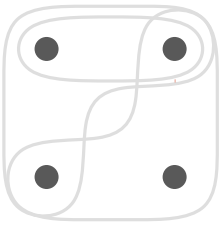
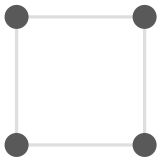
Complexity



Multiplicity

Beyond Flatland

Exploring Graphs in Many Dimensions



Expressivity

a dissertation submitted toward the degree

Doctor of Natural Sciences

of the
Faculty of Mathematics and Computer Science
of
Saarland University

by
Corinna Coupette

Descriptivity

Saarbrücken, 2023

Responsibility

Day of colloquium	23/10/2023
Dean of the faculty	Prof. Dr. Jürgen Steimle
Chair of the Committee	Prof. Dr. Markus Bläser
Reporters	Dr. Christoph Lenzen
	Prof. Dr. Kurt Mehlhorn
	Dr. Bastian Rieck
	Prof. Dr. Karsten Borgwardt
Academic Assistant	Dr. Philip Wellnitz

Beyond Flatland

Exploring Graphs in Many Dimensions

Abstract

Societies, technologies, economies, ecosystems, organisms, ... Our world is composed of *complex networks*—systems with many elements that interact in nontrivial ways. *Graphs* are natural models of these systems, and scientists have made tremendous progress in developing tools for their analysis. However, research has long focused on relatively simple graph representations and problem specifications, often discarding valuable real-world information in the process. In recent years, the limitations of this approach have become increasingly apparent, but we are just starting to comprehend how more intricate data representations and problem formulations might benefit our understanding of relational phenomena. Against this background, our thesis sets out to explore graphs in five dimensions:

descriptivity, *multiplicity*, *complexity*, *expressivity*, and *responsibility*.

Leveraging tools from graph theory, information theory, probability theory, geometry, and topology, we develop methods to (1) descriptively compare individual graphs, (2) characterize similarities and differences between groups of multiple graphs, (3) critically assess the complexity of relational data representations and their associated scientific culture, (4) extract expressive features *from* and *for* hypergraphs, and (5) responsibly mitigate the risks induced by graph-structured content recommendations. Thus, our thesis is naturally situated at the intersection of graph mining, graph learning, and network analysis.

Beyond Flatland

Exploring Graphs in Many Dimensions

Zusammenfassung

Gesellschaften, Technologien, Volkswirtschaften, Ökosysteme, Organismen, ... Unsere Welt besteht aus *komplexen Netzwerken*—Systemen mit vielen Elementen, die auf nichttriviale Weise interagieren. *Graphen* sind natürliche Modelle dieser Systeme, und die Wissenschaft hat bei der Entwicklung von Methoden zu ihrer Analyse große Fortschritte gemacht. Allerdings hat sich die Forschung lange auf relativ einfache Graphrepräsentationen und Problemspezifikationen beschränkt, oft unter Vernachlässigung wertvoller Informationen aus der realen Welt. In den vergangenen Jahren sind die Grenzen dieser Herangehensweise zunehmend deutlich geworden, aber wir beginnen gerade erst zu erfassen, wie unser Verständnis relationaler Phänomene von intrikateren Datenrepräsentationen und Problemstellungen profitieren kann. Vor diesem Hintergrund erkundet unsere Dissertation Graphen in fünf Dimensionen:

Deskriptivität, *Multiplizität*, *Komplexität*, *Expressivität*, und *Verantwortung*.

Mithilfe von Graphentheorie, Informationstheorie, Wahrscheinlichkeitstheorie, Geometrie und Topologie entwickeln wir Methoden, welche (1) einzelne Graphen deskriptiv vergleichen, (2) Gemeinsamkeiten und Unterschiede zwischen Gruppen multipler Graphen charakterisieren, (3) die Komplexität relationaler Datenrepräsentationen und der mit ihnen verbundenen Wissenschaftskultur kritisch beleuchten, (4) expressive Merkmale *von* und *für* Hypergraphen extrahieren, und (5) verantwortungsvoll den Risiken begegnen, welche die Graphstruktur von Inhaltsempfehlungen mit sich bringt. Damit liegt unsere Dissertation naturgemäß an der Schnittstelle zwischen *Graph Mining*, *Graph Learning* und Netzwerkanalyse.

Corinna Coupette

Beyond Flatland

Exploring Graphs in Many Dimensions

A SQUARE □

Preface

When I enrolled at LMU Munich in Fall 2015, I was looking for a side hustle to complement my first PhD. Little did I know that with *computer science*, I had found what I had been looking for in *law*: math married to language, providing powerful abstractions for reasoning about the real world.

By the time I started working on my second PhD in Spring 2020, I already knew that I was a scientist, and I am continually grateful to those acknowledged in my first thesis for providing the safe space I needed to discover that fact. The embodied experience from my first PhD allowed me to weather the storms of my second one—including, but not limited to, a global pandemic.

I am indebted to *Anja Feldmann* for bringing me to the Max Planck Institute for Informatics, to *Christoph Lenzen* for pulling me into theory, to *Kurt Mehlhorn* for allowing me to learn teaching, to *Bastian Rieck* for adopting me as his advisee, and to *Karsten Borgwardt* for serving on my committee. To *Jilles Vreeken* for inadvertently inspiring the centerpiece of this thesis. To *Aristides Gionis* for hosting me at KTH Stockholm, and to *Mikko Kivelä* for doing the same at Aalto University. To the coauthors of my papers, including those papers that did not make it into my thesis, and to the collaborators on my ongoing projects. To my family—especially to my brother, *Fabian Coupette*, my invaluable sibling scientist, of whose own, spirited work I am endlessly proud. And to *A Square*—for connecting my theses.

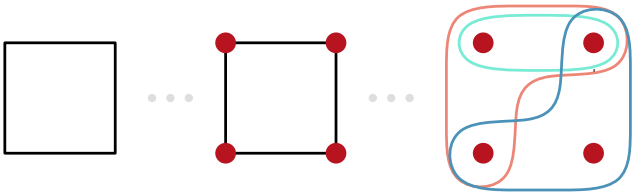
Espoo, June 2023 *Corinna Coupette*

Contents

Tables

Figures

Algorithms



1

Introduction: Toward Graphland

In Flatland [1], *A Square* living in a two-dimensional world begins to explore other dimensions. The 1884 dystopian novel constitutes both a mathematical adventure and a window into the world in which it was written. As such, Flatland is strikingly similar to research involving *graph data*: Originally limited to node sets endowed with simple binary relations, scientists are increasingly scrutinizing the limitations of their longstanding setup [32, 37, 255]. Both in pursuit of their own mathematical adventures and in response to the challenges emerging from an increasingly interconnected world, they are embracing more complex models [84]—from *multilayer networks* with several distinct node or edge sets [35, 147] to *dynamic graphs* allowing node or edge evolution [115, 117] and *hypergraphs* featuring n-ary relations [6, 281].

Standing on the shoulders of Euler [86] and Berge [26], our thesis pursues the path paved by decades of progress in *graph theory* [77, 116] and *social network analysis* [193, 265], as well as—more recently—*graph mining* [4, 47, 167], *graph learning* [39, 225, 271], and *network science* [19, 23, 201]. Our goal is to push further *beyond* Flatland*: Exploring graphs in many dimensions*, we enter Graphland, which welcomes relational data as they are—simple or complex, static or dynamic, binary or non-binary, and regardless of whether they identify as graphs or as networks. Preparing for our journey, we now delineate the territory we set out to discover, characterizing our contributions and providing details on our publications.

1.1 Contributions

Our thesis investigates Graphland in five dimensions: *descriptivity*, *multiplicity*, *complexity*, *expressivity*, and *responsibility*. Each of our contributions is primarily associated with one of these dimensions (although all contributions relate to more than one dimension), and we include them in the order in which we originally explored them.

Descriptivity: Momo Our first contribution focuses on the *descriptivity* dimension of Graphland, where *descriptivity* captures in how far our graph mining results help the analyst understand their data. Motivated by questions like "How do social networks differ across platforms?" and "How do information networks change over time?", here, our task is to compare two or more graphs. This task, also known as *graph similarity assessment*, is commonly treated as a *measurement* problem, but numerical answers give limited insight. Hence, we argue that if the goal is to gain understanding, we should treat graph similarity assessment as a *description* problem instead. We formalize this problem as a model selection task using the *Minimum Description Length principle*, capturing the similarity of the input graphs in a common model and the differences between the input graphs in transformations to individual models. To discover good models, we propose Momo, which breaks the problem into two parts and introduces efficient algorithms for each. Through extensive experiments on a wide range of synthetic and real-world graphs, we confirm that Momo works well in practice.

Multiplicity: Gragra Our second contribution focuses on the *multiplicity* dimension of Graphland, where *multiplicity* refers to a setting in which we are trying to understand multiple graphs at the same time. This scenario is motivated by questions like "How does neural connectivity in autistic children differ from neural connectivity in healthy children or autistic youths?" and "What patterns in global trade networks are shared across classes of goods, and how do these patterns change over time?". Hence, while Momo is designed to descriptively compare one graph with one or more other graphs, we would now like to descriptively compare entire *groups* of graphs: Given a set of graphs and a partition of these graphs into groups (e.g., defined by the values of a covariate associated with each individual graph), discover what graphs in one group have in common, how they systematically differ from graphs in other groups, and how multiple groups of graphs are related. We refer to this task as *graph group analysis*, which seeks to describe similarities and differences between graph groups by means of statistically significant subgraphs. To perform graph group analysis, we introduce Gragra, which uses *maximum-entropy modeling* along with *statistical testing*

to identify a non-redundant set of subgraphs with statistically significant associations to one or more graph groups. Through extensive experiments on a wide range of synthetic and real-world graph groups, we confirm that GRAGRA works well in practice.

COMPLEXITY: HYPERBARD Our third contribution focuses on the *complexity* dimension of GRAPHLAND, where *complexity* is short for both the complexity of the data we seek to capture in our graph representations and the complexity of the community in which our research is embedded. Located in the middle of the thesis, this contribution also marks a turning point in our positioning vis-à-vis our research community. To explore *data complexity*, we introduce HYPERBARD, a dataset of diverse relational data representations derived from Shakespeare's plays. Our representations range from *simple graphs* capturing character co-occurrence in single scenes to *node- and edge-weighted temporal hypergraphs* encoding complex communication settings and character contributions as hyperedges with edge-specific node weights. By making multiple intuitive representations readily available for experimentation, we facilitate rigorous representation robustness checks in graph learning, graph mining, and network analysis, highlighting the advantages and drawbacks of specific representations. Leveraging the data released in HYPERBARD, we demonstrate that many solutions to popular graph mining problems are highly dependent on the representation choice, thus calling current graph curation practices into question. As an homage to our data source, and asserting that science can also be art, we present all our points in the form of a *play*. This play is set in a microcosm modeled after our research community, and it poetically pinpoints some problematic patterns flowing from *community complexity*.

EXPRESSIVITY: ORCHID Our fourth contribution focuses on the *expressivity* dimension of GRAPHLAND, where by *expressivity*, we mean the capacity of both our graph representations and our methods to capture and conserve nuance in the data. Following the path suggested by the *data complexity* contribution of HYPERBARD, we continue our journey into the territory of *hypergraphs*. In particular, we extend the notion of *curvature*, a powerful invariant bridging geometry and topology, to the hypergraph domain. While the utility of curvature has been theoretically and empirically confirmed in the context of manifolds and graphs, its generalization to hypergraphs has remained largely unexplored. On graphs, the *Ollivier-Ricci curvature* measures differences between random walks via Wasserstein distances, thus grounding a geometric concept in ideas from probability theory and optimal transport. We develop ORCHID, a flexible framework generalizing Ollivier-Ricci curvature to hypergraphs, and prove that the resulting curvatures have fa-

vorable theoretical properties. Through extensive experiments on synthetic and real-world hypergraphs from different domains, we demonstrate that Orchid curvatures are both scalable and useful to perform a variety of hypergraph tasks in practice.

Responsibility: Gamine Our fifth contribution focuses on the *responsibility* dimension of Graphland. Here, *responsibility* refers to the necessity of computer scientists and engineers to understand and contain the risks created by their algorithms and technologies, thus extending the concerns raised in the *community complexity* contribution of Hyperbard to the impact of our research community on society at large. In particular, we are interested in learning how the recommendation algorithms deployed by digital media platforms expose users to media items that might be considered harmful, and in designing a procedure to mitigate exposure to harmful content which strikes a balance between competing stakeholder interests. Rather than *block* harmful content altogether, one promising approach here is to *minimize* the exposure to harmful content that is induced specifically by algorithmic recommendations. Hence, modeling media items and recommendations as a *directed graph*, we study the problem of reducing the exposure to harmful content by *post-processing* the recommendation graph via *edge rewiring*. We formalize this problem using *absorbing random walks*, and prove that it is NP-hard and NP-hard to approximate to within an additive error, while under realistic assumptions, the greedy method yields a $(1 - 1/e)$-approximation. Thus, we introduce Gamine, a fast greedy algorithm that can reduce the exposure to harmful content with or without quality constraints on recommendations. Through extensive experiments on synthetic data and real-world data from video recommendation and news feed applications, we confirm the effectiveness, robustness, and efficiency of Gamine in practice.

1.2 Publications

The contributions sketched above were developed in a series of publications, detailed below. For a quick overview, we list relevant metadata associated with each publication in Table 1.1. Each paper is accompanied by a reproducibility package deposited with Zenodo, in which we make all our data, code, and results publicly available (to the extent permitted by law). To all publications included in this thesis, the author of this thesis contributed as a first author, defining our problems, developing our theoretical results, designing our methods, compiling our datasets, conducting our experiments, and crafting our papers. For Gragra, first authorship was shared with Sebastian Dalleiger.

Table 1.1: Publications included in this thesis. *Rank* lists the latest available Computing Research & Education rank (CORE2021); *AR* indicates the acceptance rate in the year of publication. DSH (Digital Scholarship in the Humanities) is the flagship journal of the Alliance of Digital Humanities Organizations (ADHO) and the European Association for Digital Humanities (EADH), published by Oxford University Press.

Contribution	Dimension	Venue	Rank	AR	Year
Momo	Descriptivity	KDD	A*	15%	2021
Gragra	Multiplicity	AAAI (oral)	A*	15% (5%)	2022
Hyperbard	Complexity	DSH (submitted)	n/a	n/a	2022/3
Orchid	Expressivity	ICLR	A*	32%	2023
Gamine	Responsibility	KDD	A*	22%	2023

Ch. 2 Descriptivity: Momo

Corinna Coupette and Jilles Vreeken. "Graph similarity description: How are these graphs similar?" In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2021, pp. 185–195. DOI: 10.1145/3447548.3467257.

Reproducibility package: 10.5281/zenodo.4780912

Ch. 3 Multiplicity: Gragra

Corinna Coupette*, Sebastian Dalleiger*, and Jilles Vreeken. "Differentially describing groups of graphs." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2022, pp. 3959–3967. DOI: 10.1609/aaai.v36i4.20312.

Reproducibility package: 10.5281/zenodo.6342823

Ch. 4 Complexity: Hyperbard

Corinna Coupette, Jilles Vreeken, and Bastian Rieck. *All the world's a (hyper)graph: A data drama*. Submitted to Digital Scholarship in the Humanities (DSH). 2022. DOI: 10.48550/arXiv.2206.08225.

Reproducibility package:
10.5281/zenodo.6627158 (Dataset)
10.5281/zenodo.6627160 (Code)

Ch. 5 Expressivity: Orchid

Corinna Coupette, Sebastian Dalleiger, and Bastian Rieck. "Ollivier-Ricci curvature for hypergraphs: A unified framework." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2023. DOI: 10.48550/arXiv.2210.12048.

Reproducibility package: 10.5281/zenodo.7624573

Ch. 6 Responsibility: Gamine

Corinna Coupette, Stefan Neumann, and Aristides Gionis. "Reducing exposure to harmful content via graph rewiring." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2023, to appear. DOI: 10.1145/3580305.3599489.

Reproducibility package: 10.5281/zenodo.7936816

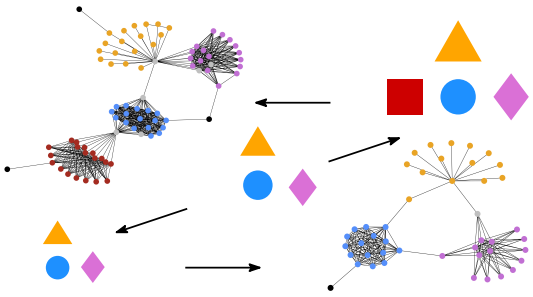
Beyond the work included in this thesis, the author contributed to a number of other papers during their computer-science dissertation phase. Below, we list these papers separated by type, in chronological order of their final appearance.

- Computer-science conference proceedings
  - Corinna Coupette and Christoph Lenzen. "A breezing proof of the KMW bound." In: *Symposium on Simplicity in Algorithms (SOSA)*. 2021, pp. 184–195. DOI: 10.1137/1.9781611976496.21
  - Corinna Coupette, Jyotsna Singh, and Holger Spamann. "Simplify your law: Using information theory to deduplicate legal documents." In: *Proceedings of the International Conference on Data Mining Workshops (ICDMW)*. 2021, pp. 631–638. DOI: 10.1109/ICDMW53433.2021.00083
- Journal articles and preprints
  - Daniel Martin Katz, Corinna Coupette, Janis Beckedorf, and Dirk Hartung. "Complex societies and the growth of the law." In: *Scientific Reports* 10 (2020), p. 18737. DOI: 10.1038/s41598-020-73623-x
  - Corinna Coupette*, Janis Beckedorf*, Dirk Hartung, Michael Bommarito, and Daniel Martin Katz. "Measuring law over time: A network analytical framework with an application to statutes and regulations in the United States and Germany." In: *Frontiers in Physics* 9 (2021). DOI: 10.3389/fphy.2021.658463
  - Corinna Coupette and Dirk Hartung. "Rechtsstrukturvergleichung [Structural comparative law]." In: *RabelsZ–The Rabel Journal of Comparative and International Private Law* 86.4 (2022), pp. 935–975. DOI: 10.1628/rabelsz-2022-0082
  - Corinna Coupette, Dirk Hartung, Janis Beckedorf, Maximilian Böther, and Daniel Martin Katz. "Law smells: Defining and detecting problematic patterns in legal drafting." In: *Artificial Intelligence and Law* 31 (2023), pp. 335–368. DOI: 10.1007/s10506-022-09315-w

- Corinna Coupette and Dirk Hartung. "Sharing and caring: Creating a culture of constructive criticism in computational legal studies." In: *MIT Computational Law Report* (2023), to appear. DOI: 10.48550/arXiv.2205.01071
- Bastian Rieck and Corinna Coupette. *Evaluating the "Learning on Graphs" conference experience*. 2023. arXiv: 2306.00586 [cs.LG]

Continuing the line of research begun in our first dissertation [55], many of these publications explore GRAPHLAND in *legal* dimensions [57, 67, 140], while others focus on *community* [58, 226], or on *simplicity* in *theory* [60] and in *practice* [59, 62]. For the purposes of this thesis, we will mostly leave our legal past behind, only occasionally leveraging law as a source of data or inspiration. This leaves us free to explore graphs in many *other* dimensions. Hence, without further ado:

Please enjoy the ride!



2

Descriptivity: Momo

Our exploration of Graphland begins with a fundamental problem: *comparing mathematical objects*. Naturally, given our territory, our mathematical objects of interest are *graphs*. What a *comparison* of graphs should output, however, depends on our objective: When our goal is to decide if two graphs are fundamentally the same, as in the case of the classic *graph isomorphism problem* [12, 109], a *binary* output suffices. When our goal is to quantify the similarity between two graphs, for example, as a subroutine in *few-shot graph learning* [127, 279], we expect a *real-valued* output. However, when our goal is to understand *how* our input graphs are (dis)similar—for example, because we have a scientific interest in the data underlying our graphs—, we desire an output that is *descriptive*. Thus motivated to explore the *descriptivity* dimension of Graphland, in this chapter, we ask:

How can we draw graph comparisons that deliver descriptivity*?*

2.1 Introduction

Comparing two or more graphs is important in many applications. For example, in biology, we might want to compare the protein interaction networks of different human tissues in order to discover common and specialized mechanisms, while in the social sciences, comparing collaboration networks over time or across fields could yield insights into knowledge dynamics. The task of comparing graphs is called *graph similarity assessment*. It is commonly treated as a *measurement* problem,

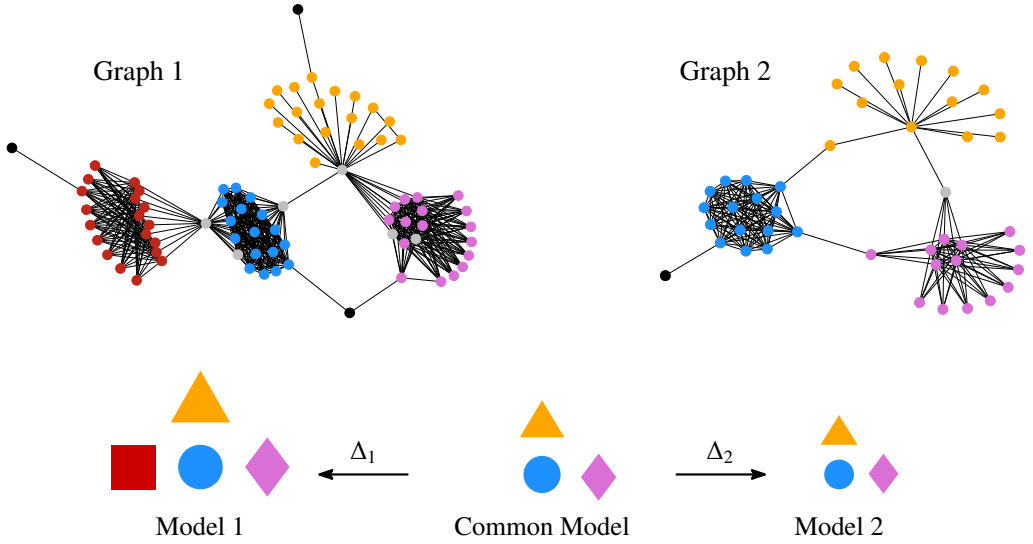


Figure 2.1: A common model captures the structure shared between the individual models of the input graphs. Here, the graphs share a clique (blue circle), a star (orange triangle), and a starclique (pink diamond), and transformations from their common model to their individual models express that the first graph features an additional biclique not present in the second graph (red square), and that the graphs in their scale (shape size).

i.e., a question to which a numerical answer suffices (e.g., 0.42). While such an answer may be useful in certain downstream tasks like classification or clustering, it provides limited insight and is thus generally dissatisfying to a domain expert.

We argue that if the goal is to gain understanding, we should not ask "how *similar* are these graphs?" but rather "*how* are these graphs similar?". That is, we propose to treat graph similarity assessment as a *description* problem, called *graph similarity description*, which demands an answer that, in easily understandable terms, characterizes what is similar and what is different between our input graphs. We formalize this problem in information-theoretic terms using the Minimum Description Length (MDL) principle, by which we are after the shortest lossless description of the input graphs using common and specific structures (e.g., stars, cliques, bicliques, and starcliques) as well as shared nodes and edges between these structures. Since we can measure how many bits we gain by compressing the graphs jointly, rather than individually, our formalization also allows for an easily interpretable quantification of differences.

An example of graph similarity description is given in Fig. 2.1, which depicts two toy graphs and the result returned by our method. Although the graphs are of different sizes, and no node alignment is given, we discover that both graphs contain a star (orange triangle) that is connected to a clique (blue circle) and a starclique (pink diamond). We further see that the left graph is different in that it

also contains a biclique (red square), and that the structures in the left graph all contain more nodes than their counterparts in the right graph (larger shapes).

Contributions. In this chapter, we make three contributions. First, we formulate the *graph similarity description* problem and formalize it using the MDL principle. Second, given that our problem features a very large and unstructured search space, we propose an algorithmic framework, called Momo (*Mo*del of *mo*dels), that breaks the problem into two parts and introduces efficient algorithms for each: Beppo discovers interpretable summaries for the individual input graphs, and Gigi uses them to unveil their shared and specific structures, from which we can also compute an informative similarity score. And third, through extensive experiments on synthetic and real-world graphs, we confirm that our algorithms perform well in practice: We discover summaries that are useful for domain experts, identify meaningful similarities between the protein interaction networks of various human tissues, and reveal distinct temporal dynamics in the collaboration networks of different scientific communities. Moreover, in practice, our approach scales near-linearly in the number of edges contained in the input graphs.

Structure. After introducing our notation and the basics of MDL in Section 2.2, we formalize our problem in Section 2.3. We develop Momo in Section 2.4 and cover related work in Section 2.5. Having validated our method through extensive experiments in Section 2.6, we conclude with a discussion in Section 2.7. We collect our notation in Section 2.A, provide further implementation details in Section 2.B, and make all code, data, and results publicly available.[1]

2.2 Preliminaries

We consider graphs $G_i = (V_i, E_i)$ with $n_i = |V_i|$ nodes, $m_i = |E_i|$ edges, and adjacency matrix $\mathbf{A}_i$, omitting the subscripts when clear from context. An alignment $\mathcal{A}_{ij}$ between the graphs G_i and G_j, denoted $G_i \|_{\mathcal{A}} G_j$, is a bijection from V_i to V_j. To allow comparisons between graphs of different sizes or graphs for which no node alignment is known, we allow this bijection to be *partial* or *empty*, i.e., there can be nodes in V_i (V_j) that have no image (preimage) in V_j (V_i) under $\mathcal{A}_{ij}$. We assume that our input graphs are *simple*, i.e., undirected, unweighted, without loops or parallel edges, and that only *two* input graphs are given, but our framework generalizes to comparisons between *more than two general* graphs.

We build on the notion of *Kolmogorov complexity*. The Kolmogorov complexity of an object x, $K(x)$, is the length in bits of the shortest program computing x on a universal Turing machine, and the *conditional* Kolmogorov complexity of

[1] 10.5281/zenodo.4780912

x given y, $\mathrm{K}(x \mid y)$, is the length of such a program with y as auxiliary input [168]. The *Information Distance* between x and y is (up to an additive logarithmic term) the length of the shortest program transforming x into y and y into x, i.e., $\mathrm{ID}(x, y) = \max\{\mathrm{K}(x \mid y), \mathrm{K}(y \mid x)\}$ [169]. Dividing by $\max\{\mathrm{K}(x), \mathrm{K}(y)\}$, we obtain the *Normalized Information Distance*.

The Kolmogorov complexity is not computable, and hence, neither is the Normalized Information Distance. To *describe* and *measure* the similarity between graphs in practice, we thus instantiate Kolmogorov complexity through the Minimum Description Length (MDL) principle [110]. MDL is a practical version of Kolmogorov complexity embracing the slogan *Induction by Compression*. Given a model class $\mathfrak{M}$ for some data $\mathcal{D}$, the best model $M \in \mathfrak{M}$ minimizes $\mathrm{L}(M) + \mathrm{L}(\mathcal{D} \mid M)$, where $\mathrm{L}(M)$ is the description length of M, $\mathrm{L}(\mathcal{D} \mid M)$ is the description length of the data when encoded using M, and both are measured in bits under our encoding. This is called *crude* MDL, and it contrasts with *refined* MDL, which encodes the model and the data together [110]. We opt for crude MDL not only because it is computable but also because we are particularly interested in the model: the structures shared by our input graphs, and the transformations necessary to derive the individual graphs from them. Finally, we require *lossless* descriptions to ensure fair comparisons between competing models.

All logarithms are to base 2, and we define $\log 0 = 0$. We use $\lfloor \cdot \rceil$ for rounding to the closest integer.

2.3 Theory

We now describe our first contribution, the MDL formulation of graph similarity assessment. Our data is $\mathcal{D} = (G_1, G_2, \mathcal{A}_{12})$, where G_1 and G_2 are our input graphs, and $\mathcal{A}_{12}$ is a (potentially partial or empty) node alignment between G_1 and G_2.

2.3.1 Graph Similarity Description, Informally

Our primary goal is to *describe* the similarity of our input graphs. That is, we aim to find the key structures that are shared between these graphs and contrast them with the structures that are specific to the individual graphs. By *structures*, we mean subgraphs whose connectivity follows distinct, interpretable patterns. Our *structure vocabulary* Ω comprises four structure types: (approximate) *cliques*, *stars*, *bicliques*, and *starcliques*. We choose these structure types because they are simple and widespread in real-world graphs from many different fields, but further structure types can easily be included, e.g., to tailor our method to a particular domain.

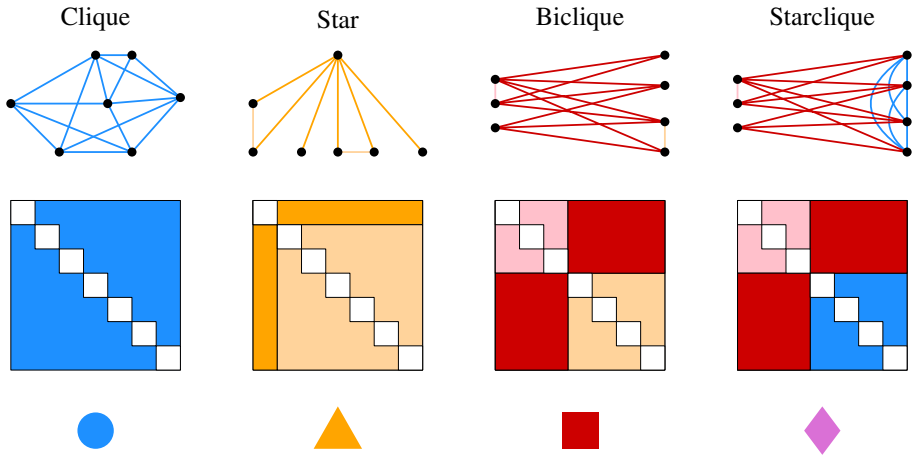


Figure 2.2: Graph structures are constraints over sets of (non-)edges. They can be visualized as induced subgraphs (top), adjacency submatrices (middle), or shapes (bottom). Each color in the adjacency submatrix is associated with a different constraint, where the white constraint enforces loop-freeness.

Intuitively, *cliques* are subgraphs with relatively homogeneous connectivity whose density stands out against the background distribution (e.g., echo chambers in social networks). *Stars* are subgraphs in which one node, the *hub*, is connected to all other nodes, the *spokes*, and the spokes are hardly connected among themselves (e.g., influencers and their followers). *Bicliques* are subgraphs whose nodes can be partitioned into two sets, *left* (L) and *right* (R), such that L and R are densely connected, the nodes in L are sparsely interconnected, and the nodes in R are sparsely interconnected (e.g., predators and prey in food webs). *Starcliques* are bicliques whose left nodes are densely, rather than sparsely, interconnected—i.e., stars whose hub is a clique (e.g., core and periphery in infrastructure networks). To describe real-world graphs accurately, we allow structures to overlap on nodes *and* on edges.

As depicted in Fig. 2.2, each structure imposes a set of constraints on the connectivity in the adjacency submatrix it identifies. We think of the node set sizes of a structure as *node fractions* (relative to a reference n) and of its connectivity constraints as *edge densities* (relative to the maximum possible number of edges).

We represent the structures we find in G_1 and G_2 individually as lists S_1 and S_2 in their *individual models* M_1 and M_2, and the structures that are shared between G_1 and G_2 as a list S_{12} in their *common model* M_{12}. To decide which structures to include in S_{12}, we construct a *matching* $\mathcal{M} \subseteq S_1 \times S_2$ (with S_1 and S_2 interpreted as sets), requiring that matched structures have the same type. For each $(s_1, s_2) \in \mathcal{M}$, we include one structure s of its type in S_{12}, writing $\varphi_1(s) = s_1$ and $\varphi_2(s) = s_2$ for the mappings from the shared structures to their counterparts. The node fractions (edge densities) of s are the averages of the node fractions (edge densities) in s_1

and s_2. For example, if $s_1 \in S_1$ is a clique with node fraction 0.1 and edge density 0.9, and $s_2 \in S_2$ is a clique with node fraction 0.2 and edge density 0.7, $s \in S_{12}$ is a clique with node fraction 0.15 and edge density 0.8.

To link the common model to the individual models, we translate M_{12} into M_1 and M_2 using *transformations* Δ_1 and Δ_2, i.e., $\Delta_1(M_{12}) = M_1$ and $\Delta_2(M_{12}) = M_2$. Our *transformation vocabulary* Σ contains edit operations to (1) add unmatched structures contained in individual models, and (2) morph structures from M_{12} into those from M_1 and M_2, i.e., reverse the averaging we perform when specifying the shared structures. For example, if a clique $s \in S_{12}$ has node fraction 0.15 and edge density 0.8, and $\varphi_1(s) = s_1 \in S_1$ has node fraction 0.1 and edge density 0.9, we need to *shrink* the node fraction and *grow* the edge density of s to match those in s_1.

To discover our common model M_{12}, individual models M_1, M_2, and transformations Δ_1, Δ_2, we leverage the MDL principle. That is, we seek to minimize

$$L(M_{12}) + L(\Delta_1, \Delta_2) + L(G_1 \|_{\mathcal{A}} G_2 \mid M_{12}, \Delta_1, \Delta_2) \ .$$

To this end, we need to define several encodings.

2.3.2 Similarity Description Encodings

In the following, we describe how we encode (1) the graphs G_1, G_2 under their individual models M_1, M_2, (2) the models M_1, M_2, (3) the common model M_{12}, and (4) the transformations Δ_1, Δ_2 in bits.

Encoding a Graph Under an Individual Model

Given a model M of a graph G, rather than using an ad-hoc encoding of the graph under the model (as is common practice), we seek to encode G *optimally*, leveraging the knowledge contained in M. As depicted in Fig. 2.2, this knowledge primarily comes as constraints on the total number of edges in the parts of the adjacency submatrix identified by the structures in M: A clique imposes one constraint, a star imposes two constraints, and a biclique or starclique imposes three constraints.

The probability distribution over the adjacency matrix $\mathbf{A}$ of G that represents the knowledge imparted by M (which includes n, m, and loop-freeness) *without any bias* is the distribution with the largest entropy among all distributions fulfilling the constraints imposed by M. Under this *maximum-entropy distribution*,

$$\Pr(a_{ij} \mid M) = \frac{\exp(\sum_{\lambda \in \Lambda(i,j)} \lambda)}{1 + \exp(\sum_{\lambda \in \Lambda(i,j)} \lambda)} \ , \tag{2.1}$$

where $\Lambda(i,j)$ is the set of Lagrange multipliers associated with the constraints covering $a_{ij} \in \boldsymbol{A}$ in the optimization problem finding the maximum-entropy distribution for $\boldsymbol{A}$ given M. The Shannon-optimal code based on this distribution minimizes the worst-case expected length of a message coming from the true distribution [75]. Hence, the length of G given M under an optimal encoding is

$$L(G \mid M) = \sum_{a_{ij} \in \boldsymbol{A}_1} -\log \Pr(a_{ij} \mid M) + \sum_{a_{ij} \in \boldsymbol{A}_0} -\log(1 - \Pr(a_{ij} \mid M)) \,, \tag{2.2}$$

where $\boldsymbol{A}_x = \{a_{ij} \in \boldsymbol{A} \mid a_{ij} = x \text{ and } i < j\}$ for $x \in \{0,1\}$.

Encoding an Individual Model

To encode an individual model M for a graph G, we communicate n, m, and $|S|$ using $L_{\mathbb{N}}$, the universal code for positive integers [227]. We then transmit the number of structures per type, and for each structure, in order, its type and its length. Thus, the length of an individual model M for a graph G is

$$\begin{aligned} L(M) = L_{\mathbb{N}}(n+1) + L_{\mathbb{N}}(m+1) + L_{\mathbb{N}}(|S|+1) + \log \binom{|S| + |\Omega| - 1}{|\Omega| - 1} \\ + \sum_{s \in S} \big(-\log \Pr(\text{type}(s) \mid S) + L(s) \big) \,. \end{aligned} \tag{2.3}$$

Each structure is defined *abstractly* by its constraints (cf. Fig. 2.2), and when we seek to find an MDL-optimal individual model, it is further identified by *concrete* node IDs (typeset in gray below). Assuming that all structures contain a positive number of nodes, the detailed encoding of our structures is as follows.

Cliques. To communicate a *clique* s, we transmit its number of nodes n_s, its number of edges m_s or non-edges $\overline{m}_s$, and the node IDs. Therefore, with $m_s^* = \frac{n_s(n_s-1)}{2}$, the length of a clique is

$$L(s) = L_{\mathbb{N}}(n_s) + 1 + \log\log \left\lfloor \frac{m_s^*}{2} \right\rfloor + \log(\min\{m_s, \overline{m}_s\}) + \log \binom{n}{n_s} \,. \tag{2.4}$$

Stars. To communicate a *star* s, we transmit its number of spokes $n_s - 1$, the number of edges between its spokes $x_s = m_s - n_s + 1$, the ID of the hub, and the IDs of the spokes. Hence, with $x_s^* = \frac{n_s-1)(n_s-2)}{2}$, the length of a star is

$$L(s) = L_{\mathbb{N}}(n_s - 1) + \log\log x_s^* + \log x_s + \log n + \log \binom{n-1}{n_s - 1} \,. \tag{2.5}$$

Bicliques and Starcliques. To communicate a *biclique* s, we transmit (1) its number of nodes n_s, (2) its number of left nodes n_L, (3) its number of edges between

left nodes m_L, (4) its number of edges between right nodes m_R, (5) its number of non-edges between left nodes and right nodes $m_{\bar{A}}^* - m_A$ (where $m_{\bar{A}}^* = n_L n_R$), and (6) the IDs of its left nodes and its right nodes. Thus, with $m_L^* = \frac{n_L(n_L-1)}{2}$ and $m_R^* = \frac{n_R(n_R-1)}{2}$, the length of a biclique is

$$\begin{aligned} L(s) = L_{\mathbb{N}}(n_s) + \log n_s + \log\log m_L^* + \log m_L + \log\log m_R^* + \log m_R \\ + \log\log m_{\bar{A}}^* + \log(m_{\bar{A}}^* - m_A) + \log \binom{n}{n_L} + \log \binom{n - n_L}{n_s - n_L} . \end{aligned} \tag{2.6}$$

To transmit a *starclique* s, we replace m_L by $\overline{m}_L = m_L^* - m_L$.

Encoding a Common Model

When communicating M_{12}, without loss of generality, we assume that $n_1 \geqslant n_2$, and we transmit the node fractions and edge densities of all shared structures with reference to n_1. Since we explicitly want to handle unaligned graphs and graphs of different sizes, their common model does not include node IDs. To encode M_{12}, we hence use the expression for individual models, with the node ID parts omitted, and the terms for n and m replaced by

$$L_{\mathbb{N}}(n_1 + 1) + L_{\mathbb{N}}(n_1 - n_2 + 1) + L_{\mathbb{N}}(m_1 + 1) + L_{\mathbb{N}}(|m_1 - m_2| + 1) + 1 . \tag{2.7}$$

Encoding Transformations

The common model M_{12} contains only structures that are shared between G_1 and G_2, and structures may be shared without being isomorphic. Consequently, M_{12} is generally different from M_1 and M_2, even if we define all models without node IDs. *Transformations* link M_{12} to M_1 and M_2 such that $\Delta_1(M_{12}) = M_1$ and $\Delta_2(M_{12}) = M_2$. That is, for $i \in \{1, 2\}$, Δ_i morphs M_{12} into M_i by *growing* or *shrinking* the node fractions and edge densities of the structures in S_{12} to match those in S_i as well as *adding* those structures from S_i that have no counterpart in S_{12}.

To derive the necessary content for the transformations, we reason as follows. The node fractions and edge densities of each structure $s \in S_{12}$ are the average of its representatives in S_1 and S_2, $\varphi_1(s)$ and $\varphi_2(s)$. Hence, for each structure in $s \in S_{12}$, we expect a structure of the same type in S_1 and S_2. For each node fraction x in s, we expect the size of its counterpart in $\varphi_i(s)$ to be $\lfloor x \cdot n_i \rceil$, and for each edge density y in s, we expect the number of edges in its counterpart in $\varphi_i(s)$ to be $\lfloor y \cdot m_y^* \rceil$, where m_y^* is the maximum number of edges in the associated area of $\boldsymbol{A}_i$ (for $i \in \{1, 2\}$).

The transformation Δ_i is the deviation of M_i from our expectation based on M_{12}, and since the node fractions and edge densities of each structure in S_{12} are the average of its representatives in S_1 and S_2, for the shared structures, we can infer Δ_2 from Δ_1. Hence, to communicate Δ_1 and Δ_2, for each node fraction x (edge density y) in each structure $s \in S_{12}$, we transmit the number of nodes (edges) we need to add or subtract from $\lfloor x \cdot n_1 \rceil$ ($\lfloor y \cdot m_y^* \rceil$) to arrive at the size of its counterpart in $\varphi_1(s)$, along with the change direction (*grow* or *shrink*). Finally, we transmit the structures in $\overline{S}_1 = S_1 \setminus \varphi_1(S_{12})$ and the structures in $\overline{S}_2 = S_2 \setminus \varphi_2(S_{12})$.

Therefore, if $L(\delta_1 : \delta_1(s) = \varphi_1(s))$ is the description length of the transformation δ_1 morphing s into $\varphi_1(s)$, and T is the total number of change directions we need to transmit, the length of the transformations Δ_1 and Δ_2 is

$$L(\Delta_1, \Delta_2) = \sum_{s \in S_{12}} L(\delta_1 : \delta_1(s) = \varphi_1(s)) + \log T + \sum_{i \in \{1,2\}} L_{\mathbb{N}}(|\overline{S}_i| + 1)$$
$$+ \sum_{i \in \{1,2\}} \left(\log \binom{|\overline{S}_i| + |\Omega| - 1}{|\Omega| - 1} + \sum_{s \in \overline{S}_i} \left(-\log \Pr(\text{type}(s) \mid \overline{S}_i) + L(s) \right) \right) .$$

Although we have defined the individual models M_1 and M_2, the common model M_{12}, and the transformations Δ_1 and Δ_2 for similarity *description*, we can also use them for similarity *measurement*.

2.3.3 Similarity Measurement

For similarity measurement, our score should reflect the extent to which the structure of the input graphs can be captured by their common model. Since graphs have many permutation-invariant representations (unlike, e.g., strings), and computable instantiations of the Normalized Information Distance typically use intransparent compressors, defining such a score is not straightforward. To guarantee computability and interpretability, we thus instantiate the Normalized Information Distance using our *models as compressors*.

To this end, let G_1 and G_2 be our input graphs with alignment $\mathcal{A}_{12}$ and individual models M_1 and M_2 (encoded without node IDs). Let M_{12} be their best $\mathcal{A}_{12}$-respecting common model, and let Δ_1 and Δ_2 be transformations such that $\Delta_1(M_{12}) = M_1$ and $\Delta_2(M_{12}) = M_2$. The *Normalized Model Distance* (NMD) between G_1 and G_2 is

$$\text{NMD}(G_1, G_2) = \frac{L(M_{12}) + L(\Delta_1, \Delta_2) - \min\{L(M_1), L(M_2)\}}{\max\{L(M_1), L(M_2)\}} . \tag{2.8}$$

The NMD is 0 if $M_{12} = M_1 = M_2$ (with $\Delta_1 = \Delta_2 = \emptyset$), and it is 1 if $M_{12} = \emptyset$ (with $\Delta_1 = M_1$ and $\Delta_2 = M_2$). It allows us to compare our method with other similarity *measurement* methods even though our primary goal is similarity *description*.

2.3.4 Similarity Description, Formally

We are now ready to formally state our main problem.

Problem 2.1 (Graph Similarity Description)**.** *Given graphs* G_1, G_2, *and a (full, partial, or empty) alignment* $\mathcal{A}_{12} : V_1 \to V_2$, *discover the individual models* M_1, M_2, *the common model* M_{12}, *and the transformations* Δ_1, Δ_2 *that together minimize*

$$L(M_{12}) + L(\Delta_1, \Delta_2) + L(G_1 \|_{\mathcal{A}} G_2 \mid M_{12}, \Delta_1, \Delta_2) \,. \tag{2.9}$$

The search space for Problem 2.1 is huge: Even if we searched for *one* individual model only, limited the number of structures to k, set the minimum size of a structure to r, and required the union of all structures to form a partition of V, we would need to search over 4^k times the number of partitions of n into k parts of size at least r. These partitions are in bijection with the partitions of $n - k(r-1)$ into k parts, and hence, there are $S(n-k(r-1), k)$ of them, where S is the Stirling number of the second kind. Since we are looking for *three* models with intricate interconnections, the search space for our problem is even larger—not to mention the NP-hard subproblems we need to solve to identify optimal structures (e.g., MaxClique). Furthermore, our search space exhibits no structure such as (weak) (anti-)monotonicity of the total description length that would allow us to search it efficiently. Hence, we resort to heuristics.

