| - 0.5 \cdot |E_k^-[U]| \geq 0 \tag{4.21}$$

$$\implies |E_k^+[U]| - |E_k^-[U]| \geq 0. \tag{4.22}$$

Consequently T_k will contribute positively to $\xi(U)$, i.e. in the subgraph induced on T_k by U the number of positive edges is greater or equal than the number of negative edges.

More generally, due to (4.2)-(4.5) we have

$$z_k = c > 0 \implies x_{i'j'}^k = c\ \forall i'j' \in E_k[U],$$

meaning that if $z_k = c$ all and only the variables x_{ij}^k associated to edges $e_{ij} \in E_k$ induced by U will get value c (any other x_{ij}^k will get value 0 for (4.2)-(4.3)).

Combining this result with (4.6) we get again (4.18) and, consequently, (4.22). Therefore, the contribution of T_k associated to $z_k > 0$ will be

$$\sum_{ij \in E_k^+} x_{ij}^k - \sum_{ij \in E_k^-} x_{ij}^k = z_k(|E_k^+[U]| - |E_k^-[U]|) \geq 0. \tag{4.23}$$

Since $z_k \in [0,1]$ the contribution of the same thread in $\xi(U)$ will be greater (if $z_k \in [0,1)$) or equal (if $z_k = 1$) to the contribution of thread T_k in the objective function.

□

This concludes the proof for Theorem 4.1.1. □

A MIP Model for $\alpha > 0.5$

Previously, we solved the ECP for $\alpha \in [0, 0.5]$. Solving the problem for $\alpha \in [0,1]$ requires the definition of additional variables and constraints.

$$\text{maximize} \quad \sum_{T_k \in \mathcal{T}_C,\, C \in \hat{\mathcal{C}}} \Big(\sum_{ij \in E_k^+} x_{ij}^k - \sum_{ij \in E_k^-} x_{ij}^k \Big) \tag{4.24}$$

subject to

$$x_{ij}^k \leq y_i \qquad \forall ij \in E_k \tag{4.25}$$

$$x_{ij}^k \leq y_j \qquad \forall ij \in E_k \tag{4.26}$$

$$x_{ij}^k \leq z_k \qquad \forall ij \in E_k \tag{4.27}$$

$$x_{ij}^k \geq -2 + y_i + y_j + z_k \qquad \forall ij \in E_k \tag{4.28}$$

$$-N_k z_k < \sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k \leq M_k(1 - z_k) \qquad \forall T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} \tag{4.29}$$

$$a_{ij}^k \geq -1 + y_i + y_j \qquad \forall ij \in E_k \tag{4.30}$$

$$a_{ij}^k \leq y_i \qquad \forall ij \in E_k \tag{4.31}$$

$$a_{ij}^k \leq y_j \qquad \forall ij \in E_k \tag{4.32}$$

$$0 \leq a_{ij}^k \leq 1 \qquad \forall ij \in E_k \tag{4.33}$$

$$y_i \in \{0,1\} \qquad \forall i \in V \tag{4.34}$$

$$0 \leq x_{ij}^k \leq 1 \qquad \forall ij \in E_k \tag{4.35}$$

$$z_k \in \{0,1\} \qquad \forall T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} \tag{4.36}$$

Where are N_k and M_k are constants of value $\alpha(|E_k^+| + 1)$ and $(1 - \alpha)(|E_k^-| + 1)$, respectively.

Note also that (4.29) involves a strict inequality, which is not allowed by the definition of MIP or LP. However, this constraint can easily be transformed into a valid and equivalent formulation by the means of a small $\epsilon > 0$ (generally we can choose $\epsilon < \alpha \cdot 10^{-10}$), so that it becomes

$$-N_k z_k \leq \sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k - \epsilon \leq M_k(1 - z_k) \qquad \forall T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}}.$$

We will however use (4.29) for simplifying notation.

This problem requires the introduction of variables a_{ij}^k, associated to edges (4.33), which generally take value one if they are associated to an edge induced by the set of vertices considered as solution (i.e. y_i with value one). Note that, in order to contribute to the score, x_{ij}^k also require that the corresponding thread $T_k \in S_C(U)$. The other variables have the same meaning as in MIP (4.1)-(4.9).

Theorem 4.1.4. *Let $G = \{G_k = (V, E_k)\}_k$ be an* Interaction Graph *and $\alpha \in [0,1]$. Then*

$$\max_{U \subseteq V} \xi(U) = OPT(MIP) \tag{4.37}$$

where $OPT(MIP)$ denotes the optimal solution to MIP (4.24)-(4.36).

Proof. We will prove the theorem by showing that $LHS \geq RHS$ and that $LHS \leq RHS$.

Claim 4.1.5. *For any $U \subseteq V$, the MIP* (4.24)-(4.36) *gets value $\geq \xi(U)$.*

It is easy to see that by choosing x_{ij}^k, y_i, z_k as in Claim 4.1.2 and

$$a_{ij}^k = \begin{cases} 1 & \text{if } e_{ij} \in E_k[U] \\ 0 & \text{otherwise} \end{cases}$$

all the constraints of the new formulation are satisfied and Claim 4.1.5 is consequently proved for MIP (4.24)-(4.36).

We will instead focus on proving the analogous of Claim 4.1.3.

Claim 4.1.6. *Given a feasible solution of MIP* (4.24)-(4.36) *with value v we can construct U s.t. $\xi(U) \geq v$ for any α.*

Proof. Let $U \coloneqq \{v_i \ s.t. \ y_i = 1\}$. We will not prove some results in Claim 4.1.3 which still hold.

Due to (4.30)-(4.32) we have that

$$a_{ij}^k = 1 \iff y_i = 1 \land y_j = 1 \tag{4.38}$$

i.e. all and only the edges induced by U will have the corresponding $a_{ij}^k = 1$. Therefore,

$$|E_k^-[U]| = \sum_{ij \in E_k^-} a_{ij}^k, \qquad\qquad |E_k[U]| = \sum_{ij \in E_k} a_{ij}^k. \tag{4.39}$$

Consider now a thread $T_k \in \mathcal{S}_C(U)$. By definition, we have that

$$\eta(T_k) \leq \alpha \implies \frac{|E_k^-[U]|}{|E_k[U]|} \leq \alpha \tag{4.40}$$

$$\implies |E_k^-[U]| - \alpha \cdot |E_k[U]| \leq 0. \tag{4.41}$$

Now suppose $z_k = 0$. This means that (4.29) resolves to

$$0 < \sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k \leq M_k$$

Which is not satisfied due to (4.41) and (4.39), since they imply that $\sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k \leq 0$. This justifies (4.29).

Thus, $z_k = 1$. This means that the constraint resolves to

$$-N_k = -\alpha(|E_k^+[U]| + 1) < \sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k$$

which is satisfied since

$$\begin{aligned}\sum_{ij\in E_k^-} a_{ij}^k - \alpha \sum_{ij\in E_k} a_{ij}^k =& |E_k^-[U]| - \alpha \cdot |E_k[U]| && (4.42)\\ =&(1-\alpha)|E_k^-[U]| - \alpha \cdot |E_k^+[U]| && (4.43)\\ \geq& -\alpha \cdot |E_k^+[U]| && (4.44)\\ >& -N_k. && (4.45)\end{aligned}$$

Consequently, because of (4.17), we have that

$$T_k \in \mathcal{S}_C(U) \iff x_{ij}^k = 1 \;\forall\, i,j \in U \tag{4.46}$$

meaning that each *non-controversial* T_k will contribute to ξ and v with the same score, i.e.

$$\sum_{ij\in E_k^+} x_{ij}^k - \sum_{ij\in E_k^-} x_{ij}^k = |E_k^-[U]| - |E_k[U]|.$$

Now consider T_k *controversial* and suppose $z_k = 1$. This means that (4.29) resolves to

$$-N_k < \sum_{ij\in E_k^-} a_{ij}^k - \alpha \sum_{ij\in E_k} a_{ij}^k \leq 0,$$

where the second inequality is not satisfied due to (4.39) and by definition of *controversial.* So z_k must be 0. Then, we have

$$\begin{aligned}\sum_{ij\in E_k^-} a_{ij}^k - \alpha \sum_{ij\in E_k} a_{ij}^k =& |E_k^-[U]| - \alpha \cdot |E_k[U]|\\ =&(1-\alpha)|E_k^-[U]| - \alpha \cdot |E_k^+[U]|\\ \leq&(1-\alpha)\cdot|E_k^-[U]|\\ \leq& M_k,\end{aligned}$$

thus satisfying (4.29). Therefore, $x_{ij}^k = 0 \;\forall\, i,j,\in V$ (because of (4.27)). This means that the contribution of a controversial T_k is the same in ξ and v and, in general, $\xi(U) = v$, proving the claim. □

Due to Claims 4.1.5 and 4.1.6 the theorem is proved. □

4.1.2 A MIP Model for the D-ECP

Similarly to the ECP, here we propose a MIP model for finding a solution for the D-ECP, $\alpha \in [0,1]$.

For simplifying notation we define $E_k := E(T_k), T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}}$.

$$\text{maximize} \sum_{T_k\in\mathcal{T}_C,\, C\in\hat{\mathcal{C}}} \Big(\sum_{ij\in E_k^+} x_{ij}^k - \sum_{ij\in E_k^-} x_{ij}^k \Big) \tag{4.47}$$

subject to

$$a_{ij}^k \leq b_i \qquad \forall ij \in E_k \tag{4.48}$$
$$a_{ij}^k \leq b_j \qquad \forall ij \in E_k \tag{4.49}$$
$$a_{ij}^k \geq -1 + b_i + b_j \qquad \forall ij \in E_k \tag{4.50}$$
$$-N_k z_k < \sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k \leq M_k(1 - z_k) \qquad \forall T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} \tag{4.51}$$
$$x_{ij}^k \leq y_i \qquad \forall ij \in E_k \tag{4.52}$$
$$x_{ij}^k \leq y_j \qquad \forall ij \in E_k \tag{4.53}$$
$$\sum_{i \in V} y_i = 1 \tag{4.54}$$
$$y_i \leq b_i \qquad \forall i \in V \tag{4.55}$$
$$y_i \geq -1 + b_i + y_j \qquad \forall i, j \in V \tag{4.56}$$
$$x_{ij}^k \geq -2 + a_{ij}^k + z_k + y_i \qquad \forall ij \in E_k \tag{4.57}$$
$$x_{ij}^k \geq -2 + a_{ij}^k + z_k + y_j \qquad \forall ij \in E_k \tag{4.58}$$
$$x_{ij}^k \leq a_{ij}^k \qquad \forall ij \in E_k \tag{4.59}$$
$$x_{ij}^k \leq z_k \qquad \forall T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} \tag{4.60}$$
$$a_{ij}^k \in \{0, 1\} \qquad \forall ij \in E_k \tag{4.61}$$
$$b_i \in \{0, 1\} \qquad \forall i \in V \tag{4.62}$$
$$y_i \geq 0 \qquad \forall i \in V \tag{4.63}$$
$$x_{ij}^k \geq 0 \qquad \forall ij \in E_k \tag{4.64}$$
$$z_k \in \{0, 1\} \qquad \forall T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} \tag{4.65}$$

Where are N_k and M_k are constants of value $\alpha(|E_k^+| + 1)$ and $(1 - \alpha)(|E_k^-| + 1)$, respectively. Again the strict inequality of (4.51) can be transformed into a valid constraint as explained in Subsection 4.1.1.