2.4 Algorithms

We now introduce our second contribution, an algorithmic framework, called Momo (*Mo*del of *mo*dels), to approximate the graph similarity description problem. To discover good models in practice, we break this problem into two parts:

1. Approximate the individual models M_1 and M_2 minimizing $L(M_1) + L(G_1 \mid M_1)$ and $L(M_2) + L(G_2 \mid M_2)$. Since these models can be thought of as graph summaries, we refer to this task as *graph summarization*.
2. Given individual models M_1 and M_2, approximate the common model M_{12} and the associated transformations Δ_1 and Δ_2 minimizing

$$L(M_{12}) + L(\Delta_1, \Delta_2) + L(G_1 \|_{\mathcal{A}} G_2 \mid M_{12}, \Delta_1, \Delta_2) \,.$$

Algorithm 2.1: Graph summarization with Beppo

Input: Graph G, structure vocabulary Ω
Output: Model M with structure list S

```
1  C ← Connected components of G from decomposition
2  S′, S ← [], []
3  forall structure types ω ∈ Ω do
4      forall components C ∈ C do
5          Generate candidate of type ω from C
6      Merge generated candidates if they have large overlap
7      Append remaining candidates to S′
8  Sort structures s ∈ S′ by (n_s, m_s) (descending)
9  forall structures s ∈ S′ do
10     if L(s) + L(G | M ∪ {s}) < L(G | M) then
11         Append s to S
12 return M
```

Since we require there to be a unique structure in both M_1 and M_2 for each structure in M_{12}, this means we search for an optimal alignment between the structures in M_1 and M_2. Hence, we refer to this task as *model alignment*.

Given M_1, M_2, M_{12}, Δ_1, and Δ_2, the NMD can be readily computed.

Our architecture is flexible in that (1) any algorithm generating graph summaries using the structure vocabulary Ω can be used in the first step, (2) any algorithm constructing a common model and transformations based on individual graph summaries using the structure vocabulary Ω and the transformation vocabulary Σ can be used in the second step, and (3) all alphabets can be replaced with other alphabets (if they are mutually compatible and the encoding is suitably amended), just as the NMD can be substituted with an alternative measure, e.g., for domain adaptation.

2.4.1 Step One: Graph Summarization (Beppo)

We begin by summarizing each of our input graphs individually. That is, our input is a single graph G with node set V and edge set E, and our output is a model M approximately minimizing $L(M) + L(G \mid M)$. Our procedure, called Beppo, is given as Algorithm 2.1.

To start, we decompose our graph into a set $\mathcal{C}$ of connected components of diameter at most three (l. 1). We do this by iteratively selecting the node v with the highest degree in the currently largest connected component to form a component $C \in \mathcal{C}$ with its neighbors, then deleting all edges incident with v, until no more components can be formed. This procedure is similar to the SlashBurn algorithm

[170], but we recurse on the *globally*, rather than on the *locally* largest connected component to ensure that all our components have small diameter. The generated components are used as seeds to produce candidates for each structure type from our structure vocabulary Ω, where we merge candidates of the same type if they overlap on a large fraction of their nodes (ll. 3–7). We sort the remaining candidates, which can overlap on nodes *and edges*, from largest to smallest (l. 8). Finally, for each structure s, in order, we add s to our model if this reduces our description length (ll. 9–11), i.e., if $L(s) + L(G \mid M \cup \{s\}) < L(G \mid M)$.

To generate a candidate of a certain structure type from a given component C with node set V_C (l. 5), we proceed as follows.

For a *clique* with node set V_s, we first find the maximum clique in C and include its nodes in V_s, then we iteratively add the node from $V \setminus V_s$ with the highest degree in G that is connected to at least 50% of the nodes in V_s until no more nodes fulfill this criterion.

For a *star* with spoke set V'_s, we declare a node with the highest degree in C to be the hub v, set $V'_s = V_C \setminus \{v\}$, and then iteratively (1) identify the nodes in V'_s that have more than $0.05 \cdot |V'_s|$ neighbors in V'_s, and (2) remove the $\min\{(0.1 + 0.01i), 1\}$ fraction of these nodes from V'_s that has the most neighbors in V'_s in iteration i.

For a *biclique* with node sets L and R, to start, we set the right node set to be the (at most) 5 nodes in a *maximal* independent set (MIS) of V_C that have the highest degree in G. We then identify the set $L' \subseteq V \setminus R$ of nodes that are connected to at least 50% of the nodes in R, and set L to be the (at most) 5 nodes in an MIS of L' that have the highest degree in G. If $|L| < 3$ or $|R| < 5$, we discard the candidate early. For the surviving candidates, we then iteratively (1) identify the set X of nodes from $V \setminus (L \cup R)$ that are connected to at most 5% of the nodes in L and at least 50% of the nodes in R, adding to L the node from X (if any) with the most neighbors in R, and (2) perform (1), switching the roles of L and R, until no more nodes satisfy our criteria for addition to L or R.

For a *starclique* with node sets L and R, to start, we set L to be the set of nodes contained in the maximum clique of C. We then identify the set $R' \subseteq V \setminus L$ of nodes that are connected to at least 50% of the nodes in L, and set $R = \text{MIS}(R')$. Subsequently, we iteratively (1) identify the set X of nodes from $V \setminus (L \cup R)$ that are connected to at least 50% of the nodes in L and to at least 50% of the nodes in R, adding to L the node from X (if any) with the most neighbors in R, and (2) identify the set Y of nodes from $V \setminus (L \cup R)$ that are connected to at most 5% of the nodes in R and to at least 50% of the nodes in L, adding to R the node from Y (if any) with the most neighbors in L, until no more nodes can be added.

Algorithm 2.2: Model alignment with GIGI

Input: Individual models M_1 and M_2 with structures S_1 and S_2, node alignment $\mathcal{A}_{12}$; transformation vocabulary Σ
Output: Common model M_{12} and transformations Δ_1, Δ_2 such that $\Delta_1(M_{12}) = M_1$, $\Delta_2(M_{12}) = M_2$

1 Compute constrained matching $\mathcal{M} \subseteq S_1 \times S_2$ `// Algorithm 2.3`
2 $M_{12}, \Delta_1, \Delta_2 \leftarrow [], [], []$
3 **forall** structures $(s_1, s_2) \in \mathcal{M}$ **do**
4 Compute the common structure s for (s_1, s_2)
5 Compute δ_i such that $\delta_i(s) = s_i$ for $i \in \{1, 2\}$
6 Append s to M_{12} and δ_i to Δ_i for $i \in \{1, 2\}$
7 **for** $i \in \{1, 2\}$ **do**
8 **forall** structures $s \in S_i \setminus \{s \in S_i \mid \exists p \in \mathcal{M} : s \in p\}$ **do**
9 Append s to Δ_i
10 **return** $M_{12}, \Delta_1, \Delta_2$

Running BEPPO on the graphs G_1 and G_2, we obtain interpretable individual models M_1 and M_2. Our next task is to align these models.

2.4.2 STEP TWO: MODEL ALIGNMENT (GIGI)

For the model alignment step, our inputs are the graphs G_1, G_2, the node alignment $\mathcal{A}_{12}$, and the models M_1, M_2. Our outputs are a common model M_{12} and the transformations Δ_1, Δ_2, which together minimize $L(M_{12}) + L(\Delta_1, \Delta_2) + L(G_1 \|_{\mathcal{A}} G_2 \mid M_{12}, \Delta_1, \Delta_2)$ approximately. Our procedure, called GIGI, is given as Algorithm 2.2.

In the critical first step, detailed below, GIGI computes a (bipartite) matching $\mathcal{M} \subseteq S_1 \times S_2$, pairing structures in S_1 with structures in S_2 (l. 1). The matching is *constrained* because we require that paired structures have the same type $\omega \in \Omega$. For each structure pair $(s_1, s_2) \in \mathcal{M}$, we then compute its common structure s as well as transformations δ_1 and δ_2 such that $\delta_1(s) = s_1$ and $\delta_2(s) = s_2$, which we add to M_{12}, Δ_1, and Δ_2, respectively (ll. 3–6). Finally, we add the unpaired structures from both S_1 and S_2 to Δ_1 and Δ_2, ensuring that $\Delta_1(M_{12}) = M_1$ and $\Delta_2(M_{12}) = M_2$ (ll. 7–9).

Typically, the matching $\mathcal{M}$ is not uniquely defined. We are interested in the matching that helps us minimize the description length. Sweeping the search space naïvely is not an option: For a structure vocabulary Ω, there exist

$$\prod_{\omega \in \Omega} (\omega_{\max} - \omega_{\min})! \cdot \binom{\omega_{\max}}{\omega_{\min}} \tag{2.10}$$

Algorithm 2.3: Structure matching with MAXIMALGREEDY

Input: Structure lists S_1, S_2, node alignment $\mathcal{A}_{12}$
Output: Structure matching $\mathcal{M} \subseteq S_1 \times S_2$

$\mathcal{M} \leftarrow \emptyset$
if $\mathcal{A}_{12} = \emptyset$ **then**
  $O_i \leftarrow (S_i, F_i, w_i)$ for $i \in \{1,2\}$, where $w_i((s,t)) = \mathrm{Jaccard}(s,t)$
  `// Build product graph G`
  $V \leftarrow \{(s_1,s_2) \in S_1 \times S_2 \mid \mathrm{type}(s_1) = \mathrm{type}(s_2)\}$
  $E \leftarrow \{((s_1,s_2),(t_1,t_2)) \mid (s_1,t_1) \in F_1, (s_2,t_2) \in F_2\}$
  $G \leftarrow (V, E, w)$, where $w\big(((s_1,s_2),(t_1,t_2))\big) = \prod_{i\in\{1,2\}} w_i\big((s_i,t_i)\big)$
  `// Select edges from product graph`
  while $E \neq \emptyset$ **do**
    $(u,v) \leftarrow \arg\max_{(u,v)\in E} w\big((u,v)\big)$
    Add u and v to $\mathcal{M}$
    $X \leftarrow \{x \in V \setminus \mathcal{M} \mid (x \cap u \neq \emptyset) \vee (x \cap v \neq \emptyset)\}$
    $E \leftarrow E \setminus \{(u,v)\}$
    $G \leftarrow G[V \setminus X]$
  `// Complete matching greedily`
  $\overline{S}_i \leftarrow S_i \setminus \{s \in S_i \mid \exists p \in \mathcal{M} : s \in p\}$ for $i \in \{1,2\}$
  forall structures $s_1 \in \overline{S}_1$ **do**
    forall structures $s_2 \in \overline{S}_2$ **do**
      if $\mathrm{type}(s_1) = \mathrm{type}(s_2)$ **then**
        Add (s_1,s_2) to $\mathcal{M}$
        $\overline{S}_i \leftarrow \overline{S}_i \setminus \{s_i\}$ for $i \in \{1,2\}$
        break
else
  $\overline{S}_i \leftarrow S_i$ for $i \in \{1,2\}$
  `// Add unmatched structures`
  while true do
    $U \leftarrow \{(s_1,s_2) \in \overline{S}_1 \times \overline{S}_2 \mid \mathrm{type}(s_1) = \mathrm{type}(s_2)\}$
    if $U = \emptyset$ **then break**
    $(s_1,s_2) \leftarrow \arg\max_{(s_1,s_2)\in U} \mathrm{Jaccard}_{\mathcal{A}}(s_1,s_2)$
    Add (s_1,s_2) to $\mathcal{M}$
    $\overline{S}_i \leftarrow \overline{S}_i \setminus \{s_i\}$ for $i \in \{1,2\}$
return $\mathcal{M}$

different *maximal* matchings alone, where, for $f \in \{\min, \max\}$, $\omega_f = f\{|\{s \in S_1 \mid \mathrm{type}(s) = \omega\}|, |\{s \in S_2 \mid \mathrm{type}(s) = \omega\}|\}$. Hence, we propose a matching heuristic, MAXIMALGREEDY, whose detailed pseudocode is given as Algorithm 2.3.

If no node alignment is available, for $i \in \{1,2\}$, MAXIMALGREEDY constructs *node overlap graphs* O_i. The nodes of these graphs are the structures in S_i, and

the weights of their edges F_i are the Jaccard similarities between the node sets of the structures (l. 3). MaximalGreedy then builds a variant of the *product graph* of O_1 and O_2, whose nodes are the subset of $S_1 \times S_2$ that agrees on type, and whose edge weights are the product of the edge weights in O_1 and O_2 (ll. 4–6). MaximalGreedy then iteratively selects the heaviest edges in the product graph and removes all nodes that are incompatible with these edges (ll. 7–12). Finally, it pairs the remaining structures of the same type in descending order of their size (ll. 13–19).

If a (partial) node alignment $\mathcal{A}_{12}$ is present, MaximalGreedy iteratively matches those structures s_1 and s_2 of the same type whose node sets have the largest average Jaccard similarity under $\mathcal{A}_{12}$ (ll. 20–27). For cliques, this equals the standard Jaccard similarity. For structures of other types, it is defined as

$$\mathrm{Jaccard}_{\mathcal{A}}(s_1, s_2) = \frac{1}{2} \cdot \sum_{i \in \{1,2\}} \frac{|\mathcal{A}_{12}(V_i(s_1)) \cap V_i(s_2)|}{|\mathcal{A}_{12}(V_i(s_1)) \cup V_i(s_2)|} ,$$

where V_1 and V_2 are the hub and spoke sets (for stars) or the left and right node sets (for bicliques and starcliques), respectively.

MaximalGreedy is designed to ensure interpretability: In the presence of a node alignment, it honors the node overlap of structures *between* graphs, and in the absence of such an alignment, it honors the node overlap of structures *within* graphs, all while respecting the constraints imposed by the structure types.

2.4.3 Computational Complexity

Having specified Beppo and Gigi as the main components of Momo, we now analyze Momo's complexity. Here, we assume that the total number of structures is $\mathcal{O}(1)$, which is required for interpretability. For Beppo, due to the set intersection operations involved, constructing structure candidates takes $\tilde{\mathcal{O}}(nm)$ time, where $\tilde{\mathcal{O}}$ hides polylogarithmic factors. To decide whether to add a candidate to our model, we need to find the maximum-entropy distribution for the adjacency matrix of the graph given that model, which takes $\mathcal{O}(1)$ time since the number of Lagrange multipliers is $\mathcal{O}(1)$. We also need to keep track of the mapping of Lagrange multipliers to potential edges, which takes $\mathcal{O}(n^2)$ time with $\mathcal{O}(1)$ candidates. Hence, Beppo runs in $\tilde{\mathcal{O}}(nm)$ time under the realistic assumption that $n \in \mathcal{O}(m)$. The complexity of Gigi is driven by $\mathcal{O}(1)$ Jaccard similarity computations, which together take $\mathcal{O}(n^2)$ time in the worst case ($\mathcal{O}(n)$ time on average), where $n = \max\{n_1, n_2\}$. Given individual models M_1, M_2, and their alignment $(M_{12}, \Delta_1, \Delta_2)$, computing the NMD takes $\mathcal{O}(1)$ basic arithmetic operations, i.e., it can be completed in $\mathcal{O}(1)$ time. Overall, the complexity of Momo is dominated by

Beppo, and hence, Momo runs in $\tilde{\mathcal{O}}(nm)$ time in the worst case. However, as we demonstrate in Section 2.6, in practice, Momo's performance is near-linear in the number of edges.

2.5 Related Work

To the best of our knowledge, we are the first to treat graph similarity assessment primarily as a *description* problem, rather than as a *measurement* problem. Related work broadly falls into two categories: graph similarity measurement and MDL-based graph summarization.

Graph Similarity Measurement. Early work on graph similarity measurement uses *global* measures that capture graph *structure*, e.g., graph edit distance and maximum common subgraphs [138, 222, 278]. Later research also explores measures that capture graph *connectivity* [151], leverage graph *decompositions* [202], or aggregate *local* similarities via node feature distributions [13, 28]. Building on prior work concerning *graph kernels* [40, 241], recent contributions investigate similarity learning via *deep* graph kernels [182, 207, 253, 274]. In contrast to the existing literature, first, our *primary goal* is graph similarity *description*, not graph similarity *measurement*. Second, our *perspective* emphasizes *interpretability*, which leads us to build on intuitive meso-level structures, rather than (overwhelmingly numerous) micro-level node features, motifs, or (opaque) macro-level graph features. Third, our *approach* is novel in that it formalizes graph similarity as a model selection task using the MDL principle. When evaluating Momo, we compare the NMD to another normalized similarity measure that is also based on information-theoretic principles: the *Network Portrait Divergence* (NPD) [13]. The NPD is the Jensen-Shannon divergence of the probability distributions of the input graphs that describe how many nodes have x neighbors at distance y. We show that the NMD and the NPD often capture similar trends, but only the NMD is intuitively interpretable.

MDL-Based Graph Summarization. Although novel in graph similarity assessment, the MDL principle has been used extensively in graph summarization. Starting with the SUBDUE system [52], a rich line of work has sought to move summarization beyond clustering using more expressive vocabularies to identify meaningful structures in *static* graphs [90, 104, 150, 170]. MDL has also been used to find partitions in graph *streams* [249] or structures ranging across multiple *aligned* snapshots of *dynamic* graphs [139, 237]. Going beyond the existing literature, first, we allow our structures to overlap not only on nodes but also on *edges*, and we can handle multiple graphs even if they are *unaligned*. Second, we improve the *method-*

Table 2.1: Our experiments are based on graph collections from highly diverse domains. N is the number of networks in the respective collection.

Coll.	Description	N	Distinction	Source
asb	AS Oregon RouteViews basic	9	2001/03/31–05/26,	[166]
asp	AS Oregon RouteViews plus	9	weekly	
bio	physical protein interactions	144	human tissues	[284]
clg	arXiv cs.LG collaborations	10	2011–2020,	[53]
csi	arXiv cs.SI collaborations	10	yearly (11/01)	
lde	German federal law	22	1998–2019,	[67]
lus	United States federal law	22	yearly	
rba	Barabási-Albert random graphs	50	10 sizes, 5 seeds	–
rer	Erdős-Rényi random graphs	50	10 sizes, 5 seeds	–

ology of previous static summarization methods, leveraging more noise-tolerant structure definitions and an optimal encoding of the data under the model. Third, in our structure search, we emphasize result *quality*, reflecting the need for accurate graph summaries as inputs to our comparison algorithm. When evaluating Momo, we compare Beppo to VoG [150], a static graph summarizer built on a similar graph decomposition method and vocabulary of interpretable structures (including cliques, bicliques, stars, and chains) that neither uses maximum entropy-modeling or component post-processing nor allows edge overlap. We show that Beppo discovers more informative summaries than VoG.

2.6 Experiments

We now present our third contribution, an extensive evaluation of the framework presented in Section 2.4. To this end, we implement Beppo in Julia and all other parts of Momo in Python. We run our experiments on Intel E5-2643 CPUs with 256 GB RAM. All data, code, and results are publicly available.[2] In the following, we answer three questions:

Q1 Graph Summarization. Does Beppo create useful graph summaries?

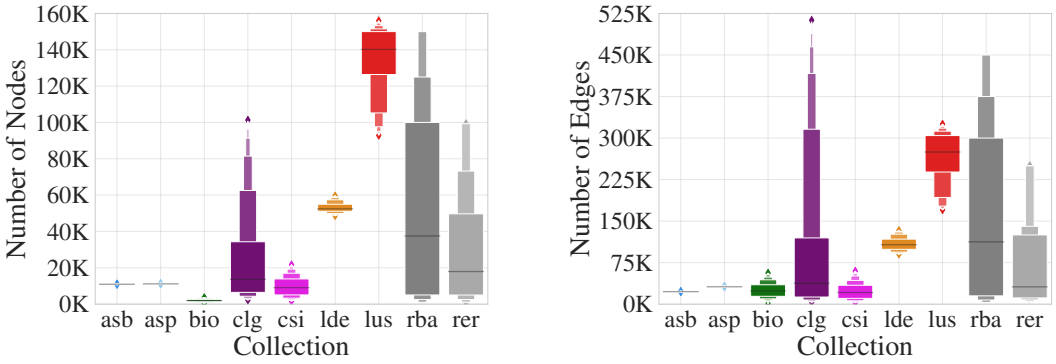
Q2 Model Alignment. Does Gigi discover interpretable common models?

Q3 Similarity Measurement. Does Momo yield informative similarity scores?

To ensure interpretability, we limit our summaries to at most 100 structures, although allowing more would give better compression.

In our experiments, we use real-world graphs from seven collections, as summarized in Table 2.1: Graphs in the *asb* and *asp* collections represent peering re-

[2] 10.5281/zenodo.4780912



(a) Distribution of n

(b) Distribution of m

Figure 2.3: We consider graphs of widely varying sizes and densities. For example, the graphs from the *bio* collection are small and dense, while the graphs from the *lus* collection are large and sparse.

lations between Autonomous Systems, each in 9 different weeks from 2001 [166]; graphs in the *bio* collection represent physical interactions between human proteins in 144 different tissues, where the protein identities induce partial node alignments between all pairs of graphs in the collection [284]; graphs in the *clg* and *csi* collections represent arXiv collaboration networks of cs.LG and cs.SI in each year from 2011 to 2020 [53]; and graphs in the *lus* and *lde* collections represent references between sections of the United States Code and the Code of Federal Regulations or their German equivalents in each year from 1998 to 2019 [67]. We also include two collections of synthetic random graphs, *rba* and *rer*, based on the Barabási-Albert (BA) model and the Erdős-Rényi (ER) model.

For all collections except *asb* and *asp*, we perform some preprocessing to transform the data provided into the graphs we use, which is documented in our codebase. All random graphs are generated with graph generators available in the Python library `networkx`. As depicted in Fig. 2.3, our graphs vary in size and density, containing up to 160 000 nodes and up to 525 000 edges.

Q1 Graph Summarization. In our context, graph summaries are useful if they capture the essence of a graph in an easily comprehensible manner. To assess whether Beppo creates such summaries, we start by comparing with VoG, which has been shown to produce useful graph summaries, on graphs from the VoG paper [150]. As shown in Table 2.2, in all experiments, Beppo saves more bits relative to the original encoding length than VoG-k for the same k, i.e., it achieves a better compression L%. Moreover, Beppo's compression is comparable to that of VoG-Greedy, although it uses much fewer structures. That is, even though our encoding of the data under the model is optimal, we manage to save more bits per structure than VoG. We also observe that while Beppo uses its entire vocabulary to

Table 2.2: Beppo compresses graphs more efficiently than VoG. |S| is the number of structures, and L% is the compression (in percent of the uncompressed encoded length). We state the n and m we found in the original input data, which sometimes differ slightly from those reported in [150].

			Beppo		VoG-k		VoG-Greedy	
Graph	n	m	|S|	L%	|S|	L%	|S|	L%
Epinions	75 879	405 740	100	20	100	5	2 746	19
Enron	79 870	288 364	100	18	100	7	2 331	25
AS-Oregon	13 579	37 448	100	28	100	21	399	29
Chocolate	2 877	5 467	55	9	100	7	101	12
Controversy	1 093	2 942	20	15	100	4	35	13

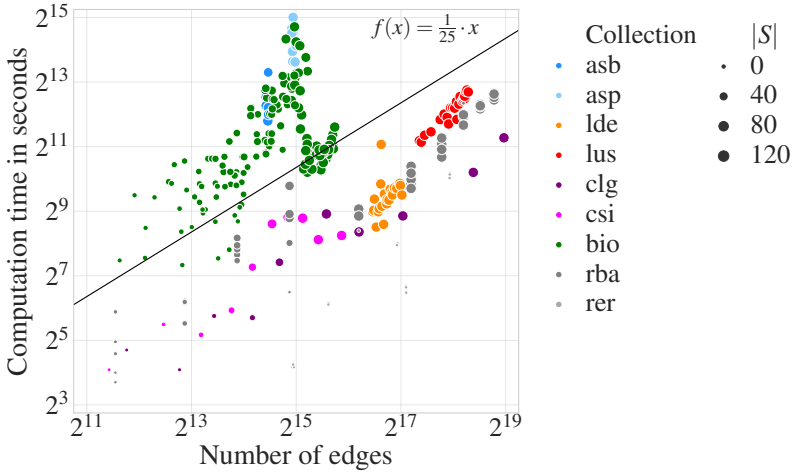


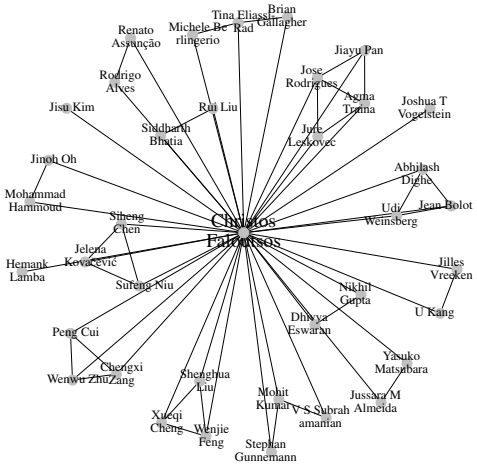
Figure 2.4: Beppo is near-linear and output-sensitive. Its computation time is shown as a function of m, with markers scaled by the number of discovered structures |S|.

summarize its input graphs, VoG finds almost only stars. As we show in Fig. 2.4, despite doing more work than VoG, Beppo is near-linear in practice.

In the left panel of Fig. 2.5, we tally how many structures of each type we find and what compression we achieve, on average, in each graph from our collections. Since the edges of ER graphs are chosen uniformly at random, and BA graphs are grown using preferential attachment, it comes as no surprise that we find at most one star (with minimal gain) in ER graphs and only stars in BA graphs, achieving no or little compression. The highest fraction of cliques occurs in the collaboration graphs (*clg*, *csi*), where papers with many authors induce cliques. (This is just one of the reasons why these data should rather be modeled as hypergraphs—but more on that in Chapters 4 and 5.) The hubs of the stars in these graphs correspond to well-known researchers with many independent collaborations, e.g., *Yoshua Bengio*, *Yang Liu*, and *Sergey Levine* in *clg* 2020. Some researchers occur in several structures, e.g., in *csi* 2020, 6 of the spokes in the star around *Christos*

Coll.	$\widehat{\text{cl}}$	$\widehat{\text{st}}$	$\widehat{\text{bc}}$	$\widehat{\text{sc}}$	$\widehat{\text{L\%}}$
asb	1	96	0	1	29
asp	3	91	0	6	29
bio	6	52	1	2	9
clg	24	36	0	2	8
csi	13	48	0	1	16
lde	0	98	1	1	1
lus	0	95	4	1	4
rba	0	68	0	0	3
rer	0	1	0	0	0

(a) Results of BEPPO



(b) Example star from *csi* 2020

Figure 2.5: BEPPO identifies meaningful structures. We show its average compression and number of structures per type (left), and an example star discovered in *csi* 2020 (right).

Faloutsos, shown in the right panel of Fig. 2.5, reappear in the star around *Danai Koutra*, and some of these spokes are also connected. This demonstrates the importance of allowing structures to overlap on nodes *and* on edges, a feature absent from other state-of-the-art summarizers like VoG. For the law graphs (*lde*, *lus*), our own analysis (supported by our PhD in law) and further discussion with legal scholars revealed that we can classify stars based on the ratio of the in-degree and out-degree of their hubs to uncover their legal function. Thus, BEPPO produces summaries that are useful to domain experts even for *directed* graphs, which sets it further apart from other methods.

As we allow structures to overlap, BEPPO's summaries can be visualized intuitively as *node overlap trees*. Node overlap trees are the maximum spanning trees of node overlap graphs, i.e., each vertex in them represents a structure, the edge weights are the Jaccard similarities between the node sets of the structures, and we remove all edges that are lightest in a cycle. To ensure connectivity, we introduce a root vertex that connects to the vertex with the largest degree inside each component. We depict the node overlap trees for selected digestive tract tissues from the *bio* collection in Fig. 2.6. Here, larger shapes indicate larger structures, and thicker edges indicate higher Jaccard similarities. From the vertices and the connectivity structure of the trees, it is immediately apparent that the top-row tissues are very similar—and indeed, the functions performed by the organs they represent are closely related.

Q2 MODEL ALIGNMENT. As GIGI builds on BEPPO, the common models it discovers are composed of easily comprehensible structures. By construction, this

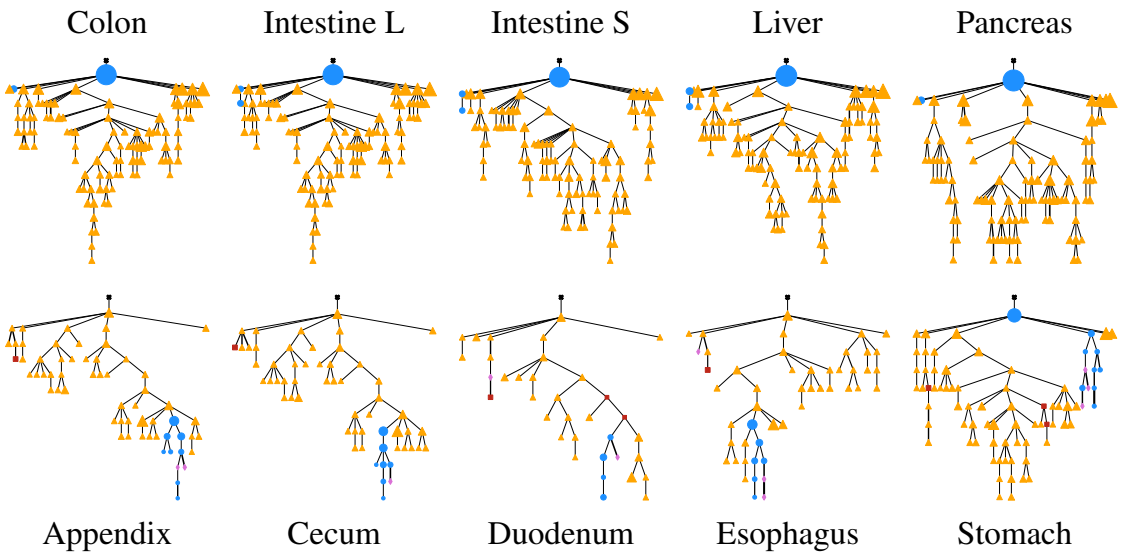
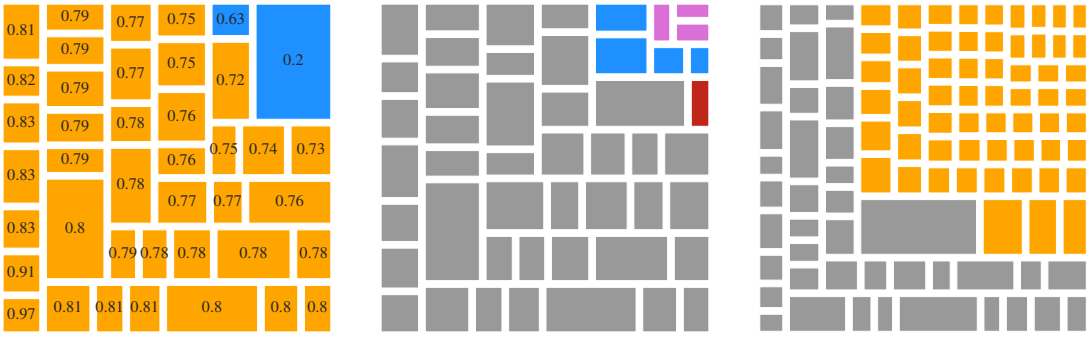


Figure 2.6: BEPPO creates similar summaries with similar node overlap structure for similar graphs. The node overlap trees for selected digestive tract organs in the *bio* collection mirror the functional (dis)similarity between these organs.

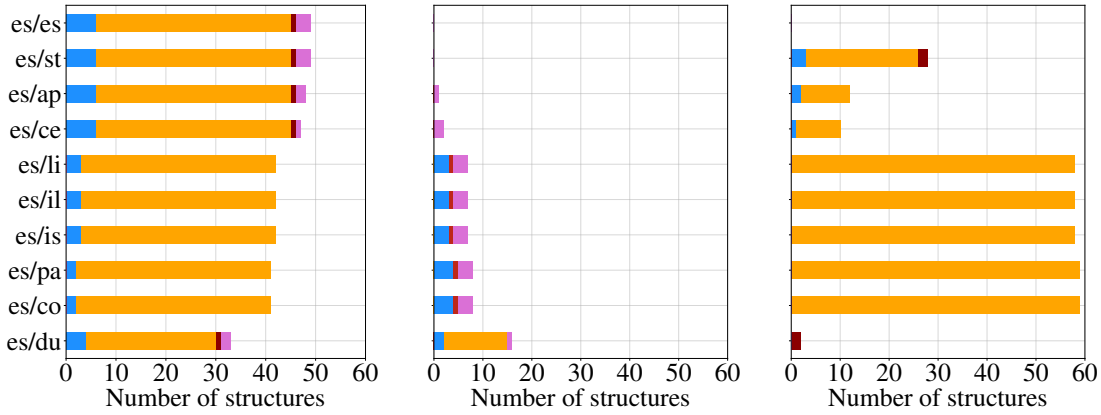
ensures a certain degree of interpretability. To understand the composition of a common model M_{12} and its relationship to individual models M_1 and M_2, we can further visualize these models using treemaps. We show an example from the *bio* collection in Fig. 2.7, contrasting the individual models for esophagus and colon with their common model. We see that esophagus and colon have many common structures, most of them stars, but the esophagus has more complex or dense structures (cliques, bicliques, and starcliques), while the colon has more simple sparse structures (stars). Using the node alignments between the *bio* graphs to annotate the shared structures with their average Jaccard similarities, we observe that all stars that are shared between esophagus and colon have a shared hub (indicated by a similarity above 0.5). Similar observations can be made for other tissues, e.g., the largest cliques in the top-row tissues from Fig. 2.6 all have a Jaccard similarity of at least 0.58. This indicates that *housekeeping proteins* might be expressed as *housekeeping structures* that recur across tissues, but a detailed investigation of this hypothesis lies outside the scope of this paper.

Beyond bilateral graph similarity assessment, GIGI's output enables comparisons between multiple graphs. As an example, in Fig. 2.8, we display the composition of the common models for comparisons of the esophagus with the tissues from Fig. 2.6 as a triptych of stacked bar charts. The graphic illustrates that the relationship between esophagus and colon, shown in Fig. 2.7, is comparable to that of the esophagus and *any* top-row organ from Fig. 2.6, and that all bottom-row organs share a biclique structure.



(a) Common model M_{12} **(b)** Esophagus model M_1 **(c)** Colon model M_2

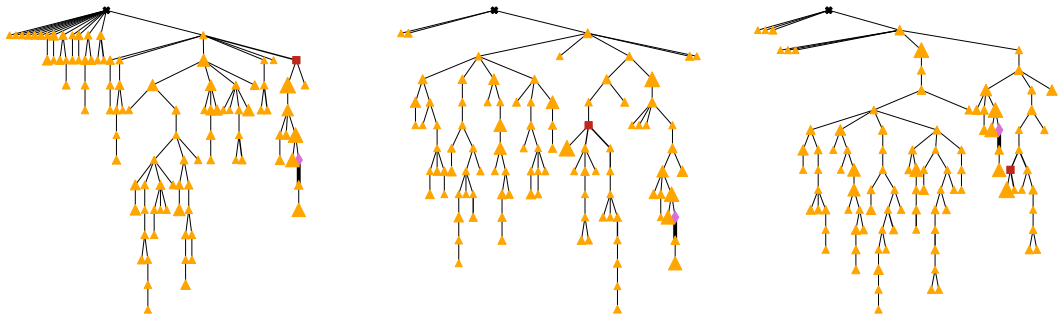
Figure 2.7: Gigi discovers interpretable common models. The common model (left) for esophagus (middle) and colon (right) contains mostly structures with high average Jaccard similarity (annotations). Each rectangle corresponds to a structure, sized proportionally to its number of nodes, and shared structures are grayed out in the individual models.



(a) Shared structures **(b)** Esophagus only **(c)** Other only

Figure 2.8: Gigi allows comparisons between multiple graphs. Here, we juxtapose shared (left) and specific (middle and right) structures for the esophagus and the other digestive tract tissues from Fig. 2.6.

To further explore the relationships between shared structures, we can leverage *common* node overlap graphs, i.e., node overlap graphs induced by our structure matching $\mathcal{M}$, with nodes $(s_1, s_2) \in \mathcal{M}$, edges $((s_1, s_2), (t_1, t_2))$, and edge weights $\prod_{i \in \{1,2\}} \text{Jaccard}(s_i, t_i)$. These graphs convey an interpretable notion of equivalence between the matched structures. To visualize common node overlap graphs, we again use node overlap trees, and Fig. 2.9 shows an example from the *lde* collection. While not all patterns from the individual trees recur in the common tree, the trees induced by the common tree in the individual node overlap graphs typically weigh a large fraction of the individual node overlap trees, i.e., the alignments discovered by Gigi respect much of the node overlap shared between the structures in our input graphs.



(a) *lde* 2018 | 2019 **(b)** *lde* 2018 **(c)** *lde* 2019

Figure 2.9: GIGI discovers common models retaining much of the node overlap shared by the structures in the individual graphs, as can be seen by comparing common (left) and individual (middle and right) node overlap trees. Here, the trees induced by the common tree in the individual node overlap graphs weigh more than $4/5$ of the individual node overlap trees.

Q3 SIMILARITY MEASUREMENT. Although our focus is similarity *description*, we can also use our similarity score, the NMD, for similarity *measurement*. To ensure that the NMD behaves as expected, we experiment with synthetic graphs. In Fig. 2.10, we show a single-linkage hierarchical clustering of the NMD values of synthetic graphs with $n \in \bigcup_{i=1}^{10}\{i \cdot 10^4\}$ nodes that contain $\lfloor 100/|\mathcal{S}| \rfloor$ structures of each type in $\mathcal{S}$, for $\mathcal{S} \in \mathcal{P}(\Omega) \setminus \emptyset$ (150 graphs in total). Fig. 2.10 highlights that the NMD is almost scale-invariant when the graphs contain rescaled versions of the same structures and their size differs within one order of magnitude, with larger size differences leading to larger NMD values. The NMD also behaves intuitively for models of varied compositions, showing a strong correlation with the number of structures that can be matched across graphs.

Next, we compare NMD values to *Network Portrait Divergence* values (NPD values), on the yearly snapshots of the IBM GitHub collaboration network from 2013 to 2017 used by Bagrow and Bollt [13]. As depicted in Fig. 2.11, the general trends are quite similar, but some years are *more* similar and others are *less* similar under NMD than under NPD. However, only our results are also interpretable: In 2014, for example, the network only has one star structure, explaining its high dissimilarity to 2015, which features one starclique and two cliques. The differences between NMD values and NPD values are likely due to the dependence of NPD on graph size, but since the underlying statistics are not intuitively comprehensible, we cannot be sure.

To understand the behavior of NMD values in real-world data at a high level, we study the distribution of NMD values for all pairwise comparisons of *different* graphs in our real-world collections, depicted in Fig. 2.12. We see that NMD values span the whole range, and their distribution differs depending on the type of

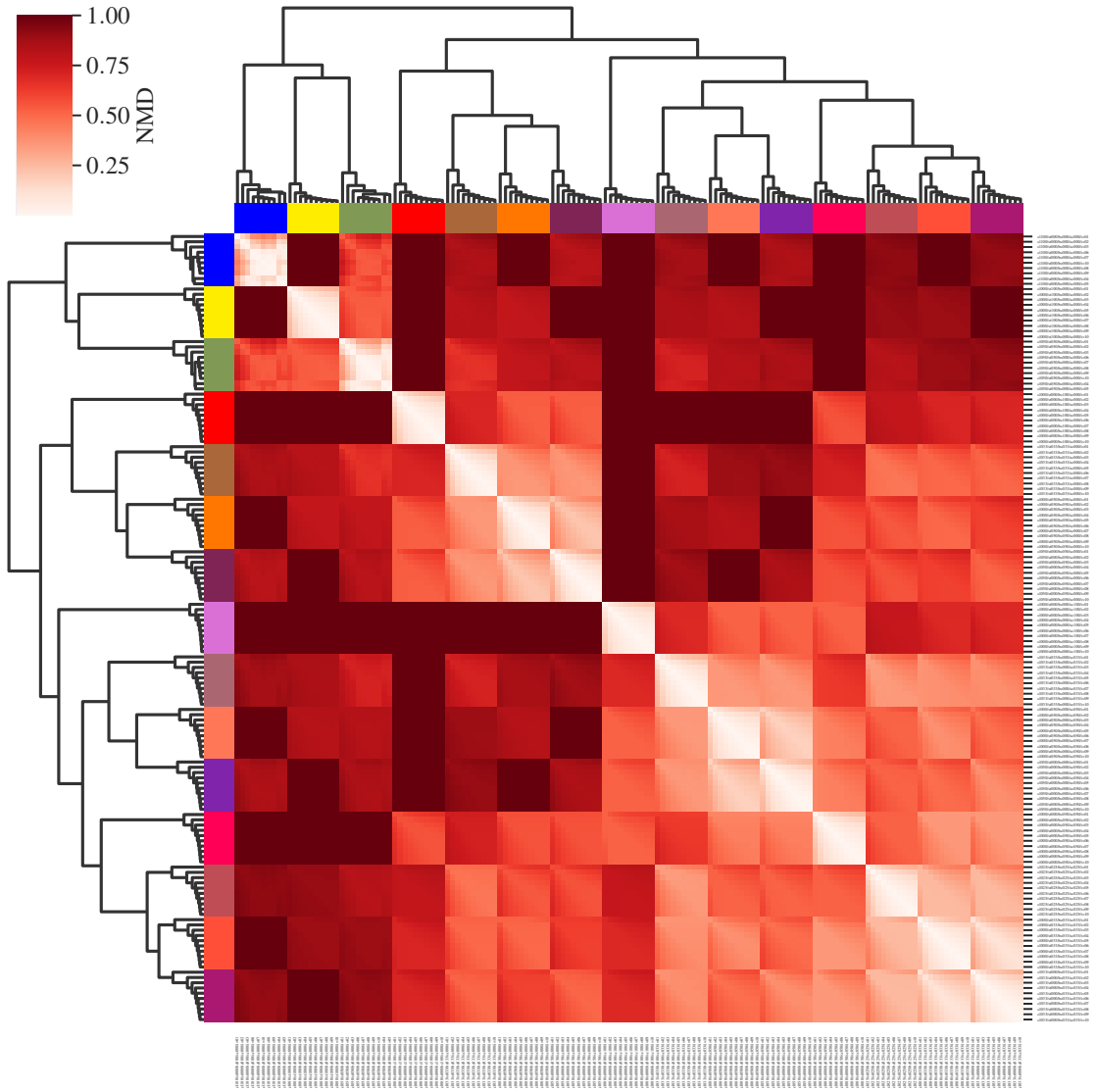
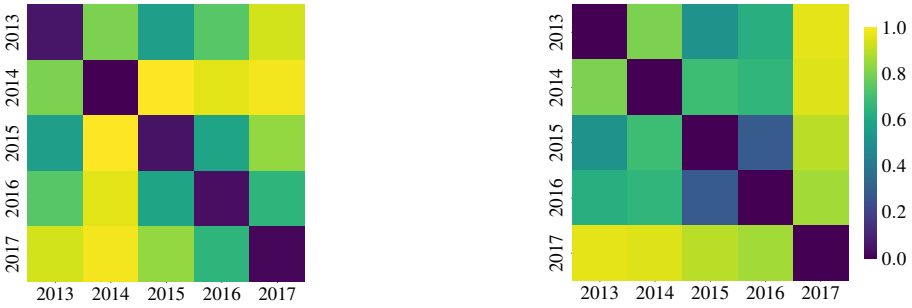


Figure 2.10: NMD values are (almost) scale-invariant (light strip along the diagonal) and correlate strongly with the number of structures that are matched across graphs (seven distinct shades of red). Row and column colors indicate model composition (mixed proportionally using blue, yellow, red, and magenta as the base colors for our structures); labels show structure counts per type and graph size (represented by i).

comparison (*cross-sectional* vs. *cross-temporal*) and the type of change (*gradual* vs. *radical*) experienced by the system we study. To illustrate radical change, we show the NMD values of the collaboration graphs (*clg*, *csi*) from 2011 to 2020 in Fig. 2.13. Both collections display the arrow of time, but self-similarity drops faster in *clg* than in *csi* from about 2015 onwards, and when comparing across collections, *csi* 2015 is most similar to *clg* 2015 but *csi* 2020 is most similar to *clg* 2017. Thus, while both communities have picked up tremendous pace in the past ten years, development in *clg* has been measurably more rapid than in *csi*.



(a) NMD values **(b)** NPD values

Figure 2.11: NMD and NPD detect similar trends, but where they differ, only NMD values are easy to interpret. Here, we compare NMD values (left) with NPD values (right) on the IBM GitHub collaboration network from Bagrow and Bollt [13].

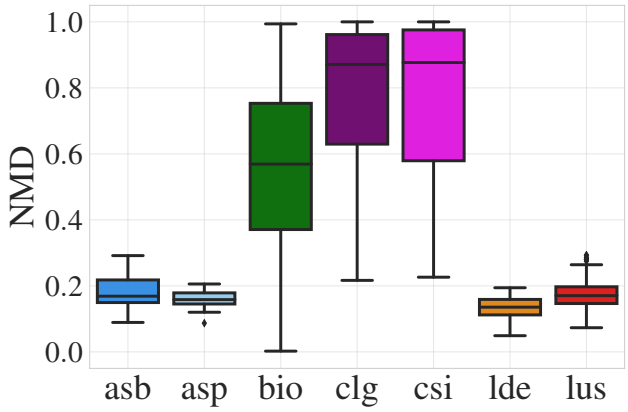
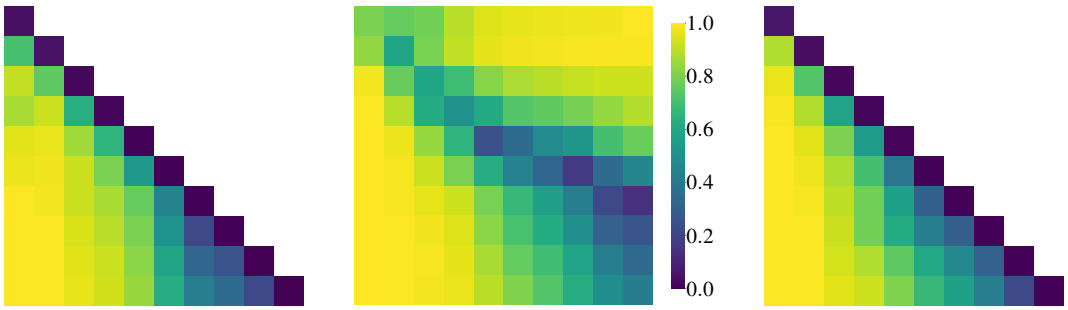


Figure 2.12: NMD values are lower for cross-temporal comparisons of systems experiencing gradual change (*asb*, *asp*, *lde*, *lus*) than for cross-temporal comparisons of systems undergoing radical change (*clg*, *csi*) or cross-sectional comparisons (*bio*).