Theorem 4.1.7. *Let $G = \{G_k = (V, E_k)\}_k$ be an* Interaction Graph *and $\alpha \in [0, 1]$.*

$$\max_{U \subseteq V} \psi(U) = OPT\text{(MIP)} \tag{4.66}$$

where $OPT(MIP)$ denotes the optimal solution to MIP (4.47)-(4.65)*.*

Proof. Similarly to Theorem 4.1.1 we will prove this equality by 2 inequalities: $RHS \geq LHS$ and $LHS \geq RHS$

Claim 4.1.8. *For any $U \subseteq V$, the MIP* (4.47)-(4.65) *gets value at least $\psi(U)$.*

Proof. Let $c = 1/|U|$. We construct a feasible solution for MIP (4.47)-(4.65) as follows:

$$y_i = \begin{cases} c, & \text{if } v_i \in U, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall v_i \in U \tag{4.67}$$

$$b_i = \begin{cases} 1, & \text{if } v_i \in U, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall v_i \in U \tag{4.68}$$

$$z_k = \begin{cases} 1, & \text{if } T_k \in \mathcal{S}_C(U),\ C \in \hat{\mathcal{C}}, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} \tag{4.69}$$

$$a_{ij}^k = \begin{cases} 1, & \text{if } e_{ij} \in E_k[U],\ T_k \in \mathcal{T}_C,\ C \in \hat{\mathcal{C}}, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall e_{ij} \in E_k, T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} \tag{4.70}$$

$$x_{ij}^k = \begin{cases} c, & \text{if } e_{ij} \in E_k[U],\ T_k \in \mathcal{S}_C(U),\ C \in \hat{\mathcal{C}}, \\ 0, & \text{otherwise.} \end{cases} \qquad \forall e_{ij} \in E_k, T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}} \tag{4.71}$$

(4.48)-(4.50) are easily satisfied since $a_{ij}^k = 1 \iff b_i = 1 \wedge b_j = 1$, meaning that an edge is induced (thus $a_{ij}^k = 1$) if and only if $v_i, v_j \in U$. The same idea applies to (4.52)-(4.53). (4.55) is trivial and (4.59)-(4.60) hold since $\mathcal{S}_C(U) \subseteq \mathcal{T}_C$. It is also easy to see that (4.56) is satisfied since we defined y_i and b_i s.t. $b_i = 1$ if and only if $y_i = c$. Furthermore, since $y_i = c$ if and only if $y_i \in U$ then

$$\sum_{i \in V} y_i = \sum_{i \in U} c = 1$$

and (4.54) is also satisfied.

Let us know consider $T_k \in \mathcal{S}_C(U)$ which by definition implies $z_k = 1$. Then,

$$\eta(T_k[U]) \leq \alpha \implies \frac{|E_k^-[U]|}{|E_k[U]|} \leq \alpha \tag{4.72}$$

$$\implies |E_k^-[U]| - \alpha(|E_k[U]|) \leq 0. \tag{4.73}$$

Thus, due to the definition of a_{ij}^k, the second inequality of (4.51) is true; the first one is also satisfied since

$$\sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k = |E_k^-[U]| - \alpha \cdot |E_k[U]| \tag{4.74}$$

$$= (1-\alpha)|E_k^-[U]| - \alpha \cdot |E_k^+[U]| \tag{4.75}$$

$$\geq -\alpha \cdot |E_k^+[U]| \tag{4.76}$$

$$> N_k. \tag{4.77}$$

(4.57)-(4.58) are true since in this case (for $z_k = 1$) by definition $a_{ij}^k = 1$ implies $x_{ij}^k = c$. If instead $T_k \notin S_c$ and $z_k = 0$ we have that

$$\eta(T_k[U]) > \alpha \implies \frac{|E_k^-[U]|}{|E_k[U]|} > \alpha \tag{4.78}$$

$$\implies |E_k^-[U]| - \alpha(|E_k[U]|) > 0 \tag{4.79}$$

and consequently the first inequality of (4.51) is satisfied. Also

$$\sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k = |E_k^-[U]| - \alpha \cdot |E_k[U]| \tag{4.80}$$

$$= (1-\alpha)|E_k^-[U]| - \alpha \cdot |E_k^+[U]| \tag{4.81}$$

$$\leq (1-\alpha) \cdot |E_k^-[U]| \tag{4.82}$$

$$< M_k. \tag{4.83}$$

Thus, also the second inequality is true. Also, (4.57)-(4.58) are trivially satisfied.

Consequently an edge contributing to $\psi(U)$ will also count in the objective function by c. So, for a given thread $T_k \in \mathcal{S}_C, C \in \hat{\mathcal{C}}$

$$\begin{aligned}\sum_{ij \in E_k^+} x_{ij}^k - \sum_{ij \in E^-(T_k)} x_{ij}^k &= \sum_{ij \in E_k^+[U]} c - \sum_{ij \in E_k^-[U]} c \\ &= c(|E_k^+[U]| - |E_k^-[U]|) \\ &= \frac{|E_k^+[U]| - |E_k^-[U]|}{|U|}.\end{aligned}$$

Thus, for each thread we have the same contribution to both $\psi(U)$ and the objective function of MIP (4.47)-(4.65). Therefore, the sum through all the threads will correspond as well. □

Claim 4.1.9. *Given a feasible solution of MIP* (4.47)-(4.65) *with value v we can construct U s.t. $\psi(U) \geq v$.*

Proof. Let us define $U \coloneqq \{v_i \ s.t. \ y_i \neq 0\}$; consider v_i s.t. $y_i \neq 0$ (if no such vertex exists then the proof is trivial) and let $c \coloneqq y_i$. By (4.56) and (4.55) we have that

$$\forall v_j \in V, \ y_j \in \{0, c\}, \tag{4.84}$$
$$\forall i, j \in V, \ b_i = 1 \wedge b_j = 1 \implies y_i = y_j = c. \tag{4.85}$$

Furthermore, due to (4.48)-(4.50) we have that

$$a_{ij}^k = 1 \iff b_i = 1 \wedge b_j = 1. \tag{4.86}$$

Now consider some $x_{ij}^k > 0$. By (4.59) $x_{ij}^k > 0$ implies $a_{ij}^k = 1$ and, thanks to (4.60), $x_{ij}^k > 0$ implies $z_k = 1$. Thus, combining this to the results in (4.84) and (4.86)

$$x_{ij}^k > 0 \implies a_{ij}^k = 1 \wedge z_k = 1, \tag{4.87}$$
$$z_k = 1 \implies x_{i'j'}^k = c, \ \forall i', j' \in U, \tag{4.88}$$
$$z_k = 1 \wedge a_{ij}^k = 1 \implies x_{ij}^k = c. \tag{4.89}$$

This means that if exists $x_{ij}^k > 0$ then all the variables $x_{i'j'}^k$ associated to edges induced by U have value c. Also, since we have that $z_k = 1$, (4.51) will correspond to

$$\sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k \leq 0. \tag{4.90}$$

We showed in (4.87) and (4.89) that $a_{ij}^k = 1 \wedge z_k = 1$ if and only if $x_{ij}^k > 0$. In other other words, the edges contributing to the objective function are part of a *non-controversial* subgraph, i.e. the corresponding thread $T_k \in \mathcal{S}_C(U)$ and it will contribute to $\psi(U)$.

Now suppose exists $T_k \in \mathcal{S}_C(U), \ C \in \hat{\mathcal{C}}, \ s.t. \ z_k = 0$. By definition of $\mathcal{S}_C(U)$ we have that

$$\eta(T_k[U]) \leq \alpha \implies |E_k^-[U]| - \alpha |E_k[U]| \tag{4.91}$$
$$\implies \sum_{ij \in E_k^-} a_{ij}^k - \alpha \sum_{ij \in E_k} a_{ij}^k \leq 0, \tag{4.92}$$

because $a_{ij}^k = 1$ for all edges induced by U. But, if $z_k = 0$ then constraint (4.51) is violated, and this would be a *contradiction*. So no such T_k exists and $T_k \in \mathcal{S}_C(U) \iff z_k = 1$.

This means that a thread contributing to the objective function of MIP (4.47)-(4.65) also counts towards $\psi(U)$. Due to (4.88) we can then write, for $T_k \in \mathcal{S}_C(U)$

$$\sum_{ij\in E_k^+} x_{ij}^k - \sum_{ij\in E^-(T_k)} x_{ij}^k = \sum_{ij\in E_k^+[U]} c - \sum_{ij\in E_k^-[U]} c$$
$$= c(|E_k^+[U]| - |E_k^-[U]|)$$
$$= \frac{|E_k^+[U]| - |E_k^-[U]|}{|U|}.$$

Thus, each threads equally contributes to $\psi(U)$ and the objective function of MIP (4.47)-(4.65). Therefore, $\psi(U) \geq v$. □

This concludes the proof of the theorem.

□

4.2 Heuristics

We now present some heuristic algorithms for solving the ECP and D-ECP. We start by describing how $\xi(U)$ and $\psi(U)$ can be computed in practice.

Let $\textsc{Score}_\xi(U)$ and $\textsc{Score}_\psi(U)$ be the functions computing the *Echo Chamber Score* and *Densest-Echo Chamber Score* of U, respectively. These subroutines iterate over the edges of the vertices in U, ignoring those that are not induced by U, and counting for each thread $T \in \mathcal{T}_C, C \in \hat{\mathcal{C}}$ the number of edges and negative edges to see which are *controversial*, then calculating their contributions (Algorithm 4.1 shows in detail $\textsc{Score}_\xi$; $\textsc{Score}_\psi$ can simply be computed as $\textsc{Score}_\xi(U)/|U|$). Note that this algorithm operates also on the weights w_{ij}^k of the edges. This is something that is "implicitely" done in the MIPs, as the sums iterate over positive and negative edges.

We present our algorithms focusing on ECP in detail. They can generally be adapted for solving the D-ECP by replacing calls to $\textsc{Score}_\xi$ with $\textsc{Score}_\psi$.

4.2.1 The β-Algorithm

The β-algorithm is an heuristic for the ECP. The β-algorithm (Algorithm 4.2) constructs a set of users U by iteratively adding the node which increases the most the score or removing from U the one which contributes the least, stopping when the score cannot be increased by adding a node. The frequencies of addition and removal are regulated through a parameter $\beta \in [0, 1]$ (for smaller values a higher density is to be expected, generally).

U is initialized by sampling one node from the graph. There are two possible approaches in doing that: one is uniformly; the other is using probabilities proportional to the number of positive edges each node has.

Algorithm 4.1: The $\textsc{Score}_\xi$ subroutine

```
Input: Interaction graph G = {G_k = (V, E_k)}_k, a set of users U ⊆ V,
       α ∈ [0, 1]
Result: ξ(U)
N⁺(T) ← 0, N⁻(T) ← 0  for all threads T ∈ 𝒯_C, C ∈ Ĉ ;
foreach v_i ∈ U do
    S_i ← edges starting from v_i ;
    foreach e_ij ∈ S_i if v_j ∈ U do
        T_ij ← thread of e_ij ;
        w_ij ← weight of e_ij ;
        if w_ij ≥ 0 then
            N⁺(T_ij) ← N⁺(T_ij) + 1 ;
        else
            N⁻(T_ij) ← N⁻(T_ij) + 1 ;
        end
    end
end
ξ(U) ← 0 ;
η(T) ← N⁻(T) / (N⁻(T) + N⁺(T))  for all threads T ∈ 𝒯_C, C ∈ Ĉ ;
foreach T ∈ 𝒯_C, C ∈ Ĉ if η(T) ≤ α do
    ξ(U) ← ξ(U) + N⁺(T) − N⁻(T)
end
return ξ(U) ;
```

Algorithm 4.2: β-algorithm

```
U = { a single random node };
ξ(U) = 0 ;
while ∃ v_j s.t. SCORE_ξ(U ∪ {v_j}) > ξ(U) do
    N(U) ← neighbours of vertices in U in the graph G ;
    Flip a coin which gives head with probability β ;
    If head   U ← U ∪ {arg max_{v_j ∈ N(U)} Score_ξ(U ∪ {v_j})} ;
    else   U ← U \ {arg max_{v_j ∈ U} Score_ξ(U \ {v_j})} ;
end
return Score_ξ(U) ;
```

In addition, one may also want to ignore a node when it is removed for the next iterations, in order to prevent the algorithm from repeatedly adding and taking out from U the same vertex.

The result is clearly dependent on the choice of the initial node. For this reason the process should be repeated for different initial nodes.

One of the limitations of this approach is that the algorithm will only find sets of users that are connected in the original graph. This is due to the fact that it will never add a node which is not connected to any of the vertices in U, as it produces an increase of the score equal to 0.

4.2.2 Peeling Algorithm

Inspired to the greedy algorithm proposed in [36], the peeling algorithm starts by considering a set $U = V$, iteratively removing the worst nodes (Algorithm 4.3).

Algorithm 4.3: Peeling algorithm

$U = V$;
$S = \textsc{Score}_\xi(U)$;
while $U \neq \emptyset$ **do**
  $v = \arg\max_{v_j \in U} \textsc{Score}_\xi(U \setminus \{v_j\})$;
  $U \leftarrow U \setminus \{v\}$;
  $S_i = \textsc{Score}_\xi(U)$;
end
return $\arg\max_i S_i$;

If many nodes produce the same score, then one of them is randomly selected (or, alternatively, the one which has the highest fraction of negative edges).

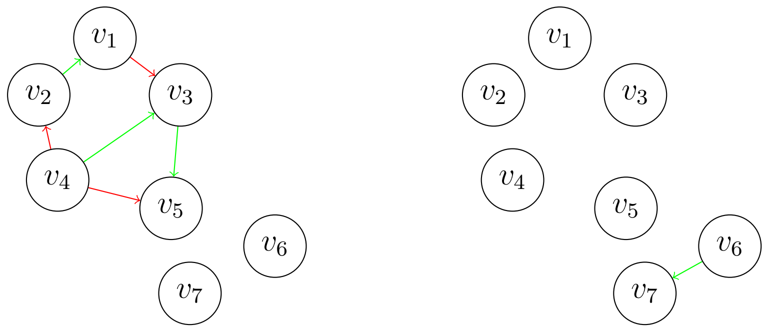
4.2.3 Rounding Algorithm

This algorithm reconstructs a solution starting from the results of the relaxation of the exact models and is again inspired by the algorithm for reconstructing the exact solution from the LP model in [36].

More specifically, our relaxation of MIP (4.24)-(4.36) replaces constraints (4.34) and (4.36) with

$$0 \leq y_i \leq 1, \tag{4.93}$$
$$0 \leq z_i \leq 1. \tag{4.94}$$

We now have to solve an LP problem.



(a) T_1 **(b)** T_2

Figure 4.1: Example original *Interaction Graph G*

Let $\tilde{E}$ be the sequence of edges ordered in descending order by x_{ij}^k. The algorithm (Algorithm 4.4) iterates over the edges in $\tilde{E}$, adding them to a *dummy* graph $\hat{G}$, also eventually adding incident nodes if not already present. At each iteration it computes the score of the vertices in the graph $\hat{G}$ and the score of the vertices of each component in the graph, keeping track of the best result.

Algorithm 4.4: Rounding algorithm

Solve the relaxation of MIP (4.24)-(4.36) ;
$\hat{G} \leftarrow$ empty graph ;
$\hat{V} \leftarrow$ vertices of $\hat{G}$;
$S = 0$
foreach $e_{ij}^k \in \tilde{E}$ *in descending order of* x_{ij}^k **do**
    $\hat{V} \leftarrow \hat{V} \bigcup \{v_i\}$ **if** $v_i \notin \hat{V}$;
    $\hat{V} \leftarrow \hat{V} \bigcup \{v_j\}$ **if** $v_j \notin \hat{V}$;
    $S \leftarrow \max(S,\ \textsc{Score}_\xi(\hat{V}))$
    foreach *component* C *in* $\hat{G}$ **do**
        $S \leftarrow \max(S,\ \textsc{Score}_\xi(C))$
    end
end
return S ;

The motivation for the algorithm can be seen in Figures 4.1-4.3: the problem relaxation involves a solution whose value assigned to the edges can be used to find subgraphs with many positive edges by using each separate component as set of users U.

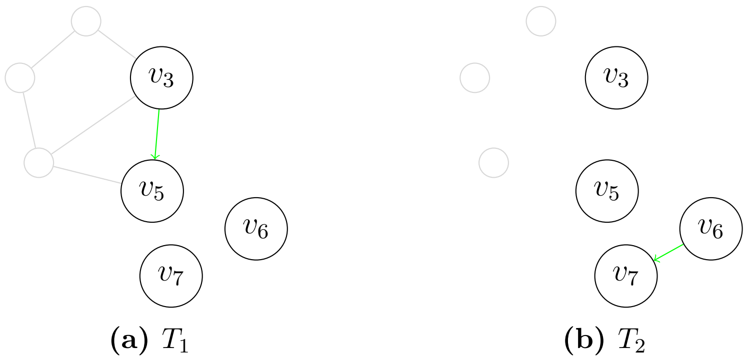


Figure 4.2: Exact solution of the example in Figure 4.1, $\alpha = 0.4$

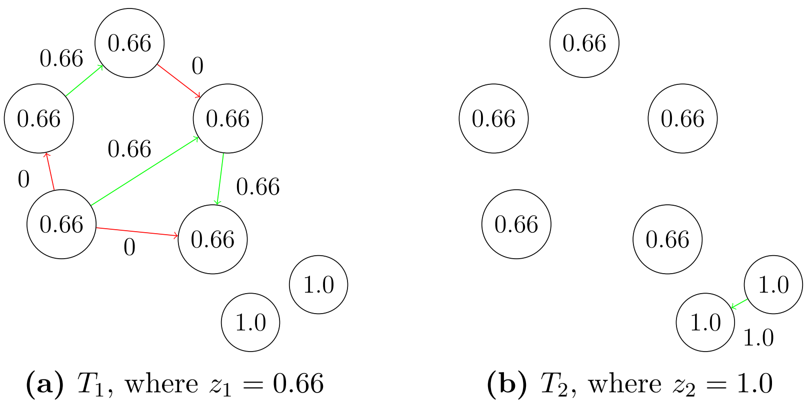


Figure 4.3: Solution of the relaxation of G of Figure 4.1, $\alpha = 0.4$

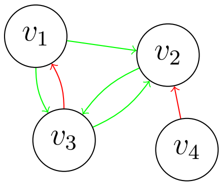


Figure 4.4: Another *Interaction graph* example, with only one thread. In this case the rounding algorithm is able to find the exact solution by selecting all the nodes except for v_4. In the result of the relaxation all the edges except for e_{42} get the value of 1.

While one may think from these examples that the relaxation trivially assigns non-zero values only to positive edges, Figure 4.4 shows a case in which a negative edge, e_{31}, gets the value of 1. Furthermore, in this example the algorithm is able to reconstruct the exact solution of the problem.

4.3 Alternative Formulations

Due to the intrinsic complexity of the problems (Chapter 3) we define variants of the ECP and D-ECP problems, for some of which we are also able to find an exact solution.

For these new problems we need to define new graphs, obtained by preprocessing the *interaction graph*.

4.3.1 The Pair Aggregated Graph

Let $G = \{G_k = (V, E_k)\}_k$ be the *interaction graph*, let E_k and E_k^- denote the edges and negative edges in thread T_k, respectively. We define $\delta(v_i, v_j)$ and $\delta^-(v_i, v_j)$ to be the sum of the edges and negative edges, respectively, associated to controversial contents between vertices v_i and v_j, i.e.

$$\delta(v_i, v_j) = \sum_{T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}}} \Big(\sum_{e_{ij} \in E_k} w_{ij} + \sum_{e_{ji} \in E_k} w_{ji} \Big), \tag{4.95}$$

$$\delta^-(v_i, v_j) = \sum_{T_k \in \mathcal{T}_C, C \in \hat{\mathcal{C}}} \Big(\sum_{e_{ij} \in E_k^-} w_{ij} + \sum_{e_{ji} \in E_k^-} w_{ji} \Big). \tag{4.96}$$

The Pair Aggregated (PA) graph $G_P = (V_P, E_P)$ is constructed as follows from G:

- For any vertex $v_i \in V$ add a corresponding vertex in V_P.

- For any pair of vertices v_i, v_j in G let $\eta(v_i, v_j) \coloneqq \frac{\delta^-(v_i,v_j)}{\delta(v_i,v_j)}$. If $\eta(v_i, v_j) \leq \alpha$, add a positive edge between v_i and v_j in G_P. If, instead, $\eta(v_i, v_j) > \alpha$ or $\delta(v_i, v_j) = 0$ then don't add any edge between the two vertices.

The problem then is finding the Densest Subgraph of G_P, i.e., if $E_P[U]$ is the set of edges induced on G_P by $U \subseteq V$, finding U maximizing

$$\xi(U) = \frac{|E_P[U]|}{|U|}. \tag{4.97}$$

4.3.2 The Thread Pair Aggregated Graph

Differently from the previous method, in this case edges are aggregated separately for each thread.

More specifically, given an *interaction graph* $G = \{G_k = (V, E_k)\}_k$, let $E(T)$ and $E^-(T)$ denote the edges and negative edges in thread T, respectively. We define $\delta_T(v_i, v_j)$ and $\delta_T^-(v_i, v_j)$ to be the sum of the edges and negative edges, respectively, associated to thread T between vertices v_i and v_j, being $T \in \mathcal{T}_C, C \in \hat{\mathcal{C}}$, i.e.

$$\delta_T(v_i, v_j) = \sum_{e_{ij} \in E(T)} w_{ij} + \sum_{e_{ji} \in E(T)} w_{ji}, \tag{4.98}$$

$$\delta_T^-(v_i, v_j) = \sum_{e_{ij} \in E^-(T)} w_{ij} + \sum_{e_{ji} \in E^-(T)} w_{ji}. \tag{4.99}$$

We will produce a graph, the Thread Pair Aggregated (TPA) Graph

$$G_{TP} = (V_{TP}, E_{TP}),$$

that, differently from the PA Graph, is a multiplex graph (each layer representing a thread). The construction of the TPA Graph is as follows:

- For any vertex $v_i \in V$ add a corresponding vertex in V_P.
- For any thread $T \in \mathcal{T}_C, C \in \hat{\mathcal{C}}$ we add a layer T to G_{TP}.
- For any thread $T \in \mathcal{T}_C, C \in \hat{\mathcal{C}}$ and pair of vertices v_i, v_j in G, let $\eta_T(v_i, v_j) \coloneqq \frac{\delta_T^-(v_i,v_j)}{\delta_T(v_i,v_j)}$. If $\eta_T(v_i, v_j) \leq \alpha$ add a positive edge between v_i and v_j in G_{TP}, in the layer associated to thread T.

We can then solve on G_{TP}:

1. The Densest Subgraph Problem (or, equivalently, the DCS-MM), which we will refer to as the Densest Thread Pair Aggregated (D-TPA) Problem (Subsection 2.3.1).

2. The O^2Bff Problem, more specifically O^2Bff-AM, which we will denote as O^2Bff Thread Pair Aggregated (O^2Bff-TPA) Problem (Subsection 2.3.3).

Chapter 5

Experiments and Discussion

In this chapter we present how data is retrieved from social media, what are the models we use for generating synthetic data and discuss the results we obtain by running the methods presented in Chapter 4.

5.1 Data Collection and Generation

We now define techniques for generating synthetic data and present how real-world data is retrieved and preprocessed.

5.1.1 Synthetic Data

Here we propose two possible methods for generating data, the Signed SBM and the Information spread model. Given $\tau \in \mathbb{N}$, both of these methods randomly generate interaction graphs with τ threads/layers. Depending on the method, they will require more parameters.

Signed SBM

This model is very similar to the Stochastic Block Model (SBM), a model commonly used for generating random graphs having some community structures [13].