(a) *clg* **(b)** *clg* | *csi* **(c)** *csi*

Figure 2.13: NMD values yield nuanced insights. The NMD values for the *clg* and *csi* graphs from 2011 (top/left) to 2020 (bottom/right) show the arrow of time within each collection (left, right) and the lag between *clg* and *csi* from 2015 onwards (middle).

2.7 Conclusion

We studied graph similarity assessment as a *description* problem, guided by the question "*how* are these graphs similar?". Formalizing this problem using the MDL principle, we captured the similarity of the input graphs in their *common*

model and the differences between them in *transformations* to *individual models*. Since our search space was huge and unstructured, we proposed a framework, Momo, which breaks the problem into two parts: Beppo creates graph summaries that are useful to domain experts, and Gigi discovers interpretable common models, from which we can also derive informative similarity scores. Through experiments on undirected and directed graphs of radically varying sizes from diverse domains, we confirmed that Momo works well and is near-linear in practice.

However, Momo also leaves room for improvement. For example, we would like to handle richer graph types, including weighted and attributed graphs, using encodings that fully leverage the available information. Ideally, Beppo and Gigi would discover their structure and transformation vocabularies on the fly, integrating domain-specific background knowledge in the process. An improved structure encoding might account for the overlap between structures, which is currently considered explicitly only by Gigi. Our NMD score focuses on the models of the input graphs, and a more comprehensive measure could integrate the data under these models.

Finally, MDL forces us to take a binary decision when considering structure candidates, which can result in large differences between models based on small differences between description lengths. To eliminate these artifacts and still retain interpretability, we could consider the full set of high-quality structure candidates and compress it using *structures of structures*. This could lead to an *interpretable graph kernel*, which—like overcoming Momo's other limitations—constitutes an interesting topic for future work.

Appendices

2.A Notation

For easy reference, we collect the notation used in this chapter in Table 2.3.

2.B Implementation Details

In the following, we provide implementation details for all components of Momo: Beppo, Gigi, and the NMD computation.

Beppo. Beppo has a size threshold, which allows us to stop decomposing connected components or discard generated candidates when they are too small. We set this threshold to 10 for all our experiments except when comparing NMD values with NPD values, where we set it to 3 because the input graphs are relatively small.

Table 2.3: Basic notation used in this chapter.

Symbol	Definition	Description
G_i	$= (V_i, E_i)$	Graph i with node set V_i and edge set E_i
n_i	$= \lvert V_i \rvert$	Number of nodes in G_i
m_i	$= \lvert E_i \rvert$	Number of edges in G_i
$\boldsymbol{A}_i$		Adjacency matrix of G_i
$\mathcal{A}_{ij}$, $G_i \Vert_{\mathcal{A}} G_j$		Alignment between G_i and G_j
$\mathfrak{D}$		Data
$\mathfrak{M}$		Model class
$M \in \mathfrak{M}$		Model M in model class $\mathfrak{M}$
$K(\cdot)$		Kolmogorov complexity
$ID(\cdot, \cdot)$		Information distance
$L(x)$		Number of bits to describe x using our encoding
$L_{\mathbb{N}}(x)$		Number of bits to describe x using the universal code for integers
$\log$		Binary logarithm with $\log(0) = 0$
$\lfloor x \rceil$		x rounded to the closest integer
Ω		Structure vocabulary
$s \in S$		Structure s in structure list (or set) S
$\mathcal{M} \subseteq S_1 \times S_2$		Matching between structures of S_1 and S_2
Σ		Transformation vocabulary
$\Delta_i(M_{12})$	$= M_i$	Transformation from M_{12} to M_i for $i \in \{1, 2\}$
$\delta_i(s)$	$= \varphi_i(s)$	Transformation of s morphing it into s_i for $i \in \{1, 2\}$
φ_i	$: S_{12} \to S_i$	Mapping from shared structures S_{12} to their counterparts in S_i for $i \in \{1, 2\}$
O	$= (S, F, w)$	Node overlap graph with $F = \{\{s, t\} \mid s, t \in S, s \neq t\}$ and $w((s, t)) = \mathrm{Jaccard}(s, t)$ for $s, t \in S$

When deciding whether to merge candidates due to large overlap between their node sets in the final candidate generation step, we choose our merge thresholds such that we can reduce redundancy among candidates without harming structure quality. For cliques, we set the merge threshold to 90% of the nodes. For bicliques and starcliques, we require both the left sets and the right sets of two candidates to overlap on 90% of the nodes. We do not merge stars even for large overlaps because this would result in structures of a different type, which we generate separately.

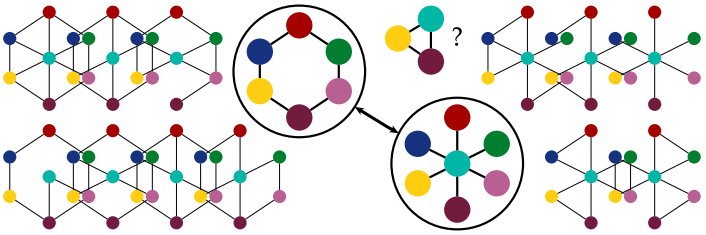
We allow Beppo to stop early if (1) it has added a given maximum number of structures to our model, or (2) we have tested a given maximum number of can-

didates *without* adding them to our model. As described at the beginning of Section 2.6, to guarantee that our summaries are interpretable, we set their maximum number of structures to 100. We set the maximum number of rejected candidates to 300, but in our experiments, this becomes relevant only for graphs from the *bio* collection. Because these graphs are relatively dense, Beppo creates many overlapping candidates, but few of them suffice to cover most of the nodes and edges. With early stopping, we can thus shorten the running time of Beppo without compromising the quality of our graph summaries.

Gigi. To speed up the computation when no node alignment is given and structures do not overlap, our implementation has a no-overlap flag which, when set, allows us to skip directly to the greedy matching (Algorithm 2.3, ll. 13–19).

NMD Computation. If we compute the NMD naïvely, it is in rare cases possible to obtain a value above 1. This occurs when the models for the two graphs are so different that encoding them individually is cheaper than encoding them using a common model and transformations, i.e., when $L(M_{12}) + L(\Delta_1, \Delta_2) > L(M_1) + L(M_2)$. As any value above 1 signals that we do not gain any bits by compressing G_1 and G_2 together, we set the NMD to 1 in this situation.

For the *bio* collection, the NMD distribution we report in Fig. 2.12 is based on structure matchings using node alignments induced by protein identities. For all other collections, the distributions reported are based on structure matchings without node alignments.



3

Multiplicity: Gragra

In Chapter 2, we explored the *descriptivity* dimension of Graphland, where we developed Momo to compare one graph to two or more other graphs by building individual and common models composed of intuitive (sub)graph structures. However, some graphs naturally occur in groups, and descriptivity alone does not suffice to characterize and compare *groups of graphs*. This motivates us to explore the *multiplicity* dimension of Graphland, such that in the present chapter, we inquire:

How can we compare a multiplicity *of graphs?*

3.1 Introduction

Differentially describing groups of graphs lies at the heart of many scientific and societal challenges. For example, neuroscientists might want to characterize brain activity in healthy subjects, elucidate how it differs from brain activity in subjects diagnosed with certain disorders or diseases (e.g., autism or Alzheimer's), and investigate whether their findings are the same across different groups of subjects (e.g., children, adolescents, and adults; or men and women). Policymakers, security experts, and epidemiologists alike could seek to understand patterns of human mobility, be it to improve the resilience of traffic infrastructure to random failures and targeted attacks, or to curb the spread of infectious diseases. And international economists might want to investigate patterns of world trade, e.g.,

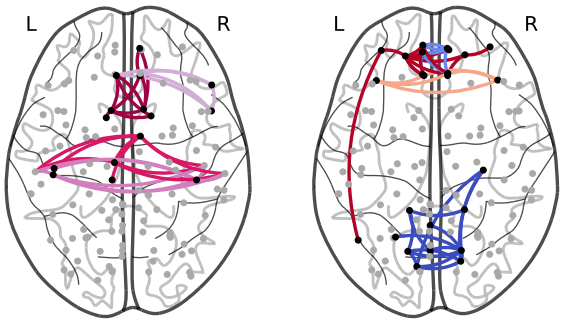


Figure 3.1: Gragra discovers common and contrastive graph patterns in noisy, heterogeneous groups of graphs, capturing, e.g., systematic similarities (left) and differences (right) between the functional brain networks of adolescents with and without autism spectrum disorder. Here, nodes represent centers of mass for brain regions from the Automated Anatomical Labeling (AAL) Atlas, and edge color classes correspond to significant subgraphs shared between (left) or specific to (right) groups, with individual edges signaling strong connectivity between regions. In the right part of the figure, reds indicate overconnectivity and blues indicate underconnectivity in autistic brains as compared to typically developed controls.

imports and exports between countries, and ask how these vary across different years and product classes.

We refer to the common task underlying these scenarios as *graph group analysis*: Given a set of graphs and a partition of this set into *graph groups*, succinctly summarize the commonalities and differences between graphs in the same group, between graphs in different groups, and between the relationships connecting the groups. We formalize graph group analysis as a *maximum-entropy modeling* problem, using *significant subgraphs* as graph patterns to factorize our probability distribution.

As a real-world example of graph group analysis, consider Fig. 3.1. Here, we show the top shared (left) and specific (right) patterns identified in resting-state functional brain networks of adolescents with and without autism spectrum disorder, where nodes in the graphs correspond to regions of interest (ROIs) from the Automated Anatomical Labeling (AAL) Atlas, and edges signal strong connectivity between regions. On the right, patterns with red edges are characteristic of autistic adolescents, and patterns with blue edges are characteristic of non-autistic adolescents. They indicate overconnectivity (reds) and underconnectivity (blues), respectively, in the brains of autistic adolescents when compared to typically developed controls. Although there is no consensus regarding the relationships between autism and neural connectivity [29, 128, 177], our method identifies graph patterns that permit neuroscientific interpretation: For example, the dark blue pattern in Fig. 3.1 indicates underconnectivity between the visual cortex, respon-

sible for processing visual information, and the lingual gyrus, involved in vision and word processing.

Graph group analysis is related to, but distinct from, several other challenges studied in the literature. *Graph classification* [30, 164, 240], for instance, assigns labels to unseen graphs by leveraging the *differences* between graph groups defined by labels in the training set. In contrast, we are interested not only in the differences but also in the *similarities* between graph groups. Graph group analysis further shares some of its motivation with *significant subgraph mining* [31, 221, 248], *graph summarization* [141, 150, 174], and *data clustering* with graphs as data points [196, 242, 277]. However, we focus on a *complete* characterization of a *set* of graphs under a *given* partition—a cornerstone of scientific discovery involving graph data.

Contributions. In this chapter, we make three contributions. First, we introduce graph group analysis as a task and formalize it as a maximum-entropy modeling problem. Second, we develop Gragra (*Gra*ph *gr*oup *a*nalysis), which jointly discovers a set of graph patterns and an assignment of these patterns to graph groups, as an algorithm to address the problem. Third, through an extensive set of experiments on a wide range of synthetic and real-world graph groups, we confirm that Gragra works well in practice.

Structure. After settling our basic notation in Section 3.2, we describe the theoretical foundations of our method in Section 3.3 and introduce our algorithm in Section 3.4. Having covered related work in Section 3.5, we demonstrate that Gragra works well in practice in Section 3.6, before concluding with a discussion in Section 3.7. We collect our notation in Section 3.A, give more details on our datasets in Section 3.B, and make all data, code, and results publicly available.[1]

3.2 Preliminaries

We consider a set $\mathcal{G} = \{G_1, \ldots, G_{|\mathcal{G}|}\}$ of $|\mathcal{G}|$ *node-aligned* graphs $G_i = (V, E_i)$ with $n = |V|$ nodes and $m_i = |E_i|$ edges, omitting the subscripts when clear from context. A partition $\Pi = \{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$ is a set of k non-empty subsets of $\mathcal{G}_i \subseteq \mathcal{G}$, called *graph groups*, of cardinalities $c_i = |\mathcal{G}_i|$, whose disjoint union is $\mathcal{G}$. Our graphs can be undirected or directed, loopy or non-loopy, and unweighted, edge-labeled, or integer-weighted, where for the purposes of our model, we treat distinct edge labels or edge weights as a set W of categories, and regard edges $e \in E_i$ as drawn from the set $\mathcal{E} = V \times V \times W$ of all possible edges. For an edge set $X \subseteq \mathcal{E}$, we denote by V_X the set of nodes incident with at least one edge in X.

[1] 10.5281/zenodo.6342823

We base our probabilistic model on the *maximum-entropy principle*, by which the distribution that best reflects a given set of constraints without introducing additional assumptions is the distribution with maximum Shannon entropy [136]. Denoting the *empirical frequency* of edge set $X \subseteq \mathcal{E}$ in group $\mathcal{G}_i$ as

$$q_i(X) = \frac{|\{(V, E) \in \mathcal{G}_i \mid X \subseteq E\}|}{c_i} , \tag{3.1}$$

the *expected frequency* of X in $\mathcal{G}_i$ under a given set of edge sets $S \subseteq \mathcal{P}(\mathcal{E})$ (for example, a set of graph patterns) is

$$p_i(X \mid S) = \mathbb{E}_f[X] = \sum_{Y \in \mathcal{P}(\mathcal{E}),\ X \subseteq Y} f(Y \mid S) , \tag{3.2}$$

where $\mathcal{P}(\mathcal{E})$ is the power set of $\mathcal{E}$, and f is the distribution satisfying

$$\operatorname*{argmax}_f \left\{ -\sum f_X \log f_X \right\} , \tag{3.3}$$

subject to linear constraints $\mathbb{E}_f[X] = q_i(X)$ for all elements in S [70]. That is,

$$f(X \mid S) = \theta_0 \prod_{Y_i \in S,\ Y_i \subseteq X} \theta_i , \tag{3.4}$$

where θ_0 and all θ_i are real-valued model parameters. Finding the distribution f is a convex problem that involves computing the expected frequency $p_i(X \mid S)$ over exponentially many elements. This is intractable if done naïvely, but there exist practical approaches that factorize p_i into a product of independent distributions [72, 73, 185].

3.3 Theory

We now lay the theoretical foundations of our method, introducing our probabilistic model, our objective function, and our statistical test. At a high level, our goal in graph group analysis is to discover a set S of *graph patterns*, i.e., edge sets of *connected subgraphs*, and an *association matrix* A assigning graph patterns to graph groups, such that S and A together reveal the similarities and differences between graphs in the same group, between graphs in different groups, and between the relationships connecting the groups. A pattern is *specific* if we assign it to only one graph group, and it is *shared* if we assign it to several graph groups. We choose which patterns to include in our model based on the information we gain from them, testing if this gain is statistically significant to rule out spurious results.

To avoid redundancy, we assign a graph pattern X to a group $\mathcal{G}_i$ iff X is *informative* for $\mathcal{G}_i$, given what we already know about all groups. More precisely, using X as a column index of A in a slight abuse of notation, we set $A_{iX} = 1$ iff X is *informative* for $\mathcal{G}_i$ under our current model (S, A). We assess whether X is informative for $\mathcal{G}_i$ by comparing the *empirical frequency* of X in group $\mathcal{G}_i$, $q_i(X)$, to its *expected frequency* in $\mathcal{G}_i$ under our current model, $p_i(X \mid S_i)$, where $S_i = \{X \in S \mid A_{iX} = 1\}$, and p_i is obtained from a practical approximation of the maximum-entropy distribution with constraint set S_i. X is *informative* for $\mathcal{G}_i$ iff $q_i(X)$ is significantly different from $p_i(X \mid S_i)$ as judged by our statistical test (detailed below), and we add X to S (and column X to A) if X is informative for *some* $\mathcal{G}_i \in \Pi$.

To identify a suitable set of graph patterns S and an adequate association matrix A, we exploit the interplay between two steps. First, we discover the best pattern X to add to S, given the current (S, A), and second, we identify the best assignment of X to graph groups for updating A, given the current (S, A) *and* the new pattern X. We now describe each of these steps in more detail.

Identifying Informative Graph Patterns. To measure the likelihood of a set $S \subseteq \mathcal{P}(\mathcal{E})$ of graph patterns, we use the *Bayesian Information Criterion* (BIC) [236],

$$\mathrm{BIC}(S) = \ell(S) + \frac{k \cdot |S|)}{2 \log |\mathcal{G}|} , \tag{3.5}$$

where $k \cdot |S|$ is the number of coefficients in our model (recall that k is the number of graph groups, i.e., the cardinality of Π), and

$$\ell(S) = \sum_i \ell_i(S) = -\sum_i \sum_{G \in \mathcal{G}_i} \log p_i(G \mid S_i) \tag{3.6}$$

is the log-likelihood of S (with $S_i \subseteq S$ derived from A), assuming that the graphs in a group are independent and identically distributed. This allows us to identify a good set of graph patterns by minimizing the BIC score, i.e.,

$$\operatorname*{argmin}_{S \subseteq \mathcal{P}(\mathcal{E})} \left\{ \mathrm{BIC}(S) \right\} . \tag{3.7}$$

Solving this problem exactly poses significant challenges in practice due to its combinatorial nature and the explosion in the number of solution candidates. Therefore, we employ a greedy search strategy, iteratively selecting the graph pattern $X \subseteq \mathcal{E}$ that best improves our current model. That is, for a given (S, A), we select the graph pattern X that maximizes our likelihood, or equivalently, maxi-

mizes the difference $\mathrm{BIC}(S) - \mathrm{BIC}(S \cup \{X\})$, which we write as

$$\Delta(X) = \ell(S) - \ell(S \cup \{X\}) - \frac{k}{2 \log |\mathcal{G}|} \,. \tag{3.8}$$

In a nutshell, the core of our approach is the procedure

$$S \leftarrow S \;\cup \left\{ \underset{X \subseteq \mathcal{E},\; \Delta(X) > 0}{\operatorname{argmax}} \; \{\Delta(X)\} \right\}, \tag{3.9}$$

by which we iteratively and greedily insert into S the pattern $X \subseteq \mathcal{E}$ that locally maximizes our information gain.

Using a model selection criterion alone, however, we cannot tell if our information gain is due to random fluctuations or due to signal, especially if we only have a limited number of samples. Thus, to avoid modeling noise, we add X to S only if its information gain $\Delta(X)$ is *statistically significant*. Therefore, we test whether we can reject the null hypothesis

$$H_0 : \; \mathrm{BIC}(S) = \mathrm{BIC}(S \cup \{X\}) \,. \tag{3.10}$$

To this end, we use Vuong's closeness test [261], a likelihood ratio test designed for model selection problems under BIC. Vuong's test statistic is defined as $2\Delta(X)$, which is asymptotically χ^2-distributed with $\mathrm{df}_\Delta(X) = \mathrm{df}\, p_i(\,\cdot \mid S \cup \{X\}) - \mathrm{df}\, p_i(\,\cdot \mid S)$ degrees of freedom. To calculate $\mathrm{df}_\Delta(X)$, we count the coefficients θ that must be changed in every distribution if we insert X into S. As we add one coefficient for X, and update at least $|X|$ edge coefficients per group, we arrive at $|X| + 1$ additional degrees of freedom.

Discovering Differential Pattern Associations. Once we have selected a new pattern $X \subseteq \mathcal{E}$ to add to S, given the current S and A, we identify a good assignment of X to graph groups $\mathcal{G}_i \in \Pi$ for updating A. Here, the significance of $\Delta(X)$, which is used to accept X into S, signals that X is informative for *some* $\mathcal{G}_i \in \Pi$, but it does not tell us for *which* $\mathcal{G}_i$. To assign X to a graph group $\mathcal{G}_i$, we hence rely on the *partial* information gain of X for $\mathcal{G}_i$,

$$\Delta_i(X) = \ell_i(S_i) - \ell_i(S_i \cup \{X\}) - \frac{k}{2 \log |\mathcal{G}|} \,. \tag{3.11}$$

Again, we use Vuong's closeness test to decide whether $\Delta_i(X)$ is significant; and if $\Delta_i(X)$ is significant for graph group $\mathcal{G}_i$, we set $A_{iX} = 1$.

Algorithm 3.1: Graph group analysis with GRAGRA

Input: Groups of graphs $\mathcal{G}_1, \ldots, \mathcal{G}_k$
Output: Set of graph patterns S, association matrix A

1 $S \leftarrow \mathcal{E}$
2 $A \leftarrow$ empty binary matrix with k rows and 0 columns
3 $C \leftarrow \big\{\{x, y\} \mid x, y \in \mathcal{E}, x \neq y, V_{\{x\}} \cap V_{\{y\}} \neq \emptyset\big\}$
4 **while** $C \neq \emptyset$ **do**
5 $\quad \hat{X}, C \leftarrow \textsc{Grow}(C)$
6 $\quad$ **if** $\exists i \in [k]$ s.t. $h_i(\hat{X})$ is significant **then**
7 $\quad\quad$ resize A
8 $\quad\quad$ $A_{i\hat{X}} = 1 \iff h_i(\hat{X})$ is significant $\forall i \in [k]$
9 $\quad\quad$ $S \leftarrow S \cup \{\hat{X}\}$
10 $\quad\quad$ estimate $p_i(\,\cdot \mid S_i)$ $\forall i \in [k]$ s.t. $A_{i\hat{X}} = 1$
11 **return** $S \setminus \mathcal{E}, A$

12 **Function** $\textsc{Grow}(C)$
13 $\quad X \leftarrow \underset{X \in C}{\operatorname{argmax}} \big\{h(X) \text{ s.t. } h(X) \text{ is significant}\big\}$
14 $\quad C \leftarrow C \cup (((V_X \times V \times W) \cup (V \times V_X \times W)) \setminus X)$
15 $\quad C \leftarrow \big\{X \in C \mid h(X) \text{ is significant}\big\}$
16 $\quad \hat{X} \leftarrow \underset{X \in C}{\operatorname{argmax}}\{h(X)\}$
17 $\quad$ **if** $h(\hat{X}) > h(X)$ **then**
18 $\quad\quad$ **return** $\textsc{Grow}(C)$
19 $\quad$ **else**
20 $\quad\quad$ **return** $\hat{X}, C \setminus \{\hat{X}\}$

3.4 ALGORITHM

Having established its theoretical foundations, we now introduce GRAGRA as an algorithm to differentially describe groups of graphs using sets of significant subgraphs. As detailed in Algorithm 3.1, GRAGRA proceeds as follows. Starting with an initial set of candidates C (l. 3), we select (l. 13) and grow (l. 14) the best candidate, and retain all significant expansions (l. 15), until we have grown X to its fullest potential (ll. 17–18). Afterwards, we test if the information gain provided by X is significant, and if so, we keep track of its graph group associations in A (l. 8), and insert X into S (l. 9). In the following, we provide more details on GRAGRA's most intricate components, candidate generation and information-gain computation, before analyzing the complexity of our algorithm.

3.4.1 Candidate Generation

At Gragra's heart lies the procedure stated in Eq. (3.9), a greedy process that iteratively selects the graph pattern candidate that best enhances our model. This *could* involve myriad searches through the exponentially-sized space of all possible (sub)graphs with nodes from V, which is computationally infeasible in most cases and also unnecessary, as most candidates will be eliminated by Vuong's test. Hence, rather than exhaustively searching for the best graph patterns, we grow graphs systematically by adding edges to graph pattern candidates.

To enable our model to infer all possible graphs, we initialize it with the set $\mathcal{E}$ of all possible edges (l. 1). As our initial graph to grow, we select the most promising graph pattern from our initial candidates, i.e., the connected triples (l. 3)

$$C = \left\{\{x, y\} \mid x, y \in \mathcal{E}, x \neq y, V_{\{x\}} \cap V_{\{y\}} \neq \emptyset\right\} . \tag{3.12}$$

The candidate growth process that follows, and is repeated for each subsequent candidate, is summarized in the function Grow (ll. 12–20).

Starting with a graph pattern X (l. 13), we explore all its expansions (l. 14),

$$\big((V_X \times V \times W) \cup (V \times V_X \times W)\big) \setminus X , \tag{3.13}$$

from which we select the best candidate pattern to grow further, as long as we gain information and our information gain is significant (ll. 15–18).

3.4.2 Information-Gain Computation

In the candidate generation process described above, we cannot afford to compute our information gain Δ exactly: The candidate growth process requires many inferences of Δ, and each inference of Δ entails many more inferences of expected frequencies p_i. Hence, Gragra instead relies on h, a practical, pessimistic heuristic for Δ. This heuristic only considers the information gain from graphs $G \in \mathcal{G}$ in which X is fully present. Starting from the exact information gain $\Delta(X)$, we arrive at our heuristic $h(X)$ as follows.

Abbreviating the constant model-cost delta as

$$z = \frac{k}{2} \cdot |S \cup \{X\}| \cdot \log|\mathcal{G}| - \frac{k}{2} \cdot |S| \cdot \log|\mathcal{G}| = \frac{k}{2} \log|\mathcal{G}| , \tag{3.14}$$

we obtain

$$\Delta(X) = \ell(S) - \ell(S \cup \{X\}) - z = -\sum_i \sum_{G \in \mathcal{G}_i} \log p_i(G \mid S) - \log p_i(G \mid S \cup \{X\}) - z \tag{3.15}$$

$$= -\sum_i \sum_{G \in \mathcal{G}_i} \log \frac{p_i(G \mid S)}{p_i(G \mid S \cup \{X\})} - z \,.$$

Constraining the sum to include only graphs in which X is fully present, we get

$$-\sum_i \sum_{G \in \mathcal{G}_i, X \subseteq G} \log \frac{p_i(X \mid S)}{p_i(X \mid S \cup \{X\})} \frac{p_i(G \setminus \{X\} \mid S)}{p_i(G \setminus \{X\} \mid S \cup \{X\})} - z \,, \tag{3.16}$$

using a factorization of p_i and G. By assuming that

$$\log \frac{p_i(G \setminus \{X\} \mid S)}{p_i(G \setminus \{X\} \mid S \cup \{X\})} \approx 0 \,, \tag{3.17}$$

and since $p_i(X \mid S \cup \{X\}) = q_i(X)$ holds, we can further simplify the above to

$$-\sum_i c_i \cdot q_i(X) \log \frac{p_i(X \mid S)}{q_i(X)} - z = \sum_i c_i \cdot q_i(X) \log \frac{q_i(X)}{p_i(X \mid S)} - z \,, \tag{3.18}$$

thus arriving at our heuristic

$$h(X) = \sum_i c_i \cdot q_i(X) \log \frac{q_i(X)}{p_i(X)} - \frac{k}{2 \log |\mathcal{G}|} \,. \tag{3.19}$$

This heuristic is computationally feasible because it involves only *one* inference of an expected frequency per graph group.

3.4.3 Computational Complexity

The computational complexity of Gragra depends on the number of candidates, which can grow to at most $|\mathcal{P}(\mathcal{E})|$. In practice, Gragra's complexity depends on the number of times we grow graph patterns, which is data-dependent and bounded by the size γ of the largest connected component observed in an input graph, as growing beyond that reduces the information gain. Multiplying γ by the initial set of candidates, Gragra achieves a complexity of $\mathcal{O}\left(\binom{n}{3}|W|\gamma\right)$ for all practical purposes, where we assume that the complexity of inferring the expected frequency is bounded.

3.5 Related Work

To the best of our knowledge, we are the first to differentially describe groups of graphs through sets of significant subgraphs that collectively capture the similarities and differences between the individual graph groups. Our method is inspired by advances in graph similarity description [Momo, 63] and explainable pattern set mining using maximum-entropy modeling [DISC, 72, 73]. However, Momo focuses on pairs and unpartitioned sets of graphs, while DISC is designed for itemset data, ignores graph structure, and does not scale on graphs. Moreover, neither method uses a statistical test to select patterns. Further related work broadly falls into two categories: statistical inference on network populations, and graph mining for groups of graphs.

Statistical Inference on Network Populations. In the statistics literature, the task of analyzing multiple graphs simultaneously is typically framed as an inference problem for network-valued random variables [82, 179, 181]. Here, Ghoshdastidar et al. [100] establish limits for distinguishing two population distributions given small sample sizes, and Lunagómez, Olhede, and Wolfe [181] propose notions of mean and dispersion for a single population of networks, where the population mean is itself a network. Maugis et al. [187] use subgraph counts to test if all graphs in a sample are drawn from the same distribution, and Signorelli and Wit [242] propose a model-based clustering approach to describe subpopulations within a population of networks. Finally, Durante, Dunson, and Vogelstein [82] extend latent-space approaches designed for single graphs to capture the probabilistic mechanism that generates multiple graphs from a single population distribution. Their model has been used to characterize and test for differences between groups of brain networks [81]—an actively studied application for which numerous statistical methods, mostly focusing on *testing* for differences, have been developed [102, 154, 165, 178, 180, 263]. Prior work in the statistics literature has focused on describing *one* network population or distinguishing *two* populations. In contrast, with Gragra, we aim to construct a *differential description* of *any* number of populations. Furthermore, we ask not only *if* these populations are different, but also *how* they are different and how they are *similar*.

Graph Mining for Groups of Graphs. In the graph mining literature, groups of graphs are studied in contexts as diverse as significant subgraph mining [175, 248], graph classification [156, 260, 273], graph clustering with graphs as data points [196], anomalous graph detection [105], and graph summarization for time series of graphs [237].

In *significant subgraph mining*, the work by Sugiyama et al. [248], who study the problem of discovering subgraphs that are statistically significantly enriched in one group of graphs but not in another, is most closely related to ours. However, this work differs both in its statistical framework and in its target output. First, in the *statistical framework* of Sugiyama et al. [248], *no model* of the input data is maintained, and hypothesis tests assess whether the *occurrence of a candidate subgraph* is independent of the group membership of the graphs in which the subgraph occurs, using *Fisher's exact test*. Consequently, the authors place particular emphasis on retaining statistical power while faced with an enormous search space and correcting for multiple hypothesis testing. In contrast, Gragra gradually builds a *maximum-entropy model* of the input data, and its hypothesis tests assess whether *adding a subgraph to our model* significantly improves the likelihood of the model, as measured by the BIC score, using *Vuong's closeness test*. Second, the *target output* of Sugiyama et al. [248] is a set of subgraphs that characterizes the *differences* between *two* graph groups, whereas with Gragra, we are after a set of subgraphs that describes the entire input data in terms of the differences *and similarities* between *any number* of graph groups. Thus, while the approach by Sugiyama et al. [248] could potentially be modified to better align with our goals, its original formulation does not address our problem of interest.

Our setup—i.e., medium-sized graphs with aligned node sets—, has also received heightened attention in the *graph classification* community, again inspired by challenges from neuroscience [156, 260, 273]. The methods that are closest to our work are *contrast subgraphs* [156] and *signal subgraphs* [260], both designed for two groups of node-aligned graphs. *Contrast subgraphs* discover the densest subgraph in the difference of the summary graphs of the input groups (obtained by adding the graphs in each group separately and then subtracting the results), where the size of this subgraph depends on a user-specified regularization parameter α. *Signal subgraphs* assume edge independence as a prior to rank edges by the p-values of an edge-wise statistical test for distributional difference (e.g., Fisher's exact test). Like *signal subgraphs*, Gragra combines ideas from structural and statistical pattern mining to produce interpretable results that—unlike *contrast subgraphs*—are built on a statistical foundation. Gragra is more exploratory and more flexible than both competitors, however, because it treats graph group description as an end in itself and can handle any number of graph groups.

3.6 Experiments

We now present an extensive evaluation of our algorithm. To this end, we implement Gragra in C++ and expose a Python interface to facilitate experimentation.

We run our experiments on Intel E5-2643 CPUs with 128 or 256 GB RAM, testing at a conservative significance level of 1×10^{-7} (or 1×10^{-5} when operating with less than 50 samples), and make all data, code, and results publicly available.[2] Our experiments revolve around two questions:

Q1 Reliability. Can Gragra reliably recover the ground truth from groups of *synthetic* graphs?

Q2 Interpretability. Does Gragra discover meaningful patterns in groups of *real* graphs?

Q1 Reliability. To assess the reliability of Gragra, we run it on groups of synthetic graphs with planted patterns. We consider three scenarios, namely,

1. summarizing *one* group of graphs,
2. differentially describing *two* groups of graphs, and
3. differentially describing *four* groups of graphs.

In all three scenarios, each graph group consists of 100 graphs with 100 nodes, and our configurations differ in their planted patterns (type, prevalence, and position) and noise levels. A detailed overview of our synthetic data configurations is given in Section 3.B.1.

For each scenario, we report the distribution of precision, recall, and F1 score, computed separately for each group of graphs, for the edges of the planted patterns across 100 graph group datasets sampled with different seeds. In all scenarios, we compare Gragra, which uses BIC with Vuong's closeness test for pattern selection, with a variant using only BIC and no statistical test to select patterns ($\text{Gragra}_{\text{BIC}}$). For configurations in the second scenario, we also compare our results with those from *contrast subgraphs* (CSG) and *signal subgraphs* (SSG), described in Section 3.5.

As shown in Fig. 3.2, $\text{Gragra}_{\text{BIC}}$ delivers good results in the four-group scenario but generally has poor precision, treating noise as signal. CSG and SSG identify only constrastive patterns, and fail even for contrastive patterns if the individual edges in planted patterns have similar occurrence probabilities across groups. Gragra, however, reliably recovers the ground truth across scenarios and configurations, which leads us to hope that it will also work well in practice.

Q2 Interpretability. To determine whether Gragra discovers meaningful patterns in groups of real graphs, we run 29 experiments on graph group data of various graph types from three domains: functional brain networks (undirected, unweighted), air transportation networks (directed, weighted), and international trade networks (directed, weighted). We compile basic statistics of these networks

[2] 10.5281/zenodo.6342823

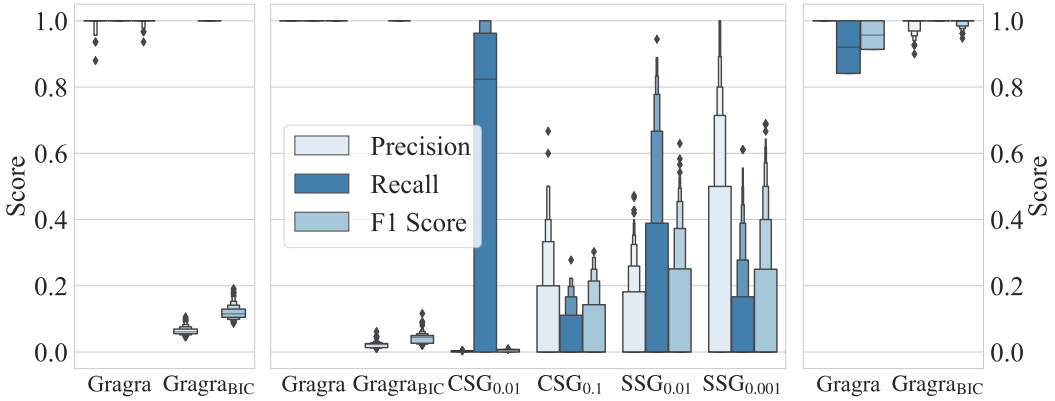


Figure 3.2: GRAGRA reliably recovers the ground truth from synthetic data. We show precision, recall, and F1 score distributions for GRAGRA, GRAGRA$_{BIC}$, *contrast subgraphs* (CSG), and *signal subgraphs* (SSG), separately for all experiments in our three different settings: one-group setting (left), two-group setting (middle), and four-group setting (right). Subscripts of CSG labels correspond to different choices of their regularization parameter α, and subscripts of SSG labels indicate different requirements for the p-values obtained from their edge-wise distributional difference test.

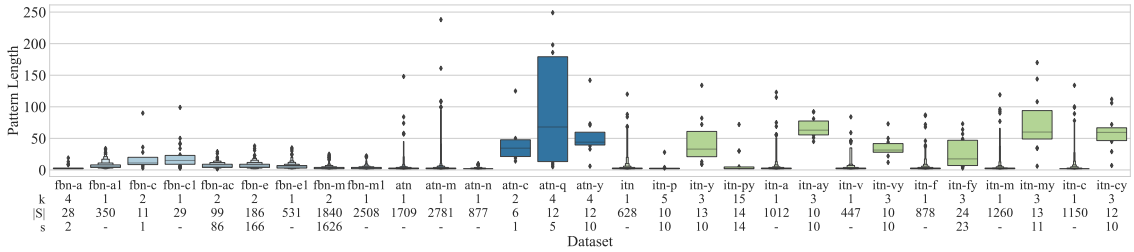


Figure 3.3: GRAGRA discovers long graph patterns in datasets with different numbers of graph groups. Here, we show the length distribution of the patterns identified in each of our experiments on real-world data, where each boxplot corresponds to a dataset. The first number below a dataset identifier states the number of graph groups k in the dataset, the second number states the total number of patterns |S|, and the third number states the number of patterns s shared between at least two graph groups.

in Section 3.B.2, and present a quantitative overview of our results in Fig. 3.3. We observe that, in line with expectations derived from theory, *more graphs* or *graphs with more potential edges*, partitioned into *fewer groups*, generally yield *more patterns*.

FUNCTIONAL BRAIN NETWORKS. *Network neuroscience* has emerged as a promising approach to understanding neurological disorders and diseases [22, 43, 94]. One of its fundamental questions is whether certain disorders are systematically associated with structural or functional connectivity alterations in the brain [257]. In particular, there is considerable uncertainty surrounding the neurological footprint of autism (and the delineation of its subtypes), and small sample sizes as well as covariates make many published findings hard to replicate [118, 145]. This calls for methods that can detect signal in the presence of considerable noise and

heterogeneity, identifying connectivity patterns that are statistically significantly associated with one or more groups of brain networks.

Motivated by this application, we obtain graphs from preprocessed functional connectomes provided by the Autism Brain Imaging Data Exchange (ABIDE) [69]. In these graphs, each node corresponds to one of the 116 *regions of interest* (ROIs) from the Automated Anatomical Labeling Atlas [AAL, 229], and each edge indicates relatively strong connectivity between two regions, as measured by their blood-oxygen-level dependent (BOLD) signal correlation during resting-state functional magnetic resonance imaging (fMRI). To facilitate comparisons, the data is processed and grouped as described by Lanciano, Bonchi, and Gionis [156], but we remove the self-loops (corresponding to perfect self-correlations) that are present in their data.

We experiment with Gragra in four two-group settings (individuals with autism spectrum disorder [ASD] and typically developed controls [TD] in the categories *adolescents*, *children*, *eyes closed during scan*, and *males*), four one-group settings (autistic individuals in each category only), and one four-group setting (autistic and non-autistic children and adolescents), operating on graphs with $m \in [1\,320, 1\,348]$ edges and graph groups $\mathcal{G}_i$ with $c_i \in [49, 420]$ graphs. Our four-group experiment identifies significant overconnectivity across multiple brain regions as characteristic of ASD children versus all other groups, paralleling the neuroscience literature [204, 250]. However, as shown in Fig. 3.4, most of the patterns we identify in the two-group setting yield similar information gains across both groups (left), and there is significant structure to be exploited even *within* individual groups (right). This indicates that the differences between autistic and non-autistic brains in the settings under study are rather subtle, and that there is considerable heterogeneity also in the one-group data. To explore this heterogeneity and delineate neurosubtypes of autism [cf. 122], our results could be used as inputs to multivariate subgroup discovery or clustering algorithms, where Gragra would effectively serve as a dimensionality reduction technique.

Air Transportation Networks. We obtain data on passenger flows between domestic airports in the United States for each month over the sixteen years from January 2005 to December 2020 from the website of the Bureau of Transportation Statistics (BTS) [44]. Restricting our analysis to United States mainland airports and carriers classified as national (100 million to 1 billion USD revenue in the previous year) or major (over 1 billion USD revenue in the previous year), we create one air transportation network per year, month, and carrier class. To this end, for each year and month, we aggregate the passenger flows between two airports by

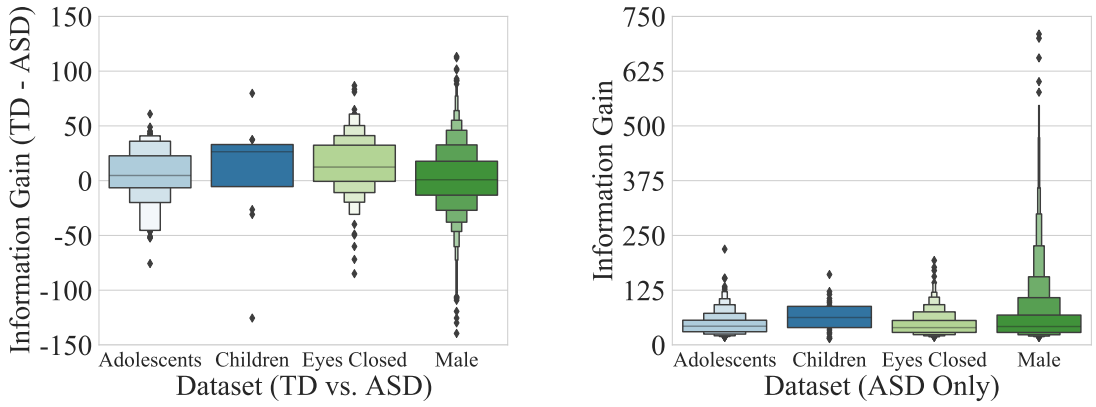


Figure 3.4: GRAGRA unveils shared and contrastive patterns in noisy and heterogeneous graph data. Here, we display the distribution of information gain differences per pattern in the two-group setting (left), and the distribution of information gains per pattern in the one-group setting (right), for our experiments on functional brain networks.

carrier class and filter edges corresponding to fewer than 3 000 passengers, which leaves edges between $n = 300$ airports (identified by three-letter IATA codes). Excluding graphs with fewer than $n-1 = 299$ edges, we arrive at 374 graphs, whose edges we discretize into ten weight categories using equal-width binning.

We are interested in discovering patterns that are shared across all graphs, identifying structures of connected routes that are specific to individual carrier classes, and unveiling both seasonal and temporal trends. Therefore, we run GRAGRA in six different settings: on all graphs as one group, on the graphs corresponding to each carrier class separately, on all graphs with carrier classes as groups, on all graphs with quarters as groups (starting from December to capture the winter holiday season), and on all graphs with consecutive four-year intervals as groups. Thus, our setup contains graphs with $m \in [335, 3\,533]$ edges and graph groups $\mathcal{G}_i$ with $c_i \in [86, 374]$ graphs. In Fig. 3.5, we depict a subset of our results from the experiments involving the distinction between carrier classes. GRAGRA reveals an air transportation backbone jointly serviced by both carrier classes (middle), and it uncovers routes that are characteristically served by national or major carriers (left and right). Overall, we find that patterns corresponding to national carrier routes are often smaller and cover shorter distances than those corresponding to major carrier routes, mirroring the relatively smaller role of national carriers in the air traffic market.

INTERNATIONAL TRADE NETWORKS. We obtain data on international trade flows from the website of the World Integrated Trade Solution [270] provided by the World Bank, for the thirty years from 1989 to 2018 (inclusive). The raw data correspond to exports of goods between (mostly) countries, classified using the Harmo-

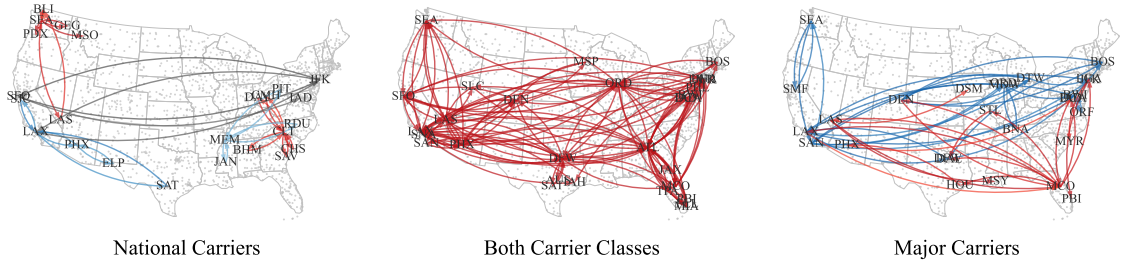


Figure 3.5: Gragra discovers large, meaningful graph patterns. Here, we depict some of the patterns discovered in the air transportation networks of national carriers (left, five patterns shown), major carriers (right, two patterns shown), and both carrier classes (middle, one pattern shown). Gray nodes represent airports, and node labels identify airports contained in at least one of the displayed patterns by their three-letter IATA codes. Directed edges represent flight segments, and edge colors are proportional to their weight bins, following different color maps (reds, blues, or grays) where necessary to make them visually distinguishable. All drawn patterns are among the top fifteen in terms of information gain for their respective experiment, and the pattern in the middle is the top shared pattern, corresponding to the United States air transportation backbone.

nized System at the four-digit level (HS-4), whose trade values we aggregate per (source, destination, HS-4 code) triple. For each year and HS-4 code, we construct one directed, weighted graph with (roughly) countries as nodes and exports as edges, discretizing the edge weights into ten categories using equal-width binning. We eliminate all trade entities above the country level but retain trade entities below the country level (and countries that do not exist anymore) if they have an ISO3 code. Restricting our attention to the WITS product groups *Animals*, *Vegetables*, *Food Products*, *Minerals*, and *Chemicals*, we arrive at 3 976 graphs with $n = 250$ nodes and at least $n - 1 = 249$ edges.