The Signed SBM is based on the following parameters:

- $k \in \mathbb{N}$, the number of communities.
- $b_i \in [k]$, the group assignment of each vertex i.
- $\omega^+_{rs} \in [0, 1]$ and $\omega^-_{rs} \in [0, 1]$, the probabilities of positive and negative edges, respectively, between users in group r and s. Vertices have also a probability

of not having an edge, which is equal to $1 - \omega_{rs}^- - \omega_{rs}^+$. For this reason it is needed that $\omega_{rs}^+ + \omega_{rs}^- \leq 1$.

- $\theta \in [0, 1]$, controlling the reduction of the probability of interacting between *inactive* communities: for the generation of each thread we will distinguish between *active* and *inactive* communities, having different probabilities of interacting.

- $\hat{k} \in \mathbb{N}$, the number of communities active in a thread.

During the generation process, we will sample the edges from a categorical distribution with three parameters which we will denote as

$$\Omega = (\Omega^+, \Omega^-, \Omega^0).$$

Here, Ω^+ and Ω^- are the probabilities of adding a positive and negative edges, respectively, while Ω^0 is the probability of not adding any edge. Note that since this is a probability distribution, we will have $\Omega^+, \Omega^-, \Omega^0 \geq 0$ and $\Omega^+ + \Omega^- + \Omega^0 = 1$.

Therefore, generating a thread layer T[1] for an interaction graph involves the following steps:

1. Sample uniformly $\hat{k}$ of the k communities. These are the *active* communities in the thread. The remaining communities are *inactive*.

2. For each node pair i, j consider their corresponding groups r and s and, if both communities are *active*, draw from

$$\Omega = (\omega_{rs}^+, \omega_{rs}^-, 1 - \omega_{rs}^+ - \omega_{rs}^-).$$

   Otherwise, if at least one of the two communities is not *active*, the distribution becomes

$$\Omega = (\theta\omega_{rs}^+, \theta\omega_{rs}^-, 1 - \theta(\omega_{rs}^+ + \omega_{rs}^-)).$$

   Then, we possibly add the edge to thread T.

Information Spread Model

Here we describe the Information spread model, which aims at simulating the process of information flowing between different users of a social network.

Like in the Signed SBM, each node has a group assignment $b_i \in [k]$ and each pair of groups has probabilities of positive and negative edges (ω_{rs}^+ and ω_{rs}^-, respectively, with $\omega_{rs}^- + \omega_{rs}^+ \leq 1$). Additionally, we have the following new parameters:

[1]In this model we will generate contents uniquely associated to threads.

- $\{\phi_{rs}\}$, the edge probabilities of a standard SBM. A standard SBM is a model for generating undirected and unweighted graphs with community-like structures. It takes as parameters the number of community k, a group assignment for each node (we will use b_i) and the probability of edge between each pair of communities (exactly $\{\phi_{rs}\}$). Then, for each pair of nodes v_i and v_j belonging to communities r and s, respectively, it adds an edge between v_i and v_j with probability ϕ_{rs}.

  This model is used for generating a graph G_f, which we will call the *friend* graph, representing the friendship relationships between the users. We will refer to neighbors in this graph as *friends*.

- β_a, the probability that a node is initially activated: we will distinguish between *active* and *inactive* nodes, that will have different probabilities of interacting.

- β_n, the probability that an *inactive* node is activated from an *active* friend.

After generating G_f from an SBM with parameters $\{\phi_{rs}\}$, the generation of each thread of an *interaction graph* goes as follows:

1. Initialize all nodes as *inactive*.

2. Activate each vertex with probability β_a.

3. Active nodes activate their inactive friends with probability β_n. This step is repeated each time a new user is activated, until the network becomes stable.

4. Similarly to the Signed SBM, if two nodes are both active, draw from

   $$\Omega = (\omega^+_{rs}, \omega^-_{rs}, 1 - \omega^+_{rs} - \omega^-_{rs})$$

   for adding a positive, negative or no edge. If, instead, at least one of them is not active, draw from

   $$\Omega = (\theta\omega^+_{rs}, \theta\omega^-_{rs}, 1 - \theta(\omega^+_{rs} + \omega^-_{rs})).$$

5.1.2 Collection and Preprocessing

Datasets are built over two social medias: Twitter[2] and Reddit[3]; the data collection process, consequently, slightly differs between them.

[2] `twitter.com`

[3] `www.reddit.com`

Personal details	
Born	October 13, 1989 (age 31) New York City, U.S.
Political party	Democratic
Domestic partner	Riley Roberts
Education	Boston University (BA)
Signature	
Website	House website

Figure 5.1: Wikipedia entry associated to Alexandria Ocasio-Cortez, a member of the U.S. House of Representatives.

Twitter. Interaction graphs from Twitter are mainly built starting from the tweets of profiles associated to well-known news sources, like The New York Times or Fox News, that typically post links to their articles: the set of shared URLs are the contents $\mathcal{C}$ of the corresponding interaction graph.

Each content $C \in \mathcal{C}$ will be represented just by its URL, e.g.

`https://www.nytimes.com/2021/03/04/us/richard-barnett-pelosi-tantrum.html`.

Then, in order to find all threads related to C, we search Twitter for the content URL to obtain the tweets containing it. Each of these tweets will correspond to a different thread.

For each thread we then construct the tree of replies through a DFS, recursevily fetching users replying to a comment.

In order to validate our methods, we also construct datasets in which users are labeled either as *democrat* or *republican* [4]. This is done by looking at the people a certain user v_i follows: for each account v_j followed by v_i, if v_j is a political representative, then we can retrieve from Wikipedia the party to which v_j belongs to (see, for example, Figure 5.1). Then, user v_i is assigned a label according to the party of the majority of the users v_j this follows.

Twitter data is retrieved with the help of Tweepy [47], a Python library for accessing the Twitter API, which has been patched for using some features available only in the beta of the new v2 Twitter API.

[4] We choose these two labels since the news sources we analyze for this purpose are based in U.S. Also, we think political discussion are a main source of controversial content and so it is an interesting criterion according to which users can be differentiated.

Reddit. Differently from Twitter, Reddit focuses on subreddits, which are pages collecting posts of users about a specific topic (e.g. r/politics, r/economics, ...). This means that in the datasets built from this social media the contents $\mathcal{C}$ is the set of URLs posted on these pages, which, differently from how we fetch data from Twitter, most likely come from different sources.

These posts are in turn *crossposted*, i.e. reposted on other subreddits. Each of these *crossposts* will correspond to another thread.

We also analyzed a very specific case, that of r/asktrumpsupporters. This subreddit is a "Q&A subreddit to understand Trump supporters, their views, and the reasons behind those views. Debates are discouraged" (from its description). We found it interesting as it provides an explicit labeling of the users who, before commenting in any of these posts, must "declare" their side by choosing a *flair*, which is shown next to the username of the person commenting. Three flairs are available: *Trump Supporter*, *Non supporter* and *Undecided*.

The PRAW library is used for retrieving Reddit data [48].

Edge weights assignment. Once the threads interactions are retrieved, they are passed to a state-of-the-art sentiment analyzer which labels them. More specifically, the model used is RoBERTa, adapted and retrained for dealing with Twitter data [49]. The model is made available by the Transformers python library [50].

The model, given a string of text, returns a probability distribution

$$(p_{neg}, p_{neu}, p_{pos})$$

whose parameters represent the probabilities of negative, neutral and positive sentiment. Note that since this is a probability distribution, we will have $p_{neg}, p_{neu}, p_{pos} \geq 0$ and $p_{neg} + p_{neu} + p_{pos} = 1$. Let

$$p_n := p_{neg}, \qquad\qquad p_p := p_{pos} + p_{neu}. \tag{5.1}$$

If $p_p > p_n$ we assign the edge weight p_p, otherwise $-p_n$.

Initial observations on the datasets. Reddit and Twitter are intrinsically different social medias. As mentioned before, Reddit focuses on subreddits, where all the discussions related to a certain topic find their place and most of the users interested in the theme gather. We think that this contributes to create community of users which are more active and more likely to discuss among each other, as they end up looking at the same posts and discussions.

Twitter, instead, is a much less "centralized" social media with a lot of *hubs* (users with many followers) that discuss similar topics but have disjoint communities. Think, as an example, of the Twitter accounts of Joe Biden and Donald Trump. We expect that most of the followers of the first one are not followers of the latter,

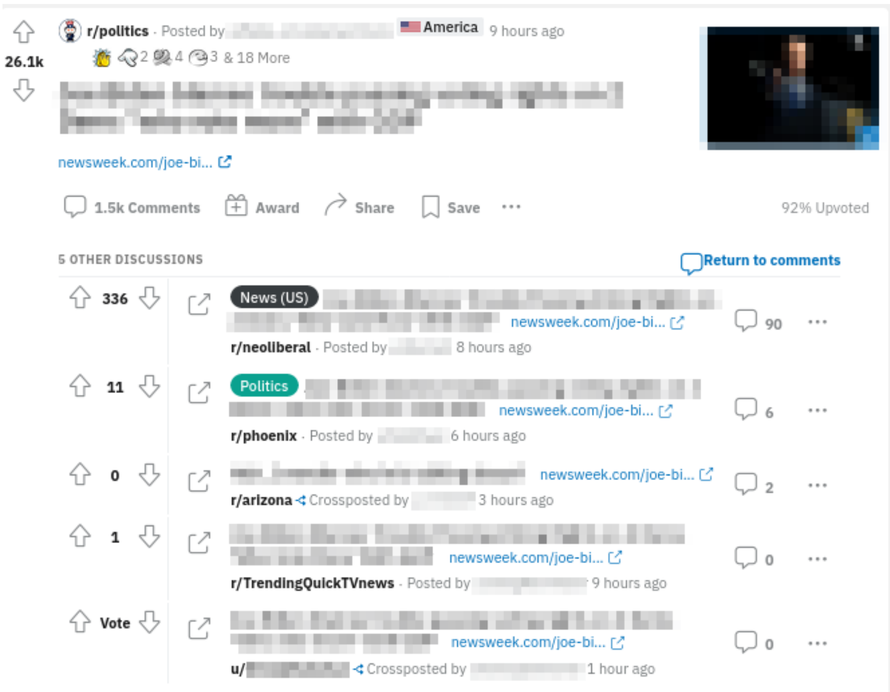


Figure 5.2: The *crossposts* on one of the most discussed article of the day on r/politics.

and vice versa, even if both of them discuss U.S. politics. This means that many users, even if they are interested in the same theme (U.S. politics in our example), will rarely interact with each other.

The "centralization" of Reddit produces, in our data model, contents that are associated with few threads. We observed that even the most discussed articles on r/politics (a subreddit focusing on U.S. politics discussions) are *crossposted* only one to five times (see, for example, Figure 5.2), while on Twitter articles of the New York Times are often shared even 100 times.

5.1.3 A Study on r/asktrumpsupporters

During the research we studied the r/asktrumpsupporters subreddit to understand if it is possible to infer the community[5]of the users by looking at how they react to contents. More specifically, in a highly polarized environment we expect that members in the same community of the author have a positive stance towards the content; conversely, members of the other communities have a negative one.

[5]In this case, we refer to community as the set of users with the same *flair*

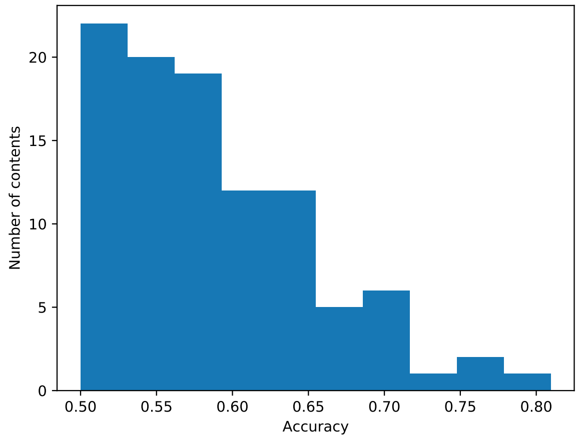


Figure 5.3: This plot shows how the accuracy of the classification process explained in Subsection 5.1.3 is distributed for the different contents $C \in \mathcal{C}$. The graph used for the analysis (built on r/asktrumpsupporters) contains 1850 user vertices and 16470 edges between them. Furthermore, we introduce 71 content nodes and 5773 edges between content and user vertices.