Leveraging the richness of our data, we ask not only what graph patterns are characteristic of international trade as a whole, but also what structures emerge when we group trade networks by product class, ten-year interval, or product class *and* ten-year interval. As Gragra allows us to inspect our data at different scales, we further investigate the trade patterns it unveils when considering each product class separately, either treating all graphs from one product class as one group or splitting them by ten-year interval. Thus, we run our experiments on graphs with $m \in [256, 11\,415]$ edges and graph groups $\mathcal{G}_i$ with $c_i \in [70, 3\,976]$ graphs. In Fig. 3.6, we illustrate five patterns discovered in the experiments that explore all graphs together, grouped by product class and ten-year interval. Although the input consists of fifteen classes, Gragra discovers not only meaningful patterns but meaningful patterns *with meaningful assignments* to graph groups that, as highlighted by the pattern labels in Fig. 3.6, can be summarized succinctly. Across all experiments, we observe that the patterns yielding the largest infor-

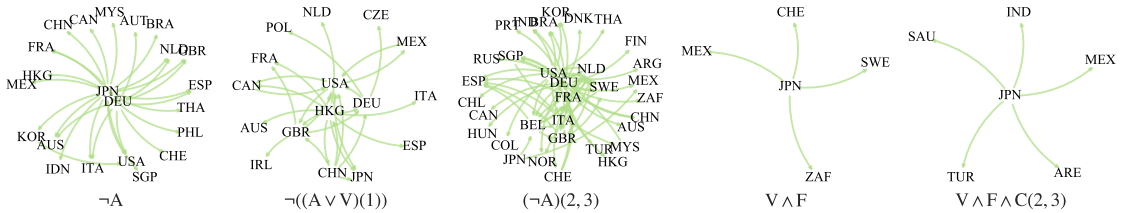


Figure 3.6: GRAGRA mines differential descriptions even when many graph groups are given as input. Here, we show the top five graph patterns identified in the international trade networks when split by product class and decade (fifteen graph groups in total). Nodes correspond to countries, which are represented by their ISO3 country codes. Directed edges correspond to trade flows between the countries, where the edge weights in all displayed patterns fall into the top weight bin. The patterns are labeled by rules identifying the graph groups in which they occur, with letters corresponding to the first letter of a product group, and numbers corresponding to the position of a ten-year interval. For example, the third pattern, labeled $(\neg A)(2, 3)$, occurs in all product classes except for *Animals*, in the second and the third ten-year interval, i.e., in $[99, 19)$.

mation gains are often composed entirely of edges in the top two weight bins. This suggests that the ranking of exporter-importer pairs is most stable on the upper end of the trade-value spectrum, which aligns with interdisciplinary research findings that international trade is highly stratified [88, 176, 232].

3.7 CONCLUSION

We studied the *graph group analysis* problem: Given a set of graphs and a partition of this set into *graph groups*, succinctly summarize the commonalities and differences between graphs in the same group, between graphs in different groups, and between the relationships connecting the groups. We introduced GRAGRA as an algorithm to solve the problem, which uses maximum-entropy modeling, paired with a model-selection criterion and a statistical test, to jointly discover a set of significant subgraphs, called graph patterns, and an assignment of these patterns to graph groups. In our experiments, we demonstrated that GRAGRA differentially describes synthetic and real-world graph groups, even when faced with heterogeneity, noise, or large group numbers. As a byproduct, we introduced two novel datasets of node-aligned graphs, which might be of independent interest to the graph mining community.

However, our work also has limitations. First of all, we modeled edge weights as categories, which works well for binned edge weights in practice but is theoretically dissatisfying. Therefore, a natural enhancement of GRAGRA would be able to handle real edge weights, possibly using a maximum-entropy model on its edge weight distribution. Second, we tested all our graph patterns at the same alpha level. While this is theoretically defensible, given that we combine our statistical test with a model selection criterion, dynamically adjusting our alpha level might be an option worth exploring. Finally, GRAGRA is currently limited to groups of

Table 3.1: Basic notation used in this chapter.

Symbol		Definition	Description
$\mathcal{G}$	$=$	$\{G_1, \ldots, G_{\lvert\mathcal{G}\rvert}\}$	Set of graphs
Π	$=$	$\{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$	Partition of $\mathcal{G}$ into k equivalence classes
k	$=$	$\lvert\Pi\rvert$	Number of equivalence classes in partition Π
$\mathcal{G}_i$	$\in$	$\Pi, \quad \mathcal{G}_i \subseteq \mathcal{G}$	Group of graphs i
c_i	$=$	$\lvert\mathcal{G}_i\rvert$	Number of graphs in $\mathcal{G}_i$
G_i	$=$	(V, E_i)	Graph i with node set V and edge set E_i
n	$=$	$\lvert V\rvert$	Number of nodes (identical across graphs)
m_i	$=$	$\lvert E_i\rvert$	Number of edges in G_i
γ			Largest-connected-component size of $G \in \mathcal{G}$
W			Edge categories or discrete edge weights
$\mathcal{E}$	$=$	$V \times V \times W$	Set of all possible edges
$\mathcal{P}(\mathcal{E})$	$=$	$\{X \mid X \subseteq \mathcal{E}\}$	Power set of $\mathcal{E}$
X	$\subseteq$	$\mathcal{E}$	Set of edges
V_X	$=$	$\{v \in V \mid \exists e \in X : v \in e\}$	Set of nodes incident with an edge in X
$q_i(X)$	$=$	$\frac{\lvert\{(V,E)\in\mathcal{G}_i \mid X\subseteq E\}\rvert}{c_i}$	Empirical frequency of X in $\mathcal{G}_i$
S	$\subseteq$	$\mathcal{P}(\mathcal{E})$	Set of edge sets
$p_i(X \mid S)$	$=$	$\mathbb{E}_f[X] = \sum_{Y\in\mathcal{P}(\mathcal{E}),\ X\subseteq Y} f(Y \mid S)$	Expected frequency of X in $\mathcal{G}_i$ under S
f			Maximum-entropy distribution
θ_i			Real-valued model parameter
A			Association matrix
S_i	$=$	$\{X \in S \mid A_{iX} = 1\}$	Set of graph patterns associated with $\mathcal{G}_i$
$\ell(S)$			Log-likelihood of S
$\Delta(X)$	$=$	$\ell(S) - \ell(S \cup \{X\}) - \frac{k}{2\log\lvert\mathcal{G}\rvert}$	Original maximization objective
$h(X)$			Heuristic approximation of $\Delta(X)$

node-aligned graphs, and extending it to other graph types constitutes an open opportunity for future work.

Appendices

3.A Notation

For easy reference, we collect the notation used in this chapter in Table 3.1.

3.B Dataset Details

In the following, we provide further information on the synthetic data and the real-world data used in our experiments.

3.B.1 Synthetic Data

For each configuration from Table 3.2, we generate 100 graph group datasets with $k \in \{1, 2, 4\}$ graph groups. Each group consists of 100 graphs with $n = 100$ nodes (labeled from 0 to 99), and edges are sampled randomly using a $G(n, p)$ random

Table 3.2: Synthetic graph group configurations. k is the number of groups, p is the edge probability in a $G(n, p)$ random graph model, P is the pattern (*cl*ique, *st*ar, or *bi*clique), and $|P|$ is the size of (the node equivalence classes in) the pattern. *Prevalence* is the occurrence probability of the pattern in the graph group, *position* is the label of the first node in the pattern, and *t* indicates the pattern *t*ype, i.e., whether it is *s*hared, *o*verlapping, or *c*ontrastive between graph groups.

k	p	P(|P|)	**Prevalence**	**Position**	t
1	0.2 0.1	$\begin{bmatrix} \mathrm{cl}(5) \\ \mathrm{st}(1,9) \\ \mathrm{bc}(5,5) \end{bmatrix}$	$\begin{bmatrix} 0.2 & 0.2 & 0.2 \end{bmatrix}^T$ $\begin{bmatrix} 0.1 & 0.2 & 0.3 \end{bmatrix}^T$	$\begin{bmatrix} 0 \\ 5 \\ 15 \end{bmatrix}$	–
2	0.2	st(1, 9)	$\begin{bmatrix} 0.2 & 0.2 \end{bmatrix}$ $\begin{bmatrix} 0.2 & 0.4 \end{bmatrix}$ $\begin{bmatrix} 0.4 & 0.4 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 10 \end{bmatrix}$	s c c
2	0.2	cl(5)	$\begin{bmatrix} 0.2 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \end{bmatrix}$ $\begin{bmatrix} 0 & 2 \end{bmatrix}$ $\begin{bmatrix} 0 & 5 \end{bmatrix}$	s o c
4	0.2 0.1	$\begin{bmatrix} \mathrm{cl}(5) \\ \mathrm{cl}(5) \\ \mathrm{st}(1,9) \\ \mathrm{st}(1,9) \\ \mathrm{bc}(5,5) \end{bmatrix}$	$\begin{bmatrix} 0.2 & 0.2 & 0 & 0 & 0 \\ 0 & 0.2 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 & 0.2 \end{bmatrix}^T$ $\begin{bmatrix} 0.1 & 0.2 & 0 & 0 & 0 \\ 0 & 0.1 & 0.2 & 0 & 0 \\ 0 & 0 & 0.3 & 0.2 & 0 \\ 0 & 0 & 0 & 0.3 & 0.2 \end{bmatrix}^T$	$\begin{bmatrix} 0 \\ 5 \\ 10 \\ 20 \\ 30 \end{bmatrix} * 4$	$\begin{bmatrix} c \\ s \\ s \\ s \\ c \end{bmatrix}$

graph model, edge probability $p \in \{0.1, 0.2\}$, and different seeds. We then plant *cliques* (i.e., complete graphs) of size 5, *stars* (i.e., one hub node connected to pairwise nonadjacent spoke nodes) of size 10, and balanced *bicliques* (i.e., two equally-sized independent node sets A and B such that every node in A is connected to every node in B) of size 10 as patterns into these random graphs, using the prevalence and position parameters given in the fourth and fifth columns of Table 3.2. Here, each column in the prevalence and position matrices corresponds to a graph group, and repeated columns in the four-group setting are condensed as $[\cdot] * 4$.

For example, for the second one-group setting (Table 3.2, row 2), we plant a clique starting at node 0 with prevalence 0.1, a star starting at node 5 with prevalence 0.2, and a biclique starting at node 15 with prevalence 0.3, into 100 graphs generated using $G(100, 0.1)$.

As described in Section 3.6 and mirrored in the layout of Table 3.2, we distinguish three scenarios: the one-group, the two-group, and the four-group scenario. In each scenario, we evaluate the performance of GRAGRA, GRAGRA$_{\mathrm{BIC}}$, and—in the two-group setting—its competitors (contrast subgraphs and signal subgraphs),

using precision, recall, and F1 score for the edges of the planted patterns. We compute these statistics based on the edge sets of the planted patterns for each graph group dataset separately, and report the resulting distributions in Fig. 3.2.

3.B.2 Real-World Data

We use real-world data from three different domains: functional brain networks (fbn), air transportation networks (atn), and international trade networks (itn). Functional brain networks are modeled as undirected, unweighted graphs, whereas both air transportation networks and international trade networks are modeled as directed, weighted graphs, with ten discrete weight categories created using equal-width binning.

The *functional brain network data* stem from the Autism Brain Imaging Data Exchange (ABIDE). In the graphs representing these data, each node corresponds to a region of interest (ROI) from the automated anatomical labeling (AAL) atlas, and each unweighted, undirected edge corresponds to a relatively strong blood-oxygen-level dependent (BOLD) signal correlation between the time series of these regions obtained during a resting-state functional magnetic resonance imaging (fMRI) scanning session. Here, our data consists of one graph per subject. Subjects can be partitioned by their *diagnostic status* (either ASD if diagnosed with autism spectrum disorder or TD if typically developed), and they can be grouped or selected by other attributes, such as *sex* (the only options being male and female), *age*, or *scanning modality* (eyes open or eyes closed).

The *air transportation network data* are taken from the website of the Bureau of Transportation Statistics (BTS). In the graphs representing these data, each node corresponds to an airport in the United States, and each weighted, directed edge corresponds to the volume of a passenger flow. Here, our data consists of one graph per *carrier class* and *month* from 2005 to 2020 (374 graphs in total).

The *international trade network data* are sourced from the World Integrated Trade Solution (WITS) provided by the World Bank. In the graphs representing these data, each node corresponds to a country (or similar unit), and each weighted, directed edge corresponds to the value of a trade flow. Here, our data consists of one graph per *product group* (Animals, Vegetables, Food Products, Minerals, or Chemicals) and *month* from 1989 to 2018 (3 976 graphs in total).

We run Gragra on different subsets and splits of our datasets, as shown in the three sections of Table 3.3.

Table 3.3: Real-world graph group data used in our experiments. n is the number of nodes, $[m]$ specifies the range of the number of edges per graph, k is the number of graph groups, and $[c_i]$ specifies the range of the graph group cardinalities. *TD* stands for *Typically Developed*, and *ASD* stands for *Autism Spectrum Disorder*. For the brain networks, which are sparsified during preprocessing, we use a minimum support of 2, and for the airline transportation networks, we use an adaptive threshold of 0.1 times the cardinality of the smallest group in the experiment for sparsification. In all experiments, we use Vuong's test at a conservative significance level of 1×10^{-7} (or 1×10^{-5} when operating with less than 50 samples). (Table continued on next page.)

Dataset	Description	k	$[c_i]$
	Functional Brain Networks (undirected, unweighted) $n = 116$; $m \in [1\,320, 1\,348]$		
fbn-a	TD vs. ASD, age $[15, 20]$	2	$[116, 121]$
fbn-a1	ASD, age $[15, 20]$	1	$[116]$
fbn-c	TD vs. ASD, age $\leqslant 9$	2	$[49, 52]$
fbn-c1	ASD, age $\leqslant 9$	1	$[49]$
fbn-ac	TD vs. ASD $\times$ a vs. c	4	$[49, 121]$
fbn-e	TD vs. ASD, eyes closed	2	$[136, 158]$
fbn-e1	ASD, eyes closed	1	$[136]$
fbn-m	TD vs. ASD, males only	2	$[418, 420]$
fbn-m1	ASD, males only	1	$[420]$
	Air Transportation Networks (directed, weighted) $n = 300$; $m \in [335, 3\,533]$		
atn	all (2005–2020)	1	$[374]$
atn-m	major carriers	1	$[191]$
atn-n	national carriers	1	$[183]$
atn-c	carrier classes	2	$[183, 191]$
atn-q	quarters $[12, 3, 6, 9)$	4	$[92, 95]$
atn-y	four-year intervals	4	$[86, 96]$

Table 3.3: Real-world graph group data used in our experiments. n is the number of nodes, $[m]$ specifies the range of the number of edges per graph, k is the number of graph groups, and $[c_i]$ specifies the range of the graph group cardinalities. For the international trade networks, we use an adaptive threshold of 0.1 times the cardinality of the smallest group in the experiment for sparsification. In all experiments, we use Vuong's test at a conservative significance level of 1×10^{-7} (or 1×10^{-5} when operating with less than 50 samples). (Table continued from previous page.)

Dataset	Description	k	$[c_i]$
	International Trade Networks (directed, weighted) $n = 250$; $m \in [256, 11\,415]$		
itn	all (1989–2018)	1	[3 976]
itn-p	product class	5	[210, 1 530]
itn-y	ten-year intervals	3	[1 314, 1 332]
itn-py	product class × intervals	15	[70, 510]
itn-a	animals	1	[210]
itn-ay	animals in intervals	3	[70, 70]
itn-v	vegetables	1	[796]
itn-vy	vegetables in intervals	3	[247, 262]
itn-f	food products	1	[1 137]
itn-fy	food products in intervals	3	[377, 380]
itn-m	minerals	1	[330]
itn-my	mineral in intervals	3	[110, 110]
itn-c	chemicals	1	[1 530]
itn-cy	chemicals in intervals	3	[510, 510]

4

Complexity: Hyperbard

Our contributions in Chapters 2 and 3 followed the rules: We adopted the standard structure of a graph mining paper and worked with graph data as if there was only one way—*our* way—to define it. In doing so, we *reduced* two types of complexity: the complexity of the *data* we sought to capture in our graph representations, and the complexity of the *community* in which we conducted our research. While this expectably produced the expected results (A* conference papers), it also nurtured a nagging question:

What if we embraced complexity *instead?*

DRAMATIS PERSONÆ

Authors. Reviewer, a reader. } Persons in the Induction.
Creature, a curious mind.
Hyperbard, a faun, sovereign of spirits.
Graph, a gentle spirit.

Professor, Senior Researcher, Colleague. } Part of the Community.
Tutor, Secretary, Deadlines. } Serving the Community.

Scene.—*Sometimes in the Community; and sometimes in the forest.*

INDUCTION.

Scene I.—*Between submission and decision.*

Enter Reviewer *and* Authors.

Rev. What is this? Is this not against the rules?
Auth. The columns? These are only simple tables.
They serve to help us implement blank verse.
The script-sized numbers count the spoken lines,
They disappear when folks use prose at times.
We introduce a novel dataset,
With full documentation as Appendix.
Raw data stem from all of Shakespeare's plays [195],
We model them as graphs in many ways,
And demonstrate representations matter.
The data readily accessible [65],
All code is publicly available [66].
What follows, to avoid redundancy,
Conveys our main ideas, as you will see
A tragedy in the Community.

ACT I.

Scene I.—*The Community.* Professor*'s office.*

Enter Senior Researcher *and* Tutor*, bearing a barrow. On the barrow, a swooning* Creature*, feeble but breathing.*

Tut. They must have hit a rock while on our problem.
Sen. R. Did they get hurt? Who are they, anyway?
They put down the barrow.
Enter Professor *and* Secretary.
Prof. What is this fuss? Did they get an appointment?
Another rescue? Do they know to code?
Tut. Should we employ them?
Prof. What, you mean by contract?
Sen. R. It seems that they are really good with graphs.
Prof. All right—
Creature *moves.*
Tut. Be quick, they wake!
Sec. I'll get the forms.
Exit Secretary.
Creature *shuffles, sighs, and sits up.*
Prof. Welcome to the Community!
[They smile generously.]
Cre. The what?
Where am I—Why is everything so clean? Wasn't I chasing bugs, out in the woods? Or roaming pastures, playing in the mud? I fail to recollect; I must be dreaming. So is this but a nightmare? Or a prison?
Am I a hostage?
Prof. Fellow, you are free!
Re-enter Secretary*, handing* Professor *the forms.*
Prof. Just sign here, will you?
They point to a field in the forms.
Creature *signs.*
Prof. and Sen. R. [in synchrony] Welcome to your PhD!
Exeunt.

Scene II.—Creature*'s office.*

Enter Creature*, closing the door. They pace about the room, then settle before the window.*

Cre. So here I stand; and I can do no other? Little do I remember of my roots. Well, elsewhere sure they lie, but must I cut them? How will I learn this play, and play my part?
Knocking.
Cre. Come in!—*[Aside]* Stay out!
Enter Colleague.
Col. Hello, how are you? You must be the new one!
You work on graphs, or that's what I've been told?
They said you came from outside, from the forest.
Well, better not go back—here, we do trees.
Cre. What does that mean?
Col. We like to operate with clear-cut questions,
Employing very powerful abstractions.
To be successful, publish many units,
At top-ranked venues, making single points.
Evaluate on standard datasets,
And over-promise, then, to hedge your bets.
Cre. So, this is science?
Col. It is how things work.
Exit Colleague.
Enter Professor *with* Deadlines.
Prof. So let me introduce you to your guardians.

We call them Deadlines—never mind the name.
They form the circle of scientific life,
And soon will be your greatest motivators.
As papers pave your path to graduation,
Your thinking becomes music to their beat.
Cre. But why?
First Dea. Why what?
Sec. Dea. It's pressure making diamonds.
Third Dea. We set incentives, we're just here to help!
Prof. I'll leave you with them, then, you'll get familiar.
And when you're done, make sure to put my name.
Exit. Deadlines *surround* Creature*, who shakes.*
Cre. Fie, get thee off me!
First Deadline *comes closer, breathing down* Creature*'s neck. The* Creature *freezes.*
Cre. I said no!
They strike at First Deadline*, then faint in fatigue.* First Deadline *staggers and retreats. All other* Deadlines *disappear into the distance.*

Scene III.—*The forest, in* Creature*'s dream.*

Enter Hyperbard*, with a lute.*

Hyp. What beauty are these woods! In every tree
Lives past enshrined and calling the observant.
The devil? Angels lie in all these details.
Look at the fragile bark, the fractal branching,
The posture, parasites—And see the leaves!
Colors, shapes, textures—all varieties.
The fauna—beetles, rodents, insects, birds,
Thriving together in their interaction.
They strike a chord on their lute.
Hyp. From all there is, let there be data!
As data points, we demarcate these trees
And put them into known categories.
They mark the selected trees with leaves of various shapes (Fig. 1).
Hyp. Each tree is full of life, full of relations,
To capture this, we need representations.
They strike another chord. Enter Graph.
Gra. You called me, honor?
Hyp. Will you, docile spirit,
Transform these trees to yield discoveries?
Gra. Your honor, master, mistress, sure I can
But there are many different transformations
Among the flurry, which one do you choose?
Hyp. Why choose but one when there exist so many?
How do we even know which one to pick?
Gra. Sir, madam, with respect, your speech is madness!
Did you not call me to produce your truth?
Hyp. What truth? Your transformations are but shadows
Of essence vested with complexity
Cast on the narrow walls of our perception
And varied as you shift and change your light.
Gra. I hear your words but struggle with their meaning.
Which output do you want me to obtain?
Hyp. To every data point associate
A set of transformations as its data.
Such that in all our future inquiries
We treat not only one but many shadows.
Each partly blind, together they create
A truer truth than commonly considered.
Gra. Your honor, as a practicality
We can't enumerate exhaustively.
Among the myriad possibilities
You still will have to choose some transformations.
Hyp. Fair spirit, as an overarching goal,
All our representations should be faithful.
Among the transformations that you see,
How do they differ systematically?
Screaming heard. Hyperbard *and* Graph *vanish.*
Creature *wakes.*

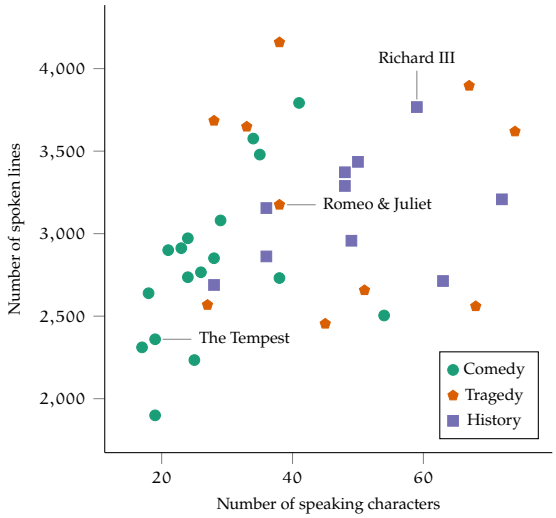


Figure 4.1: Number of spoken lines vs. number of speaking characters in the 37 plays by William Shakespeare. Each point corresponds to a play for which we provide 18 different (hyper)graph representations.

Act II.

Scene I.—*The Community. In the dining hall.*

Professor, Senior Researcher, *and* Colleague *seated at a table. Enter* Creature, *carrying a tray.*

Col. Hey fellow, please come join us, have a seat!

Creature, *jolted from their thoughts, obeys with reluctance.*

Sen. R. They told me you submitted, so, good cheer!

Col. Next time, though, try to not scare Secretary.

Prof. Now fate lies with the review gods, almighty
And they select not just for quality.
Regardless of their upcoming decision,
You'll get this published, well, eventually.

Cre. That's comforting.

Col. Well, it is how things go.

Prof. My admin work is calling.

Sen. R. And mine, too!

Exeunt Professor *and* Senior Researcher.

Awkward silence.

Cre. May I ask you something? Here in the Community, how do you get your data? You hardly go outside...

Col. What do you mean? We grab it from the shelves.
There's shelves for almost every data type.
For graphs, e.g., there's OGB [124], and SNAP [167],

KONECT [155], and TUD [194], and Netzschleuder [215],
And finally, Network Repository [230].

Cre. Hold on, you are confusing me. How do the graph shelves get their data, then?

Col. You really ask the weirdest things. I guess
They send some hunter-gatherers to catch
Or pick the graphs they find out in the wild.

Cre. You make it sound like graphs exist, for real. But are they not defined by their observers?

Col. Who are you? Not the Spanish Inquisition?
All graphs have nodes and edges, that's what matters.
Sometimes they come with weights or attributes.
Semantics—God, who cares?—graphs are abstractions,
And abstract data is our working truth.

Exeunt.

Scene II.—Creature's *office.*

In a corner, on the floor, Creature, *in contemplation.*

Cre. What canny creatures met my febrile mind. That friendly faun, the gentle spirit, exchanging such profound considerations. I wish I could have stayed a little longer—instead, I'm left to draw my own conclusions. What graph shadows could I create by shining different lights on what there is? It seems the sensible depends on the semantics.

They close their eyes, following their thoughts.

Cre. When we transform reality to math,
Graphs are but outputs, in—phenomena.

The myriad transformations that we see,
How do they differ systematically?
For now, we shall distinguish three dimensions.
First, our *semantic mapping*—Nodes and edges:
What types of entities do we assign?
Second, our *granularity*—What are
Our modeling units for semantic mapping?
And third, our *expressivity*: What more
Do we attach to all our modeling units?
Directions, weights, and multiplicities,
Or attributes and cardinalities...
What universe! *Haec facta, fiant data.*

Tracing coordinate axes with their fingers, they sigh.

Cre. All these distinctions, it appears, are known in the Community [255]. And yet, the knowledge seldom heeded—graph data shelves are filled with all these captive singular truths. We hardly hold what that free faun foresaw: For every data point, a set of transformations as its data. I wonder why.

Exit.

Scene III.—Colleague's *office.*

Colleague, *trimming a bonsai with scissors.*

Col. Alas, they really want documentation?

Creature *steps into the door frame, unnoticed.*

Col. A datasheet [99]? Well—all the world is data,

And all we care for merely data points;
They get created, updated, deleted,
And every data point plays many parts,
Its fate being seven stages. First, *motivation*
Defining purpose or specific tasks.
Then *composition*, sketching the raw data
And telling people where it was obtained,
If anything's amiss. And then *collection*,
How did we get each single data point,
And what else did we check. Then *preprocessing*,
Full of strange quirks and idiosyncrasies,
But made that it looks principled. Then *uses*,
What all things did we do, what could have been,
And what should not be done. Then *distribution*,
If, when, and how will we make data public,
Restrictions by third parties, if imposed,
And also all the laws. Last stage of all,
That ends this template documentary,
Is *maintenance* and hosting and support,
Sans updates, sans errata, sans comment.

Creature *retires, flabbergasted.*

Colleague *stashes the stunted bonsai into a shelf.*

Exit.

Scene IV.—Creature's *office.*

Enter Creature, *restless.*

Cre. This stream of observations leaves me drowning in confusion. If *Is* is not what *Ought*, how can *Is* be? What is this thing they call Community? Am I misguided, am I wrong—to doubt that I belong?
Two souls, alas, are dwelling in my breast,
And each one seeks to rule without the other.
The one a falcon, fierce and fighting fetters,
That's dreaming of faun's forest, flying free,
The other a caged chary canary,
That calmly, coyly, cheerfully chants chatters.
They open the window and balance on the window sill.
Cre. Should there be spirits roaming through the air,
I beg they lift the spell of my despair.
They jump.

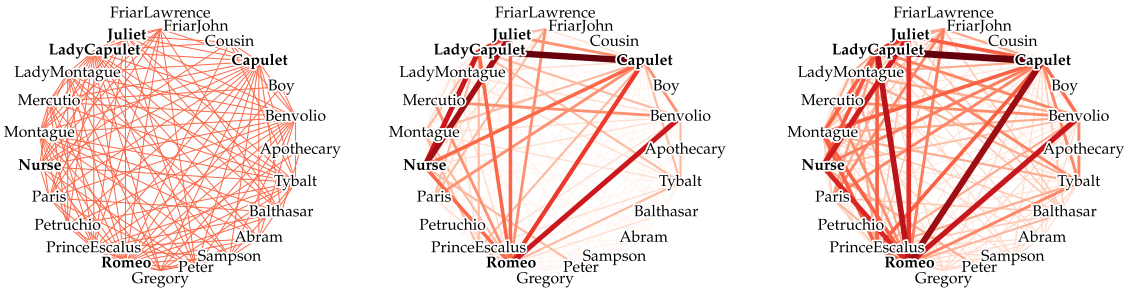
Scene V.—*The forest.*

Graph *tending to a mat of moss. On the mat,* Creature, *somnolent. Enter* Hyperbard.

Hyp. So few return once captured by Its magic!
Gra. Playing that dream was worth it, after all.
Cre. Is this a dream no more? Do you exist?
Hyp. Depends on your philosophy. But see,
My Graph says you have interesting ideas.
So tell me, how would *you* transform these trees
To bear the fruit of new discoveries?
Cre. Did you not eavesdrop on my ruminations,
Distinguishing between those three dimensions?
Semantic mapping, granularity,
And expressivity—put abstractly?
Hyp. I heard, but what does it all mean in practice?
Cre. Let's walk through an example. Take this tree:
The Tragedy of R. and J.—a play.
When modeled *Les Misérables*-y [148], the nodes

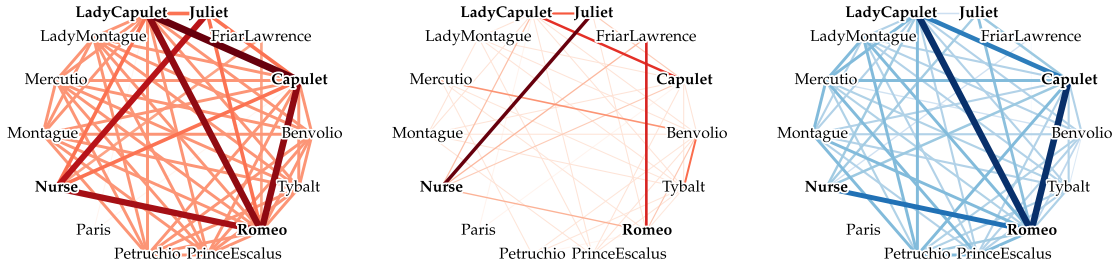
Are characters, and edges—co-occurrence.
That's one semantic mapping, hold this fixed.
Then, as to granularity, we ask
What unit should determine co-occurrence?
The first—most common—option is: a scene.
And here, much modeling ends, unfortunately:
Max simple graphs, min expressivity.
Hyp. But does this not reveal essential structure?
Cre. It smudges all the details, Fig. 2a!
Do the play's namesake heroes co-occur
No more than Montague and Capulet?
Hyp. So should we count-weight edges, Fig. 2b?
Cre. Or introduce edge multiplicity.
The multigraph perspective would allow us
To treat—Fig. 2c—co-occurrence weights.
In our setting, this could, e.g., mean
The count of spoken lines in every scene.
But that is basic expressivity—
We yet have to treat granularity.
To illustrate, in Fig. 3a, we draw
The co-occurrence only for Act III.
The Capulets and Romeo appear
To interact too much—this sparks suspicion.
Hyp. You mean we're introducing information?
Cre. And hiding what there really is to see!
The scene is far too coarse a modeling unit,
Quite often is there movement in between.
We must keep track of entries and of exits
To capture interactions faithfully.
Each part confined by any two such changes,
A *stage group*, separately defines an edge.
Accounting now for expressivity,
These edges may be binary or multi,
Or weighted by lines spoken, Fig. 3b.
The outcome, evident from Fig. 3c,
Is far from what we had initially.
Thus, even for just one semantic mapping,
And R. and J. as a specific case:
We see at least six decent transformations,
Statistics differing tremendously.
Hyp. So is this all?
Cre. Oh, that is but the start!
Thus far, we've had just characters as nodes.
One possible complaint with this approach
Is that it gives us artificial cliques.
Instead, we could in our semantic mapping
Consider also parts of plays as nodes,
Transforming plays into bipartite graphs,
Whose edges signal character occurrence.
Then granularity, Fig. 4a–b,
Concerns the nodes, but sometimes also edges.
In terms of expressivity, we could
Again attend to weights, and represent
Directionality, see Fig. 4c,
With greater ease than in the one-mode case—
To model single *speech acts*, too, as edges.
Hyp. Now, that is quite a lot—so are you finished?
Cre. Respectfully, the best is yet to come!
Conceptually, all I have just described
Can be derived from a more general model.
All graphs, regarding expressivity
Force '$\in \{1, 2\}$' on cardinality
Of edges—
Hyp. Marvelous mathematically!
Cre. But artificial, thinking critically.
The interactions in your vivid woods—
How many of them are bilateral?
This common cardinality constraint:
Let's do away with it!
Hyp. Then what remains?
Cre. A set system—a *hypergraph*, they say [27],

We visualize its power in Fig. 6.
Confusingly: All graphs are hypergraphs
But not vice versa.
Hyp. Do we need this, Graph?
Gra. Well, some found hypergraphs to be quite handy
To capture higher-order interactions [6, 15, 23].
They certainly are more intuitive
Than making cliques of multi-arities,
Or else treating relations, too, as nodes.
Cre. We can go far with *graphs* but don't know yet
Just how much further we can get with *hyper*.
Observe the beauty in these hypergraphs:
They readily entail *all* transformations!
From their perspective, what first we discussed
Are *clique expansions*, and our next ideas
Are known as *star expansions* [246]—see, in sum,
Fig. 5, and our proposals in Tab. 1.
Hyp. Things hyper, in their generality,
They seem to suit my woods quite naturally.
Gra. But sovereign, as a practicality,
There's hardly any software letting us
Compute with hypergraphs conveniently!
Hyp. and Cre. [in synchrony] Who are you, the Community?
Gra. I'm sorry.
Exeunt.



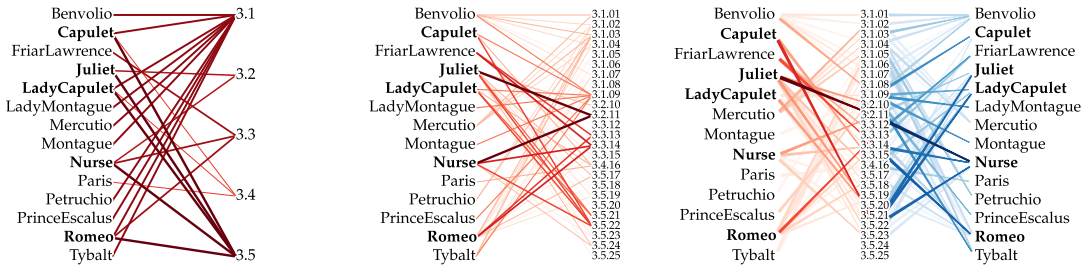
(a) ce-scene-b **(b)** ce-scene-mb ∼ ce-scene-w **(c)** ce-scene-mw

Figure 4.2: Relationships between the named characters in *Romeo and Juliet* when modeled as binary (a), count-weighted (b), and line-weighted (c) co-occurrence networks, resolved at the scene level, where we highlight the protagonists appearing in Act III, Scene V. The binary representation is a classic hairball, while the count-weighted representation and the line-weighted representation provide more nuance. In (c), the strikingly strong connection between Romeo and Capulet is partly due to Act III, Scene V, where both characters appear but *do not meet* on stage.



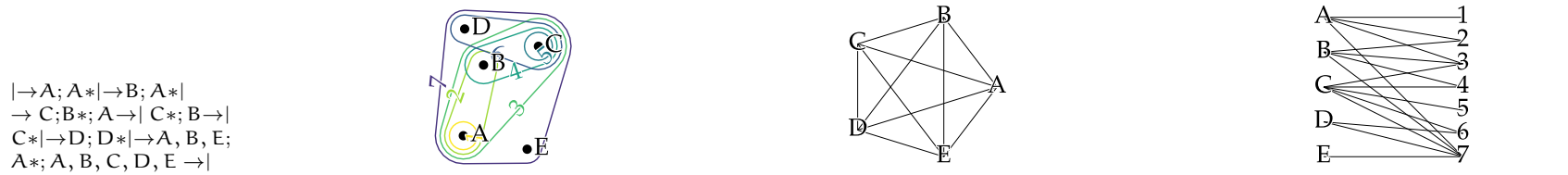
(a) ce-scene-mw **(b)** ce-group-mw **(c)** (ce-scene-mw) − (ce-group-mw)

Figure 4.3: Line-weighted co-occurrence network of the named characters in Act III of *Romeo and Juliet*, resolved at the scene level (a) and at the stage group level (b), as well as the difference network between the two (c), where we highlight the protagonists appearing in Act III, Scene V. The coarse-grained representation overestimates the co-occurrence between Romeo and Juliet's parents, i.e., Capulet and Lady Capulet (a and c), while the fine-grained representation emphasizes Juliet's bond with the Nurse and Romeo's interaction with Friar Lawrence (b).



(a) se-scene-w **(b)** se-group-w **(c)** se-speech-wd

Figure 4.4: Weighted bipartite graph of named character occurrences in Act III of *Romeo and Juliet*, resolved at the scene level (a) and at the stage group level (b), as well as the directed weighted bipartite graph resolved at the speech act level, with character nodes split up into speakers and listeners for visual clarity (c), where we highlight the protagonists appearing in Act III, Scene V. While the coarse-grained representation overestimates Romeo's role in Act III, Scene V (a), the finer-grained representation again highlights Juliet's bond with the Nurse (b), and the directed representation reveals the hierarchical structure of their communication (c).



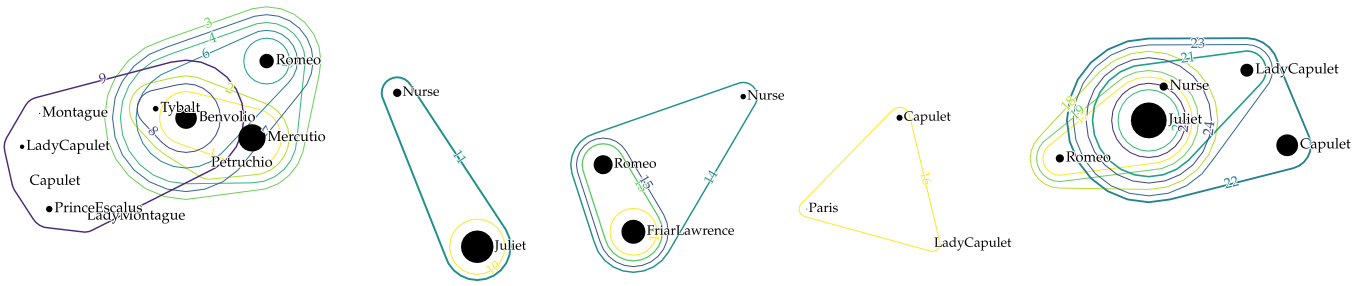
(a) Toy drama **(b)** Hypergraph ($n = 5$, $m = 7$) **(c)** Clique expansion ($n = 5$, $m = 10$) **(d)** Star expansion ($n = 12$, $m = 16$)

Figure 4.5: Relationship between hypergraphs, clique expansions, and star expansions, illustrated for a toy drama. In the toy drama, characters are capital letters, →X denotes entry, X→ denotes exit, ∗ denotes speech, | marks scene boundaries, ; marks activity boundaries, and , indicates several characters acting together.

Table 4.1: Overview of relational data representations provided with Hyperbard for each play attributed to William Shakespeare, based on the TEI simple-encoded XMLs provided by Folger Digital Texts [195]. Unidirectional arrows indicate assignment; bidirectional arrows indicate bijection. We highlight the transformations most commonly used in the literature.

Representation	Semantic Mapping		Granularity	Expressivity
ce-scene-b	Nodes ← Characters Edges ← Co-occurrence		Edges ↔ Scenes	—
ce-scene-mb				Edge order
ce-scene-mw				Edge order, edge weights
ce-group-b			Edges ↔ Stage groups	—
ce-group-mb				Edge order
ce-group-mw				Edge order, edge weights
se-scene-b	Edges ← Occurrence	Nodes (1) ← Characters Nodes (2) ← Play parts	Nodes (2) ↔ Scenes	Partial node and edge order
se-scene-w				Partial node and edge order; edge weights
se-group-b			Nodes (2) ↔ Stage groups	Partial node and edge order
se-group-w				Partial node and edge order; edge weights
se-speech-wd	Edges ← Information flow		Nodes (2) ↔ Stage groups Edges ↔ Speech acts	Partial node order; edge weights, edge directions
se-speech-mwd				Partial node and edge order; edge weights, edge directions
hg-scene-mb	Edges ← Co-occurrence	Nodes ← Characters	Edges ↔ Scenes	Edge order
hg-scene-mw				Edge order, edge weights; edge-specific node weights
hg-group-mb			Edges ↔ Stage groups	Edge order
hg-group-mw				Edge order, edge weights; edge-specific node weights
hg-speech-wd	Edges ← Information flow		Edges ↔ Speech acts	Edge directions, edge weights
hg-speech-mwd				Edge order, edge directions, edge weights

Representation abbreviations follow the pattern <model>-<aggregation>-<properties>, where model ∈ {ce: clique expansion, se: star expansion, hg: hypergraph}, aggregation ∈ {scene: play scene, group: stage group, speech: speech act}, and properties ⊆ {b: binary edges, d: directed edges, m: multi-edges allowed, w: weighted edges}. Binary multigraph representations of clique expansions (ce-∗-mb) can be transformed into weighted graph representations of clique expansions without multiedges (ce-∗-w) using edge counts as weights, but only the multigraph representations can retain order information on edges.



(a) Scene I **(b)** Scene II **(c)** Scene III **(d)** Scene IV **(e)** Scene V

Figure 4.6: Line-weighted hypergraph resolved at the stage group level, separated by scene and restricted to named characters, for Act III of *Romeo and Juliet*. Edge labels denote stage groups, edge colors indicate edge order, and node sizes and edge widths are proportional to the number of spoken lines. From (e), it is visually clear that Romeo and Juliet's parents never meet in the scene.

Act III.

Scene I.—*The forest.*

Creature, *squatting on a rock, sorting leaves.*
Enter Deadlines, *invisible, at a distance.*

First Dea. I told you! I could see from their submission!
Third Dea. The poor thing—they adore the real world!
Sec. Dea. No way that they will make me if we don't
Now intervene. So let us capture them,
Surround them, and restore the rhythmic cords,
By which we subjugate our sovereigns. Go!
They encroach on Creature, *in silence, settling in a triangle around them.*

First Deadline *sings.*

Come back to the office lands,
Don't take a chance:
Meta fair but be aware
In camera, better prepare
Fix your figures here and there;
And review two the burden bear.

Cre. Where should this music be? I know the beat.
It sounds no more? No, it begins again.

Second Deadline *sings.*

To taller skies your metrics rise;
Publish, perish, stars are made;
Do not whine, stay in line,
Otherwise your glory fade.
Dutifully use your wit
And then submit.

Exeunt all but Creature.

Cre. The ditty does remind me of my paper,
And all the future work yet to be done.
They rise.
Cre. To flee or PhD—that is the question:
Whether our destiny lies in the system,
To cling onto scientific ladder's rungs,
Or to renounce the reign of rules unwritten
And, by opposing, vanish. To flee, to think—
To think, perchance discover. Ay, there's the rub,
For once outside the pithy paywalled castles,
The giant's shoulders quickly out of reach,
For lack of funding. There's cautiousness
That crafts careers of so long strive,
And makes us rather swarm the conference streams
Than swim the savage seas so far uncharted.
Thus mellow meal the mighty mills of science,
And conscience can coerce our compliance.
Exit.

Act IV.

Scene I.—*The Community.* Creature*'s Office.*

Enter Graph, *invisible, floating, trailed by* Creature *and* Hyperbard.

Gra. What are we doing here? Did you not exit
Precisely through this window here to flee
From all these straining office-worldly fights
To think, explore, discover, to be free?
Hyp. Don't tease them, spirit! We've discussed at length
The ends to which we undertook this trip.
You've seen the acts of hunter-gatherers
As they bereave our natural habitat.
If we ignore them, they will seize control,
And colonize our forest with their views
Of graph data as unambiguous truths.
Cre. I'm confident we'll make them understand
The problem once they see our transformations.
That future work in the Community
May operate with more representations!
Enter Professor.
Prof. What's all this noise? The rules! No visitations!
Cre. Let me explain—
Prof. Save me your explanations!
I want you in my office, now! And when
We're done, this dirty stray thing must be gone!
Exeunt Professor *and* Creature.
Gra. Your honor, I foresaw this would be dangerous.
Hyp. You see their wielding of authority?
So far up in the hierarchy, so long,
And funeral their only honest feedback.
I'm not afraid, but let us maybe make
Our data case not at the top to start with.
Gra. When floating down the hall I think I saw
The perfect target for us to attack.
Hyp. What's with this war rhetoric?
Gra. I'll be back.
Exit Graph. Hyperbard *settles by the office plant.*

Scene II.—Professor*'s Office.*

Enter Professor *and* Creature.

Prof. The judgment's in, you have no time to spare:
They hand Creature *a sheet of paper.*
Prof. Accept, well done, but now in camera's near.
Cre. They're taking months, and now we're given days?
Additional experiments? But how?
No space! What should I do about R2?
Prof. That's up to you—it will not change a thing.
Cre. [Aside] That's comforting.
Exeunt.

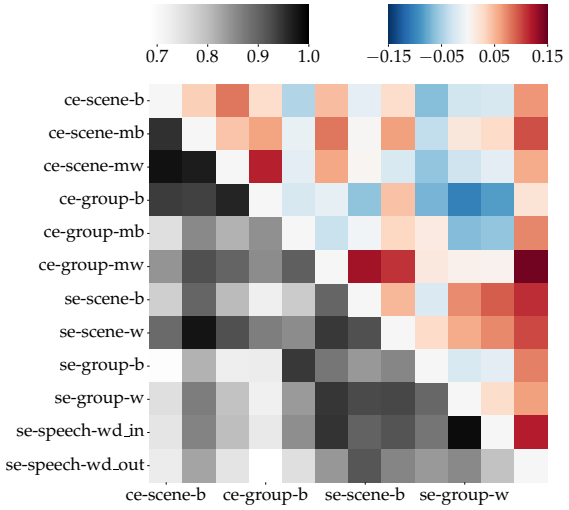


Figure 4.7: Spearman's correlations of degree rankings in the clique and star expansions from Tab. 4.1 for *Romeo and Juliet* (bottom), and residuals after subtracting the average correlations in the Hyperbard corpus (top).

Scene III.—Creature's *Office.*

Hyperbard, *engaging the office plant.*

Gra. [Within] Watch out, they'll be here any minute now!
Enter Colleague.
Col. Congrats on that acceptance—wait! Who's this?
Hyp. What's in a name? I heard you work with data,
We're colleagues, in a sense—I do the same
But mostly in the wild.
Col. So you're a hunter?
Hyp. Far off! I roam reality's realms
In search of structure that persists across
Perspectives.
Col. By perspectives, you mean tools?
Hyp. I mean representations, as for each
Phenomenon there's many paths to data.
I like to call each path a transformation,
And transformation is my tested trade.
Col. Can you elaborate? What good is that?
Hyp. Let's take a look at, you would say, *graph data.*
Imagine that you have a tree—say, R. and J.—
Col. That famous play?
Hyp. —And that you want to model
The structure of its story as a graph.
Col. Well, obviously, each character's a node
And there's an edge between two nodes in case
They co-occur in more than zero scenes.
Hyp. But this is only one of many options.
And without dwelling on the details here,
Fig. 8 reveals how even simplest things
Such as degree ranks differ with our choices.
The variations vary, too, Fig. 7,
Within a set of trees as data raw.
And—to conclude representation matters—
Less simple transformations may support
More nuanced inquiries as in Fig. 9,
Or exploration over time, Fig. 10.
Col. You worry well, but then, so why should I?
What's in it for my publication record?
Enter Professor.
Prof. What fool is this?
Col. and Hyp. [in synchrony] O that I were a fool!
Enter Creature.
Cre. Did you discuss the problem with *the* data?
Hyp. I laid it out for them, to no avail.
Col. You surely got me thinking, but—
Prof. Enough!
My patience is exhausted. Think? Produce!
[To Col.] You, give productive treatment to that thinker.
Exit Colleague *with* Hyperbard.
[To Cre.] And you, fix these few figures; faugh R2.
Exeunt.

Act V.

Scene I.—*The Community.* Colleague's *Office.*

Graph, *invisible, floating by the window.*
Enter Colleague, *carrying a jar.*
Col. Those fecund thoughts shall find their fertile soil.
They empty the content of the jar onto the bonsai.
Col. To ashes, ashes—dust to dust. Not me—
Thus goes the system, let the system be.
Exit. Graph *caresses the bonsai.*
Gra. Full many a transformation have I seen
Flatter the flora with their sovereign hand,
And sovereign's hand in spirit I'll have been
To help evaluate their promised land.
Community, defined as uninvolved
With hideous beauty born by Mother Earth
Begets solutions without problems solved
And burns the flame of wonder in Its dearth.
When culture counters nature, it prevails,
And builds its truths from rigid rigor bricks,
As myriad feeble fledglings it derails
Into the cave of engineering tricks.
For in the trenches of discovery,
To shatter shadows, meet obscurity.
Exit.

Scene II.—Creature's *Office.*

Enter Creature.
A deadline, and a deadline, and a deadline,

Creeps in this petty pace to publication,
To the last syllable of our defense.
They slew my Graph and choked my inspiration,
Our work is but a walking shadow thence.
The curiosity that drew me in
Now lies in dust. The lofty dreams I had
Of mindful monasterial devotion
To just the cause—no more. Out, out, sore studies!
Should I give up that which I know I love—to save my love for it? And go in silence, not disturbing the Machine? Or should I stay to salvage my beloved—to, once on top, speak out, let nature in?
My story, so it seems, a tragedy
In the Community:
All the world's a (hyper)graph.
Thus, I'll begin.
They write.

1. Graph data do not exist, they are defined.
2. Semantic mapping, granularity, and expressivity are key ingredients to define graph representations.
3. Many phenomena permit several graph representations.
4. Graph data context matters for graph representations.
5. Graph data representations matter for graph methods.
6. Hypergraphs are powerful.
7. Hyperbard is free.

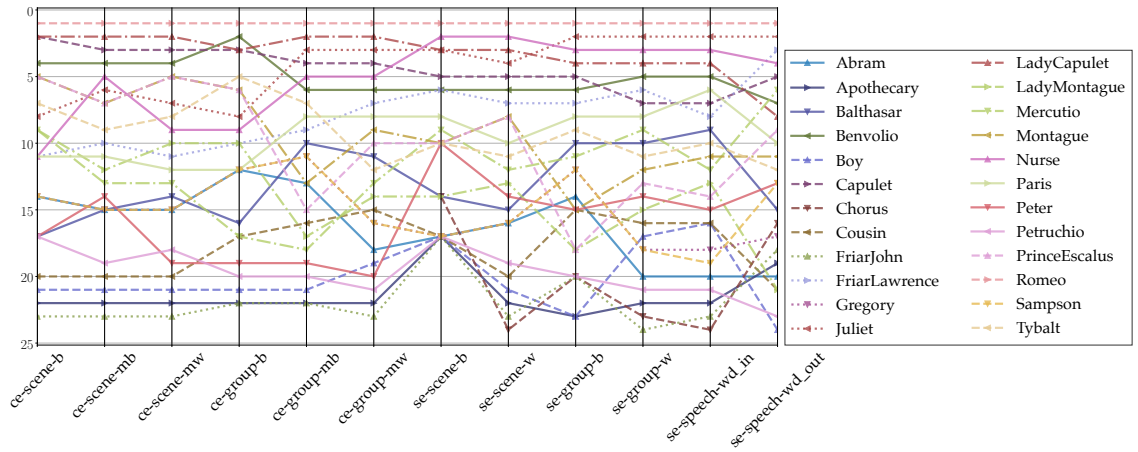


Figure 4.8: Named characters in *Romeo and Juliet*, ranked by their degree in the clique expansion (ce) and star expansion (se) representations from Tab. 4.1. We omit the se-speech-mwd representation because its ranking is equivalent to that of the se-speech-wd representation by construction. While Romeo is ranked first under all representations, the rankings differ, inter alia, in the prominence assessment of side characters, such as the Nurse or Friar Lawrence.

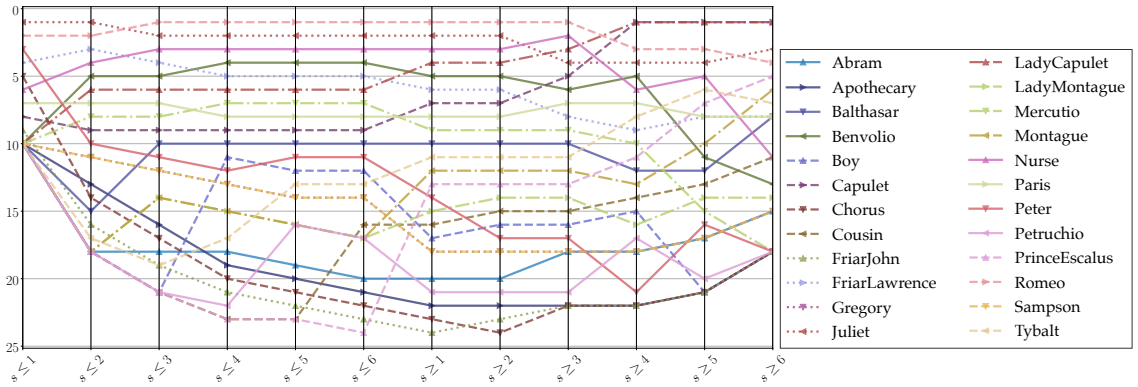


Figure 4.9: Named characters in *Romeo and Juliet*, ranked by their degree in the weighted hypergraph representation aggregated at the stage group level (hg-group-mw) when considering only hyperedges of cardinality at most s or at least s, for $s \in \{1, 2, 3, 4, 5, 6\}$. Hyperedges of cardinality at most 1 correspond to monologues. While Romeo and Juliet rank highest when including hyperedges of low cardinality, Capulet and Lady Capulet dominate when considering only less private settings.

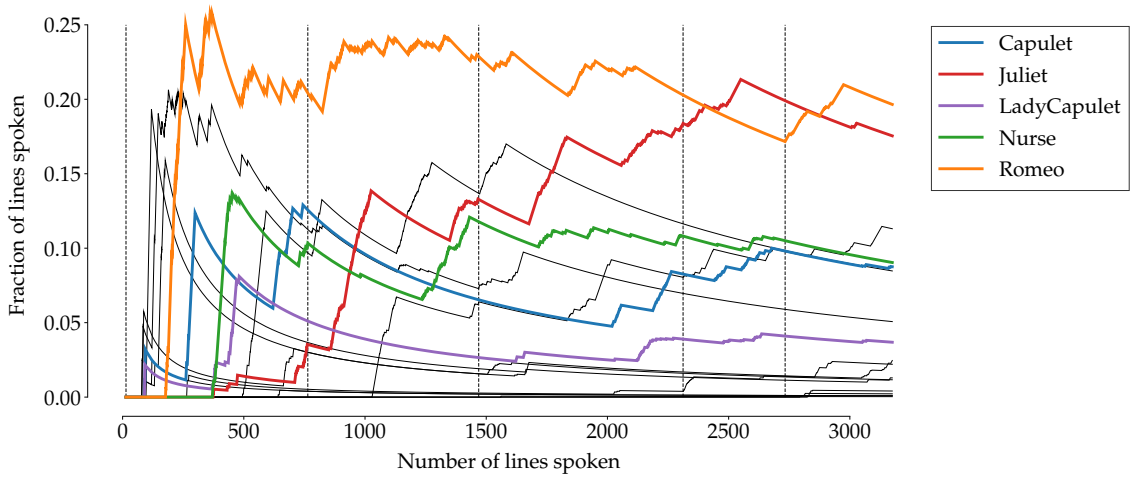


Figure 4.10: Prominence of named characters in *Romeo and Juliet* over time (excluding named servants), as measured by their fraction of spoken lines, derived from the hypergraph representation resolved at the speech act level (hg-speech-mwd). Dashed vertical lines mark the beginning of each act, and colored lines indicate protagonists of Act III, Scene V. From this perspective, Romeo is most prominent for most of the play, temporarily replaced only by Juliet for a period in Act IV and V.

Appendices

4.A Contribution Documentation

Embracing complexity, we make two contributions: (1) a *dataset*, called Hyperbard, of diverse relational data representations derived from Shakespeare's plays, ranging from simple graphs to complex hypergraphs, and (2) a *critique* of our research community. In the main text, owing to the unconventional source of the first contribution and the unconventional content of the second contribution, both contributions are interwoven and presented in the style of a Shakespeare play. In Appendices 4.B and 4.C as well as the online materials [65, 66], the dataset is introduced in full detail in conventional forms; Appendix 4.D explains the inspirations for and the style of the play. For accessibility, the story of the play and its two main themes, the dataset and the community critique, are summarized below.

4.A.1 The Story

Induction, Scene I. Confronted by Reviewer, Authors explain their first contribution. *Act I, Scene I.* Creature gets drawn into the Community by Senior Researcher and Tutor. Welcomed by Professor, they sign their PhD contract. *Act I, Scene II.* Creature quarrels with their new role. They meet Colleague, their office mate, and three Deadlines, introduced by Professor. They submit to First Deadline. *Act I, Scene III.* Creature dreams of Hyperbard, a faun caring for raw data, and Graph, one of their spirits. They discuss how to obtain insights from raw data via transformations, and that each raw data point permits several relational representations. *Act II, Scene I.* Creature converses with Colleague, Professor, and Senior Researcher over lunch. They ask Colleague about the provenance of graph data used in the Community, and they learn about graph data repositories. *Act II, Scene II.* Creature revisits their dream. They identify semantic mapping, granularity, and expressivity as the dimensions in which several graph representations of the same raw data may differ. *Act II, Scene III.* Creature secretly observes Colleague as they mechanically prepare a graph dataset and produce a datasheet in the process. *Act II, Scene IV.* Confused and depressed by the practices they witness in the Community, Creature attempts suicide. *Act II, Scene V.* Outside the Community, Creature is cared for by Graph and Hyperbard. Together, the three of them develop the graph and hypergraph representations of Shakespeare's plays included in the Hyperbard dataset. *Act III, Scene I.* Creature gets haunted by the three Deadlines, who remind them of their ignoble academic incentives. They contemplate quitting their PhD. *Act IV, Scene I.* Accompanied by Graph and Hyperbard, Creature returns to the Community. They meet Pro-

fessor, who calls Creature into their office and demands that Hyperbard leaves. *Act IV, Scene II.* From Professor, Creature learns that their paper got accepted. *Act IV, Scene III.* In the absence of Creature, Hyperbard and Graph try to convey their message that representations matter to Colleague. Professor and Creature return, and Professor orders Colleague to eliminate Hyperbard. *Act V, Scene I.* Having cremated Hyperbard, Colleague pours their ashes onto the graph dataset prepared earlier. Graph mourns the death of their sovereign and sketches its implications. *Act V, Scene II.* Creature wrestles with their experience in the Community. Instead of leaving in silence, they decide to tell their own story.

4.A.2 The Dataset

The Hyperbard dataset comprises 666 graphs and hypergraphs: 18 relational representations for each of 37 plays by William Shakespeare (Fig. 4.1). From the TEI Simple XMLs provided by Folger Digital Texts [195], for each play, we derive 6 hypergraphs, 6 clique expansions (i.e., interaction graphs), and 6 star expansions (i.e., bipartite graphs) that differ along 3 dimensions (Tab. 4.1, Fig. 4.5): *semantic mapping*, *granularity*, and *expressivity*. As we show for *Romeo and Juliet*, the representations we provide emphasize different aspects of the underlying raw data (Fig. 4.2–4.4, 4.6), and they yield widely varying results even for simple measurements of character importance (Fig. 4.7–4.10). Thus, Hyperbard *enables* and *demonstrates the need for* research on how representation choices impact the outputs and performance of graph learning, graph mining, and network analysis methods.

4.A.3 The Critique

The Community is designed as a microcosm of *our community*, including all levels of academic seniority as well as common supporting roles. The characters *inside* the Community exhibit cognitive, behavioral, and interaction patterns that frequently afflict people with corresponding roles in our community. The characters *outside* the Community appear as their antidotes, challenging the status quo and engaging in free-spirited scientific inquiry. As the play progresses, Creature gets caught up between both worlds, and we witness the force of community dynamics acting upon individuals that do not fit in. Examples of community phenomena featured in the play (there are many more): a struggling PhD student (Creature), abuse of power and difficulties of criticism in hierarchical organizations (Professor), administrative overload at the top of the pyramid (Professor and Senior Researcher), cynical resignation, disillusionment, and complicitness (Colleague), publish or perish (Deadlines), academia vs. "freedom" (Community vs. forest), mental health (Creature attempts *suicide*), uncomfortable viewpoints being shut down (Hyperbard is *cremated*).

4.B Data Documentation

All accessibility, hosting, and licensing information for Hyperbard is summarized in Table 4.2.

Table 4.2: Accessibility, hosting, and licensing information for Hyperbard.

Dataset Hosting Platform	Zenodo
Dataset Homepage	https://hyperbard.net
Dataset Tutorials	https://github.com/hyperbard/tutorials
Dataset DOI (original version)	10.5281/zenodo.6627159
Dataset DOI (latest version)	10.5281/zenodo.6627158
Dataset License	CC BY-NC 4.0
Code Hosting Platform	GitHub (maintenance), Zenodo (releases)
Code Repository	https://github.com/hyperbard/hyperbard
Code Documentation	https://hyperbard.readthedocs.io/en/latest/
Code DOI (original release)	10.5281/zenodo.6627161
Code DOI (latest release)	10.5281/zenodo.6627160
Code License	BSD 3-Clause

4.B.1 Datasheet

Our documentation follows the *Datasheets for Datasets* framework [99], omitting the questions referring specifically to data related to people.[1] For conciseness, unless otherwise indicated, the term *graph* refers to both *graphs* and *hypergraphs*.

Motivation

For what purpose was the dataset created? *Was there a specific task in mind? Was there a specific gap that needed to be filled? Please provide a description.*

Hyperbard was created to study the effects of modeling choices in the graph data curation process on the outputs produced by graph learning, graph mining, and network analysis algorithms.

There was no specific task in mind; rather, all classic graph learning, graph mining, and network analysis tasks were considered to be in scope. These tasks include, e.g., centrality ranking, outlier detection, clustering, similarity assessment, and standard statistical summarization, each for nodes, edges, and graphs, as well as variants of node classification, link prediction, or graph classification.

[1] When construed broadly (as suggested by Gebru et al. [99]), our raw data relates to people because the plays were written by William Shakespeare. The people-specific datasheet questions, however, are ill-suited for our scenario, in which the raw data consists of literary works conceived by someone who died several centuries ago.

HYPERBARD was designed to fill a specific gap: Although there were myriad freely available graph datasets, to the best of our knowledge, none of them contained

- several different relational data representations,
- of the *same* underlying raw data,
- derived in a principled and well-documented manner,
- from each of several raw data instances belonging to a natural collection,
- where the raw data is intuitive and interpretable.

WHO CREATED THE DATASET (E.G., WHICH TEAM, RESEARCH GROUP) AND ON BEHALF OF WHICH ENTITY (E.G., COMPANY, INSTITUTION, ORGANIZATION)?

Corinna Coupette and Bastian Rieck created the dataset as part of their research.

WHO FUNDED THE CREATION OF THE DATASET? *If there is an associated grant, please provide the name of the grantor and the grant name and number.*

The creation of the dataset was indirectly funded by the institutions employing the dataset authors, i.e., the Max Planck Institute for Informatics (Corinna Coupette) and the Institute of AI for Health, Helmholtz Munich. There are no associated grants.

ANY OTHER COMMENTS?

None.

COMPOSITION

WHAT DO THE INSTANCES THAT COMPRISE THE DATASET REPRESENT (E.G., DOCUMENTS, PHOTOS, PEOPLE, COUNTRIES)? *Are there multiple types of instances (e.g., movies, users, and ratings; people and interactions between them; nodes and edges)? Please provide a description.*

Each instance represents a play attributed to William Shakespeare as a graph, and there are multiple different graph representations per play. In some graphs (i.e., hypergraphs and graphs derived from clique expansions of hypergraphs), nodes represent characters, and (hyper)edges represent that characters were on stage at the same time in some part of the play. In other graphs (i.e., graphs derived from star expansions of hypergraphs), nodes represent characters or parts of a play, and an edge indicates that a character was on stage in that part of the play. The representations provided differ not only in their semantic mapping (what are the nodes and edges) but also in their granularity (what parts of the play are modeled as edges resp. nodes) and in their expressivity (what additional information is associated with nodes and edges); see Table 4.1 in the HYPERBARD paper.

How many instances are there in total (of each type, if appropriate)?

There are 37 plays in the raw data; 17 comedies, 10 historical plays, and 10 tragedies. Each play is represented as a graph in (at least) 18 different ways, for a total of 666 graph representations.

Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set? *If the dataset is a sample, then what is the larger set? Is the sample representative of the larger set (e.g., geographic coverage)? If so, please describe how this representativeness was validated/verified. If it is not representative of the larger set, please describe why not (e.g., to cover a more diverse range of instances, because instances were withheld or unavailable).*

The dataset contains graph representations of all plays attributed to William Shakespeare by the Folger Shakespeare Library (see https://folgerpedia.folger.edu/William_Shakespeare%27s_plays), with the exception of lost plays and the comedy *The Two Noble Kingsmen*—a collaboration between Shakespeare and John Fletcher that is not currently provided in the TEI simple format by Folger Digital Texts.

What data does each instance consist of? *"Raw" data (e.g., unprocessed text or images) or features? In either case, please provide a description.*

Each instance, i.e., each of Shakespeare's plays, is represented by a set of files: one raw data file containing the text of the play as an XML encoded using the TEI Simple format, taken from Folger Digital Texts without modification, three CSV files containing preprocessed data, and 19 CSV files containing node lists and edge lists to construct different graph representations.

Consequently, dataset is distributed using the following folder structure:

- `rawdata`: contains 37 raw data XML files encoded in TEI simple.
- `data`: contains 3·37 preprocessed data files derived from files in `rawdata`.
- `graphdata`: contains 19·37 node and edge lists to construct graph representations from the files in `data`.
- `metadata`: contains `playtypes.csv`, mapping play identifiers to play types (comedy, history, or tragedy).

Python code to reproduce all graph representations and load them as *networkx* or *hypernetx* graphs is maintained in a GitHub repository, https://github.com/hyperbard/hyperbard, and code releases are archived via Zenodo (10.5281/zenodo.6627160).

Is there a label or target associated with each instance? *If so, please provide a description.*

There are labels corresponding to the type of play (one of {comedy, history, tragedy}), which could be used to partition the data for exploration, or as targets in classification tasks.

Is any information missing from individual instances? *If so, please provide a description, explaining why this information is missing (e.g., because it was unavailable). This does not include intentionally removed information, but might include, e.g., redacted text.*

There is no missing information.

Are relationships between individual instances made explicit (e.g., users' movie ratings, social network links)? *If so, please describe how these relationships are made explicit.*

When considering plays as instances, no relationships between individual instances are made explicit. When considering characters or parts of plays as instances, however, relationships between characters, or between characters and parts of plays are made explicit in the graph representations, exploiting the TEI Simple encoding of that data and the annotations provided in the XML attributes.

Are there recommended data splits (e.g., training, development/validation, testing)? *If so, please provide a description of these splits, explaining the rationale behind them.*

There are no recommended data splits for the current release.

Are there any errors, sources of noise, or redundancies in the dataset? *If so, please provide a description.*

The raw data contain some errors and redundancies in the XML encoding. Errors include redundant XML tags (e.g., doubly-wrapped tags), but also character entries or exits not explicitly annotated. Redundancies result from the choice, made by the creators of Folger Digital Texts, to encode some information conveyed in the raw text also as attributes or separate XML tags (e.g., a character who speaks is encoded both as an attribute of the tag wrapping the speech and as an XML tag wrapping the name of the speaker).

There are two notable sources of noise affecting the preprocessed data and the graph data, both of which relate to our handling of stage directions—i.e., our processing of the XML attributes of `<stage>` tags in the raw data.

First, to determine which characters are on stage when a word is spoken, we primarily rely on the contents of `who` attributes in the `<stage>` tags of the raw data marked with `type="entry"` resp. `type="exit"`. The `who` attributes, however, are sometimes *semantically* incomplete, i.e., they may reflect Shakespeare's orig-

inal stage directions accurately, but the original stage directions do not mention implied character movements (such as the exit of a side character or the exit of characters that died or fell unconscious at the end of a scene). To limit the impact of this noise source on our graph representations, we "flush" characters when a new scene starts (to handle missing exits) and ensure that the speaker is always on stage (to handle missing entries, some of which are also introduced by our character flushing policy).

Second, in our directed graph representations, where edges encode speaking and being spoken to, we equate being on stage while a word is spoken with hearing the word. Thus, we do not account for the impact of some stage directions concerning delivery, e.g., stage directions indicating that speech is inaudible for some or all other characters on stage, on the information flow our directed graph representations purport to capture. In the TEI simple encoding of our raw data, such stage directions are annotated with `type="delivery"`, but there is no indication of who can hear the words so delivered in the XML annotations. There are 2 200 XML tags annotated with `type="delivery"` (i.e., 60 delivery modifications per play on average). As modifications to delivery are sometimes crucial to drive the plot (e.g., by setting up misunderstandings), the impact of this noise source should not be underestimated, but it affects only our directed graph representations, which might be cautiously interpreted as "upper bounds" on the information flow between the characters on stage.

These sources of noise detailed above could likely be eliminated, to a large extent, by a more sophisticated parsing of the stage directions. This parsing could leverage, e.g., natural language processing methods to supplement the XML annotations. We plan to implement this improvement for a future dataset release.

Is the dataset self-contained, or does it link to or otherwise rely on external resources (e.g., websites, tweets, other datasets)? *If it links to or relies on external resources, a) are there guarantees that they will exist, and remain constant, over time; b) are there official archival versions of the complete dataset (i.e., including the external resources as they existed at the time the dataset was created); c) are there any restrictions (e.g., licenses, fees) associated with any of the external resources that might apply to a dataset consumer? Please provide descriptions of all external resources and any restrictions associated with them, as well as links or other access points, as appropriate.*

The dataset is self-contained. The raw data stem from Folger Digital Texts, maintained by the Folger Shakespeare Library and released under the CC BY-NC 3.0 Unported license, and they are redistributed without modifications as part of the Hyperbard dataset. All other data are derived from the raw data, and the CC

BY-NC 3.0 Unported license does not impose any additional restrictions. As part of our dataset maintenance (see below), we will regularly check Folger Digital Texts for modifications, and we will recompute and redistribute an updated Hyperbard dataset under a versioned DOI whenever we detect changes.

Does the dataset contain data that might be considered confidential (e.g., data that is protected by legal privilege or by doctor-patient confidentiality, data that includes the content of individuals' non-public communications)? *If so, please provide a description.*

The dataset does not contain data that might be considered confidential.

Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety? *If so, please describe why.*

The raw data, i.e., Shakespeare's plays, contain scenes that might be considered offensive, insulting, threatening, or otherwise anxiety-inducing from a contemporary perspective. For example, there is considerable controversy in the humanities around whether *The Taming of the Shrew* is misogynistic, and the main female protagonist's final speech on female submissiveness (Act V, Scene 2, ll. 136–179) might cause discomfort to modern readers. Moreover, the corpus uses words that might be considered derogatory or offensive from a contemporary perspective. The preprocessed data, however, disassembles the original text, such that (offensive) play content is no longer immediately apparent when the data is viewed directly.

Any other comments?

The entire dataset takes up roughly 365 MB when uncompressed, and 30 MB when compressed.

Collection Process

How was the data associated with each instance acquired? *Was the data directly observable (e.g., raw text, movie ratings), reported by subjects (e.g., survey responses), or indirectly inferred/derived from other data (e.g., part-of-speech tags, model-based guesses for age or language)?*

The raw data associated with each instance was acquired from Folger Digital Texts as XML files encoded in TEI Simple format. This format contains both raw text and structural, linguistic, and semantic annotations embedded in XML tags or XML attributes. Hence, it was partially directly observable (e.g., the raw text and its structure) and partially derived from other data (e.g., the XML tags and

their attributes). The preprocessed data and the graph data were derived from the raw data.

If the data was reported by subjects or indirectly inferred/derived from other data, was the data validated/verified? *If so, please describe how.*

To the extent that the raw data were indirectly inferred or derived from other data, validation was performed by the specialists from Folger Digital Texts. The preprocessed data and the graph data were validated by unit tests and manual inspection aided by visualizations (which also led us to discover the noise sources detailed above).

What mechanisms or procedures were used to collect the data (e.g., hardware apparatuses or sensors, manual human curation, software programs, software APIs)? *How were these mechanisms or procedures validated?*

The raw data was bulk downloaded in TEI Simple format as a ZIP archive from the Folger Digital Texts downloads section, and Folger Digital Texts compiled the raw data through computer-assisted manual curation. The bulk download was checked manually to ensure that the extracted archive contained one XML file per play, as expected. The code creating the preprocessed data from the raw data and the graph representations from the preprocessed data is almost completely unit tested.

If the dataset is a sample from a larger set, what was the sampling strategy (e.g., deterministic, probabilistic with specific sampling probabilities)?

The data is not a sample from a larger set.

Who was involved in the data collection process (e.g., students, crowdworkers, contractors) and how were they compensated (e.g., how much were crowdworkers paid)?

Only Corinna Coupette and Bastian Rieck, the dataset authors, were involved in the data collection process.

Over what time frame was the data collected? *Does this time frame match the creation time frame of the data associated with the instances (e.g., recent crawl of old news articles)? If not, please describe the time frame in which the data associated with the instances was created.*

The raw data was collected through one download call to https://shakespeare.folger.edu/downloads/teisimple/shakespeares-works_TEIsimple_FolgerShakespeare.zip in June 2022, and the preprocessed data and the graph data were derived from the raw data by running a code pipeline, also in June 2022. This time frame does not match the creation time frame of the raw data, which, though

internal to the Folger Shakespeare Library, spans at least several months in 2020. It also does not match the creation time frame of Shakespeare's plays, which spans several decades in the 16th and 17th centuries.

Were any ethical review processes conducted (e.g., by an institutional review board)? *If so, please provide a description of these review processes, including the outcomes, as well as a link or other access point to any supporting documentation.*

No ethical review processes were conducted.

Any other comments?

None.

Preprocessing/Cleaning/Labeling

Was any preprocessing/cleaning/labeling of the data done (e.g., discretization or bucketing, tokenization, part-of-speech tagging, SIFT feature extraction, removal of instances, processing of missing values)? *If so, please provide a description. If not, you may skip the remaining questions in this section.*

Our data preprocessing consists of two steps.

1. Transform raw XML data into preprocessed CSV data (`rawdata`→`data`).
   Script: `run_preprocessing.py`
   (a) Extract the cast list from the TEI Simple XML and store it as a CSV. (This is technically unnecessary to generate our graph representations, but it gives a convenient overview of the characters occurring in the play.)
   Function: `get_cast_df`
   Artifact: `data/{play}.cast.csv`
   (b) Parse the TEI Simple XML into a table containing one row per descendant of the TEI Simple `<body>` tag, and the tag names and XML attributes of all XML tags of interest (eliminating redundant XML elements), plus the text content of all XML tags that are leaves, as columns. Annotate the result with information on the act and scene in which the tag occurs, the characters on stage when the tag occurs, and the speaker(s), if any.
   Function `get_raw_xml_df`
   Artifact: `data/{play}.raw.csv`
   (c) Transform the artifact from the previous step into a table with one row per setting on stage, where a setting is a stretch of the play without changes to the speaker or to the group of characters on stage, and information on the setting as well as the number of lines and tokens spoken

in that setting as columns.
Artifact: `data/{play}.agg.csv`

2. Transform preprocessed CSV data into node and edge CSV files for graph construction (`data`→`graphdata`).
The artifacts resulting from this step are generally labeled `{play}_{semantic mapping}_{granularity}_{expressivity}.{list type}.csv`, omitting the expressivity (and granularity) components in node lists if all different graph representations with a given semantic mapping (and granularity) use the same set of nodes.

   (a) Create node lists and edge lists for different graph representations in CSV format from `data/{play}.agg.csv` artifacts.
   Script: `create_graph_representations.py`
   Artifacts:
   - `graphdata/{play}_ce-group-mw.edges.csv`
   - `graphdata/{play}_ce-group-w.edges.csv`
   - `graphdata/{play}_ce-scene-mw.edges.csv`
   - `graphdata/{play}_ce-scene-w.edges.csv`
   - `graphdata/{play}_ce.nodes.csv`
   - `graphdata/{play}_se-group-w.edges.csv`
   - `graphdata/{play}_se-group.nodes.csv`
   - `graphdata/{play}_se-scene-w.edges.csv`
   - `graphdata/{play}_se-scene.nodes.csv`
   - `graphdata/{play}_se-speech-mwd.edges.csv`
   - `graphdata/{play}_se-speech-wd.edges.csv`
   - `graphdata/{play}_se-speech.nodes.csv`

   (b) Create node lists and edge lists for different hypergraph representations in CSV format from `data/{play}.agg.csv` artifacts.
   Script: `create_hypergraph_representations.py`
   Artifacts:
   - `graphdata/{play}_hg-group-mw.edges.csv`
   - `graphdata/{play}_hg-group-mw.node-weights.csv`
   - `graphdata/{play}_hg-scene-mw.edges.csv`
   - `graphdata/{play}_hg-scene-mw.node-weights.csv`
   - `graphdata/{play}_hg-speech-mwd.edges.csv`
   - `graphdata/{play}_hg-speech-wd.edges.csv`
   - `graphdata/{play}_hg.nodes.csv`

Was the "raw" data saved in addition to the preprocessed/cleaned/labeled data (e.g., to support unanticipated future uses)? *If so, please provide a link or other access point to the "raw" data.*

The raw data was saved, and it is distributed along with the preprocessed data in the dataset available from Zenodo under a versioned DOI: 10.5281/zenodo.6627158.

Is the software that was used to preprocess/clean/label the data available? *If so, please provide a link or other access point.*

The software used to transform the raw data into the preprocessed data, and the preprocessed data into the graph data representations, is available on GitHub in the following repository:

https://github.com/hyperbard/hyperbard.

All code releases are also available on Zenodo under a versioned DOI: 10.5281/zenodo.6627160.

Any other comments?

All data preprocessing can be completed in a couple of minutes even on older commodity hardware. We used a 2016 MacBook Pro with a 2.9 GHz Quad-Core Intel Core i7 processor and 16 GB RAM.

Uses

Has the dataset been used for any tasks already? *If so, please provide a description.*

In the paper introducing Hyperbard, the dataset has been used to demonstrate the differences between rankings of characters by degree that result from different modeling choices made when transforming raw data into graphs.

Is there a repository that links to any or all papers or systems that use the dataset? *If so, please provide a link or other access point.*

Papers or systems known to use dataset will be collected on https://hyperbard.net and on GitHub.

What (other) tasks could the dataset be used for?

Hyperbard was designed for inquiries into the stability of algorithmic results under different reasonable representations of the underlying raw data, i.e., to enable *representation robustness checks* for graph learning, graph mining, and network analysis methods. In this role, it could generally be used for all graph learning, graph mining, and network analysis tasks identified as *in scope* in the motivation section.

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses? *For example, is there anything that a dataset consumer might need to know to avoid uses that could result in unfair treatment of individuals or groups (e.g., stereotyping, quality of service issues) or other risks or harms (e.g., legal risks, financial harms)? If so, please provide a description. Is there anything a dataset consumer could do to mitigate these risks or harms?*

The quality and expressivity of the dataset is limited by the quality and expressivity of Folger Digital Texts encoded using the TEI Simple format, which could restrict usage in the digital humanities, e.g., when they are interested in the minute details of character interactions described in stage directions.

Hyperbard contains relational data representations of Shakespeare's plays, which were written more than four centuries ago. Hence, there are no risks or harms associated with the dataset beyond the risks or harms also associated with the ongoing study of Shakespeare's works in the humanities, and the risks or harms associated with the decontextualization or overinterpretation of any dataset.

At https://hyperbard.net and on GitHub, we keep a continuously-updated list of all known dataset limitations for dataset consumers to review when deciding whether Hyperbard is appropriate for their use case.

Are there tasks for which the dataset should not be used? *If so, please provide a description.*

Outside *representation robustness checks*, Hyperbard should not be used in tasks that have no reasonable semantic interpretation in the domain of the raw data.

Any other comments?

None.

Distribution

Will the dataset be distributed to third parties outside of the entity (e.g., company, institution, organization) on behalf of which the dataset was created? *If so, please provide a description.*

The dataset was not created on behalf of any entity, and it will be distributed freely.

How will the dataset will be distributed (e.g., tarball on website, API, GitHub)? *Does the dataset have a digital object identifier (DOI)?*

The dataset will be distributed as a ZIP archive via Zenodo, based on code hosted on GitHub. Each dataset version and each code release will have a versioned DOI, generated automatically by Zenodo. See also Table 4.2.

When will the dataset be distributed?

The dataset was distributed when the paper introducing it was originally submitted.

Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)? *If so, please describe this license and/or ToU, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms or ToU, as well as any fees associated with these restrictions.*

The dataset will be distributed under a CC BY-NC 4.0 license, according to which others are free to

- *share*, i.e., copy and redistribute, and
- *adapt*, i.e., remix, transform, and build on the material,

provided they

- *give attribution*, i.e., give appropriate credit, provide a link to the license, and indicate if changes were made,
- do *not* use the material for *commercial purposes*, and
- *add no restrictions* limiting others in doing anything the license permits.

The code constructing the dataset will be distributed under a permissive BSD 3-Clause license.

Have any third parties imposed IP-based or other restrictions on the data associated with the instances? *If so, please describe these restrictions, and provide a link or other access point to, or otherwise reproduce, any relevant licensing terms, as well as any fees associated with these restrictions.*

The Folger Shakespeare Library has released the source of our raw data, Folger Digital Texts, under the CC BY-NC 3.0 Unported license, which has essentially the same usage conditions as our CC BY-NC 4.0 license.

Do any export controls or other regulatory restrictions apply to the dataset or to individual instances? *If so, please describe these restrictions, and provide a link or other access point to, or otherwise reproduce, any supporting documentation.*

No export controls or other regulatory restrictions apply.

Any other comments?

None.

Maintenance

Who will be supporting/hosting/maintaining the dataset?

Corinna Coupette and Bastian Rieck will be supporting, hosting, and maintaining the dataset.

How can the owner/curator/manager of the dataset be contacted (e.g., email address)?

In the interest of transparency, the preferred method to contact the dataset maintainers is by opening GitHub issues at https://github.com/hyperbard/hyperbard. Alternatively, the dataset maintainers can be reached by email to info@hyperbard.net.

Is there an erratum? *If so, please provide a link or other access point.*

Errata will be documented at https://hyperbard.net and on GitHub.

Will the dataset be updated (e.g., to correct labeling errors, add new instances, delete instances)? *If so, please describe how often, by whom, and how updates will be communicated to dataset consumers (e.g., mailing list, GitHub)?*

The dataset will be updated as needed, and updates will be labeled using *semantic versioning*.

- A *patch version* (e.g., 0.0.1 → 0.0.2) is a recomputation of the latest dataset version following a non-breaking change in the underlying raw data.
- A *minor version* (e.g., 0.0.1 → 0.2.0) is an update of the latest dataset version that increases the expressivity of existing representations while maintaining all of their previously present features.
- Any other update is a *major version* (e.g., 0.0.1 → 1.0.0). This includes, e.g., responses to breaking changes in the underlying source data, additions of new representations, and changes to existing representations that might break dataset consumer code.

Patch versions will be created automatically using GitHub actions. Minor versions and major versions will be created by the dataset maintainers, potentially accepting pull requests or implementing feature requests filed at:

https://github.com/hyperbard/hyperbard.

New releases will be communicated at https://hyperbard.net and on GitHub, and they will be available for download under a versioned DOI on Zenodo, with 10.5281/zenodo.6627158 always resolving to the latest release.

If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were the individuals in question told that their data would be retained for a fixed period of time and then deleted)? *If so, please describe these limits and explain how they will be enforced.*

There are no data retention limits.

Will older versions of the dataset continue to be supported/hosted/maintained? *If so, please describe how. If not, please describe how its obsolescence will be communicated to dataset consumers.*

Older versions of the dataset will remain hosted on Zenodo, with the relevant version of the code needed to reproduce them available in an associated GitHub release, also archived on Zenodo.

There will be basic support for older versions of the dataset, and as Hyperbard is derived from century-old literary works, dataset maintenance amounts to dataset updates (see the paragraph on dataset updates).

If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so? *If so, please provide a description. Will these contributions be validated/verified? If so, please describe how. If not, why not? Is there a process for communicating/distributing these contributions to dataset consumers? If so, please provide a description.*

Others can extend, augment, build on, and contribute to the dataset through the engagement mechanisms provided by GitHub.
See also:
https://github.com/hyperbard/hyperbard/blob/main/CONTRIBUTING.md.

Extensions, augmentations, and contributions provided via pull requests will be validated and verified by the dataset maintainers in a regular code and data review process, while changes made in independent forks will not be checked.

Contributions integrated with the Hyperbard code repository will be visible on GitHub, and they trigger new dataset releases, in which contributors will be specifically acknowledged.

Any other comments?

None.

4.B.2 Hosting, License, and Maintenance Plan

For hosting and licensing information, see Table 4.2 and Section 4.B.1. For the maintenance plan, see Section 4.B.1.

4.B.3 Author Responsibility Statement

The dataset authors, Corinna Coupette and Bastian Rieck, bear all responsibility in case of violation of rights, etc., and they confirm that the data is released under the CC BY-NC 4.0 license, and that the code is released under the BSD 3-Clause license.

4.C Usage Documentation

The Hyperbard dataset is distributed in four folders: `rawdata`, `data`, `graphdata`, and `metadata`. See Section 4.B.1 for more details on the composition of the dataset. The dataset can be reproduced by cloning the GitHub repository and running `make` (this will also generate most figures included in the Hyperbard paper).

In addition to the written documentation, we provide Jupyter notebook tutorials for interactive data exploration. The tutorials are hosted on GitHub at https://github.com/hyperbard/tutorials, and they can be run both locally and in a Binder, i.e., a fully configured remote environment accessible through the browser without any local setup. Launching the Binder usually takes around thirty seconds.

In the following, we explain the structure of the files in Hyperbard's folders and detail how these files can be read. All file examples are taken from *Romeo and Juliet*, and for CSV files, all columns are described in alphabetical order.

4.C.1 rawdata

This folder contains XML files encoded in TEI Simple as provided by Folger Digital Texts. These files can be read with any XML parser, such as the parser from the `beautifulsoup4` library in Python. All file names follow the pattern

`{play}_TEIsimple_FolgerShakespeare.xml`.

The XML encoding is designed to meet the needs of the (digital) humanities, and hence, it is very detailed and fine-grained. For example, every word, whitespace character, and punctuation mark is contained in its own tag.

The encoding practices followed by Folger Digital Texts are described in the `<encodingDesc>` tag of each text. To summarize:

- The major goal of the TEI Simple encoding is to achieve interoperability with a large corpus of early modern texts derived from the Early English Books Text Creation Partnership transcriptions (i.e., it is different from *our* goal).
- The encoding is completely faithful to the readings, orthography, and punctuation of the source texts (i.e., the Shakespeare texts edited by Barbara Mowat and Paul Werstine at Folger Shakespeare Library).
- All `xml:ids` are corpuswide identifiers (i.e., they are unique across all our plays, too).
- Words, spaces, and punctuation characters are numbered sequentially within each play, incremented by 10 (XML attribute: `n`).
- Most other elements begin with an element-specific prefix, followed by a reference to the Folger Through Line Number, a sequential numbering of the numbered lines in the text. (Details omitted.)

– Spoken words are linguistically annotated with a lemma and POS tag.

Running the script `compute_rawdata_xml_statistics.py` in the HYPERBARD GitHub repository, which computes basic XML tag, path, and attribute statistics for the entire corpus and writes the results to the `metadata` folder as CSV files, provides some intuition regarding the structure of the raw data. This script also pulls the descriptions of all tags from the current TEI specification. For more information on the TEI Simple format, which has been integrated with the main TEI specification, see https://github.com/TEIC/TEI-Simple.

Example:

```
...
<sp xml:id="sp-0015" who="#SERVANTS.CAPULET.Sampson_Rom">
<speaker xml:id="spk-0015">
<w xml:id="fs-rom-0002610">SAMPSON</w>
</speaker>
<p xml:id="p-0015">
<lb xml:id="ftln-0015" n="1.1.1"/>
<w xml:id="fs-rom-0002620" n="1.1.1" lemma="Gregory" ana="#n1-nn">Gregory</w>
<pc xml:id="fs-rom-0002630" n="1.1.1">,</pc>
<c> </c>
<w xml:id="fs-rom-0002650" n="1.1.1" lemma="on" ana="#acp-p">on</w>
<c> </c>
<w xml:id="fs-rom-0002670" n="1.1.1" lemma="my" ana="#po">my</w>
<c> </c>
<w xml:id="fs-rom-0002690" n="1.1.1" lemma="word" ana="#n1">word</w>
<c> </c>
<w xml:id="fs-rom-0002710" n="1.1.1" lemma="we|will" ana="#pns|vmb">we'll</w>
<c> </c>
<w xml:id="fs-rom-0002730" n="1.1.1" lemma="not" ana="#xx">not</w>
<c> </c>
<w xml:id="fs-rom-0002750" n="1.1.1" lemma="carry" ana="#vvi">carry</w>
<c> </c>
<w xml:id="fs-rom-0002770" n="1.1.1" lemma="coal" ana="#n2">coals</w>
<pc xml:id="fs-rom-0002780" n="1.1.1">.</pc>
</p>
</sp>
...
```

4.C.2 DATA

This folder contains CSV files, which can be read with any CSV parser, such as the parser from the `pandas` library in Python.

There are three types of files:
`{play}.cast.csv` files, `{play}.raw.csv` files, and `{play}.agg.csv` files.

{play}.cast.csv

A {play}.cast.csv file contains the XML identifiers and attributes of all <castItem> tags found in a {play}_TEIsimple_FolgerShakespeare.xml file. It gives an overview of the characters occurring in a play, and it can be used to count the number of characters (including characters that do not speak) or to build a hierarchy of characters and character groups.

Rows correspond to characters or character groups.

Columns in alphabetical order:

- corresp: group (i.e., another cast item) to which a given cast item belongs, if any (XML attribute abbreviating "corresponds").
  Type: String or NaN (if the cast item does not belong to any other cast item).
- xml:id: unique identifier of the cast member.
  Type: String.