For this analysis we add to our interaction graph another type of vertices, the *content nodes* which we uniquely associate to a content. More formally, for each content $C \in \mathcal{C}$, we add a vertex v_C. In order to add links between users and contents, we consider the sequence of comments leading to a specific user comment and multiply the sign of the associated edges to calculate the sign of the content-user edge.

For example, consider a user v_i replying positively to a post related to content C and user v_j replying negatively to v_i. We will add a positive edge between v_i and v_C and a negative edge between v_j and v_C.

We then assign to the users linked to a content the same label of the author of the post if the user is connected to the content by a positive edge and the opposite one[6]otherwise. Then, we measure the accuracy of this classification.

We show in Figure 5.3 the histogram of the accuracy of classification of the contents in the dataset. We see that in very few cases it is possible to discriminate users better than by just using the majority label (79%), while a big part of the contents achieve an accuracy between 0.5 and 0.6.

We explain this with the following reasons:

- The majority of the posts in the subreddit are open questions, e.g. “What do

[6]We ignore the *Undecided* label

you think of ...' and, similarly, "What's your idea on...". This means that our initial hypothesis that positive and negative stances can be used for inferring the positions of the users is not correct: for this type of posts most of the users will just answer with their opinion, without being either friendly or hostile.

- This community of users is not a representative sample: people attending the subreddit are open to discuss with people of different opinion and for this reason we generally expect less hostility in the comments.

5.2 Experiments

The following presented results have been obtained from a Python implementation of the methods described in the previous chapters. The library used for handling and manipulating graphs is graph-tool, which has been chosen because of its efficiency [51].

5.2.1 Initial Real-World Data Analysis

We did an initial analysis of the data to understand basic properties of the real-world datasets we fetched.

We can gain some insights about the existence of echo chambers by comparing the distribution of $\eta(C)$ and $\eta(T)$ for different interaction graphs. Intuitively, echo chambers may correspond to threads with a high fraction of positive edges. Consequently, given a certain $\eta(C)$ distribution for the interaction graph, in presence of Echo Chambers we expect an increase for low values of $\eta(T)$, when compared to the distribution of $\eta(C)$.

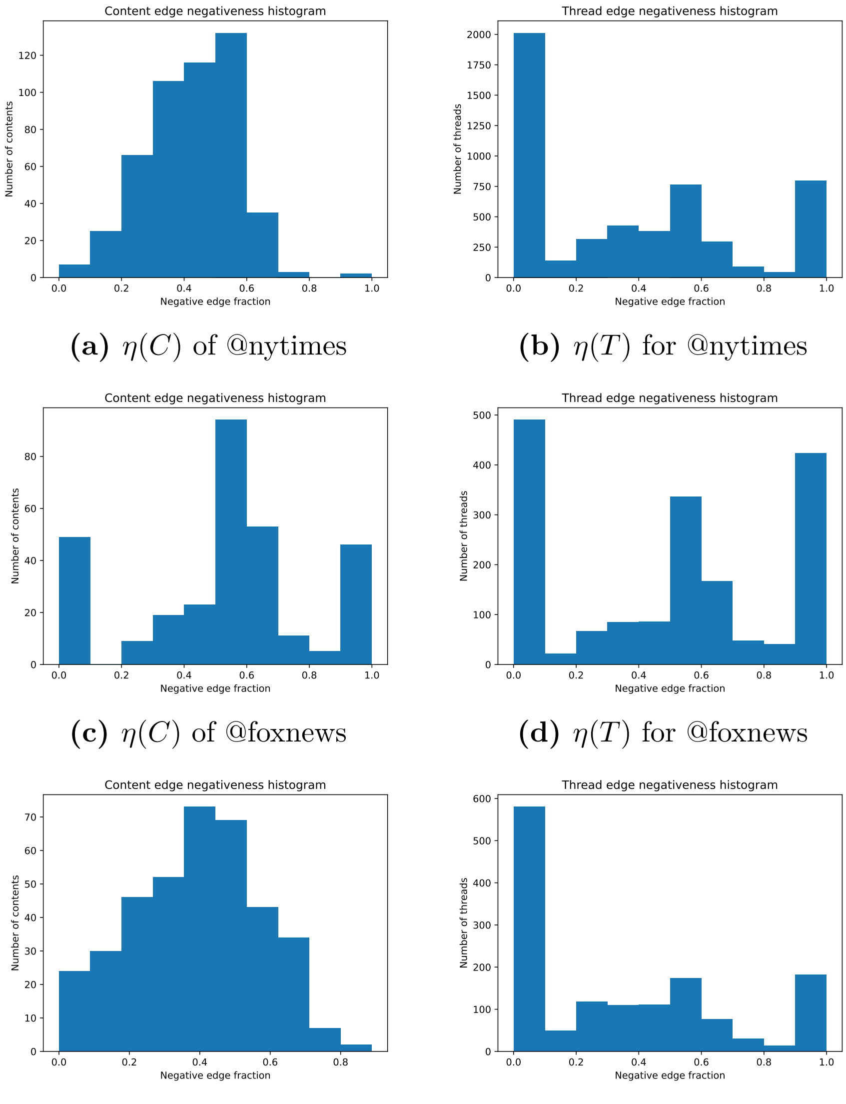
We report these results for three datasets, a first built over @nytimes, a second over @foxnews and a third one over @bbcnews Twitter accounts[7]. The basic statistics of these graphs are listed in Table 5.1. Histograms, obtained by distributing values in 10 equal-sized buckets, are shown in Figure 5.4.

By looking at these plots it is evident, as we were expecting, that when moving from contents to threads there is a significant increase in the percentage of threads with a very small η, meaning that it is possible that contents which globally have a non-negligible amount of negative edges produce also threads that have very few or no negative edges. These are the subgraphs in which we expect to find the echo chambers. This is especially evident in the @nytimes and @bbcnews datasets, while in @foxnews this effect is less visible.

[7]As explained above (Subsection 5.1.2), we are referring to the accounts that are used to retrieve the contents of the graph.

Table 5.1: Basic statistics for analyzed datasets. Threads with no replies are excluded from the counts. @nytimes dataset is built from contents between the 13th and 21th of May, @foxnews between the 22nd and 29th of April and @bbcnews between the 26th and 31st of May.

Dataset	$\|V\|$	$\|E\|$	$\|\mathcal{C}\|$	Threads	Fraction of neg. edges
@foxnews	45509	82494	311	1922	0.588
@nytimes	81318	118876	492	6246	0.462
@bbcnews	16875	26636	380	1566	0.438



(a) $\eta(C)$ of @nytimes **(b)** $\eta(T)$ for @nytimes

(c) $\eta(C)$ of @foxnews **(d)** $\eta(T)$ for @foxnews

(e) $\eta(C)$ of @bbcnews **(f)** $\eta(T)$ for @bbcnews

Figure 5.4: $\eta(C)$ and $\eta(T)$ distribution for many datasets.

Table 5.2: Fraction of negative edges for datasets built on different subreddits, each for 200 contents between December 14, 2020 and March 11, 2021.

Dataset	Description	Fraction of neg. edges
r/cats	Pictures and videos about cats	0.169
r/Covid19	Scientific discussion of the pandemic	0.298
r/programming	Computer Programming discussions	0.303
r/climate	News about climate and related politics	0.392
r/Football	News, Rumors, Analysis of football	0.411
r/Economics	News and discussion about economics	0.417
r/Politics	News and discussion about U.S. politics	0.511
r/AskTrumpSupporters	Q&A between Trump supporters and non supporters	0.533

For verifying the reliability of the definition of *controversial* content, we also looked at the fraction of negative edges for different datasets, each associated to contents of the same topic, which we report in Table 5.2.

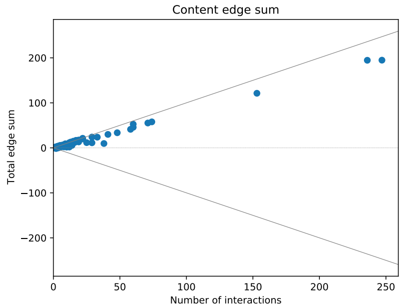
These results show an intuitive association between the fraction of negative edges in the graph and the topic discussed: graphs dealing with well-known *controversial* contents, like r/politics and r/asktrumpsupporters, are the one producing a higher fraction of negative edges. Also, as expected, they are followed by related topics (r/economics and r/climate) and football, while subreddits in which discussions over technologies and sciences predominate generally have less negative interactions between the users.

Furthermore, for each content C in an interaction graph, we plotted in Figure 5.5 the relationship between its number of interactions and the sum of the weights of its edges.

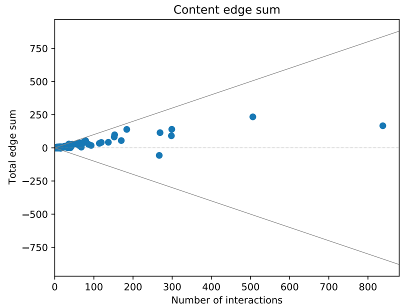
This relationship is closely related to the $\eta(C)$ of a content: let $E(C)$ be the set of edges associated to content $C \in \mathcal{C}$. We have that

$$\begin{aligned}\sum_{e_{ij} \in E(C)} w_{ij} &= \sum_{e_{ij} \in E^+(C)} |w_{ij}| - \sum_{e_{ij} \in E^-(C)} |w_{ij}| \\ &= \sum_{e_{ij} \in E(C)} |w_{ij}| - 2 \sum_{e_{ij} \in E^-(C)} |w_{ij}|.\end{aligned}$$

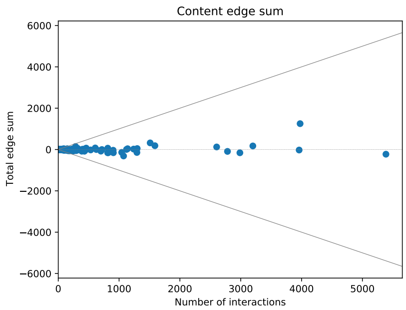
Thus, if we take the ratio with the number of interactions and suppose $|w_{ij}| \approx 1$



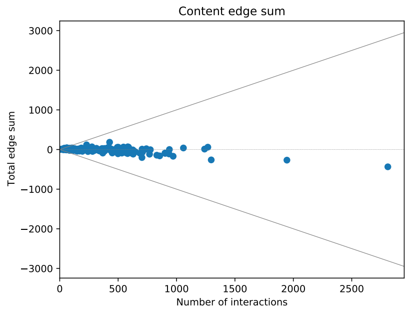
(a) r/cats



(b) r/covid19



(c) r/politics



(d) r/asktrumpsupporters

Figure 5.5: Plots of sum of the edge weights over the number of interactions for contents from different datasets/subreddits.

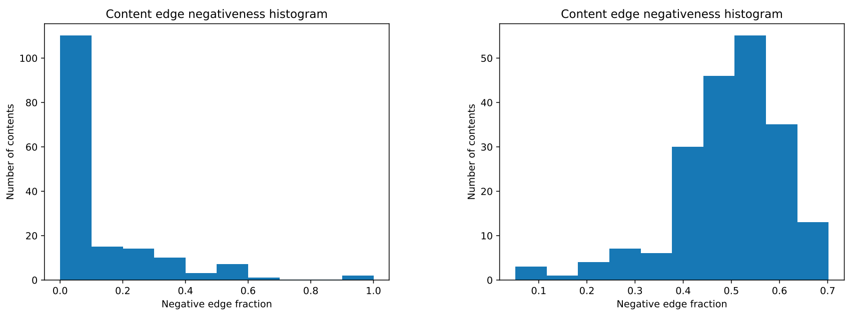
we obtain

$$\frac{\sum_{e_{ij} \in E(C)} |w_{ij}| - 2 \sum_{e_{ij} \in E^{-}(C)} |w_{ij}|}{|E(C)|} \approx 1 - 2\eta(C). \tag{5.2}$$

We can see in the plots that content distributes in a pattern which is very similar to that of a line with rare or no outliers. Due to (5.2) this means that $\eta(C)$ is very similar for different contents C, thus the points create a line whose angular coefficient is exactly (5.2).