Note that the data in each of these columns does *not* start with a # sign. This contrasts with *references* to the xml:ids in the attributes of other XML tags in the raw data XML files, which *do* start with a # sign (to indicate the referencing).

Example:

```
xml:id,corresp
ATTENDANTS.PRINCE_Rom,ATTENDANTS_Rom
ATTENDANTS_Rom,
Apothecary_Rom,
Benvolio_Rom,
Boy_Rom,
...
```

{play}.raw.csv

A {play}.raw.csv file contains the descendants of the <body> tag found in a {play}_TEIsimple_FolgerShakespeare.xml file, with redundancies resulting from the encoding format eliminated, and additional information to build graph representations annotated. It provides a *disaggregated* tabular overview of the information underlying our graph representations, and it serves as the basis of its corresponding {play}.agg.csv file.

Rows correspond to instances of XML tags.

Columns in alphabetical order:

- act: Derived attribute. The number of the act in which the tag occurs. An integer in [5] for all tags in the main part of the play. 0 for tags occurring before the first act (e.g., in a prologue or an induction), 6 for tags occurring

after the fifth act (e.g., in an epilogue).
Type: Non-negative integer.

- `ana`: Original attribute. If the tag wraps a spoken word, the POS tag of that word (XML attribute abbreviating "analysis").
  Type: String or NaN (if the tag does not wrap a spoken word).
- `lemma`: Original attribute. If the tag wraps a spoken word, the lemma of that word.
  Type: String or NaN (if the tag does not wrap a spoken word).
- `n`: Original attribute. A label for the element, not necessarily unique.
  Type: String, positive integer (for `<div>` tags representing acts or scenes), or NaN (e.g., for `<c>` tags wrapping whitespace characters).
- `onstage`: Derived attribute. Whitespace-separated list of characters on stage when the tag occurs.
  Type: String or NaN.
- `part`: Original attribute. Rare and not of interest for graph building.
  Type: String or NaN.
- `prev`: Original attribute. Rare and not of interest for graph building.
  Type: String or NaN.
- `rendition`: Original attribute. Rare and not of interest for graph building.
  Type: String or NaN.
- `scene`: Derived attribute. The number of the scene in which the tag occurs. `0` if the tag does not occur in a scene.
  Type: Non-negative integer.
- `speaker`: Derived attribute. Whitespace-separated list of characters who are speaking when a tag occurs. Note that several characters can speak at the same time, although the overwhelming majority of speech in the corpus is uttered by only one speaker.
  Type: String or NaN.
- `stagegroup_raw`: Derived attribute. Number stating how many changes in the set of characters on stage we have already witnessed when a tag occurs (i.e., the same set of characters can occur in different stage groups). Relevant for sorting and aggregation.
  Type: Non-negative integer.
- `tag`: Original entity. The name of the XML tag to which the row corresponds.
  Type: String.
- `text`: Original text content.
  Type: String or NaN (if a tag is not a leaf in the XML tree).

- `type`: Original attribute. Used to give details on `<div>` and `<stage>` tags, e.g., distinguish between acts and scenes, and mark stage directions as, e.g., character entry or exit.
  Type: String or NaN.
- `who`: Original attribute giving information on characters who act, transformed into a set. Will become whitespace-separated list in future releases.
  Type: Set of strings or NaN.
- `xml:id`: Original XML identifier. Note that instances of some XML tags, including `<div>` and `<c>` tags, do not have XML identifiers.
  Type: String or NaN.

Example:

```
tag,type,n,text,xml:id,who,lemma,ana,part,rendition,prev,act,
   scene,onstage,stagegroup_raw,speaker
...
sp,,,,sp-0015,{'#SERVANTS.CAPULET.Sampson_Rom'},,,,,,1,1,#
   SERVANTS.CAPULET.Gregory_Rom #SERVANTS.CAPULET.Sampson_Rom
   ,3,#SERVANTS.CAPULET.Sampson_Rom
p,,,,p-0015,,,,,,,1,1,#SERVANTS.CAPULET.Gregory_Rom #SERVANTS.
   CAPULET.Sampson_Rom,3,
lb,,1.1.1,,ftln-0015,,,,,,,1,1,#SERVANTS.CAPULET.Gregory_Rom #
   SERVANTS.CAPULET.Sampson_Rom,3,#SERVANTS.CAPULET.
   Sampson_Rom
w,,1.1.1,Gregory,fs-rom-0002620,,Gregory,#n1-nn,,,,1,1,#
   SERVANTS.CAPULET.Gregory_Rom #SERVANTS.CAPULET.Sampson_Rom
   ,3,#SERVANTS.CAPULET.Sampson_Rom
pc,,1.1.1,",",fs-rom-0002630,,,,,,,1,1,#SERVANTS.CAPULET.
   Gregory_Rom #SERVANTS.CAPULET.Sampson_Rom,3,#SERVANTS.
   CAPULET.Sampson_Rom
c,,, ,,,,,,,,1,1,#SERVANTS.CAPULET.Gregory_Rom #SERVANTS.
   CAPULET.Sampson_Rom,3,
w,,1.1.1,on,fs-rom-0002650,,on,#acp-p,,,,1,1,#SERVANTS.CAPULET
   .Gregory_Rom #SERVANTS.CAPULET.Sampson_Rom,3,#SERVANTS.
   CAPULET.Sampson_Rom
c,,, ,,,,,,,,1,1,#SERVANTS.CAPULET.Gregory_Rom #SERVANTS.
   CAPULET.Sampson_Rom,3,
...
```

{PLAY}.AGG.CSV

A {play}.agg.csv file contains a condensed and filtered view of its corresponding {play}.raw.csv file, focusing only on spoken words. It provides an *aggregated* tabular overview of the information underlying our graph representations, and it serves as the basis of all files in the graphdata folder. In contrast to the {play}.raw.csv file, which contains some original attributes, {play}.agg.csv contains only derived attributes.

Rows correspond to *settings* (or *speech acts*), i.e., maximal sequences of words in which neither the speaker(s) nor the group of characters on stage change.

Columns in alphabetical order:

- act: The same as act in {play}.raw.csv.
- n_lines: The number of lines spoken in a setting.
  Type: Positive integer.
- n_tokens: The number of tokens spoken in a setting.
  Type: Positive integer.
- onstage: The same as onstage in {play}.raw.csv.
- scene: The same as scene in {play}.raw.csv.
- setting: Number stating how many changes in the tuple (set of characters on stage, speaker) we have seen when the words summarized in this row occur, plus 1 (for consistency with the numbering in stagegroup).
  Type: Positive integer.
- speaker: The same as speaker in {play}.raw.csv.
- stagegroup: The contents of the stagegroup_raw column, renumbered to be consecutive in {play}.agg.csv, starting with 1.
  Type: Positive integer.
- stagegroup_raw: The same as stagegroup_raw in {play}.raw.csv.

Example:

```
act,scene,stagegroup,stagegroup_raw,setting,onstage,speaker,
   n_lines,n_tokens
0,0,1,1,1,#Chorus_Rom,#Chorus_Rom,14,106
1,1,2,3,2,#SERVANTS.CAPULET.Gregory_Rom #SERVANTS.CAPULET.
   Sampson_Rom,#SERVANTS.CAPULET.Sampson_Rom,1,8
1,1,2,3,3,#SERVANTS.CAPULET.Gregory_Rom #SERVANTS.CAPULET.
   Sampson_Rom,#SERVANTS.CAPULET.Gregory_Rom,1,7
1,1,2,3,4,#SERVANTS.CAPULET.Gregory_Rom #SERVANTS.CAPULET.
   Sampson_Rom,#SERVANTS.CAPULET.Sampson_Rom,1,9
```

```
1,1,2,3,5,#SERVANTS.CAPULET.Gregory_Rom #SERVANTS.CAPULET.
    Sampson_Rom,#SERVANTS.CAPULET.Gregory_Rom,2,10
```

4.C.3 graphdata

This folder contains CSV files, which can be read with any CSV parser, such as the parser from the `pandas` library in Python.

For each play, the folder holds all files needed to generate the representations listed in Table 4.1, i.e.:

- Files to construct *clique expansions* (`ce`, i.e., character co-occurrence networks):
  - `{play}_ce-group-mw.edges.csv`:
    Weighted multi-edges for clique expansions aggregated at the stage group level.
    Use to generate ce-group-{mb,mw} representations.
  - `{play}_ce-group-w.edges.csv`:
    Count-weighted edges for clique expansions aggregated at the stage group level.
    Use to generate ce-group-b representations (or ce-group-w representations for easier plotting of ce-group-mb representations if the edge order does not matter).
  - `{play}_ce-scene-mw.edges.csv`:
    Weighted multi-edges for clique expansions aggregated at the scene level.
    Use to generate ce-scene-{mb,mw} representations.
  - `{play}_ce-scene-w.edges.csv`:
    Count-weighted edges for clique expansions aggregated at the scene level.
    Use to generate ce-scene-b representations (or ce-scene-w representations for easier plotting of ce-scene-mb representations if the edge order does not matter).
  - `{play}_ce.nodes.csv`:
    Nodes for all clique expansions.
    Use to generate all ce-$*$ representations.
- Files to construct *star expansions* (`se`, i.e., bipartite graphs with characters and text units as node sets):
  - `{play}_se-group-w.edges.csv`:
    Edges for star expansions aggregated at the stage group level.
    Use to generate se-group-{b,w} representations.

  - `{play}_se-group.nodes.csv`:
    Nodes for star expansions aggregated at the stage group level.
    Use to generate se-group-{b,w} representations.
  - `{play}_se-scene-w.edges.csv`:
    Edges for star expansions aggregated at the scene level.
    Use to generate se-scene-{b,w} representations.
  - `{play}_se-scene.nodes.csv`:
    Nodes for star expansions aggregated at the scene level. The character nodes are the same as for `{play}_se-group.nodes.csv`, but the text unit nodes differ.
    Use to generate se-scene-{b,w} representations.
  - `{play}_se-speech-mwd.edges.csv`:
    Directed multi-edges for star expansions aggregated at the speech act level. Multi-edges can occur because there exists one edge per speech act, but text unit nodes are resolved at the stage group level, and one stage group can contain several speech acts.
    Use to generate the se-speech-mwd representation.
  - `{play}_se-speech-wd.edges.csv`:
    Directed edges for star expansions aggregated at the speech act level, with multi-edges aggregated into edge weights.
    Use to generate the se-speech-wd representation.
  - `{play}_se-speech.nodes.csv`:
    Nodes for star expansions aggregated at the speech act level. The same as `{play}_se-group.nodes.csv`; provided separately to facilitate the matching between node and edge files.
    Use to generate se-speech-{wd,mwd} representations.
- Files to construct *hypergraphs* (`hg`, i.e., generalized graph representations allowing edges with cardinalities in $\mathbb{N}$):
  - `{play}_hg-group-mw.edges.csv`:
    Edges for hypergraph representations resolved at the stage group level.
    Use to generate hg-group-{mb,mw} representations.
  - `{play}_hg-group-mw.node-weights.csv`:
    Edge-specific node weights for hypergraph representations resolved at the stage group level.
    Use to generate hg-group-{mb,mw} representations with edge-specific node weights.

- `{play}_hg-scene-mw.edges.csv`:
  Edges for hypergraph representations resolved at the scene level.
  Use to generate hg-scene-{mb,mw} representations.
- `{play}_hg-scene-mw.node-weights.csv`:
  Edge-specific node weights for hypergraph representations resolved at the scene level.
  Use to generate hg-scene-{mb,mw} representations with edge-specific node weights.
- `{play}_hg-speech-mwd.edges.csv`:
  Directed, weighted multi-edges for hypergraph representations resolved at the speech act level, where both the source and the target can contain multiple nodes.
  Use to generate the hg-speech-mwd representation.
- `{play}_hg-speech-wd.edges.csv`:
  Directed, weighted edges for hypergraph representations resolved at the speech act level, where both the source and the target can contain multiple nodes, with multi-edges aggregated into edge weights
  Use to generate the hg-speech-wd representation.
- `{play}_hg.nodes.csv`:
  Nodes for all hypergraph representations. Technically redundant because hyperedges can have cardinality 1, too, such that all nodes can be derived from the edge lists. Provided with global node weights for convenience.
  Use to generate all hg-$*$ representations.

The rows in each file represent either nodes or edges.

The columns in the individual files differ depending on the *semantic mapping*, the *granularity*, and the *expressivity* of the file contents, all of which are expressed in the file name (cf. Table 4.1), but the column semantics should be intuitive in light of the details on the `{play}.agg.csv` file columns given above. Note the following conventions for column names in edge lists:

- For clique and star expansions, if the graph is undirected, the nodes are called `node1` and `node2`, and if the graph is directed, the nodes are called `source` and `target`.
- If edges are count-weighted, the weight column is called `count`, otherwise, the columns `n_tokens` and `n_lines` can both serve as edge weights.
- For multi-edges in clique and star expansions, the column `edge_index` ensures that there are no duplicate rows. In hypergraphs, this is ensured by the `setting` column.

Finally, when working with the edge lists, please refer to the *expressivity* column in Table 4.1 to check whether the edge ordering in any particular file is intrinsically meaningful.

Examples:

– Nodes for clique expansions:

```
node
#ATTENDANTS.PRINCE_Rom
#ATTENDANTS_Rom
#Apothecary_Rom
#Benvolio_Rom
#Boy_Rom
...
```

– Edges for clique expansions (here: ce-group-mw):

```
node1,node2,key,act,scene,stagegroup,n_tokens,n_lines,edge_index
#SERVANTS.CAPULET.Gregory_Rom,#SERVANTS.CAPULET.Sampson_Rom,0,1,1,2,254,33,2
#SERVANTS.CAPULET.Gregory_Rom,#SERVANTS.CAPULET.Sampson_Rom,1,1,1,3,149,25,3
#SERVANTS.CAPULET.Gregory_Rom,#SERVANTS.MONTAGUE.1_Rom,0,1,1,3,149,25,3
#SERVANTS.CAPULET.Gregory_Rom,#SERVANTS.MONTAGUE.Abram_Rom,0,1,1,3,149,25,3
#SERVANTS.CAPULET.Sampson_Rom,#SERVANTS.MONTAGUE.1_Rom,0,1,1,3,149,25,3
...
```

– Nodes for star expansions (here: se-group):

```
node,node_type
#ATTENDANTS.PRINCE_Rom,character
#ATTENDANTS_Rom,character
#Apothecary_Rom,character
...
0.00.0001,text_unit
1.01.0002,text_unit
1.01.0003,text_unit
...
```

– Edges for star expansions (here: se-speech-mwd):

```
source,target,key,n_lines,n_tokens,edge_index,edge_type
#Chorus_Rom,0.00.0001,0,14,106,1,active
#SERVANTS.CAPULET.Sampson_Rom,1.01.0002,0,1,8,2,active
1.01.0002,#SERVANTS.CAPULET.Gregory_Rom,0,1,8,2,passive
#SERVANTS.CAPULET.Gregory_Rom,1.01.0002,0,1,7,3,active
1.01.0002,#SERVANTS.CAPULET.Sampson_Rom,0,1,7,3,passive
...
```

– Nodes for hypergraphs:

```
node,n_tokens_onstage,n_tokens_speaker,
    n_lines_onstage,n_lines_speaker
#ATTENDANTS.PRINCE_Rom,1147,0,150,0
#ATTENDANTS_Rom,905,0,121,0
#Apothecary_Rom,224,53,29,7
#Benvolio_Rom,5671,1160,771,161
#Boy_Rom,905,0,121,0
...
```

– Edge-specific node weights for hypergraphs (here: hg-scene-mw):

```
act,scene,node,n_tokens_speaker,n_lines_speaker,
    n_tokens_onstage,n_lines_onstage
0,0,#Chorus_Rom,106,14,106,14
1,1,#Benvolio_Rom,376,52,1403,189
1,1,#CITIZENS_Rom,16,2,237,32
1,1,#Capulet_Rom,26,3,221,30
1,1,#LadyCapulet_Rom,10,2,221,30
...
```

– Edges for hypergraphs (here: hg-speech-mwd):

```
act,scene,stagegroup,setting,speaker,onstage,n_tokens,n_lines
0,0,1,1,#Chorus_Rom,#Chorus_Rom,106,14
1,1,2,2,#SERVANTS.CAPULET.Sampson_Rom,#SERVANTS.CAPULET.
   Gregory_Rom #SERVANTS.CAPULET.Sampson_Rom,8,1
1,1,2,3,#SERVANTS.CAPULET.Gregory_Rom,#SERVANTS.CAPULET.
   Gregory_Rom #SERVANTS.CAPULET.Sampson_Rom,7,1
...
```

4.C.4 metadata

This folder currently contains exactly one CSV file, which maps play identifiers to play types. The file can be read with any CSV parser, such as the parser from the `pandas` library in Python, but since its provenance is documented as a comment at the start of the file, the # character needs to be passed to the parser as a comment character. Rows correspond to plays. Columns in alphabetical order:

– `play_name`: The name of the play, as used to fill the `{play}` placeholder in all play-specific file names.
Type: String.

– `play_type`: The type of the play. One of {comedy, history, tragedy}.
Type: String.

4.D Play Documentation

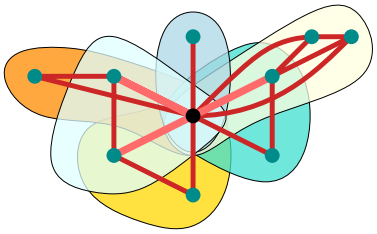
4.D.1 Inspiration

The play deliberately adopts and adapts ideas and text fragments from Shakespeare's works and other popular texts. These are:

– Dramatis Personæ: Three deadlines ~ three witches from Shakespeare's *Macbeth*
– Induction: Framing device used in Shakespeare's *The Taming of the Shrew*
– Act I, Scene II, l. 32: A phrase famously *attributed* to Martin Luther
– Act II, Scene I, l. 123: Allusion to a series of sketches from Monty Python's Flying Circus
– Act II, Scene III, ll. 154–174: Jon's speech from Shakespeare's *As You Like It*
– Act II, Scene IV, ll. 179–186: Faust's speech from Goethe's Faust I
– Act III, Scene I, ll. 298–311: Ariel's Song from Shakespeare's *The Tempest*
– Act III, Scene I, ll. 314–327: Hamlet's monologue from Shakespeare's *Hamlet*
– Act IV, Scene III, l. 365: Juliet addressing Romeo in Shakespeare's *Romeo and Juliet*
– Act IV, Scene III, l. 395: Pieces from Jon's interactions in Shakespeare's *As You Like It*
– Act V, Scene I, ll. 405–418: Shakespeare's *Full Many a Glorious Morning Have I Seen* (Sonnet 33)
– Act V, Scene II, ll. 419–427: Macbeth's monologue from Shakespeare's *Macbeth*

4.D.2 Style

Our layout follows the Oxford Shakespeare from 1916 [238] (whose text sometimes differs from the Folger Shakespeare underlying our data [195], especially in the stage directions). We adopt the basic language patterns characteristic of Shakespeare's plays, using primarily blank verse, i.e., non-rhyming verse in iambic pentameter with feminine endings allowed, but also prose and rhyming verse. Our main character switches between blank verse and prose depending on their internal state. Longer passages of rhyming verse occur in song and sonnet adaptations (see Section 4.D.1); shorter passages of rhyming verse are scattered throughout the play. We generally use Modern American English, sprinkled with brief interludes of Old British English.



5

Expressivity: Orchid

One of the main insights from Chapter 4 is that *representations matter*, and that some relational phenomena cannot be adequately represented as *graphs* because ordinary edges can only model binary relations. *Hypergraphs* presented themselves as an alternative—capable, inter alia, of capturing nuances in the multilateral character interactions present in Shakespeare's plays. However, there are comparatively few methods for hypergraph analysis that truly leverage the richness of hypergraph data. This motivates us to explore the *expressivity* dimension of Graphland, such that we now investigate:

How can we exploit the expressivity *of hypergraphs in our analyses?*

5.1 Introduction

Hypergraphs generalize graphs by allowing any number of nodes to participate in an edge. They enable us to faithfully represent complex relations, such as co-authorship of scientific papers, multilateral interactions between chemicals, or group conversations, which cannot be adequately captured by graphs. While hypergraphs are more expressive than graphs and other relational objects like simplicial complexes, they are harder to analyze both theoretically and empirically, and many concepts that have proven useful for understanding graphs have yet to be transferred to the hypergraph setting.

Curvature has established itself as a powerful characteristic of Riemannian manifolds, as it permits the description of *global properties* through *local measurements* by harmonizing ideas from geometry and topology. For graphs, *graph curvature* measures to what extent the neighborhood of an edge deviates from certain idealized model spaces, such as cliques, grids, or trees. It has proven helpful, for example, in assessing differences between real-world networks [233], identifying bottlenecks in real-world networks [106], and alleviating oversquashing in graph neural networks [254]. One prominent notion of graph curvature is *Ollivier-Ricci curvature* (ORC). ORC compares random walks based at specific nodes, revealing differences in the information diffusion behavior in the graph. As the sizes of edges and edge intersections can vary in hypergraphs, there are many ways to generalize ORC to hypergraphs. While some notions of hypergraph ORC have been previously studied in isolation [e.g., 10, 83, 158], a unified framework for their definition and computation is still lacking.

Contributions. In this chapter, we make three contributions. First, we disentangle the individual building blocks of hypergraph ORC and identify its desirable properties. Second, we introduce Orchid, a unified framework for Ollivier-Ricci curvature on hypergraphs, which integrates and generalizes existing approaches to hypergraph ORC. And third, we perform a rigorous theoretical and empirical analysis of Orchid curvatures, establishing that our notions of hypergraph ORC are aligned with our geometric intuition while still efficient to compute, and that these notions are also useful to perform a variety of hypergraph tasks in practice.

Structure. After providing the necessary background on graphs and hypergraphs and recalling the definition of Ollivier-Ricci curvature for graphs in Section 5.2, we introduce Orchid, our framework for hypergraph ORC, and analyze the theoretical properties of Orchid curvatures in Section 5.3. Having discussed related work in Section 5.4, we assess the empirical properties and practical utility of Orchid curvatures through extensive experiments in Section 5.5. We conclude with a discussion in Section 5.6, provide further supplementary material in Sections 5.A to 5.D, and make all code, data, and results publicly available.[1]

5.2 Preliminaries

We begin by stating our terminology for graphs and hypergraphs (Section 5.2.1) as well as defining Ollivier-Ricci curvature for graphs (Section 5.2.2). For convenience, we collect our basic notation in Table 5.4.

[1] 10.5281/zenodo.7624573

5.2.1 Graphs and Hypergraphs

A *simple graph* $G = (V, E)$ is a tuple containing n nodes (vertices) $V = \{v_1, \ldots, v_n\}$ and m edges $E = \{e_1, \ldots, e_m\}$, with $e_i \in \binom{V}{2}$ for all $i \in [m]$. Here, for a set S and a positive integer $k \leqslant |S|$, $\binom{S}{k}$ denotes the set of all k-element subsets of S, and for $x \in \mathbb{N}$ with $0 \notin \mathbb{N}$, $[x] = \{i \in \mathbb{N} \mid i \leqslant x\}$. In *multi-graphs*, edges can occur multiple times, and hence, $E = (e_1, \ldots, e_m)$ is an indexed family of sets, with $e_i \in \binom{V}{2}$ for all $i \in [m]$. Generalizing simple graphs, a *simple hypergraph* $H = (V, E)$ is a tuple containing n nodes V and m hyperedges $E \subseteq \mathcal{P}(V) \setminus \emptyset$, i.e., in contrast to edges, hyperedges can have any cardinality $r \in [n]$. In a *multi-hypergraph*, $E = (e_1, \ldots, e_m)$ is an indexed family of sets, with $e_i \subseteq V$ for all $i \in [m]$. We assume that all our hypergraphs are multi-hypergraphs, and we drop the prefix *hyper* from *hypergraph* and *hyperedge* where it is clear from context.

We denote the *degree* of node i, that is, the number of edges containing i, as $\deg(i) = |\{e \in E \mid i \in e\}|$, write $i \sim j$ if i is adjacent to j (i.e., there exists $e \in E$ such that $\{i, j\} \subseteq e$), and use $\mathcal{N}(i)$ ($\mathcal{N}(e)$) for the *neighborhood* of i (e), i.e., the set of nodes adjacent to i (edges intersecting edge e). While $\deg(i) = |\mathcal{N}(i)|$ in simple graphs and $\deg(i) \geqslant |\mathcal{N}(i)|$ in multi-graphs, these relations do not generally hold for hypergraphs. Two distinct nodes $i \neq j$ are *connected* in H if there is a sequence of nodes $i = v_1, v_2, \ldots, v_{k-1}, v_k = j$ such that $v_l \sim v_{l+1}$ for all $l \in [k]$. Every such sequence is a *path* in H, whose *length* is the cardinality of the set of edges used in the adjacency relation. We refer to the length of a shortest path connecting nodes i and j as the *distance* between them, denoted as $d(i, j)$. We assume that all (hyper)graphs are *connected*, i.e., there exists a path between all pairs of nodes. This turns H into a metric space (H, d) with *diameter* $\operatorname{diam}(H) = \max\{d(i, j) \mid i, j \in V\}$.

(Hyper)graphs in which all nodes have the same degree k ($\deg(i) = k$ for all $i \in V$) are called k-*regular*. Three properties of hypergraphs that distinguish them from graphs give rise to additional (ir)regularities. First, *hyperedges* can vary in cardinality, and a hypergraph in which all hyperedges have the same cardinality r ($|e| = r$ for all $e \in E$) is called r-*uniform*. Second, *hyperedge intersections* can have cardinality greater than 1, and we call a hypergraph s-*intersecting* if all nonempty edge intersections have the same cardinality s ($e \cap f \neq \emptyset \Leftrightarrow |e \cap f| = s$ for all $e, f \in E$). Third, nodes can *co-occur in any number of hyperedges*; we call a hypergraph c-*co-occurrent* if each node co-occurs c times with any of its neighbors (that is, for all $i, j \in V$, we have $i \sim j \Leftrightarrow |\{e \in E \mid \{i, j\} \subseteq e\}| = c$). Using this terminology, simple graphs are 2-uniform, 1-intersecting, 1-co-occurrent hypergraphs.

Given a hypergraph $H = (V, E)$, the *unweighted clique expansion* of H is $G^\circ = (V, E^\circ)$ with $E^\circ = \{\{i, j\} \mid \{i, j\} \subseteq e \text{ for some } e \in E\}$, where two nodes are adjacent

in G° if and only if they are adjacent in H. The *weighted clique expansion* of H is G° endowed with a weighting function $w\colon E^\circ \to \mathbb{N}$, where $w(e) = |\{e \in E \mid \{i,j\} \subseteq e\}|$ for each $e \in E^\circ$, i.e., an edge $\{i,j\}$ is weighted by how often i and j co-occur in edges from H. Both of these transformations are lossy, i.e., we cannot uniquely reconstruct H from G°. The *unweighted star expansion* of H is the bipartite graph $G' = (V', E')$ with $V' = V \dot\cup E$ and $E' = \{\{i,e\} \mid i \in V, e \in E, i \in e\}$, and we can uniquely reconstruct H from G' if we know which of its parts corresponds to the original node set of H.

5.2.2 Ollivier-Ricci Curvature for Graphs

Ollivier-Ricci curvature (ORC) extends the notion of Ricci curvature, defined for Riemannian manifolds, to metric spaces equipped with a probability measure or, equivalently, a random walk [208, 209]. On graphs, which are metric spaces with the shortest-path distance $d(\cdot,\cdot)$, the ORC κ of a pair of nodes $\{i,j\}$ is defined as

$$\kappa(i,j) = 1 - \frac{1}{d(i,j)} W_1(\mu_i, \mu_j) \text{ , and hence, } \kappa(i,j) = 1 - W_1(\mu_i,\mu_j) \text{ if } i \sim j \text{ , } \quad (5.1)$$

where μ_i is a probability measure associated with node i that depends measurably on i and has finite first moment, and W_1 is the *Wasserstein distance* of order 1, which captures the amount of work needed to transport the probability mass from μ_i to μ_j in an optimal coupling. The use of the shortest-path distance is necessary to ensure that ORC is also well-defined for pairs of non-adjacent nodes. This definition on edges or pairs of nodes alludes to the fact that Ricci curvature is associated to tangent vectors of a manifold. A common strategy to measure curvature at a node i is to average over the curvatures of all edges incident with i [18, 137], i.e.,

$$\kappa(i) = \frac{1}{\deg(i)} \sum_{\{i,j\} \in E} \kappa(i,j) \text{ .} \quad (5.2)$$

A popular probability measure that easily generalizes to weighted graphs and multigraphs is

$$\mu_i^\alpha(j) = \begin{cases} \alpha & j = i \\ (1-\alpha)\frac{1}{\deg(i)} & i \sim j \\ 0 & \text{otherwise ,} \end{cases} \quad (5.3)$$

where α serves as a smoothing parameter [171]. With this definition, stacking the probability measures yields the transition matrix of an α-lazy random walk.

5.3 Theory

Having introduced the concept of hypergraphs and the definition of Ollivier-Ricci curvature (ORC) for graphs in Section 5.2, we now develop our framework for ORC on hypergraphs, called Orchid (Ollivier-Ricci Curvature for Hypergraphs In Data). We focus our exposition on undirected, unweighted multi-hypergraphs, but Orchid straightforwardly generalizes to other hypergraph variants.

5.3.1 Ollivier-Ricci Curvature for Hypergraphs (Orchid Curvatures)

As mentioned in Section 5.2.1, hypergraphs differ from graphs in that edges can have any cardinality, and consequently, edges can intersect in more than one node, and nodes can co-occur in more than one edge. When generalizing ORC as defined in Section 5.2.2 to hypergraphs, these peculiarities become relevant in two places: (1) in the generalization of the measure μ for nodes, and (2) in the generalization of the distance metric W_1. Construing the distance metric as a function *aggregating* measures (`Agg`), with $\texttt{Agg}\colon V^+ \to \mathbb{R}$, we can rewrite Eq. (5.1) for pairs of nodes $\{i, j\}$ as

$$\kappa(i, j) = 1 - \frac{\texttt{Agg}(\mu_i, \mu_j)}{d(i, j)}\,, \tag{5.4}$$

which facilitates its generalization; we will also use $\kappa(e)$ for (hyper)edges as a shorthand notation for Eq. (5.4). When defining probability measures μ and aggregation functions `Agg` on hypergraphs, we would like to retain as much flexibility as possible while also ensuring the following conditions:

I. Mathematical generalization. For graphs, `Agg` simplifies to the original ORC on graphs.
II. Permutation invariance. $\texttt{Agg}(e) = \texttt{Agg}(\sigma(e))$ for edges e and all node index permutations σ.
III. Scalability. The probability measures and aggregation functions should be efficiently computable.

Beyond these properties, we would also like to have the following *interpretability* features to ascertain that a hypergraph curvature measure is a *conceptual generalization* of ORC:

A. Probabilistic intuition. The probability measures assigned to nodes should correspond to a semantically sensible random walk on the hypergraph.
B. Optimal transport intuition. The generalization of the distance metric (`Agg`) should have a semantically sensible interpretation in terms of optimal transport.

C. Geometric intuition. Edges in hypercliques should have positive curvature, edges in hypergrids should have curvature zero, and edges in hypertrees should have negative curvature.

We now specify probability measures and `Agg` functions for which the conditions above hold.

Probability Measures (μ). In graphs, the most natural probability measures are induced by the α-lazy random walk given in Eq. (5.3): With probability α, we stay at the current node i, and with probability $\frac{(1-\alpha)}{\deg(i)}$, we move to one of its neighbors. There are at least three direct extensions of this formulation to hypergraphs that all retain this probabilistic intuition, thus fulfilling the requirement of Feature A. These extensions, illustrated in Fig. 5.1, differ only in how they distribute the $(1-\alpha)$ probability mass in Eq. (5.3) from node i to the nodes in i's neighborhood. Given a hypergraph H, for i and j with $i \sim j$, first, we could define

$$\mu_i^{\texttt{EN}}(j) = (1-\alpha)\frac{1}{|\mathcal{N}(i)|}\,, \tag{5.5}$$

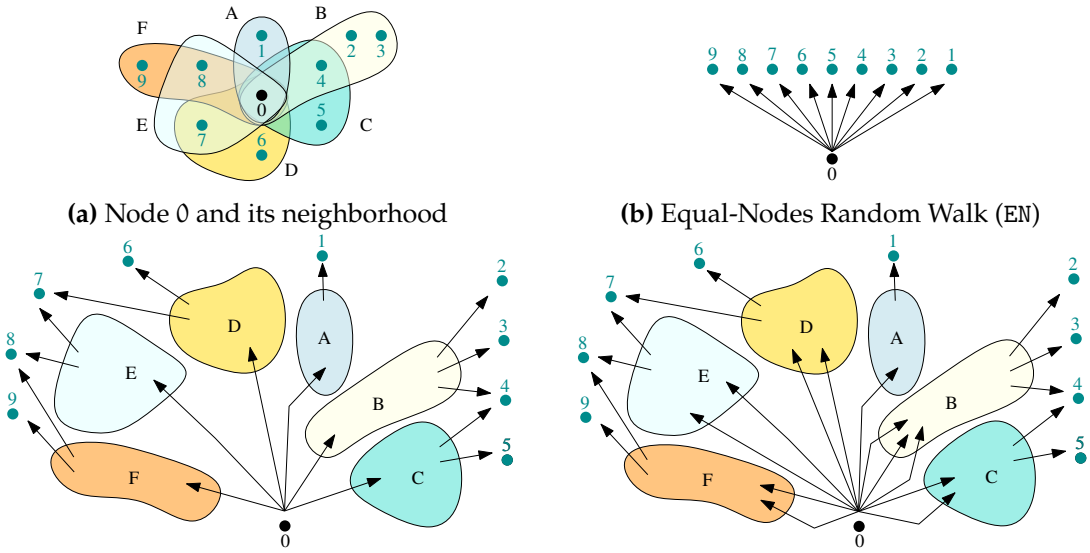
by which we pick a neighbor j of node i uniformly at random. We call this the *equal-nodes random walk* (`EN`), which is a random walk on the *unweighted clique expansion* of H. Second, we could set

$$\mu_i^{\texttt{EE}}(j) = (1-\alpha)\frac{1}{\deg(i) - |\{e \ni i \mid |e| = 1\}|}\sum_{e \supseteq \{i,j\}}\frac{1}{|e|-1}\,, \tag{5.6}$$

which first picks an edge $e \ni i$ with $|e| \geqslant 2$, then picks a node $j \in e \setminus \{i\}$, both uniformly at random. We call this the *equal-edges random walk* (`EE`), which is a two-step random walk on the *unweighted star expansion* of H, starting at a node $i \in V$, and non-backtracking in the second step. It underlies the curvatures studied by Asoodeh, Gao, and Evans [10] and Banerjee [18]. Third, we could define

$$\mu_i^{\texttt{WE}}(j) = (1-\alpha)\sum_{e \supseteq \{i,j\}}\frac{|e|-1}{\sum_{f \ni i}\big(|f|-1\big)}\frac{1}{|e|-1} = (1-\alpha)\frac{|\{e \in E \mid \{i,j\} \subseteq e\}|}{\sum_{f \ni i}\big(|f|-1\big)}\,, \tag{5.7}$$

first picking an edge e incident with i with probability proportional to its cardinality, then picking a node $j \in e \setminus \{i\}$ uniformly at random. We call this the *weighted-edges random walk* (`WE`): a two-step random walk from a node $i \in V$ on a specific *directed weighted star expansion* of H whose second step is non-backtracking—or equivalently, a random walk on a *weighted clique expansion* of H.



(a) Node 0 and its neighborhood

(b) Equal-Nodes Random Walk (EN)

(c) Equal-Edges Random Walk (EE)

(d) Weighted-Edges Random Walk (WE)

Figure 5.1: The probability measures underlying ORCHID are based on random walks, depicted here for the neighborhood of a node 0. Arrows outgoing from the same node or edge are traversed with uniform probability.

AGGREGATION FUNCTIONS (AGG). In the original formulation of ORC, i.e., Eq. (5.1), when determining the curvature of an edge $\{i, j\}$, the Wasserstein distance W_1 is used to aggregate the probability measures of i and j. There are at least three different extensions of this aggregation scheme to hypergraphs that retain an optimal transport intuition, as required by Feature B. Leveraging that an edge $e \subseteq V$ is simply a set of nodes, the easiest extension is to leave the aggregation function unchanged. We continue determining the curvature for pairs of nodes, and account for the edges in H only in the definition of our probability measure. In this case, we could derive a curvature for an edge e as the average over all curvatures of node pairs contained in e, i.e., we could define AGG as

$$\text{AGG}_A(e) = \frac{2}{|e|(|e|-1)} \sum_{\{i,j\} \subseteq e} W_1\left(\mu_i, \mu_j\right) . \tag{5.8}$$

This is equivalent to computing the curvature of e based on the average over all W_1 distances of probability measures associated with nodes contained in e:

$$\kappa_A(e) = 1 - \text{AGG}_A(e) = 1 - \frac{2}{|e|(|e|-1)} \sum_{\{i,j\} \subseteq e} W_1(\mu_i, \mu_j) = \frac{2}{|e|(|e|-1)} \sum_{\{i,j\} \subseteq e} \kappa(i, j) . \tag{5.9}$$

Intuitively, this definition assesses the mean amount of work needed to transport the probability mass from one node in e to another node in e. Alternatively, and

still keeping with the intuition from optimal transport, we can define $\texttt{Agg}$ as

$$\texttt{Agg}_{\texttt{B}}(e) = \frac{1}{|e|-1} \sum_{i \in e} W_1(\mu_i, \bar{\mu}) \ , \quad \text{and consequently,} \quad \kappa_{\texttt{B}}(e) = 1 - \texttt{Agg}_{\texttt{B}}(e) \ , \tag{5.10}$$

where $\bar{\mu}$ denotes the Wasserstein *barycenter* of the probability measures of nodes contained in e, and the denominator generalizes the original $d(i,j)$. Asoodeh, Gao, and Evans [10] use this aggregation function. Intuitively, $\texttt{Agg}_{\texttt{B}}$ is proportional to the minimum amount of work needed to transport all probability mass from the probability measures of the nodes to one place, with the caveat that this place need not correspond to a node in the underlying hypergraph. Finally, we can capture the maximum amount of work needed to transport all probability mass from one node in e to another node in e as

$$\texttt{Agg}_{\texttt{M}}(e) = \max\{W_1(\mu_i, \mu_j) \mid \{i,j\} \subseteq e\} \ , \quad \text{and consequently,} \quad \kappa_{\texttt{M}}(e) = 1 - \texttt{Agg}_{\texttt{M}}(e) \ . \tag{5.11}$$

Independent of the choice of $\texttt{Agg}$, the curvature at a node i can be defined as the mean of all curvatures of meaningful directions containing i, i.e.,

$$\kappa^{\mathcal{N}}(i) = \frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \kappa(i,j) \ , \tag{5.12}$$

or it can be derived as the mean of all curvatures of edges containing i, i.e.,

$$\kappa^{E}(i) = \frac{1}{\deg(i)} \sum_{e \ni i} \kappa(e) \ . \tag{5.13}$$

Finally, since H is connected, we can define the curvature of an arbitrary subset of nodes $s \subseteq V$ as

$$\kappa(s) = 1 - \frac{\texttt{Agg}(s)}{d(s)} \ , \tag{5.14}$$

where $\texttt{Agg}$ can be any of our aggregation functions, and

$$d(s) = \max\{d(i,j) \mid \{i,j\} \subseteq s\} \tag{5.15}$$

refers to the *extent* of the subset s. Note that for $s \in E$, $d(s) = 1$, and thus, Eq. (5.14) is consistent with our previous definitions of hyperedge curvatures.

5.3.2 Properties of Orchid Curvatures

Having introduced our probability measures (μ) and aggregation functions ($\texttt{Agg}$), we now analyze their properties and the properties of the resulting curvatures.

First, we note that μ^{EN}, μ^{EE}, and μ^{WE} are equivalent for certain hypergraph classes, and all aggregation functions coincide for graphs.

Lemma 5.1. *For graphs and* r*-uniform,* k*-regular,* c*-co-occurrent hypergraphs, we have* $\mu^{\text{EN}} = \mu^{\text{EE}} = \mu^{\text{WE}}$.

Proof. For notational simplicity, without loss of generality, we assume that $\alpha = 0$. In an r-uniform, k-regular, c-co-occurrent hypergraph $H = (V, E)$, each node i has degree k and $\frac{(r-1)k}{c}$ neighbors, and each edge has cardinality r. Hence, for nodes $i, j \in V$ with $i \sim j$,

$$\begin{aligned} \mu_i^{\text{EN}}(j) = \frac{1}{|\mathcal{N}(i)|} = \frac{c}{(r-1)k} = \frac{1}{k} \cdot c \cdot \frac{1}{r-1} = \frac{1}{\deg(i)} \sum_{e \ni i,j} \frac{1}{|e|-1} = \mu_i^{\text{EE}}(j) \\ = \frac{c}{k(r-1)} = \frac{|\{e \in E \mid \{i,j\} \subseteq e\}|}{\sum_{f \ni i} \big(|f|-1\big)} = \mu_i^{\text{WE}}(j) \,. \end{aligned} \tag{5.16}$$

Graphs are 2-uniform and 1-co-occurrent (but not generally regular), and hence, $|\mathcal{N}(i)| = \deg(i)$. Using this to simplify the probability measure expressions, the claim follows. □

Lemma 5.2. *For graphs, i.e., 2-uniform hypergraphs, we have* $\texttt{AGG}_A(e) = \texttt{AGG}_B(e) = \texttt{AGG}_M(e)$ *for all edges* $e \in E$.

Proof. Given probability distributions $\mu_1, \mu_2, \ldots, \mu_n$, their Wasserstein barycenter is defined as the distribution $\bar{\mu}$ that minimizes $f(\bar{\mu}) = \frac{1}{n} \sum_{i=1}^{n} W_1(\bar{\mu}, \mu_i)$. Since $|e| = 2$, we minimize $W_1(\bar{\mu}, \mu_1) + W_1(\bar{\mu}, \mu_2)$. The Wasserstein distance is a metric, so it satisfies the triangle inequality. Thus, $W_1(\mu_1, \mu_2) \leqslant W_1(\bar{\mu}, \mu_1) + W_1(\bar{\mu}, \mu_2)$ for all choices of $\bar{\mu}$. Hence, f is minimized by either μ_1 or μ_2. Evaluating both cases yields $\texttt{AGG}_A(e) = \texttt{AGG}_B(e)$, and observing that $\texttt{AGG}_M(e) = W_1(\mu_i, \mu_j)$ for $e = \{i, j\}$ by definition, the claim follows. □

Taken together, Lemma 5.1 and Lemma 5.2 imply that for graphs, ORCHID simplifies to ORC, regardless of the choice of probability measure and aggregation function. This fulfills Condition I. Moreover, *all* our aggregation functions are permutation-invariant by construction, thus satisfying Condition II. Concerning Condition III, κ_A and κ_M exhibit better scalability than κ_B, as Wasserstein barycenters are harder to compute than individual distances [71]. Another reason to prefer κ_A and κ_M over κ_B is the existence of upper and lower bounds that are easy to calculate. To this end, let $d_{\min}(H) = \min\{d(u, v) \mid u \neq v \in V\}$ be the smallest nonzero distance in H, and let $\|\cdot\|_1$ refer to the L_1 norm of a vector. We then obtain the following bounds for κ_A and κ_M.

Theorem 5.3. *For any probability measure* μ *and* $C(e) = \frac{2}{|e|(|e|-1)}$*, the curvature* $\kappa_A(e)$ *of an edge* $e \in E$ *is bounded by*

$$1-\mathrm{diam}(H)C(e)\sum_{\{i,j\}\subseteq e}\|\mu_i-\mu_j\|_1 \leqslant \kappa_A(e) \leqslant 1-\mathrm{d_{min}}(H)C(e)\sum_{\{i,j\}\subseteq e}\|\mu_i-\mu_j\|_1 \ . \quad (5.17)$$

Proof. We bound each of the summands in the curvature calculation. Given probability measures μ_i, μ_j, a result by Gibbs and Su [101, Theorem 4] states that

$$\mathrm{d_{min}}(H)\,\mathrm{d_{TV}}(\mu_i,\mu_j) \leqslant W_1(\mu_i,\mu_j) \leqslant \mathrm{diam}(H)\,\mathrm{d_{TV}}(\mu_i,\mu_j)\ , \quad (5.18)$$

where $\mathrm{d_{TV}}$ refers to the *total variation distance*. The intuition behind this bound is that the total variation distance represents a specific type of transport plan between the two probability measures; the factors arising from the minimum (maximum) distance in a space indicate the minimum (maximum) distance that realizes this transport plan. Since all our measures are defined over a finite space, we have $\mathrm{d_{TV}}(\mu_i,\mu_j) = \frac{1}{2}\|\mu_i-\mu_j\|_1$. The claim follows by considering that pairwise distances are being *subtracted* to calculate our curvature measure. □

Theorem 5.4. *For any probability measure* μ*, the curvature* $\kappa_M(e)$ *of an edge* $e \in E$ *is bounded by*

$$1-\mathrm{diam}(H)\max_{\{i,j\}\subseteq e}\|\mu_i-\mu_j\|_1 \leqslant \kappa_M(e) \leqslant 1-\mathrm{d_{min}}(H)\max_{\{i,j\}\subseteq e}\|\mu_i-\mu_j\|_1 \ . \quad (5.19)$$

Proof. For $\texttt{Agg}_M$, Eq. (5.18) applies for a single pairwise distance only. We thus only obtain a single bound based on the maximum total variation distance between two probability measures. □

Directly from our definitions, we further obtain the following relationships between κ_A, κ_B, and κ_M, and between Orchid curvatures on hypergraphs and ORC on their unweighted clique expansions.

Corollary 5.5. *Given a hypergraph* $H = (V, E)$*,* $\kappa_M(e) \leqslant \kappa_A(e)$ *and* $\kappa_M(e) \leqslant \kappa_B(e)$ *for all* $e \in E$*.*

Corollary 5.6. *Given a hypergraph* $H = (V, E)$ *and its unweighted clique expansion* $G^\circ = (V, E^\circ)$*, for* $\{i, j\} \in E^\circ$*, the ORC* $\kappa(i, j)$ *in* G° *equals its Orchid curvature* $\kappa(i, j)$ *of direction* $\{i, j\} \subseteq V$ *in* H *with* μ^{EN}*, and the ORC* $\kappa(i)$ *of* $i \in V$ *in* G° *equals its Orchid curvature* $\kappa^{\mathcal{N}}(i)$ *in* H *with* μ^{EN}*.*

Corollary 5.6 clarifies that the equal-nodes random walk establishes the connection between ORCHID and ORC on graphs. Moreover, ORCHID curvatures capture relations between *global* properties and *local* measurements, similar to the Bonnet–Myers theorem in Riemannian geometry [199].

Theorem 5.7. *Given a subset of nodes* $s \subseteq V$ *and an arbitrary probability measure* μ*, let* δ_i *denote a Dirac measure at node* i*, and let* $J(\mu_i) = W_1(\delta_i, \mu_i)$ *denote the* jump probability *of* μ_i*. If (i) all curvatures based on* μ *are strictly positive, i.e.,* $\kappa(s) \geqslant \kappa > 0$ *for all* $s \subseteq V$*, and (ii)* $W_1(\mu_i, \mu_j) \leqslant \texttt{Agg}(s)$ *for* $\{i, j\} = \operatorname{argmax}(d(s))$*, then*

$$d(s) \leqslant \frac{J(i) + J(j)}{\kappa(s)} \,. \tag{5.20}$$

Proof. Let $\{i, j\} = \operatorname{argmax}(d(s))$ as required in the theorem. We then have following chain of (in)equalities:

$$d(s) = d(i, j) = W_1(\delta_i, \delta_j) \leqslant W_1(\delta_i, \mu_i) + W_1(\mu_i, \mu_j) + W_1(\mu_j, \delta_j) \,. \tag{5.21}$$

Rearranging Eq. (5.14), we have $(1 - \kappa(s))\, d(s) = \texttt{Agg}(s)$. According to our assumptions, $W_1(\mu_i, \mu_j) \leqslant \texttt{Agg}(s) = (1 - \kappa(s))\, d(i, j)$. Inserting this into Eq. (5.21) yields

$$\begin{aligned} & d(i, j) \leqslant J(\mu_i) + J(\mu_j) + (1 - \kappa(s))\, d(i, j) && (5.22) \\ \Leftrightarrow \quad & d(i, j) - (1 - \kappa(s))\, d(i, j) \leqslant J(\mu_i) + J(\mu_j) \\ \Leftrightarrow \quad & d(i, j) \leqslant \frac{J(i) + J(j)}{\kappa(s)} \,, \end{aligned}$$

where the last step is only valid since $\kappa(s) \geqslant \kappa > 0$ by assumption. □

Note that condition (ii) of Theorem 5.7 is always satisfied by $\texttt{Agg}_M$.

Finally, we generalize the concepts of cliques, grids, and trees (prototypical positively curved, flat, and negatively curved graphs) to hypergraphs, which enables us to ensure that ORCHID curvatures respect our geometric intuition, as required by Feature C.

Definition 5.8 (Hypercliques, hypergrids, hypertrees)**.** *A simple, connected hypergraph* $H = (V, E)$ *is*

- *a* hyperclique *if* $E = \binom{V}{r}$ *for some* $r \leqslant |V|$*,*
- *a* hypergrid *if* H *is an* r*-uniform hypergraph for which there exists a lattice* $L = (V, E_L)$ *s.t.* $E = \{e \in \binom{V}{r} \mid e$ *corresponds to a path of length* r *in* $L\}$*, and*

– *a* hypertree *if there exists a tree* $T = (V, E_T)$ *s.t. each edge* $e \in E_T$ *induces a subtree in* T.

Corollary 5.9. *Cliques are hypercliques, grids are hypergrids, and trees are hypertrees.*

Corollary 5.10. *If* $H = (V, E)$ *is a hyperclique, a hypergrid, or an* r*-uniform,* k*-regular,* 1*-intersecting hypertree, for* $i, j \in V$*, the sets* $S_i = \{e \in E \mid i \in e\}$ *and* $S_j = \{e \in E \mid j \in e\}$ *are isomorphic, i.e., there exists* $\varphi : \mathcal{N}(i) \cup \{i\} \to \mathcal{N}(j) \cup \{j\}$ *such that* $\{\{\varphi(x) \mid x \in e\} \mid e \in S_i\} = S_j$.

For hypercliques, hypergrids, and hypertrees with certain regularities, $\texttt{Agg}_A(e)$ and $\texttt{Agg}_M(e)$ are constants.

Lemma 5.11 (Hypercliques, hypergrids, hypertrees)**.** *If* $H = (V, E)$ *is a hyperclique, a hypergrid, or an* r*-uniform,* k*-regular,* 1*-intersecting hypertree, we have* $\texttt{Agg}_A(e) = \texttt{Agg}_M(e) = W_1(\mu_i, \mu_j) = w$ *for* $w \in \mathbb{R}$*,* $e \in E$*, and* $i, j \in V$ *with* $i \sim j$.

Proof. By Corollary 5.10, we have $w = W_1(\mu_i, \mu_j) = W_1(\mu_p, \mu_q)$ for $i, j, p, q \in V$ with $i \sim j$ and $p \sim q$. Hence $\texttt{Agg}_M(e) = w$, and

$$\texttt{Agg}_A(e) = \frac{2}{|e|(|e|-1)} \sum_{\{i,j\} \subseteq e} W_1(\mu_i, \mu_j) = \frac{2}{|e|(|e|-1)} \frac{|e|(|e|-1)}{2} w = w \,, \tag{5.23}$$

for $e \in E$. □

Corollary 5.12. *If* $H = (V, E)$ *is a hyperclique, a hypergrid, or an* r*-uniform,* k*-regular,* 1*-intersecting hypertree,* $\texttt{Agg}_A(e) = \texttt{Agg}_M(e)$.

Using Lemma 5.11, we now prove that under $\texttt{Agg}_A$ and $\texttt{Agg}_M$, hypercliques are positively curved, hypergrids are flat, and hypertrees are negatively curved, as desired.

Theorem 5.13 (Hyperclique curvature)**.** *For an edge* e *in a hyperclique* $H = (V, E)$ *on* n *nodes with edges* $E = \binom{V}{r}$ *for some* $r \leqslant n$*, with* $\alpha = 0$*,*

$$\kappa(e) = 1 - \frac{1}{n-1}, \text{ i.e., } \lim_{n \to \infty} \kappa(e) = 1, \text{ independent of } r.$$

Proof. A hyperclique is r-uniform, $(n-1)$-regular, and $(r-2)$-co-occurrent, so $\mu_i^{\texttt{EN}} = \mu_i^{\texttt{EE}} = \mu_i^{\texttt{WE}}$ for each node $i \in V$ by Lemma 5.1. Thus, considering $\mu_i^{\texttt{EN}}$, each node $i \in V$ has $n-1$ neighbors to which it distributes its probability mass equally, and we have $W_1(\mu_i, \mu_j) = \frac{1}{n-1}$ for $i, j \in V$ with $i \sim j$. The claim now follows from Lemma 5.11. □

Theorem 5.14 (Hypergrid curvature)**.** *For an edge e in a r-uniform, k-regular hypergrid, with $\alpha = 0$, $\kappa(e) = 0$, independent of r and k.*

Proof. By Corollary 5.10, the sets $S_i = \{e \in E \mid i \in e\}$ and $S_j = \{e \in E \mid j \in e\}$ are isomorphic, and due to the symmetries in the hypergrid, the isomorphism $\varphi\colon \mathcal{N}(i) \cup \{i\} \to \mathcal{N}(j) \cup \{j\}$ minimizing the cost $\sum_{x \in \mathcal{N}(i) \cup \{i\}} d(x, \varphi(x))$ corresponds to the coupling minimizing $W_1(\mu_i, \mu_j)$. The cost of φ equals the minimum cost of an isomorphism in H's underlying lattice L between the inclusive $(r-1)$-hop neighborhoods of two nodes adjacent in L, which is $|\mathcal{N}(i) \cup \{i\}|$. Hence, $W_1(\mu_i, \mu_j) = \frac{|\mathcal{N}(i) \cup \{i\}|}{|\mathcal{N}(i) \cup \{i\}|} = 1$ for $i, j \in V$ with $i \sim j$ and all choices of μ, and the claim then follows from Lemma 5.11. □

Theorem 5.15 (Hypertree curvature)**.** *For an edge e in a r-uniform, k-regular, 1-intersecting hypertree, with $\alpha = 0$,*

$$\kappa(e) = 1 - \left(\frac{3(k-1)}{k} + \frac{1}{(r-1)k}\right), \text{ i.e., } \lim_{k \to \infty} \kappa(e) = -2, \text{ independent of } r.$$

Proof. An r-uniform, k-regular, 1-intersecting hypertree is 1-co-occurrent, so we have $\mu_i^{\texttt{EN}} = \mu_i^{\texttt{EE}} = \mu_i^{\texttt{WE}}$ for each node $i \in V$ by Lemma 5.1. Each node $i \in V$ has $(r-1)k$ neighbors, such that $\mu_i^{\texttt{EN}}$ distributes a fraction $\frac{1}{(r-1)k}$ of the probability mass to each of i's neighbors. Nodes $i, j \in V$ with $i \sim j$ share $(r-2)$ neighbors (those in the unique edge e satisfying $\{i, j\} \subseteq e$), and the probability mass allocated by μ_i to j can be matched with the probability mass allocated by μ_j to i at cost 1. Because H is a hypertree, the remaining probability mass, $\frac{(r-1)(k-1)}{(r-1)k} = \frac{k-1}{k}$, needs to be transported from the neighborhood of i to the neighborhood of j at cost 3. Hence,

$$W_1(\mu_i, \mu_j) = 1 \cdot \frac{1}{(r-1)k} + 3 \cdot \frac{k-1}{k} \tag{5.24}$$

for $i, j \in V$ with $i \sim j$. Again, the claim follows from Lemma 5.11. □

Thus, ORCHID curvatures fulfill all our requirements for generalizing Ollivier-Ricci curvature from graphs to hypergraphs.

5.4 Related Work

Work related to Orchid broadly falls into three categories: hypergraph curvature, graph curvature, and hypergraph learning, mining, and analysis.

Hypergraph Curvature. Most closely related to our work is the literature on hypergraph curvatures. Much of this literature focuses on defining notions of ORC and Forman-Ricci Curvature (FRC) specifically for *directed* hypergraphs and studying some of their mathematical and empirical properties [e.g., 157–159, 234]. Notably, the directed hypergraph ORC introduced by Eidi and Jost [83] is an instantiation of our framework with μ^{EE} and $\texttt{Agg}_A$. Curvature notions for *undirected* hypergraphs are comparatively less explored, and especially the literature generalizing ORC is almost entirely theoretical. The generalization of ORC proposed by Asoodeh, Gao, and Evans [10] and the equivalent measure used by Banerjee [18] are instantiations of our framework using μ^{EE} and $\texttt{Agg}_B$. Akamatsu [5] propose (α, h)-ORC using cost functions based on structured optimal transport, and Ikeda et al. [131] define λ-coarse Ricci curvature using a λ-nonlinear Kantorovich difference based on a submodular hypergraph Laplacian as a generalization of ORC as introduced by Lin, Lu, and Yau [171]. Both works define curvature exclusively for pairs of nodes, rather than for hyperedges. Beyond ORC, Yadav, Samal, and Saucan [272] study FRC for undirected hypergraphs defined via poset representations, and Murgas, Saucan, and Sandhu [198] explore hypergraphs constructed from protein-protein interactions using a different notion of FRC based on the Hodge Laplacian. To the best of our knowledge, with Orchid, we are the first to introduce a flexible framework generalizing ORC to hypergraphs, and to demonstrate the utility of hypergraph ORC in practice.

Graph Curvature. Beyond the Ollivier-Ricci concepts, there are also curvature concepts based on the contractivity of operators [16], which could be considered a spiritual precursor to Ollivier's work. This contractivity-based view has been used to provide a predominantly *spectral perspective* on curvature [173, 197], whereas ORC can foremost be seen as a *probabilistic concept*. Recently, Kempton, Lippner, and Münch [142] defined a hybrid between Ollivier and Bakry-Émery curvature on graphs. A more combinatorial perspective is assumed by FRC, which is motivated by defining equivalent formulations of curvature on structured spaces, such as CW complexes or simplicial complexes. Originally described by Forman [93], FRC has since been improved in the context of explaining the learning behavior of graph neural networks [254], with other recent work focusing on fusing it with topological graph properties [231]. ORC was first developed for general Markov chains [208, 209], but has quickly been adopted to characterize graphs [137] and

networks [266]. With numerous follow-up publications elucidating the relationship between structural properties of a graph and ORC [24, 233], the initial concept has also been substantially updated [41, 171]. As an emerging research direction, we identified the combination of ORC (and FRC) with concepts from computational topology, leading to an inherent multi-scale perspective on data. This has led, *inter alia*, to promising results for treating biomedical graph data [267, 268].

Hypergraph Learning, Mining, and Analysis. Work tackling certain hypergraph learning tasks such as hypergraph clustering [7, 258] has existed for many years [262, 281]. Some approaches make use of intrinsic structural properties of hypergraphs, leading to hypergraph neural network architectures [126] and message passing formulations [96], whereas others focus on developing similarity measures, i.e., *kernels* [14, 34, 186]. Methods from the rich literature on *graph* kernels [39, 152] can also be employed to address hypergraph learning tasks, namely, by transforming the hypergraph into a graph. However, most popular transformations are lossy and may drastically increase the size of the object under study, such that the practicality and utility of this approach is unclear. Beyond hypergraph learning, in recent years, there has been a renewed interest in hypergraph mining and hypergraph analysis. Notably, there is work developing novel hypergraph descriptors [6], extending motif discovery to hypergraphs [162, 163], solving classic graph mining tasks in the hypergraph setting [183], or identifying patterns in real-world hypergraphs [78]. However, to the best of our knowledge, none of this work draws on curvature concepts to address the mining and analysis tasks we are interested in.

5.5 Experiments

Having established in Section 5.3 that Orchid curvatures have our desired theoretical properties, and finding that they strictly generalize both ORC on graphs and existing definitions of hypergraph ORC, we now seek to ascertain that they are also meaningful in practice. To this end, we ask the following questions:

Q1 Parametrization. How do our choices of α, μ, and Agg impact Orchid curvatures?

Q2 Hypergraph exploration. How can Orchid curvatures help us in exploring hypergraphs?

Q3 Hypergraph learning. How can Orchid curvatures support hypergraph learning tasks?

To address these questions, we experiment with data from different domains, spanning several orders of magnitude. We investigate four *individual real-world hypergraphs* in which edges represent co-authorship (aps-a, dblp) and FDA-registered

Table 5.1: Hypergraphs used in Orchid experiments cover several domains and orders of magnitude. n and m are node and edge counts, n/m is the aspect ratio, c is the number of filled cells in the node-to-edge incidence matrix, c/nm is the density, and N is the number of hypergraphs in a collection.

(a) Individual Hypergraphs

	Nodes	Edges	n	m	n/m	c	c/nm
aps-a	Authors	APS Papers	505 827	688 707	0.7345	2 480 373	0.000007
dblp	Authors	DBLP Papers	3 108 658	6 011 388	0.5171	19 411 479	0.000001
ndc-ai	Active Ingr.	NDC Drugs	7 090	131 450	0.0539	224 084	0.000240
ndc-pc	Pharm. Classes	NDC Drugs	1 263	70 101	0.0180	273 088	0.003084

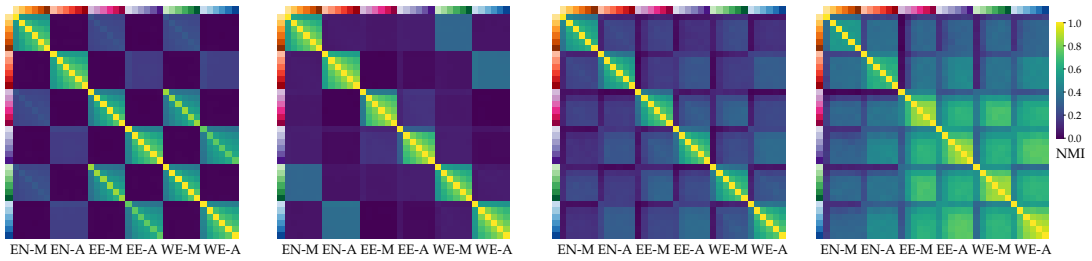
(b) Hypergraph Collections

	Nodes	Edges	Graphs	N	$(n/m)_{\max}$	$(c/nm)_{\max}$
aps-av	Authors	APS Papers	Journals	19	4.698182	0.005216
aps-cv	APS Cited P.	APS Citing P.	Journals	19	1.396552	0.028430
dblp-v	Authors	DBLP Papers	(Groups of) Venues	1 193	5.599424	0.002443
mus	Frequencies	Chords	Music Pieces	1 944	1.454545	0.375000
stex	Tags	Questions	StackExchange Sites	355	1.233449	0.121528
sha	Characters	Stage Groups	Shakespeare's Plays	37	0.554054	0.304688
syn-c	Hypergraph Configuration Models			250	0.5	0.005
syn-r	Erdős-Rényi Random Hypergraph Models			250	0.5	0.005
syn-s	Hypergaph Stochastic Block Models			250	0.5	0.005

drugs (ndc-ai, ndc-pc), six *collections of real-world hypergraphs* in which edges represent questions on Stack Exchange Sites (stex), co-authorship by venues (aps-av, dblp-v), co-citation by venues (aps-cv), chords in music pieces (mus), and character co-occurrence on stage in Shakespeare's plays (sha), as well as three *collections of synthetic hypergraphs* based on different generative models (syn-c, syn-r, syn-s), for a total of 4 321 hypergraphs. We summarize their basic properties in Table 5.1 and give more details on their statistics, semantics, and provenance in Section 5.C. We implement Orchid in Julia and Python. Our experiments are run on AMD EPYC 7702 CPUs with up to 256 cores. We discuss our implementation in more detail in Section 5.D, and make all our code, data, and results publicly available.[2]

Q1 Parametrization. To understand how our choices of α, μ, and `Agg` impact Orchid curvatures, we first compute the pairwise mutual information between Orchid edge curvatures with 36 different parametrizations. As illustrated in Fig. 5.2, while changing α for the same combination of μ and `Agg` has similar effects across hypergraphs, there is no uniform pattern in the relationships between different combinations of μ and `Agg`. This underscores the fact that the various

[2] 10.5281/zenodo.7624573



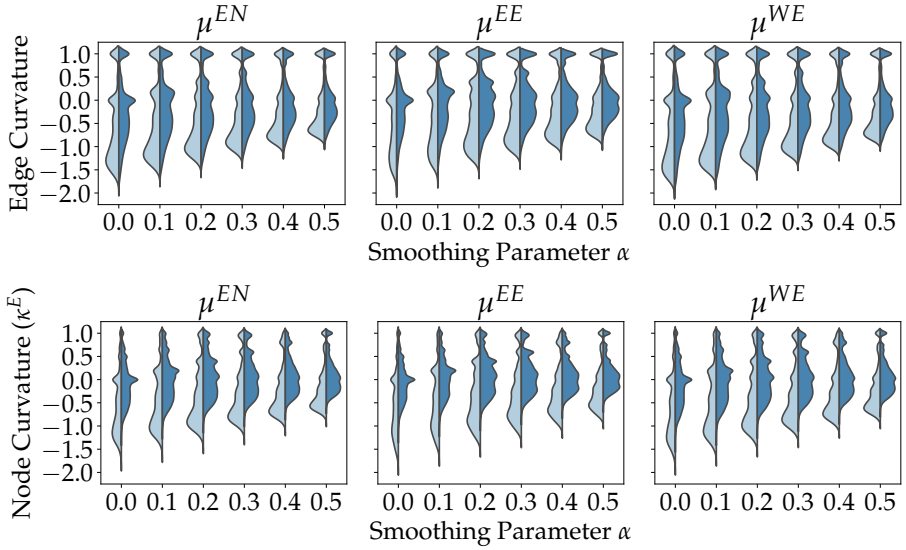
(a) (5,10)-regular syn-c **(b)** 2-community syn-s **(c)** ndc-ai **(d)** ndc-pc

Figure 5.2: ORCHID curvature notions are non-redundant. We show the Min-Max-Normalized Mutual Information (NMI) between ORCHID edge curvatures with 36 different parametrizations, using probability measures μ^{EN} (EN), μ^{EE} (EE), or μ^{WE} (WE), aggregations $\texttt{Agg}_M$ (M) or $\texttt{Agg}_A$ (A), and $\alpha \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ (ordered $\rightarrow, \downarrow$), for two synthetic and two real-world hypergraphs.

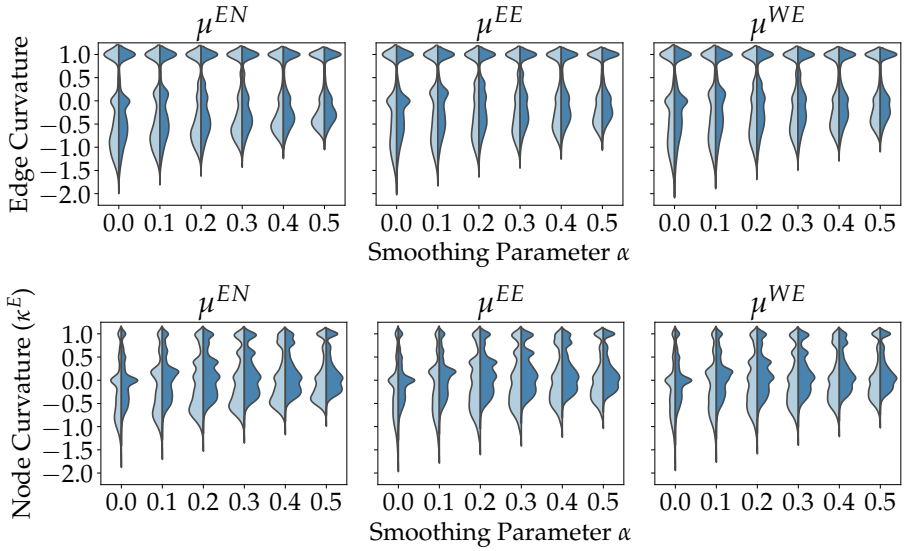
notions of ORCHID curvature are not redundant but rather emphasize distinct aspects of hypergraph structure. For a fine-grained view of the differences between parametrizations, we inspect the distributions of our four curvature types, (i) edge curvature $\kappa(e)$, (ii) edge-averaged node curvature $\kappa^E(i)$, (iii) directional curvature $\kappa(i,j)$ for all $\{i,j\} \subseteq e \in E$, and (iv) direction-averaged node curvature $\kappa^{\mathcal{N}}(i)$, for each of our 36 parametrizations.

By construction, directional curvature and direction-averaged node curvature do not vary with the choice of $\texttt{Agg}$, and κ_M lower-bounds κ_A for edge curvatures and edge-averaged node curvatures. However, the differences between κ_M and κ_A vary across graphs, while consistently, the larger α, the more concentrated our curvature distributions. To see this, Fig. 5.3 shows the distributions of edge curvatures and edge-averaged node curvatures for two hypergraphs from the dblp-v collection, representing top conferences in machine learning and theoretical computer science, respectively. The figure highlights the consistently concentrating effect of increasing α, and it elucidates the differential effects of moving from maximum aggregation (left parts of the split violins) to mean aggregation (right parts of the split violins), from almost no shifts to large shifts in probability mass (compare, e.g., Fig. 5.3b, top right panel, with Fig. 5.3b, bottom left panel).

Fig. 5.3 might convey the impression that, other parameters being equal, the distributions of curvatures based on μ^{EN} and μ^{WE} are more similar to each other than to μ^{EE}. This does not hold in general, however, as demonstrated for ndc-pc in Fig. 5.4a, where node curvature distributions based on μ^{WE} are more similar to those based on μ^{EE} than to the node curvature distributions based on μ^{EN}. Comparing Fig. 5.4a (ndc-pc) to Fig. 5.4b (ndc-ai), we further observe that rather similar distributions of edge curvature and directional curvature can be accompanied by rather different distributions of edge-averaged and direction-averaged node curvatures, even for hypergraphs originating from the same domain. Finally, when



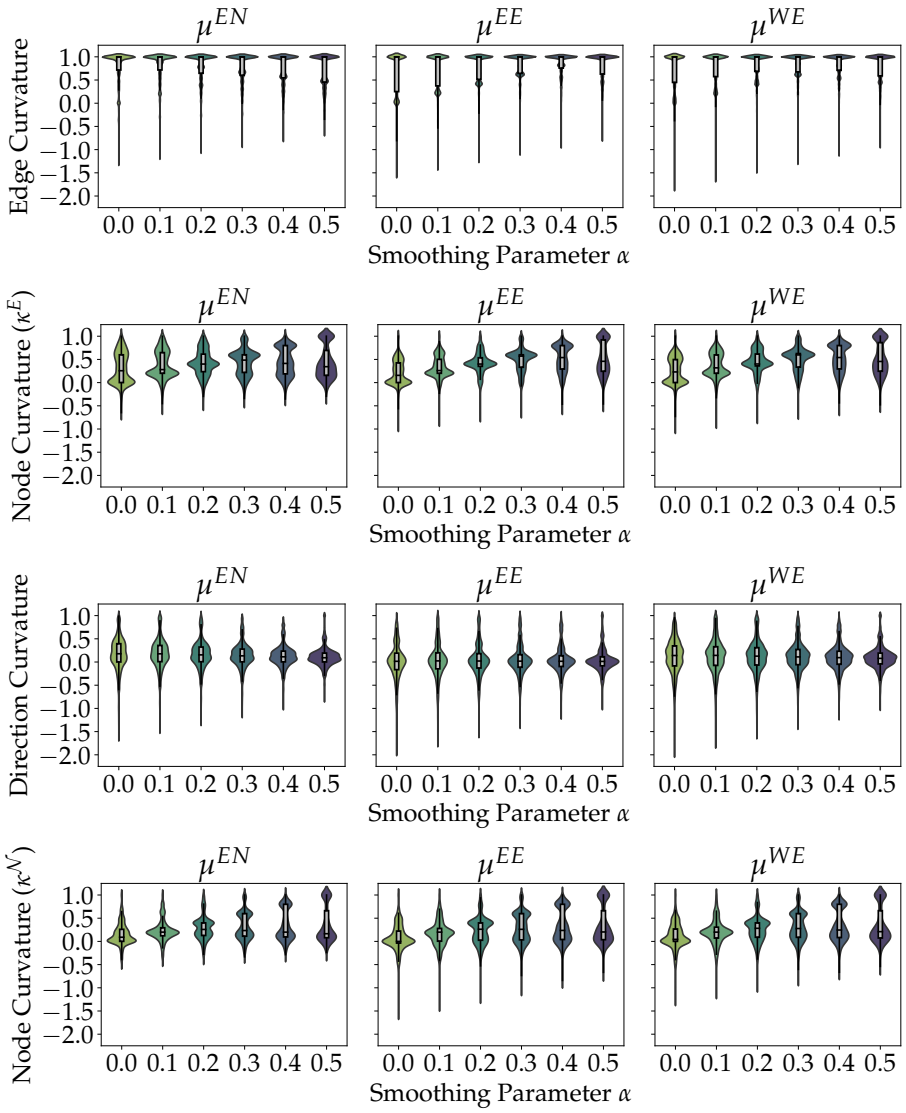
(a) Top Conferences in Machine Learning



(b) Top Conferences in Theoretical Computer Science

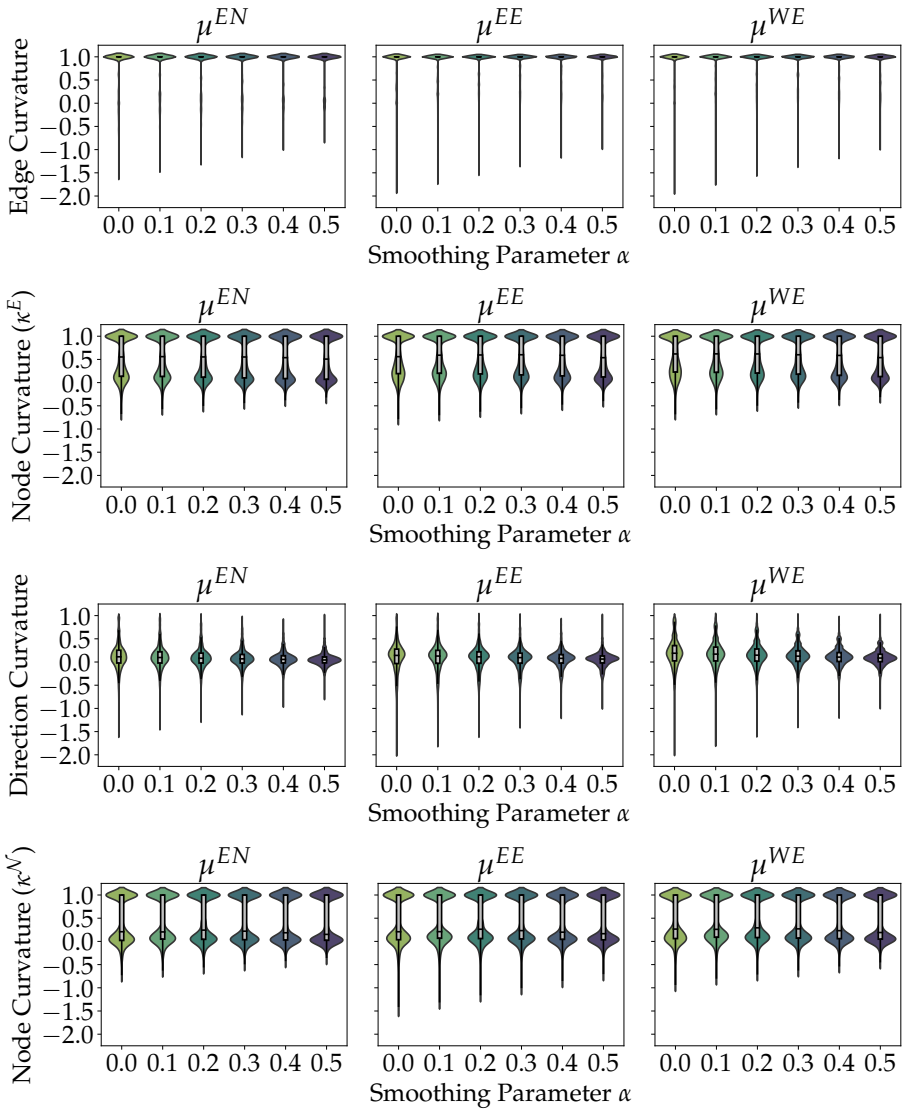
Figure 5.3: $\texttt{Agg}_{\texttt{M}}$ and $\texttt{Agg}_{\texttt{A}}$ capture different aspects of the underlying hypergraph data. We show distributions of Orchid edge curvatures (top) and edge-averaged node curvatures (bottom) using probability measures $\mu^{\texttt{EN}}$, $\mu^{\texttt{EE}}$, and $\mu^{\texttt{WE}}$ with smoothing α, for the aggregation functions $\texttt{Agg}_{\texttt{M}}$ (light blue) and $\texttt{Agg}_{\texttt{A}}$ (dark blue) on dblp-v hypergraphs representing top conferences in machine learning (Fig. 5.3a) and in theoretical computer science (Fig. 5.3b).

visualizing curvatures for hypergraphs in the same collection or across collections with related semantics, as illustrated in Fig. 5.5, we can identify several distinct prototypical shapes of curvature distributions and relationships between curvatures based on different probability measures.



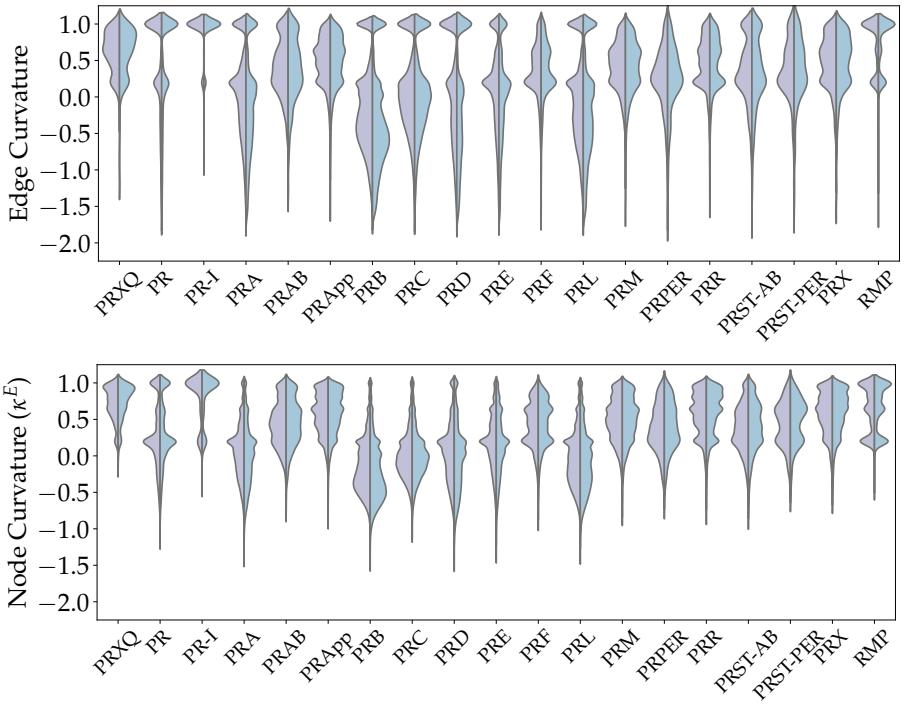
(a) ndc-pc

Figure 5.4: Hypergraphs with similar distributions of one curvature type may differ in their distributions of other curvature types. We show ORCHID curvatures computed using $\texttt{AGG}_\texttt{A}$, for all curvature types, probability measures, and $\alpha \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. (Figure continued on next page.)

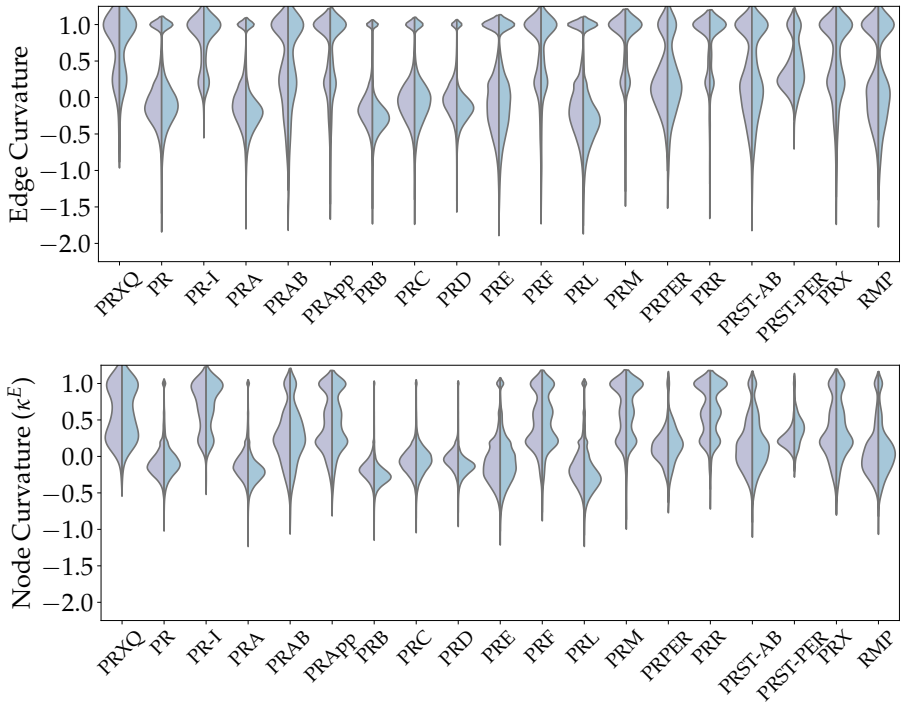


(b) ndc-ai

Figure 5.4: Hypergraphs with similar distributions of one curvature type may differ in their distributions of other curvature types. We show Orchid curvatures computed using $\texttt{Agg}_\texttt{A}$, for all curvature types, probability measures, and $\alpha \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. (Figure continued from previous page.)



(a) aps-av



(b) aps-cv

Figure 5.5: ORCHID curvature distributions within the same collection and across semantically related collections exhibit prototypical shapes, accompanied by varying types of relationships between probability measures. We show distributions of ORCHID edge curvatures (top) and edge-averaged node curvatures (bottom) computed using $\alpha = 0.1$ and $\texttt{Agg}_A$, for μ^{EE} (violet) and μ^{WE} (blue), for all hypergraphs in the aps-av and aps-cv collections. Recall that the edges in aps-av and aps-cv as well as the nodes in aps-cv represent essentially the same set of APS papers, but in aps-av, they connect co-authors, and in aps-cv, they connect co-cited papers (edges) or are connected by citing papers (nodes).

Q2 Hypergraph Exploration. To explore *individual graphs*, we perform case studies on graphs from the aps-cv collection, leveraging that most nodes in these graphs also occur as edges. We scrutinize the relationships between node and edge curvatures, other local node and edge statistics, and article metadata. We observe that curvature values span a considerable range even for articles with otherwise comparable statistics, but the curvature distributions of influential papers appear to differ systematically from those of less influential papers.

To illustrate this finding, we focus on a case study of the citation hypergraph of the journal Physical Review E (PRE), which regularly publishes, inter alia, interdisciplinary work on graphs and networks. In this hypergraph, which has 45 504 nodes and 52 574 edges, nodes represent PRE articles *cited* by at least one other PRE article, edges represent PRE articles *citing* at least one other PRE article, and each edge i comprises the nodes j cited by the paper corresponding to i. Therefore, the *edge* curvature of a (citing) paper i can be interpreted as an indicator of its *breadth of content*: The more *positive* the edge curvature, the stronger the general tendency of the papers jointly cited by paper i to be cited together, suggesting that these papers are topically related. Similarly, the *node* curvature of a (cited) paper j can be interpreted as an indicator of its *breadth of impact*: The more *negative* the node curvature, the more diversely the paper has been cited in the literature.

With these interpretations in mind, we compute all curvatures for the PRE citation hypergraph, using $\alpha = 0.1$, μ^{WE}, and $\mathtt{AGG}_{\mathtt{A}}$. We find that for all 54 articles with at least 100 citations (top articles), the edge-averaged node curvature is larger than the direction-averaged node curvature, which is always negative, although only 36% of all PRE articles exhibit this feature combination. This matches the intuition that from highly cited articles, the literature should diverge in many different directions. At the same time, we observe that curvatures span a considerable range, even among top articles. In Table 5.2, we record the top articles with extreme curvature values, and in Fig. 5.6, we display the pairwise relationships between curvature features and other local features for *all* PRE articles. In line with the interpretations sketched above, the top article with the largest node curvatures is a classic reference for community detection in the highly integrated field of network science, whereas the articles with the smallest node curvatures address topics relevant to a broader range of approaches to collective phenomena in many-body systems (which are the focus of PRE).

Table 5.2: Top articles display varying relationships between different curvature values. We list the PRE articles that, out of all PRE articles cited at least 100 times, exhibit the most extreme curvature-related values.

	$\kappa^E(i)$	$\kappa^N(i)$	$\Delta(\kappa(i))$	$\kappa(e)$	Title
$\max \kappa^E(i)$, $\max \kappa^N(i)$	0.220092	−0.006001	0.226093	0.425336	Finding community structure in very large networks (10.1103/PhysRevE.70.066111)
$\min \kappa^E(i)$	−0.319638	−0.555431	0.235793	0	Scale-invariant motion in intermittent chaotic systems (10.1103/PhysRevE.47.851)
$\min \kappa^N(i)$	−0.241216	−0.704752	0.463536	0	Extended self-similarity in turbulent flows (10.1103/PhysRevE.48.R29)
$\max \Delta(\kappa(i))$	−0.131542	−0.668266	0.536724	0.038477	Determining the density of states for classical statistical models: A random walk algorithm to produce a flat histogram (10.1103/PhysRevE.64.056101)
$\min \Delta(\kappa(i))$	−0.015495	−0.191193	0.175697	−0.156824	Amorphous systems in athermal, quasistatic shear (10.1103/PhysRevE.74.016118)
$\max \kappa(e)$	0.129557	−0.251635	0.381192	0.610123	Topological defects and interactions in nematic emulsions (10.1103/PhysRevE.57.610)
$\min \kappa(e)$	−0.191094	−0.552908	0.361815	−0.644446	Fast Monte Carlo algorithm for site or bond percolation (10.1103/PhysRevE.64.016706)

Exploring *graph collections*, we run kernel principal component analysis (kPCA) [235] with a radial basis function kernel (RBF kernel) and curvatures or other local features known to be powerful baselines [45], e.g., node degrees and neighborhood sizes, as inputs to jointly embed graphs from a collection. We statistically bootstrap the maximum mean discrepancy (MMD) [108] to test the null hypothesis that the feature distributions of two graphs are equal. As shown in Fig. 5.7, Orchid curvatures result in more interpretable embeddings and more discriminative tests than other local features.