Consequently, as we would intuitively say, contents related to politics are generally controversial, and most of them, as we can see in Figure 5.5c, have a high $\eta(C)$.

This is even more clearly visible when plotting the histogram of $\eta(C)$ for the contents in the dataset (Figure 5.6), with most of the contents having a $\eta(C)$ which is very close to the fraction of negative edges in the graph reported in Table 5.2.



(a) r/cats **(b)** r/asktrumpsupporters

Figure 5.6: $\eta(C)$ distribution for 2 of the datasets shown in Figure 5.5.

5.2.2 Experiments on Synthetic Data

For studying how the model behaves in controlled situations we define a parametrized model based on the Information Spread model (Subsection 5.1.1).

We will generate graphs with four communities, i.e. $k = 4$. Also, We choose $\beta_a = 1$, meaning that all nodes will be active in each thread. This a simplifying assumption which allows us to have a better grasp of the results. Because of this choice the values $\beta_n = 1$, $\theta = 1$ and

$$\phi_{rs} = \begin{cases} 1, & \text{if } r = s, \\ 0, & \text{otherwise} \end{cases} \qquad \text{for all } r, s \textit{ groups} \tag{5.3}$$

do not influence the structure of the resulting graph.

We also choose ω^-_{rs} and ω^+_{rs} to be dependent on a *noise* variable x. More specifically, we choose

$$\omega^+_{rs} = \begin{cases} 1 - x, & \text{if } r = s, \\ \frac{x}{4}, & \text{otherwise,} \end{cases} \qquad \text{for all } r, s \textit{ groups} \tag{5.4}$$

and

$$\omega^-_{rs} = \begin{cases} x, & \text{if } r = s, \\ \frac{1-x}{4}, & \text{otherwise,} \end{cases} \qquad \text{for all } r, s \textit{ groups}. \tag{5.5}$$

In absence of noise ($x = 0$) we will generate threads whose communities are positive cliques and all the edges between vertices in different communities (which will be present with probability 1/4) are negative.

We will compare different techniques for finding echo chambers (which, in this case, we will consider as corresponding to a community).

The approach, described in detail in Algorithm 5.1, involves calling an algorithm (generally any of the methods presented in Chapter 4) returning a set of users $U \subseteq V$ which will be labeled according to the majority of its members (by looking at the ground-truth assignment). Let E_k be the edges of thread T_k. We then remove the edges contributing to $\xi(U)$, i.e

$$\{e_{ij} \in E_k[U], T_k \in \mathcal{S}_C(U), C \in \hat{\mathcal{C}}\}.$$

After repeating k times this process, where k is the number of communities in the model, we compare the ground-truth labels and the predictions through the Jaccard coefficient and the Adjusted RAND index. We already introduced the Jaccard coefficient in Subsection 2.3.3.

The Adjusted RAND index is a measure of similarity between different clusterings. It is based on the RAND index, which compares the number of agreeing pairs in

Algorithm 5.1: Clustering process

Input: $G = \{G_k = (V, E_k)\}_k \leftarrow$ interaction graph, $\alpha \in [0, 1]$, $\mathcal{L}$ ground truth labels of V, $\mathcal{I}$ possible labels

Output: Jaccard and Adjusted RAND index

// Initialize predicted labels $\mathcal{P}$ with -1 (no label);

$\mathcal{P}[v] \leftarrow -1$ for all $v \in V$;

foreach $i \in \mathcal{I}$ **do**

- $U \leftarrow$ solve ECP on G ;
- // Remove edges contributing to $\xi(U)$;
- $E \leftarrow E \setminus \{e_{ij} \in E_k, T_k \in \mathcal{S}_C(U), C \in \hat{\mathcal{C}}\}$;
- $l \leftarrow$ majority label of users U in $\mathcal{L}$;
- // Do not re-label previously labeled nodes ;
- $U' \leftarrow U \setminus \{v \in U \ s.t. \ \mathcal{P}[v] \neq -1\}$;
- $\mathcal{P}[v] = l$ for all $v \in U'$;

end

// Compute Jaccard for each label and take the average ;

$J[l] \leftarrow Jaccard(\{v \in V \ s.t. \ \mathcal{P}[v] = l\},\ \{v \in V \ s.t. \ \mathcal{L}[v] = l\})$ for each $l \in \mathcal{I}$;

Jaccard score $\leftarrow \sum_{l \in \mathcal{I}} J[l]/|\mathcal{I}|$;

Adjusted RAND index $\leftarrow$ Adjusted RAND($\mathcal{P}$, $\mathcal{L}$) ;

return Jaccard score, Adjusted RAND index;

Table 5.3: Running times on generated graphs with 12 threads and four communities, each of six nodes, for different values of the noise variable x. The times are expressed in seconds.

	x					
	0	0.1	0.2	0.3	0.4	0.5
MIP	1762	2441	7247	24543	38992	41345
Rounding Algorithm	26	30	40	39	41	40

the two solutions; the Adjusted RAND index corrects the RAND Index "by chance", i.e. compares it to the expected index (i.e. of a random assignment). For more details we refer to [46].

What we expect is that, as the value of x increases, the produced threads will have generally more negative edges inside a community and more positive edges between different communities, making it more difficult for our algorithms to find the set of vertices corresponding to one of the Echo Chambers.

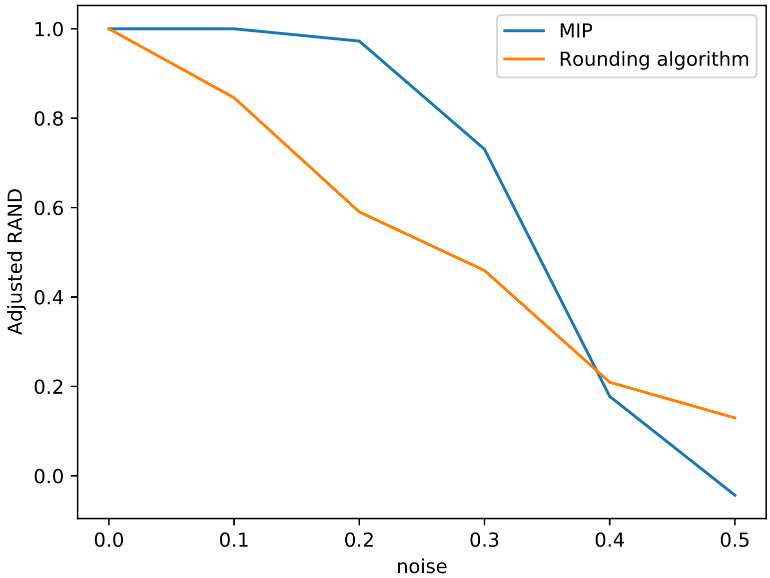
We report the scores obtained with the MIP for the ECP (Subsection 4.1.1) and the Rounding algorithm (Section 4.2). Due to the use of the MIP model these experiments have been carried out on small graphs with four communities, each of six nodes. The interaction graph contains 12 threads and we choose $\alpha = 0.2$. We experimentally saw that this choice of parameters produces controversial contents, thus allowing our methods to be applied.

Note that since we choose $\alpha = 0.2$, our analysis is partially limited for $x > 0.2$ since we may produce graphs that have a fraction of negative *intra-community* edges higher than α, although it is smaller than the fraction of negative *inter-community* edges. Nonetheless, we may be able to reconstruct the communities at least partially.

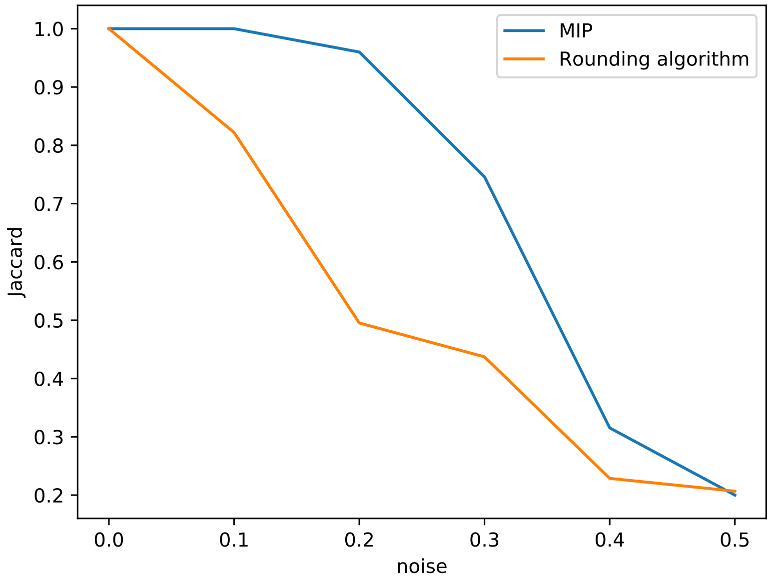
The performances of the two approaches are shown in Figure 5.7. We can see that the MIP model:

- Reconstructs the communities perfectly for values of $x \in \{0, 0.1\}$.
- Predicts the labels almost perfectly for $x = 0.2$.
- Reconstructs the communities partially for $x = 0.3$, achieving both a Jaccard coefficient and an Adjusted RAND Index around 0.7.
- Fails to find the original groups of users for $x \geq 0.4$.

The MIP formulation is generally better at finding the communities since, if the noise x is not too large, the best score is still achieved by selecting mostly nodes in



(a) Adjusted RAND index



(b) Jaccard Score

Figure 5.7: Clustering scores on generated graphs with 12 threads and four communities, each of six nodes, for different values of the noise variable x. Running times are reported in Table 5.3.

the same community (choosing nodes from other communities will generally add many more negative edges to the subgraph).

Conversely, the rounding algorithm is more affected by noise than MIP. More specifically, the rounding algorithm:

- Reconstructs the communities perfectly for $x = 0$.
- Partially recognizes the communities for $x = 0.1$, obtaining a Jaccard coefficient and Adjusted RAND Index around 0.8.
- Finds few users belonging to the same community for $x = 0.2$, with scores around 0.6.
- Fails to find the original groups of users for $x \geq 0.3$.

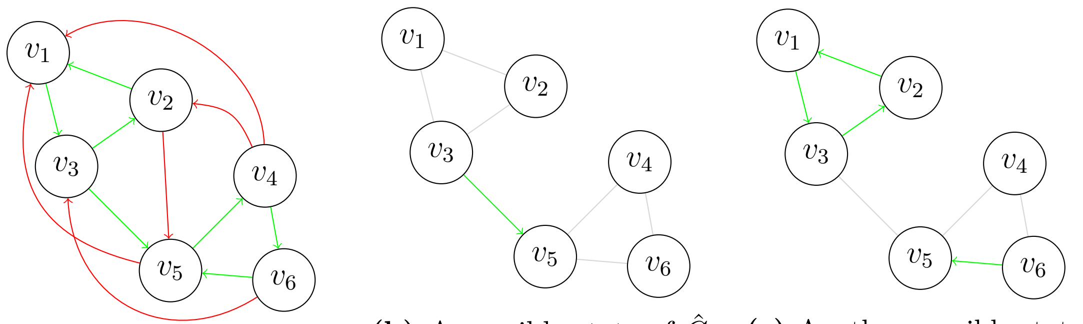
We illustrate its limitations with one example. Consider the graph in Figure 5.8a for $\alpha = 0.1$: the MIP model will find one of the two communities as optimal solution. Now consider the rounding algorithm (Section 4.2): the relaxation of the MIP will assign to all positive edges value 0.66, while the negative edges get value 0. This means that the algorithm will initially iterate over the positive edges, choosing randomly among them (since they have the same value). We illustrate in Figure 5.8b one possible iteration: in this case the algorithm will not be able to reconstruct one of the communities exactly since the considered edge will not allow the heuristic to have a single component of $\hat{G}$ associated to one of the communities. Conversely, in Figure 5.8c we can see a "luckier" iteration in which it is able to find one of the communities.

More generally, we can say that the rounding algorithm is less robust to noise than the MIP, especially if the noise produces an increase in the number of positive *inter-community* edges, as we saw in the example of Figure 5.8.

This is due to the fact that it may need to pick among positive edges with the same value (in the solution of the relaxation) during its execution: if the picked edge connects different communities, this will most likely prevent the algorithm from having communities as separate component in the dummy graph $\hat{G}$ (Section 4.2).

Consequently, we could improve the performances of the algorithm by decreasing the probability of picking inter-community edges, or, equivalently, increasing the probability of picking intra-community edges. For example, we could increase the latter probability by rising the number of threads in an interaction graph.

We repeated this experiment with a different number of threads while maintaining the same set of parameters as before. We show the results obtained by the rounding algorithm in Figure 5.9: we obtain better clustering performances as the number of threads increases, especially for values of $x \in \{0.0, 0.1\}$.



(a) An example of interaction graph G

(b) A possible state of $\hat{G}$ when running the rounding algorithm

(c) Another possible state of $\hat{G}$ when running the rounding algorithm

Figure 5.8: Possible rounding algorithm executions on an interaction graph with a one thread and one content. The two communities are represented by $\{v_1, v_2, v_3\}$ and $\{v_4, v_5, v_6\}$, respectively. Negative and positive edges are coloured in red and green, respectively. For $\alpha = 0.1$ we will have that its content is controversial and the exact solution returns either one of the two communities. The rounding algorithm may fail to reconstruct communities if the edge between the communities (e_{35}) is added early in the iterations (Figure 5.8b).

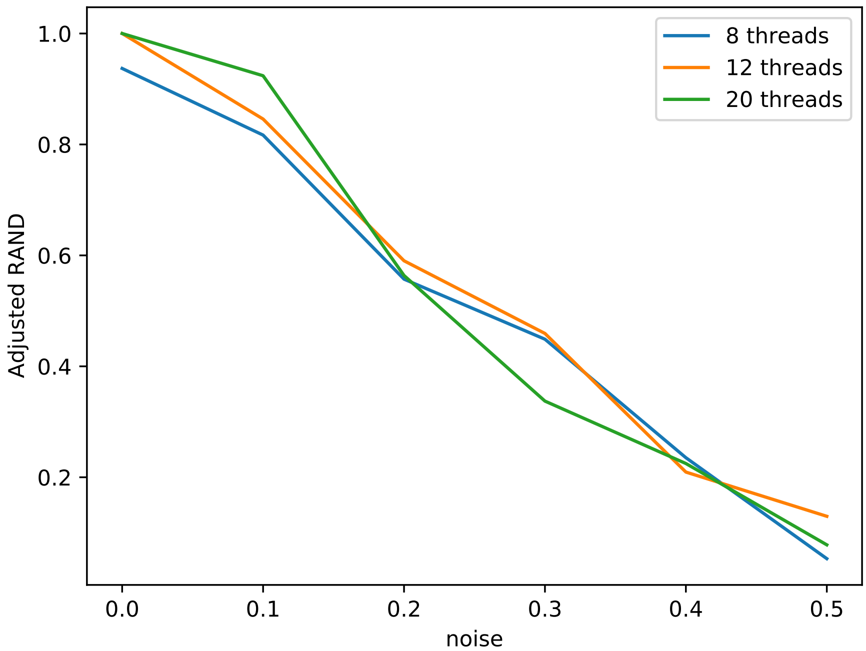


Figure 5.9: Adjusted RAND indices for graphs with four communities, each of six nodes, and different number of threads, obtained with the rounding algorithm.

5.2.3 Detecting Real-World Echo Chambers

We measured the performances of the rounding algorithm on real-world data by classifying the nodes of labeled datasets, similarly to what has been done with synthetic data. Recall from Section 5.1, that for these datasets we have retrieved a label for each user.

We will refer to a group of users with the same label as *community*.

We ran the experiments on the r/asktrumpsupporters and @nytimes datasets. In the first case users label themselves either as Trump Supporters (19%), Non Supporters (79%) or Undecided (2%). This last group of users was ignored in the analysis, i.e. these vertices were removed from the graph.

In the @nytimes dataset users are labeled either as democrats (80%) or republican (20%). In this case, in order to decrease the sparsity, we selected the 4-core.

We cluster the nodes as shown in Algorithm 5.1 and choose α as the median of the $\eta(C), C \in \mathcal{C}$. We run the rounding algorithm to find the Echo Chambers. By looking at the poor results, which we show in Table 5.4, it is clear that the algorithm is not able to correctly separate the communities. We motivate this with the following reasons:

- **Non-validity of the data model.** In trying to classify the nodes with our ECP solver, we are assuming that the data contains a clear separation of the users, in which one chamber corresponds to a single community. Furthermore, we are assuming that there are only two communities in the datasets we chose, which may also be a limiting assumption, since:
  - @nytimes may contain echo chambers related to different topics, as the set of contents does not only take into account U.S. political discussions.
  - r/asktrumpsupporters may be a non-representative dataset of discussion of polarized communities (we discuss this more in details in Subsection 5.1.3).
- **Complexity of sentiment analysis of social media language.** Social medias often involve messages which are not easily classifiable as either friendly and hostile, both because users often use jargon and because sometimes messages are aided by pictures and GIFs which are not taken into account by the sentiment analyzer.
- **Limitations of the rounding algorithm.** Since we are using an approximation algorithm we are not solving exactly the ECP: this may introduce limitations to the solutions which is used to cluster the nodes. More specifically, since at each iteration it uses a set of users connected by positive edges as possible solution (see Subsection 4.2.3), it is likely to return a set U with just one connected component.

Table 5.4: Classification scores obtained with the rounding algorithm on two labeled datasets. α is chosen as the median $\eta(C)$, $C \in \mathcal{C}$. For the @nytimes dataset we report the statistics related to its 4-core. $|\{T\}|$ indicates the number of threads. The contents of @nytimes belong to the period between the 2nd and 8th of May, while the contents of r/asktrumpsupporters are between December 27, 2020 and May 7, 2021.

Dataset	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\{T\}\rvert$	Adjusted RAND	Jaccard
r/asktrumpsupporters	11640	83038	357	357	0.095	0.016
@nytimes	1074	4921	139	254	0.022	0.420

- **Sparsity of the data**. We can see from Table 5.4 that the two datasets are sparse. This, as we discussed in Subsection 5.2.2, is important factor in achieving good performances, especially for the rounding algorithm. More generally, this is a limitation of real-world data: we observed, especially on Twitter, that even increasing the number of contents does not produce denser graphs, as the average degree remains between one and two. Furthermore, analyzing the k-core, a denser part of the graph, may affect the results since we expect that the echo chamber effect is especially visible in small and isolated components, maybe a small "bubble" of users sharing the same opinion, which may get excluded by the k-core selection.

5.3 Further Discussion of the Results

We focused our experiments on the rounding algorithm. We did this since we observed that, when running the other heuristic algorithms on smaller datasets (with less than 3000 nodes), its *time* performances were generally better than those of the peeling algorithm (Section 4.2). Also, when compared to the β-approach, the rounding algorithm is more expressive: we already discussed in Section 4.2 that the β-approach is able to find only group of nodes that are connected.

We report in Table 5.5 the execution times on some datasets. While execution *times* of the peeling approach explode with @BBCTech and r/cats, the rounding algorithm and the β-approach show a more stable trend, with most of the experiments of both of them being completed in less than 6 seconds.

We summarize our results for the rounding algorithm as follows. It achieves good performances on data with polarized communities, showing also to be more robust as the available data increases: a larger number of threads, as we discussed

Table 5.5: Execution time in seconds of the heuristics. Here, α is chosen to be the median of the $\eta(C)$ for each dataset. The contents of the datasets belong to the period between the 26th of May and the 1st of June.

Dataset	$\|V\|$	$\|E\|$	Rounding	Peeling	β
@EMA_News	1226	1842	0.933	24.050	36.058
@bbcsciencenews	447	388	4.917	175.360	0.796
@BBCNewsEnts	220	183	4.618	138.259	0.438
@BBCTech	793	719	86.203	2798.377	0.982
r/cats	2493	4028	5.440	140 844.752	0.733

in Subsection 5.2.2, helps the algorithm selecting *intra-community* edges, which allows it to correctly classify the nodes. Conversely, the rounding algorithm was not able to produce a good clustering of the nodes in real-world data.

Chapter 6

Conclusions and Future Work

In this research we proposed new methods for detecting polarization and echo chambers in social media, the ECP and D-ECP. We initially showed that these problems cannot be approximated even within a non-trivial factor $n^{1-\epsilon}$ and proposed methods for solving and approximating them, focusing on the rounding algorithm. We observed that it is able to find echo chambers in synthetically generated datasets but has limitations on real-world data. We motivate the poor performances on social media datasets with noise introduced by edge classification, sparsity of the data and possible limitations of the specific analyzed datasets. Nonetheless, our formulation paves the way for a richer and more expressive analysis of social media interactions, with more focus on the concepts of contents and threads.

Future works on the field should take into account these limitations which may require enhancing the graph through additional information like the use of a *follow* graph or changing the problem formulation to take into account the structure of real-world data and overcome the intrinsic complexity of the problem.

Moreover, we proposed alternative formulations and approximation algorithms whose performances and results could be analyzed in future research to get a better grasp of the problem. Also, we leave open the matter regarding the choice of α and how it affects the results. Finally, we leave as future challenge the study of methods for approximating the D-ECP, as well as a comparison with the results obtained with the ECP.

The further development and improvement of these methods will allow implementing techniques for reducing Echo Chambers, which are nowadays radicate into social media. However, we should note that the these methods could be also used for the opposite purpose, i.e. amplifying the Echo Chambers.

Bibliography

[1] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. «Quantifying Controversy on Social Media». In: *ACM Trans. Soc. Comput.* 1.1 (2018), 3:1–3:27. DOI: 10.1145/3140565 (cit. on p. 1).

[2] Pedro Henrique Calais Guerra, Wagner Meira Jr., Claire Cardie, and Robert Kleinberg. «A Measure of Polarization on Social Media Networks Based on Community Boundaries». In: *Proceedings of the Seventh International Conference on Weblogs and Social Media, ICWSM 2013, Cambridge, Massachusetts, USA, July 8-11, 2013*. Ed. by Emre Kiciman, Nicole B. Ellison, Bernie Hogan, Paul Resnick, and Ian Soboroff. The AAAI Press, 2013. URL: http://www.aaai.org/ocs/index.php/ICWSM/ICWSM13/paper/view/6104 (cit. on p. 1).

[3] Michael Conover, Jacob Ratkiewicz, Matthew Francisco, Bruno Gonçalves, Filippo Menczer, and Alessandro Flammini. «Political polarization on twitter». In: *Proceedings of the International AAAI Conference on Web and Social Media*. Vol. 5. 1. 2011 (cit. on p. 1).

[4] Anatoliy Gruzd and Jeffrey Roy. «Investigating political polarization on Twitter: A Canadian perspective». In: *Policy & internet* 6.1 (2014), pp. 28–45 (cit. on p. 1).

[5] Julie Jiang, Xiang Ren, and Emilio Ferrara. «Social media polarization and echo chambers: A case study of COVID-19». In: *CoRR* abs/2103.10979 (2021). arXiv: 2103.10979. URL: https://arxiv.org/abs/2103.10979 (cit. on p. 1).

[6] Jon Green, Jared Edgerton, Daniel Naftel, Kelsey Shoub, and Skyler J Cranmer. «Elusive consensus: Polarization in elite communication on the COVID-19 pandemic». In: *Science Advances* 6.28 (2020), eabc2717 (cit. on p. 1).

[7] Julie Jiang, Emily Chen, Shen Yan, Kristina Lerman, and Emilio Ferrara. «Political polarization drives online conversations about COVID-19 in the United States». In: *Human Behavior and Emerging Technologies* 2.3 (2020), pp. 200–211 (cit. on p. 1).

[8] Jun Lang, Wesley W Erickson, and Zhuo Jing-Schmidt. «# MaskOn!# MaskOff! Digital polarization of mask-wearing in the United States during COVID-19». In: *PloS one* 16.4 (2021), e0250817 (cit. on p. 1).

[9] Alessandro Cossard, Gianmarco De Francisci Morales, Kyriaki Kalimeri, Yelena Mejova, Daniela Paolotti, and Michele Starnini. «Falling into the Echo Chamber: The Italian Vaccination Debate on Twitter». In: *Proceedings of the Fourteenth International AAAI Conference on Web and Social Media, ICWSM 2020, Held Virtually, Original Venue: Atlanta, Georgia, USA, June 8-11, 2020*. Ed. by Munmun De Choudhury, Rumi Chunara, Aron Culotta, and Brooke Foucault Welles. AAAI Press, 2020, pp. 130–140. URL: `https://aaai.org/ojs/index.php/ICWSM/article/view/7285` (cit. on p. 1).

[10] Cass R Sunstein. «The law of group polarization». In: *University of Chicago Law School, John M. Olin Law & Economics Working Paper* 91 (1999) (cit. on p. 1).

[11] John C Turner, Michael A Hogg, Penelope J Oakes, Stephen D Reicher, and Margaret S Wetherell. *Rediscovering the social group: A self-categorization theory.* Basil Blackwell, 1987 (cit. on p. 1).

[12] Han Xiao, Bruno Ordozgoiti, and Aristides Gionis. «Searching for polarization in signed graphs: a local spectral approach». In: *Proceedings of The Web Conference 2020*. 2020, pp. 362–372 (cit. on p. 1).

[13] Mark Newman. *Networks*. Oxford University Press, July 2018. 800 pp. ISBN: 0198805098. URL: `https://www.ebook.de/de/product/32966014/mark_newman_networks.html` (cit. on pp. 2, 3, 55).

[14] Filippo Menczer, Santo Fortunato, and Clayton A. Davis. *A First Course in Network Science*. CAMBRIDGE, Jan. 2020. 300 pp. ISBN: 1108471137. URL: `https://www.ebook.de/de/product/37322811/filippo_menczer_santo_fortunato_clayton_a_davis_a_first_course_in_network_science.html` (cit. on p. 2).

[15] Boston) Barabasi Albert-Laszlo (Northeastern University. *Network Science*. Cambridge University Press, July 2016. 475 pp. ISBN: 1107076269. URL: `https://www.ebook.de/de/product/24312547/albert_laszlo_northeastern_university_boston_barabasi_network_science.html` (cit. on p. 2).

[16] Oded Goldreich. *Computational Complexity*. Cambridge University Press, Jan. 2015. 632 pp. ISBN: 052188473X. URL: `https://www.ebook.de/de/product/7102195/oded_goldreich_computational_complexity.html` (cit. on pp. 11, 22).

[17] Luca Trevisan. «Inapproximability of Combinatorial Optimization Problems». In: (Sept. 2004). arXiv: `cs/0409043 [cs.CC]` (cit. on pp. 11, 12).

[18] Vijay Vazirani. *Approximation Algorithms*. Springer-Verlag GmbH, Dec. 2002. ISBN: 3540653678. URL: `https://www.ebook.de/de/product/2147383/vijay_vazirani_approximation_algorithms.html` (cit. on p. 12).

[19] Giorgio Ausiello. «Approximability preserving reduction». In: (Sept. 2005) (cit. on p. 12).

[20] P. Crescenzi and V. Kann. «Approximation on the web: A compendium of NP optimization problems». In: *Randomization and Approximation Techniques in Computer Science*. Ed. by José Rolim. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 111–118. ISBN: 978-3-540-69247-8 (cit. on p. 13).

[21] Erik Demaine. *6.890 Algorithmic Lower Bounds: Fun with Hardness Proofs*. Fall 2014. URL: `https://ocw.mit.edu` (cit. on p. 12).

[22] P. Crescenzi. «A short guide to approximation preserving reductions». In: *Proceedings of Computational Complexity. Twelfth Annual IEEE Conference* (1997), pp. 262–273 (cit. on pp. 13, 14).

[23] Thomas Edgar. *Optimization of chemical processes*. New York: McGraw-Hill, 2001. ISBN: 0070393591 (cit. on pp. 14–16, 18, 19).

[24] Robert Vanderbei. *Linear programming : foundations and extensions*. New York: Springer, 2008. ISBN: 0387743871 (cit. on pp. 14–16).

[25] George Dantzig. *Linear Programming and Extensions*. Princeton University Press, Aug. 1998. 650 pp. ISBN: 0691059136. URL: `https://www.ebook.de/de/product/3326434/george_dantzig_linear_programming_and_extensions.html` (cit. on p. 14).

[26] Richard Kipp Martin. *Large Scale Linear and Integer Optimization: A Unified Approach*. Springer US, Nov. 1998. 762 pp. ISBN: 0792382021. URL: `https://www.ebook.de/de/product/2670475/richard_kipp_martin_large_scale_linear_and_integer_optimization_a_unified_approach.html` (cit. on p. 14).

[27] L.G. Khachiyan. «Polynomial algorithms in linear programming». In: *USSR Computational Mathematics and Mathematical Physics* 20.1 (1980), pp. 53–72. ISSN: 0041-5553. DOI: `https://doi.org/10.1016/0041-5553(80)90061-0`. URL: `https://www.sciencedirect.com/science/article/pii/0041555380900610` (cit. on p. 14).

[28] N. Karmarkar. «A new polynomial-time algorithm for linear programming». In: *Combinatorica* 4.4 (Dec. 1984), pp. 373–395. DOI: 10.1007/bf02579150 (cit. on p. 14).

[29] Dimitris Bertsimas. *Introduction to linear optimization*. Belmont, Mass: Athena Scientific, 1997. ISBN: 1886529191 (cit. on pp. 14, 16).

[30] Jiri Matousek Bernd Gärtner. *Understanding and Using Linear Programming*. Springer-Verlag GmbH, Sept. 2006. ISBN: 3540306978. URL: https://www.ebook.de/de/product/5832404/bernd_gaertner_jiri_matousek_understanding_and_using_linear_programming.html (cit. on p. 17).

[31] Laurence A. Wolsey Wolsey. *Integer Programming*. John Wiley & Sons, Sept. 1998. 288 pp. ISBN: 0471283665. URL: https://www.ebook.de/de/product/3599835/wolsey_laurence_a_wolsey_integer_programming.html (cit. on p. 16).

[32] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer Programming*. Springer International Publishing, Sept. 2016. 468 pp. ISBN: 3319384325. URL: https://www.ebook.de/de/product/26764375/michele_conforti_gerard_cornuejols_giacomo_zambelli_integer_programming.html (cit. on pp. 16, 18).

[33] Ravindran Kannan and Clyde L. Monma. «On the Computational Complexity of Integer Programming Problems». In: *Lecture Notes in Economics and Mathematical Systems*. Springer Berlin Heidelberg, 1978, pp. 161–172. DOI: 10.1007/978-3-642-95322-4_17 (cit. on p. 18).

[34] Leo Liberti. «Undecidability and hardness in mixed-integer nonlinear programming». In: *RAIRO - Operations Research* 53.1 (Jan. 2019), pp. 81–109. DOI: 10.1051/ro/2018036 (cit. on p. 18).

[35] Alexander Schrijver. *Theory of Linear Integer Programming*. John Wiley & Sons, June 1998. 484 pp. ISBN: 0471982326. URL: https://www.ebook.de/de/product/3055966/alexander_schrijver_theory_of_linear_integer_programming.html (cit. on p. 18).

[36] Moses Charikar. «Greedy approximation algorithms for finding dense components in a graph». In: *International Workshop on Approximation Algorithms for Combinatorial Optimization*. Springer. 2000, pp. 84–95 (cit. on pp. 19, 20, 48).

[37] Yuichi Asahiro and Kazuo Iwama. «Finding dense subgraphs». In: *International Symposium on Algorithms and Computation*. Springer. 1995, pp. 102–111 (cit. on p. 19).

[38] Yuichi Asahiro, Kazuo Iwama, Hisao Tamaki, and Takeshi Tokuyama. «Greedily finding a dense subgraph». In: *Journal of Algorithms* 34.2 (2000), pp. 203–221 (cit. on p. 19).

[39] Uriel Feige and Michael Seltser. «On the Densest K-Subgraph Problem». In: *Algorithmica* 29 (1997), p. 2001 (cit. on p. 19).

[40] Giorgio Gallo, Michael D. Grigoriadis, and Robert E. Tarjan. «A Fast Parametric Maximum Flow Algorithm and Applications». In: *SIAM Journal on Computing* 18.1 (Feb. 1989), pp. 30–55. DOI: `10.1137/0218003` (cit. on p. 19).

[41] Vinay Jethava and Niko Beerenwinkel. «Finding dense subgraphs in relational graphs». In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2015, pp. 641–654 (cit. on p. 20).

[42] Alexander Reinthal Anton Törnqvist Arvid Andersson and Erik Norlander Philip Stalhammar Sebastian Norlin. «Finding the densest common subgraph with linear programming». In: *Unpublished manuscript* (2016) (cit. on p. 20).

[43] Moses Charikar, Yonatan Naamad, and Jimmy Wu. «On finding dense common subgraphs». In: *arXiv preprint arXiv:1802.06361* (2018) (cit. on pp. 20, 22, 23).

[44] Konstantinos Semertzidis, Evaggelia Pitoura, Evimaria Terzi, and Panayiotis Tsaparas. «Finding lasting dense subgraphs». In: *Data Mining and Knowledge Discovery* 33.5 (2019), pp. 1417–1445 (cit. on pp. 20, 22–24).

[45] Guy Kortsarz. «On the hardness of approximating spanners». In: *Algorithmica* 30.3 (2001), pp. 432–450 (cit. on p. 22).

[46] Chandan K. Reddy Charu C. Aggarwal. *Data Clustering: Algorithms and Applications*. 0th ed. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Chapman and Hall/CRC, 2013. ISBN: 978-1-4665-5822-9,978-1-4665-5821-2 (cit. on pp. 26, 70).

[47] *Tweepy. An easy-to-use Python library for accessing the Twitter API.* URL: `https://www.tweepy.org/` (cit. on p. 58).

[48] *PRAW: The Python Reddit API Wrapper.* URL: `https://praw.readthedocs.io/en/latest/` (cit. on p. 59).

[49] Francesco Barbieri, Jose Camacho-Collados, Leonardo Neves, and Luis Espinosa-Anke. «TweetEval: Unified Benchmark and Comparative Evaluation for Tweet Classification». In: (Oct. 2020). arXiv: `2010.12421 [cs.CL]` (cit. on p. 59).

[50] Thomas Wolf et al. «Transformers: State-of-the-Art Natural Language Processing». In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: `https://www.aclweb.org/anthology/2020.emnlp-demos.6` (cit. on p. 59).

[51] Tiago P. Peixoto. «The graph-tool python library». In: *figshare* (2014). DOI: 10.6084/m9.figshare.1164194. URL: http://figshare.com/articles/graph_tool/1164194 (visited on 09/10/2014) (cit. on p. 62).