Q3 Hypergraph Learning. To explore the utility of curvatures for *hypergraph learning*, we focus on learning with *hypergraph collections*. To this end, we spectrally cluster the collection using RBF or exponential Wasserstein kernel matrices, $\exp(-\gamma\, W(\mu_x, \mu_y))$, on node and edge curvatures or other local features [76]. Lacking ground-truth labels, we evaluate the clustering quality in an *unsupervised* manner, using what we call the *Wasserstein Clustering Coefficient* (WCC). This measure compares averaged *intra*-cluster Wasserstein distances to averaged *inter*-cluster Wasserstein distances, such that a *lower* WCC corresponds to a higher-quality clustering. Given c clusters $\mathcal{X} = \{X_1, \ldots, X_c\}$ of hypergraphs H represented by their

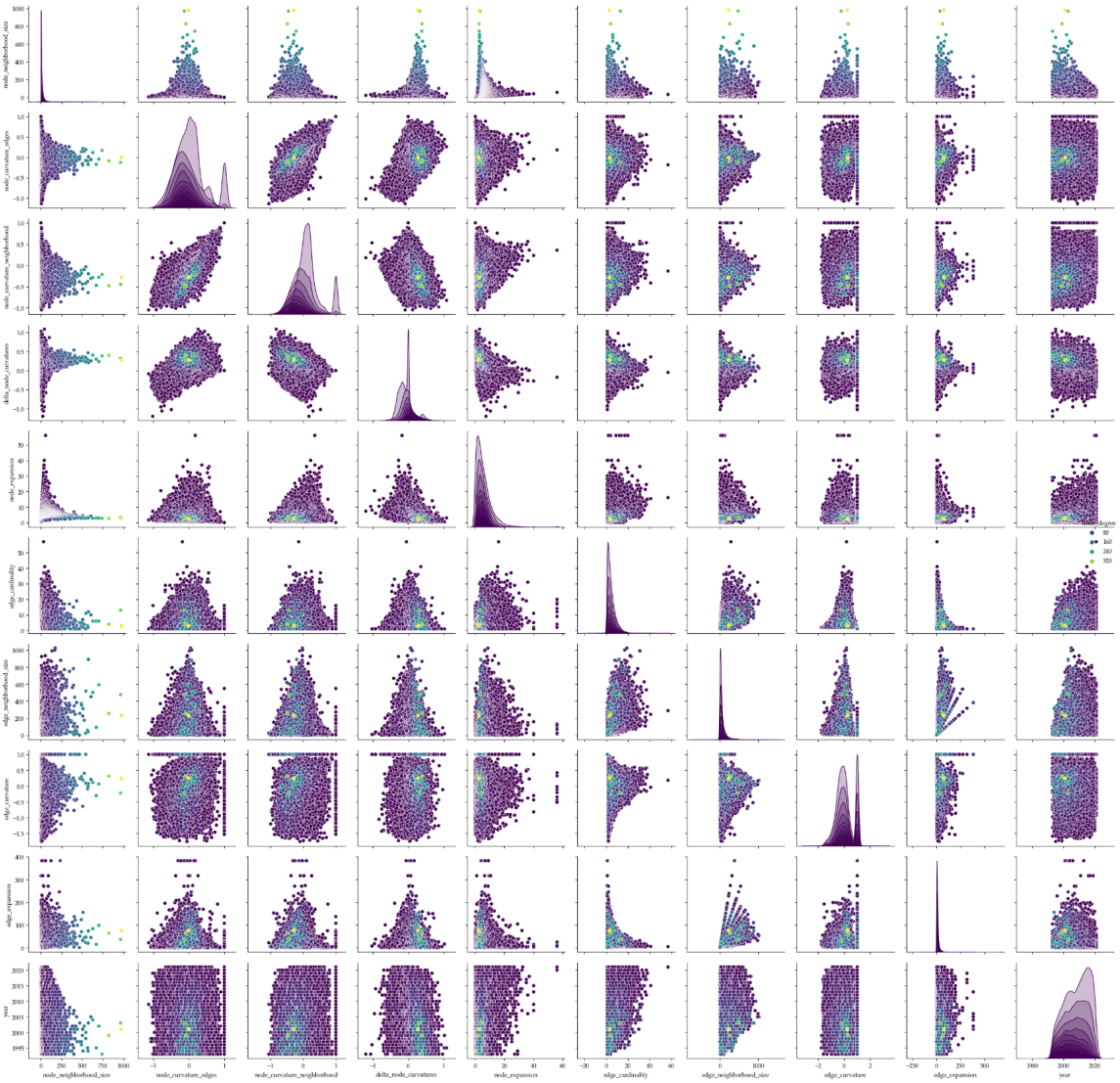


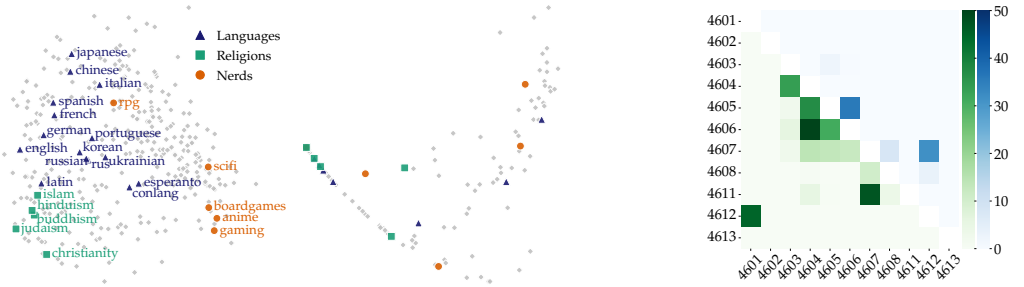
Figure 5.6: Highly cited articles have distinct curvature distributions. Pairwise relationships between (left-to-right, top-to-bottom) node neighborhood size, edge-averaged node curvature, direction-averaged node curvature, curvature delta, node expansion $= {}^{\deg(i)}/_{|\mathcal{N}(i)|}$, edge cardinality, edge neighborhood size, edge curvature, edge expansion $= {}^{\deg(e)}/_{|\mathcal{N}(e)|}$, and (as an additional metadata feature) publication year, for all PRE articles cited at least once by another PRE article, colored by node degree (number of citations within PRE), where brighter colors signal larger node degrees.

feature distributions $\vec{\chi}_H$, we define

$$\mathrm{WCC}(\mathcal{X}) = \frac{\sum_{X\in\mathcal{X}} \omega(X)}{1 + \sum_{X\neq Y\in\mathcal{X}} \omega(X,Y)} \,, \tag{5.25}$$

with

$$\omega(X) = \binom{|X|}{2}^{-1} \sum_{x\neq y\in X} W(\vec{\chi}_x, \vec{\chi}_y) \,, \tag{5.26}$$



(a) kPCA (directional curvature) **(b)** kPCA (edge neighborhood size) **(c)** MMD (cardinality vs. curvature)

Figure 5.7: Curvatures carry more information than other local features. We show a 2-dimensional embedding of graphs from the stex collection based on kPCA, using an RBF kernel with curvature distributions computed using $\alpha = 0.1$, μ^{WE}, and $\mathtt{AGG}_{\mathrm{A}}$ (5.7a) or edge neighborhood size distributions (5.7b) as input features. We see that only curvatures yield a meaningful and discriminative grouping. Corroborating this finding, we also depict Bonferroni-adjusted p-values of testing for significant differences in feature distributions—i.e., p-values multiplied by the number h of hypothesis tests, as Bonferroni [38] correction requires $p \leqslant \alpha/h$ for some desired Type I-error rate α—using MMD on distributions of edge curvatures computed with the same parameters as for (5.7a) (upper triangle) or edge cardinality (lower triangle), for the subset of the dblp-v collection corresponding to top conferences grouped by areas of research (5.7c).

Table 5.3: ORCHID curvatures lead to better clusterings than other local features. We show $\mathrm{WCC}_{\kappa(i,j)}$ for clusterings of hypergraph collections computed using RBF or exponential Wasserstein kernels with edge curvatures, edge neighborhood sizes, edge-averaged node curvatures, or node neighborhood sizes as inputs.

	$\mathrm{RBF}_{\kappa(e)}$	$\mathrm{W}_{\kappa(e)}$	$\mathrm{RBF}_{\lvert\mathcal{N}(e)\rvert}$	$\mathrm{W}_{\lvert\mathcal{N}(e)\rvert}$	$\mathrm{RBF}_{\kappa^{\mathrm{E}}(i)}$	$\mathrm{W}_{\kappa^{\mathrm{E}}(i)}$	$\mathrm{RBF}_{\lvert\mathcal{N}(i)\rvert}$	$\mathrm{W}_{\lvert\mathcal{N}(i)\rvert}$
dblp-v	0.2151	0.1908	0.3309	0.2358	0.2273	**0.0445**	0.0910	0.1285
mus	0.1955	0.1758	0.2609	0.2723	0.2062	**0.1606**	0.2774	0.2458
stex	0.2651	0.2877	0.3018	0.2950	**0.2393**	0.2577	0.3067	0.2689
sha	0.5984	0.6390	0.6716	0.6597	**0.5021**	0.6526	0.6236	0.6641

and

$$\omega(X,Y) = (|X||Y|)^{-1} \sum_{x,y \in X \times Y} \mathrm{W}(\vec{\chi}_x, \vec{\chi}_y)\,. \tag{5.27}$$

As illustrated in Table 5.3, when evaluated using WCC with directional curvature distributions as $\vec{\chi}$, i.e., $\mathrm{WCC}_{\kappa(i,j)}$, ORCHID curvatures consistently yield better clusterings than other local features.

5.6 CONCLUSION

We introduced ORCHID, the first unified framework for Ollivier-Ricci curvature on hypergraphs that integrates and generalizes existing approaches to hypergraph ORC. ORCHID disentangles the common building blocks of all notions of hypergraph ORC, yielding curvature notions that are provably aligned with our geometric intuition. We performed a rigorous theoretical and empirical analysis of

Orchid curvatures, demonstrating their practical utility and scalability through extensive experiments. Thus, we hope to have laid the groundwork for future work seeking to leverage the power of Ollivier-Ricci curvature for hypergraphs in hypergraph learning and hypergraph analysis.

Given that our work still has limitations, we see potential for future research in several directions. First, ORC on graphs is defined for *any* probability measure, but we only consider measures corresponding to a single step of a random walk. Follow-up work could thus harness higher-order random walks or alternative probability measures, and it could analyze the relationships between such probability measures and other structural hypergraph properties. Second, hyperedge intersections can vary in cardinality, but this variation is not currently reflected in our probability measures. This could be addressed by integrating Orchid with the s-walk framework proposed by Aksoy et al. [6]. Alternatively, one could define persistent Orchid curvatures based on hypergraph filtrations, extending work on persistent ORC for graphs [268]. Third, like the original ORC, Orchid curvatures are *static*, but many hypergraphs evolve over time, suggesting a need to develop *dynamic* curvature notions. Fourth, despite its comprehensive scope, our study only scratches the surface regarding the theoretical and empirical analysis of Orchid curvatures, and we believe that there are many more connections between Orchid curvatures and other hypergraph descriptors to be uncovered, and many additional use cases to be explored. For instance, Orchid generalizes ORC, but not Forman–Ricci curvature (FRC), and we believe that a framework for FRC could help uncover new relations between combinatorial curvature notions and hypergraph structure. Finally, we imagine that incorporating hypergraph curvature into models as an additional inductive bias could prove useful in hypergraph learning more broadly.

Appendices

5.A Ethics Statement

Our main contribution is Orchid, a unified mathematical framework yielding theoretically sound hypergraph descriptors that are also practically useful for hypergraph exploration and hypergraph learning. As such, Orchid comes with the caveats applicable to hypergraph exploration and hypergraph learning methods more generally. Most importantly, it should be used with caution on data related to people, and its results should not be decontextualized. We adhered to these principles in our experiments, and selected our datasets accordingly.

Table 5.4: Basic notation used in this chapter.

Symbol	Definition	Description
G	$= (V, E)$ with…	
	$\dots E \subseteq \binom{V}{2}$	Graph G with node set V and edge set E
	$\dots E = (e_1, \dots, e_m)$	Multi-Graph with node set V and edge set E
H	$= (V, E)$ with…	
	$\dots E \subseteq \mathcal{P}(V) \setminus \emptyset$	Hypergraph with node set V and edge set E
	$\dots$ with $E = (e_1, \dots, e_m)$	Multi-Hypergraph with node set V and edge set E
n	$= \lvert V \rvert$	Number of nodes in G or H
m	$= \lvert E \rvert$	Number of edges in G or H
$\binom{S}{k}$	$= \{X \subseteq S \mid \lvert X \rvert = k\}$	Set of all k-element subsets of S
$[x]$	$= \{i \in \mathbb{N} \mid i \leqslant x\}$	Set of positive integers not greater than x
$\deg(i)$	$= \lvert\{e \in E \mid i \in e\}\rvert$	Degree of node $i \in V$
$i \sim j$	$\Leftrightarrow \exists e \in E : \{i, j\} \subseteq e$	Node i is adjacent to node j
$\mathcal{N}(i)$	$= \{j \in V \mid i \sim j\}$	Neighborhood of node $i \in V$
$\mathcal{N}(e)$	$= \{f \in E \mid e \cap f \neq \emptyset\}$	Neighborhood of edge $e \in E$
$d(i, j)$		Distance between nodes i and j
$\operatorname{diam}(H)$	$= \max\{d(i, j) \mid i, j \in V\}$	Diameter of (hyper)graph H
μ_i		Probability measure associated with node i
α	$\in [0, 1)$	Smoothing parameter (laziness of random walk)
$W_1(\mu_i, \mu_j)$		Wasserstein distance between measures μ_i and μ_j
$d_{TV}(\mu_i, \mu_j)$		Total variation distance
δ_i		Dirac measure at node i
$J(i)$	$= W_1(\delta_i, \mu_i)$	Jump probability of μ_i

5.B Notation

For easy reference, we collect the notation used in this chapter in Table 5.4.

5.C Dataset Details

At a high level, our workflow to produce and work with the datasets used in our experiments (Section 5.5) was as follows:

1. Obtain raw data in a variety of different formats, e.g., CSV, JSON, or XML.
2. Transform the raw data into a hypergraph CSV that retains as much of the raw data semantics as possible. This CSV is guaranteed to contain one row per edge, one column with unique edge identifiers, and one column with the nodes contained in each edge. It may also contain additional columns holding further metadata associated with individual edges. Column names may differ between datasets to reflect dataset semantics.
3. Provide a unified loading interface to the datasets in Python.
4. Transform semantics-laden hypergraph CSV files into semantics-free one-based integer edge lists and sparse matrices for curvature computations in Julia, compute curvatures in Julia, and store the results in JSON files.

5. Map results back to original dataset semantics in Python for further examination.

In the following, we give more details on the provenance, semantics, and statistics of our datasets. Unless if otherwise noted, we make our datasets publicly available with our online materials, along with the raw data and all preprocessing code.[3]

aps-a, aps-av, aps-cv: American Physical Society Journal Articles

The American Physical Society (APS), a nonprofit organization working to advance the knowledge of physics, publishes several peer-reviewed research journals. The APS makes two datasets based on its publications available to researchers: (i) an edge list containing (citing, cited) pairs of articles contained in its collection, and (ii) a JSON dataset containing the metadata for each article in its collection. These datasets are updated on a yearly basis, and researchers can request access by filling out a web form located on the APS website. We made a data access request and were granted access to the 2021 versions of the APS datasets within two weeks.

From the APS datasets, we derived the following hypergraphs and hypergraph collections:

(i) aps-a: Each node corresponds to an author who published at least one article in an APS journal. Each edge e corresponds to an article in an APS journal, and it contains as nodes all authors of e. This hypergraph is derived from the JSON data.

(ii) aps-av: aps-a, split up by journal, for a total of 19 hypergraphs. For each journal j, the edge set of aps-a is restricted to articles from j, and the node set of aps-a is restricted to nodes authoring at least one article from j.

(iii) aps-cv: We derive one hypergraph for each of the 19 journals represented in the edge list data. For each journal j, the edge set comprises articles from j citing at least one article in j, and the node set consists of articles in j cited by at least one article in j.

Access. Due to the terms and conditions associated with data access, we cannot make the APS datasets or the hypergraphs derived from them publicly available, and researchers seeking to work with this data will have to request data access from APS directly as outlined above. However, we make our preprocessing code publicly available, such that researchers who have obtained access to the APS datasets can easily reproduce our hypergraphs from the raw data.

[3] 10.5281/zenodo.7624573

CAVEATS. When doing our case studies on the aps-cv dataset, we observed that some DOIs present in the edge list had no associated metadata in the JSON files provided by APS. This does not affect our curvature computations, but it might constrain the interpretability of results, e.g., when inspecting node clustering results based on article categories present only in the metadata.

DBLP, DBLP-V: DBLP JOURNAL ARTICLES AND CONFERENCE PROCEEDINGS

The DBLP computer science library provides high-quality bibliographic information on computer science publications. All DBLP data is released under a CC0 license and freely available in one XML file that is updated regularly. We obtained the XML dump dated September 1, 2022 from the DBLP website and preprocessed it into a CSV file containing only entries corresponding to the XML tags `article` and `inproceedings`, with one row per entry and the following columns:

- key: unique identifier of the entry, e.g., `conf/iclr/XuHLJ19` or `journals/cacm/Savage16c`.
- tag: XML tag associated with the entry, one of {`inproceedings`, `article`}.
- crossref: cross-reference to a venue, e.g., `conf/iclr/2019`. Sometimes missing although a venue should be present.
- author: semicolon-separated list of DBLP author names, e.g., `Keyulu Xu; Weihua Hu;Jure Leskovec;Stefanie Jegelka`. Sometimes missing (we discard entries without authors when loading the data).
- year: entry publication year, e.g., `2019`.
- title: entry title, e.g., `How Powerful are Graph Neural Networks?`.
- publtype: if present, the type of publication, e.g., `informal`. Mostly missing.
- journal: for article entries, the name of the publishing journal, e.g., `Commun. ACM`.
- booktitle: for inproceedings entries, the name of the publishing venue, e.g., `ICLR`.
- volume: if present, the publication volume, e.g., `59`.
- number: if present, the publication number, e.g., `7`.
- pages: if present, the entry pages, e.g., `12-14`.
- mdate: modification date, e.g., `2019-07-25`.

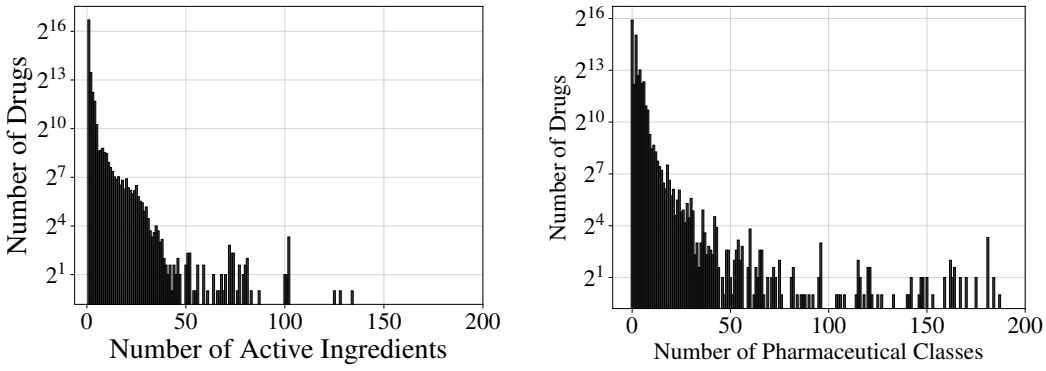
This constitutes our individual hypergraph dblp, in which each edge represents a paper, and each node represents an author. From this hypergraph, we additionally derived the dblp-v hypergraph collection, which contains different subsets of dblp by venue or group of venues. More precisely, we distinguish 1 193 hypergraphs as follows:

(i) `dblp_journal-all`, `dblp_inproceedings-all`: partition of dblp into entries published in journals and entries published as part of proceedings.

(ii) `dblp_journal-{journal}`: one hypergraph per journal, for all journals with at least 1 000 articles in the DBLP dataset.

(iii) `dblp_proceedings-{venue}`: one hypergraph per venue (grouped by `booktitle`), for all venues with at least 1 000 papers in the DBLP dataset.

(iv) `dblp_proceedings_area-{area}_{venues}`: one hypergraph per each of the FoR (field of research) areas 4601–4608, 4611–4613 as used in the CORE ranking (4609 and 4610 were not present in the ranking), where each area is represented by all conferences (grouped by `booktitle`) with CORE rank A* and A that have at least 1 000 papers in the DBLP dataset. These areas and associated top conferences are as follows:
   - 4601: Applied computing – AIED, ICCS
   - 4602: Artificial intelligence – AAAI, AAMAS, ACL, AISTATS, CADE, CIKM, COLING, COLT, CP, CogSci, EACL, EC, ECAI, EMNLP, GECCO, ICAPS, IJCAI, IROS, KR, UAI
   - 4603: Computer vision and multimedia computation – AAAI, CVPR, ECAI, ICCV, ICME, IJCAI, IROS, WACV
   - 4604: Cybersecurity and privacy – AsiaCCS, CCS, CRYPTO, DSN
   - 4605: Data management and data science – CIKM, ECIR, EDBT, ICDAR, ICDE, ICDM, ISWC, KDD, MSR, PODS, RecSys, SDM, SIGIR, VLDB, WSDM, WWW
   - 4606: Distributed computing and systems software – ASPLOS, CCGRID, CLUSTER, CONCUR, DISC, DSN, HPCA, HPDC, ICCAD, ICDCS, ICNP, ICPP, ICS, ICWS, INFOCOM, IPDPS, IPSN, PODC, SC, SIGCOMM, SPAA, WWW
   - 4607: Graphics, augmented reality and games – ISMAR, SIGGRAPH, VR, VRST
   - 4608: Human-centred computing – ASSETS, CHI, CSCW, ITiCSE, IUI, SIGCSE, UIST
   - 4611: Machine learning – AAAI, AISTATS, COLT, ECAI, ICDM, ICLR, ICML, IJCAI, KDD, NeurIPS, PPSN, WSDM
   - 4612: Software engineering – ASE, ASPLOS, CAV, ICSE, ICST, ISCA, ISSRE, MSR, OOPSLA, PLDI, POPL, RE, SIGMETRICS
   - 4613: Theory of computation – EC, ESA, FOCS, ICALP, ICLP, ISAAC, ISSAC, KR, LICS, MFCS, SODA, STACS, STOC, WG



(a) ndc-ai **(b)** ndc-pc

Figure 5.8: Edge cardinality distributions for hypergraphs derived from NDC data.

Caveats. For about 0.1% of all records, our XML parser failed, which originally resulted in "None" as one of the authors of all problematic records. We then redid the preprocessing (and all subsequent computations) *excluding* those records, but the records were still counted when determining the venues to include in dblp-v.

ndc-ai, ndc-pc: Drugs Approved by the U.S. Food and Drug Administration

The U.S. Food and Drug Administration (FDA) collects information on all drugs manufactured, prepared, propagated, compounded, or processed by registered drug establishments for commercial distribution in the United States. The FDA maintains the National Drug Code (NDC) Directory, which is updated daily and contains the listed NDC numbers and all information submitted as part of a drug listing. We downloaded the NDC data from the FDA website on August 21, 2022, and transformed it into a CSV file, an example record of which is shown in Table 5.5. From this CSV file, we derived two hypergraphs. In both hypergraphs, edges correspond to FDA-registered drugs. In ndc-ai, nodes correspond to the active ingredients used in these drugs, and in ndc-pc, nodes correspond to the pharmaceutical classes assigned to these drugs. The edge cardinality distributions resulting from both semantics are shown in Fig. 5.8.

mus: Music Pieces

`music21` is an open-source Python library for computer-aided musicology that comes with a corpus of public-domain music in symbolic notation. Using the `music21` library, we extracted a collection of hypergraphs from the `music21` corpus. In this collection, each hypergraph corresponds to a music piece, each edge corresponds to a chord sounding for a specific duration at a particular offset from the start of the piece, and each node corresponds to a sound frequency. Note that hypergraphs in the mus collection are node-aligned, which distinguishes them

Table 5.5: Example record from the data underlying the ndc-ai and ndc-pc hypergraphs.

Column Name	Record Value
product_ndc	71930-020
active_ingredients_names	[ACETAMINOPHEN, HYDROCODONE BITARTRATE]
active_ingredients_strengths	[325 mg/1, 7.5 mg/1]
pharm_class	[Opioid Agonist [EPC], Opioid Agonists [MoA]]
marketing_category	ANDA
dea_schedule	CII
finished	True
packaging	[{'package_ndc': '71930-020-12', 'description': '100 TABLET in 1 BOTTLE (71930-020-12)', 'marketing_start_date': '20180713', 'sample': False}, {'package_ndc': '71930-020-52', 'description': '500 TABLET in 1 BOTTLE (71930-020-52)', 'marketing_start_date': '20180713', 'sample': False}]
dosage_form	TABLET
product_type	HUMAN PRESCRIPTION DRUG
spl_id	58b53a57-388e-40d0-9985-048e5af09b0d
route	[ORAL]
product_id	71930-020_58b53a57-388e-40d0-9985-048e5af09b0d
application_number	ANDA210211
labeler_name	Eywa Pharma Inc
generic_name	Hydrocodone Bitartrate and Acetaminophen
brand_name	Hydrocodone Bitartrate and Acetaminophen
brand_name_base	Hydrocodone Bitartrate and Acetaminophen
brand_name_suffix	
listing_expiration_date	2022-12-31
marketing_start_date	2018-07-13
marketing_end_date	
openfda	{'manufacturer_name': ['Eywa Pharma Inc'], 'rxcui': ['856999', '857002', '857005'], 'spl_set_id': ['fcd2b59e-8087-475e-9e6b-911bd846ea96'], 'is_original_packager': [True], 'upc': ['0371930021121', '0371930020124', '0371930019128'], 'unii': ['NO70W886KK', '362O9ITL9D']}

from the hypergraphs in all other collections. In Table 5.6, we show the cardinality decomposition of selected music hypergraphs that include the largest edges. There, we include edges of cardinality 0 for completeness (they correspond to pauses in the music), but they are discarded in our curvature computations.

Caveats. When constructing our hypergraph collection from the `music21` corpus, we excluded pieces that are primarily monophonic. After exploring the corpus manually and evaluating the chord statistics of individual pieces, we decided to use only music with the following prefixes (corresponding to names of composers or collections): bach, beethoven, chopin, haydn, handel, monteverdi, mozart, palestrina, schumann, schubert, verdi, joplin, trecento, weber. Some pieces are included in several editions (e.g., BWV 190.7, the chorale by Johann Sebastian

Table 5.6: Selection of hypergraphs from the mus collection. n is the number of nodes, m is the number of edges, and the columns labeled i for $i \in \{0, 1, \ldots, 12\}$ record the number of edges of cardinality i in the hypergraph. Identifiers correspond to abbreviated `music21` identifiers and generally have the shape {composer}-{work identifier}-{suffix}, where *o* stands for *opus*, *m* stands for *movement*, and *inst* stands for *instrumental*.

	n	m	0	1	2	3	4	5	6	7	8	9	10	11	12
bach-bwv190.7-inst	38	233	1	0	0	4	25	60	56	72	9	6	0	0	0
bach-bwv190.7	38	233	1	0	0	4	25	60	56	72	9	6	0	0	0
bach-bwv248.23 − 2	35	155	1	0	0	12	45	90	0	3	1	2	1	0	0
bach-bwv248.42 − 4	38	386	3	1	11	42	147	106	54	14	7	1	0	0	0
beethoven-o133	88	5 140	236	565	828	1 515	1 758	168	42	21	5	2	0	0	0
beethoven-o18no1-m1	70	1 979	28	295	165	472	761	244	7	6	0	0	1	0	0
beethoven-o18no1-m4	77	2 669	13	338	438	678	1 032	134	33	1	1	1	0	0	0
beethoven-o18no4	81	4 730	95	465	674	977	1 940	521	50	3	3	1	1	0	0
beethoven-o59no1-m4	75	2 338	27	80	231	338	1467	168	18	4	4	0	1	0	0
beethoven-o59no2-m1	86	2 338	60	127	398	427	1 065	203	18	30	4	5	0	0	1
beethoven-o59no3-m4	81	3 292	19	381	529	734	1 219	255	139	14	1	1	0	0	0
beethoven-o74	82	6 492	112	440	922	1 448	2 886	538	119	21	5	1	0	0	0
monteverdi-madrigal.3.6	35	480	1	9	40	194	151	76	4	3	1	1	0	0	0
schumann-clara-o17-m3	63	819	5	12	133	208	151	108	83	74	25	13	5	2	0
schumann-o41no1-m5	72	2 410	51	130	208	592	919	366	117	18	2	4	0	2	1

Bach occupying the first two lines of Table 5.6, which is included in both the original and an instrumental version).

stex: StackExchange Sites

StackExchange is a platform hosting Q&A communities also known as sites. Each question is assigned at least one and at most five tags. In the second half of August 2022, we used the StackExchange API to download all questions asked on all StackExchange sites listed on the StackExchange data explorer, along with their associated tags and other metadata (including question titles and, for smaller sites, also question bodies). From our downloads, we created the stex hypergraph collection, in which each hypergraph corresponds to a StackExchange site, each edge corresponds to a question asked on a site, and each node corresponds to a tag used at least once on a site. Table 5.7 lists the basic statistics for each hypergraph from the stex collection.

Caveats. While our curvature computations uniformly include only questions asked no later than August 15, midnight GMT, the metadata associated with these questions stems from snapshots at different times in the second half of August 2022. We also excluded `stackoverflow.com` and `math.stackexchange.com` from our downloads because they could not be downloaded within one day due to API quota limitations, and `ru.stackoverflow.com` because it was large but we

Table 5.7: Basic statistics of hypergraphs derived from StackExchange sites (Part 1). n is the number of nodes, m is the number of edges, and columns labeled $i \in [5]$ count edges of cardinality i.

	n	m	n/m	1	2	3	4	5
3dprinting	416	4 902	0.084863	1 003	1 617	1 367	649	266
3dprinting.meta	45	197	0.228426	65	85	38	5	4
academia	457	39 270	0.011637	6 428	11 831	11 360	6 294	3 357
academia.meta	91	1 237	0.073565	396	486	249	95	11
ai	980	10 204	0.096041	767	1 805	2 696	2 427	2 509
ai.meta	49	315	0.155556	100	132	67	11	5
alcohol	154	1 138	0.135325	415	406	229	56	32
alcohol.meta	28	94	0.297872	28	42	14	8	2
android	1 517	56 403	0.026896	12 890	18 313	14 406	6 996	3 798
android.meta	103	996	0.103414	159	447	281	97	12
anime	1 528	12 122	0.126052	9 510	2 215	348	43	6
anime.meta	83	900	0.092222	234	384	215	56	11
apple	969	121 999	0.007943	15 822	34 777	37 243	22 652	11 505
apple.meta	108	1 452	0.074380	354	601	393	90	14
arduino	445	23 616	0.018843	5 838	7 357	6 027	2 858	1 536
arduino.meta	50	255	0.196078	101	110	34	10	0
askubuntu	3 137	393 266	0.007977	68 310	104 529	105 601	68 907	45 919
astronomy	566	12 773	0.044312	2 781	3 812	3 284	1 777	1 119
astronomy.meta	63	339	0.185841	115	93	76	43	12
aviation	1 024	22 701	0.045108	4 294	7 193	6 384	3 231	1 599
aviation.meta	73	752	0.097074	247	295	155	46	9
bicycles	548	18 873	0.029036	4 884	6 267	4 652	2 097	973
bicycles.meta	74	442	0.167421	150	197	76	15	4
bioacoustics	354	287	1.233449	20	50	101	54	62
bioacoustics.meta	36	49	0.734694	4	24	16	5	0
bioinformatics	490	4 998	0.098039	922	1 420	1 335	782	539
bioinformatics.meta	29	112	0.258929	44	53	15	0	0
biology	745	27 348	0.027241	5 487	8 618	7 093	3 742	2 408
biology.meta	88	814	0.108108	280	331	145	44	14
bitcoin	936	28 882	0.032408	6 677	8 927	7 432	3 766	2 080
bitcoin.meta	58	434	0.133641	142	202	71	16	3
blender	371	98 724	0.003758	31 012	30 861	22 200	9 614	5 037
blender.meta	69	716	0.096369	273	291	108	35	9
boardgames	1 000	13 166	0.075953	9 800	2 779	500	75	12
boardgames.meta	75	659	0.113809	197	289	144	27	2
bricks	202	4 220	0.047867	1 391	1 669	805	266	89
bricks.meta	52	211	0.246445	45	95	51	17	3
buddhism	487	7 956	0.061212	2 381	2 357	1 730	896	592
buddhism.meta	59	491	0.120163	104	252	94	30	11
cardano	285	2 248	0.126779	585	664	548	277	174
cardano.meta	24	43	0.558140	18	15	10	0	0
chemistry	370	41 571	0.008900	9 725	14 183	10 803	4 790	2 070
chemistry.meta	90	1 034	0.087041	250	441	243	88	12

would not have been able to interpret our results. For future work, we recommend using the StackExchange data dump hosted by the Internet Archive, which we only became aware of after completing our analyses, instead of the API.

Table 5.7: Basic statistics of hypergraphs derived from StackExchange sites (Part 2). n is the number of nodes, m is the number of edges, and columns labeled $i \in [5]$ count edges of cardinality i.

	n	m	n/m	1	2	3	4	5
chess	387	7 864	0.049212	1 646	2 682	2 069	985	482
chess.meta	62	368	0.168478	102	183	72	9	2
chinese	166	10 298	0.016120	4 467	3 438	1 628	543	222
chinese.meta	60	349	0.171920	93	170	67	12	7
christianity	1 129	14 955	0.075493	1 739	3 571	4 205	2 967	2 473
christianity.meta	110	1 579	0.069664	593	589	285	88	24
civicrm	507	14 324	0.035395	4 639	5 150	3 085	1 083	367
civicrm.meta	18	69	0.260870	43	18	6	2	0
codegolf	257	13 228	0.019428	1 360	4 586	4 379	2 106	797
codegolf.meta	128	2 276	0.056239	559	848	549	245	75
codereview	1 114	76 105	0.014638	6 306	20 542	23 777	16 106	9 374
codereview.meta	133	1 947	0.068310	190	615	688	345	109
coffee	114	1 381	0.082549	492	524	260	78	27
coffee.meta	27	90	0.300000	45	30	13	2	0
communitybuilding	74	559	0.132379	148	219	112	55	25
communitybuilding.meta	27	132	0.204545	36	67	24	4	1
computergraphics	259	3 600	0.071944	883	1 024	877	489	327
computergraphics.meta	34	150	0.226667	55	66	27	2	0
conlang	96	448	0.214286	109	204	91	32	12
conlang.meta	21	61	0.344262	16	34	7	4	0
cooking	834	25 877	0.032229	6 568	9 266	6 344	2 682	1 017
cooking.meta	83	866	0.095843	241	410	178	34	3
craftcms	523	13 756	0.038020	3 738	4 912	3 410	1 263	433
craftcms.meta	20	50	0.400000	22	11	15	1	1
crafts	193	2 039	0.094654	706	828	397	84	24
crafts.meta	49	184	0.266304	40	88	45	11	0
crypto	506	27 447	0.018436	6 448	9 056	6 960	3 283	1 700
crypto.meta	74	542	0.136531	139	237	127	27	12
cs	656	44 794	0.014645	8 624	14 332	12 644	6 336	2 858
cs.meta	86	603	0.142620	90	247	185	68	13
cseducators	210	1 080	0.194444	297	378	252	116	37
cseducators.meta	29	146	0.198630	52	68	26	0	0
cstheory	498	11 959	0.041642	1 653	3 384	3 495	2 052	1 375
cstheory.meta	80	608	0.131579	157	262	156	30	3
datascience	663	33 997	0.019502	4 110	8 028	9 305	6 753	5 801
datascience.meta	51	237	0.215190	80	97	38	16	6
dba	1 197	96 887	0.012355	15 956	29 750	27 361	15 610	7 682
dba.meta	76	800	0.095000	280	334	140	38	8
devops	431	5 025	0.085771	1 070	1 647	1 340	616	352
devops.meta	40	144	0.277778	45	63	31	5	0
diy	919	71 007	0.012942	19 347	22 079	17 371	8 399	3 811
diy.meta	68	603	0.112769	227	233	118	21	4
drones	220	731	0.300958	114	240	193	115	69
drones.meta	28	62	0.451613	11	31	17	3	0
drupal	149	86 283	0.001727	25 218	37 599	18 867	4 075	524
drupal.meta	75	1 014	0.073964	361	432	186	35	0
dsp	509	24 850	0.020483	4 460	6 779	6 565	4 081	2 965
dsp.meta	48	307	0.156352	153	108	30	14	2
earthscience	424	6 329	0.066993	1 111	1 778	1 698	1 094	648
earthscience.meta	54	321	0.168224	100	145	63	12	1
ebooks	180	1 466	0.122783	364	489	339	163	111
ebooks.meta	39	99	0.393939	31	37	23	6	2

Table 5.7: Basic statistics of hypergraphs derived from StackExchange sites (Part 3). n is the number of nodes, m is the number of edges, and columns labeled $i \in [5]$ count edges of cardinality i.

	n	m	n/m	1	2	3	4	5
economics	494	13 690	0.036085	3 488	4 426	3 160	1 678	938
economics.meta	60	444	0.135135	241	151	40	7	5
electronics	2 318	175 731	0.013191	31 201	46 423	46 974	29 107	22 026
electronics.meta	107	1 685	0.063501	698	628	282	62	15
elementaryos	314	8 471	0.037068	3 043	2 910	1 669	619	230
elementaryos.meta	29	107	0.271028	60	28	17	2	0
ell	533	99 970	0.005332	46 764	31 310	14 644	5 147	2 105
ell.meta	93	1 224	0.075980	448	489	226	52	9
emacs	891	23 939	0.037220	7 561	9 371	4 980	1 590	437
emacs.meta	51	216	0.236111	34	112	59	10	1
engineering	468	13 867	0.033749	3 582	4 121	3 315	1 770	1 079
engineering.meta	47	217	0.216590	71	87	45	10	4
english	984	125 848	0.007819	48 232	38 850	23 112	10 111	5 543
english.meta	182	3 589	0.050711	1 224	1 305	733	249	78
eosio	241	2 422	0.099505	766	766	533	245	112
eosio.meta	19	27	0.703704	6	14	4	2	1
es.meta.stackoverflow	168	1 817	0.092460	310	665	568	230	44
es.stackoverflow	2 960	179 452	0.016495	38 027	58 218	47 343	23 415	12 449
esperanto	99	1 592	0.062186	1 050	422	96	16	8
esperanto.meta	20	84	0.238095	37	38	9	0	0
ethereum	891	46 678	0.019088	8 449	12 402	12 327	7 687	5 813
ethereum.meta	63	259	0.243243	98	71	59	26	5
expatriates	304	7 182	0.042328	1 068	2 178	2 163	1 156	617
expatriates.meta	48	157	0.305732	41	72	41	2	1
expressionengine	603	12 447	0.048445	3 724	4 239	2 901	1 150	433
expressionengine.meta	35	123	0.284553	59	49	15	0	0
fitness	402	9 667	0.041585	2 123	2 864	2 427	1 289	964
fitness.meta	54	315	0.171429	126	123	57	7	2
freelancing	125	1 946	0.064234	632	654	394	177	89
freelancing.meta	33	132	0.250000	36	64	25	5	2
french	324	12 413	0.026102	3 368	4 126	2 923	1 390	606
french.meta	73	290	0.251724	58	127	80	24	1
gamedev	1 096	54 182	0.020228	7 381	16 130	15 996	9 433	5 242
gamedev.meta	78	910	0.085714	300	430	148	27	5
gaming	5 883	98 355	0.059814	72 655	20 708	4 120	758	114
gaming.meta	177	4 062	0.043575	478	1 853	1 219	425	87
gardening	526	16 629	0.031631	3 725	5 390	4 122	2 097	1 295
gardening.meta	60	320	0.187500	95	157	49	17	2
genealogy	465	3 572	0.130179	421	742	1 037	902	470
genealogy.meta	56	485	0.115464	133	273	70	8	1
german	265	16 022	0.016540	6 003	5 915	2 914	927	263
german.meta	69	540	0.127778	177	224	107	30	2
gis	2 829	150 205	0.018834	13 868	36 527	45 339	32 527	21 944
gis.meta	91	1 016	0.089567	174	361	317	125	39
graphicdesign	612	34 820	0.017576	7 542	10 789	9 364	4 821	2 304
graphicdesign.meta	83	851	0.097532	253	338	187	58	15
ham	334	4 299	0.077692	927	1 287	1 199	610	276
ham.meta	45	156	0.288462	39	65	32	18	2
hardwarerecs	246	3 945	0.062357	1 201	1 366	823	378	177
hardwarerecs.meta	42	255	0.164706	81	100	58	16	0
hermeneutics	422	12 563	0.033591	2 819	3 720	3 074	1 772	1 178
hermeneutics.meta	63	581	0.108434	256	212	84	22	7

Table 5.7: Basic statistics of hypergraphs derived from StackExchange sites (Part 4). n is the number of nodes, m is the number of edges, and columns labeled $i \in [5]$ count edges of cardinality i.

	n	m	n/m	1	2	3	4	5
hinduism	825	15 771	0.052311	2 597	4 337	3 976	2 876	1 985
hinduism.meta	89	827	0.107618	196	295	200	98	38
history	843	13 784	0.061158	2 071	3 757	3 839	2 436	1 681
history.meta	68	746	0.091153	340	265	107	31	3
homebrew	415	6 113	0.067888	1 393	1 976	1 593	803	348
homebrew.meta	50	172	0.290698	67	63	35	4	3
hsm	252	3 898	0.064649	982	1 272	928	464	252
hsm.meta	32	146	0.219178	61	44	37	4	0
interpersonal	280	3 890	0.071979	342	1 030	1 307	790	421
interpersonal.meta	76	825	0.092121	214	328	205	62	16
iot	241	2 103	0.114598	560	754	504	193	92
iot.meta	36	136	0.264706	30	74	27	5	0
iota	148	1 023	0.144673	300	352	248	84	39
iota.meta	18	38	0.473684	10	20	8	0	0
islam	562	13 792	0.040748	3 018	4 990	3 557	1 519	708
islam.meta	103	864	0.119213	240	358	206	47	13
italian	94	3 590	0.026184	1 296	1 376	636	206	76
italian.meta	27	151	0.178808	77	57	14	2	1
ja.meta.stackoverflow	74	1 115	0.066368	193	386	306	204	26
ja.stackoverflow	1 145	28 785	0.039778	10 077	10 518	5 624	1 946	620
japanese	354	26 365	0.013427	9 325	8 869	5 191	2 020	960
japanese.meta	75	817	0.091799	270	351	147	43	6
joomla	374	7 190	0.052017	1 289	2 221	2 058	1 072	550
joomla.meta	41	150	0.273333	81	46	19	4	0
judaism	1 264	36 511	0.034620	3 753	8 116	10 854	8 042	5 746
judaism.meta	147	1 455	0.101031	108	576	489	222	60
korean	118	1 716	0.068765	767	596	264	69	20
korean.meta	30	80	0.375000	38	28	8	5	1
languagelearning	216	1 287	0.167832	225	466	354	176	66
languagelearning.meta	52	195	0.266667	31	103	48	12	1
latin	370	5 400	0.068519	1 223	1 603	1 371	797	406
latin.meta	46	192	0.239583	34	80	49	25	4
law	938	23 649	0.039663	4 483	7 573	6 329	3 381	1 883
law.meta	66	499	0.132265	117	216	120	36	10
lifehacks	140	2 928	0.047814	1 024	1 052	595	190	67
lifehacks.meta	59	268	0.220149	65	122	72	6	3
linguistics	605	10 003	0.060482	1 947	2 836	2 556	1 627	1 037
linguistics.meta	59	363	0.162534	118	159	58	23	5
literature	2 335	5 614	0.415924	703	1 621	2 249	830	211
literature.meta	63	462	0.136364	56	292	99	15	0
magento	1 811	110 316	0.016416	15 598	28 805	32 671	20 873	12 369
magento.meta	66	575	0.114783	251	227	78	17	2
martialarts	205	2 199	0.093224	461	696	529	326	187
martialarts.meta	40	218	0.183486	66	97	46	9	0
math.meta	232	9 169	0.025303	1 051	3 485	2 919	1 312	402
matheducators	225	3 360	0.066964	696	1 118	903	435	208
matheducators.meta	57	255	0.223529	64	119	61	8	3
mathematica	705	85 069	0.008287	25 896	31 653	18 182	6 542	2 796
mathematica.meta	75	914	0.082057	416	341	130	25	2
mathoverflow.net	1 530	137 735	0.011108	20 381	37 763	38 643	24 597	16 351
mattermodeling	449	2 422	0.185384	169	547	668	495	543
mattermodeling.meta	61	142	0.429577	25	41	29	37	10

Table 5.7: Basic statistics of hypergraphs derived from StackExchange sites (Part 5). n is the number of nodes, m is the number of edges, and columns labeled $i \in [5]$ count edges of cardinality i.

	n	m	n/m	1	2	3	4	5
mechanics	1 430	25 243	0.056649	4 196	6 245	7 592	4 673	2 537
mechanics.meta	52	387	0.134367	124	182	66	13	2
medicalsciences	1 435	7 586	0.189164	1 423	1 970	1 754	1 261	1 178
medicalsciences.meta	65	501	0.129741	171	191	102	27	10
meta.askubuntu	196	5 698	0.034398	1 625	2 308	1 257	397	111
meta	1 250	97 114	0.012871	4 599	25 289	34 007	23 233	9 986
meta.mathoverflow.net	133	1 687	0.078838	272	601	504	229	81
meta.serverfault	139	2 173	0.063967	767	799	463	119	25
meta.stackoverflow	622	47 387	0.013126	5 297	15 301	15 792	8 233	2 764
meta.superuser	207	5 000	0.041400	1 010	1 914	1 474	510	92
monero	400	4 285	0.093349	1 193	1 424	969	481	218
monero.meta	23	85	0.270588	40	26	19	0	0
money	1 002	36 187	0.027690	3 788	8 036	10 340	8 450	5 573
money.meta	67	672	0.099702	220	260	147	40	5
movies	4 537	21 829	0.207843	4 857	11 430	4 546	877	119
movies.meta	75	1 285	0.058366	302	519	391	63	10
music	516	23 424	0.022029	4 754	7 644	6 370	3 117	1 539
music.meta	81	992	0.081653	391	387	166	40	8
musicfans	237	2 990	0.079264	1 209	1 169	465	111	36
musicfans.meta	42	218	0.192661	62	95	38	18	5
mythology	303	1 953	0.155146	484	723	439	215	92
mythology.meta	35	162	0.216049	43	87	31	1	0
networkengineering	453	15 624	0.028994	2 988	4 240	3 835	2 496	2 065
networkengineering.meta	53	375	0.141333	192	115	48	17	3
opendata	302	5 990	0.050417	1 562	2 002	1 492	670	264
opendata.meta	26	180	0.144444	73	76	30	1	0
opensource	203	4 226	0.048036	845	1 442	1 094	528	317
opensource.meta	53	225	0.235556	35	109	61	19	1
or	255	2 865	0.089005	351	809	848	496	361
or.meta	44	114	0.385965	21	61	23	5	4
outdoors	555	5 908	0.093940	934	2 017	1 791	806	360
outdoors.meta	52	512	0.101562	169	276	60	7	0
parenting	304	6 636	0.045811	1 182	2 175	1 873	1 004	402
parenting.meta	61	473	0.128964	96	217	125	31	4
patents	2 102	4 381	0.479799	1 421	1 211	879	481	389
patents.meta	46	167	0.275449	55	69	34	8	1
pets	289	7 874	0.036703	781	2 706	2 350	1 305	732
pets.meta	62	407	0.152334	60	194	112	26	15
philosophy	606	17 915	0.033826	4 898	5 399	4 079	2 089	1 450
philosophy.meta	61	793	0.076923	355	258	127	38	15
photo	1 156	25 961	0.044528	3 395	6 960	7 848	4 936	2 822
photo.meta	107	1 095	0.097717	289	500	239	60	7
physics	892	209 515	0.004257	21 914	42 808	53 150	45 705	45 938
physics.meta	114	3 228	0.035316	713	1 085	872	403	155
pm	283	6 198	0.045660	1 379	1 850	1 592	870	507
pm.meta	64	315	0.203175	81	129	73	27	5
poker	131	2 051	0.063871	763	659	372	181	76
poker.meta	29	122	0.237705	74	30	15	3	0
politics	793	14 628	0.054211	1 294	4 022	4 663	3 062	1 587
politics.meta	80	1 067	0.074977	249	436	259	103	20
portuguese	169	2 349	0.071946	703	898	509	174	65
portuguese.meta	35	137	0.255474	45	61	25	5	1

Table 5.7: Basic statistics of hypergraphs derived from StackExchange sites (Part 6). n is the number of nodes, m is the number of edges, and columns labeled $i \in [5]$ count edges of cardinality i.

	n	m	n/m	1	2	3	4	5
proofassistants	223	434	0.513825	80	175	116	42	21
proofassistants.meta	37	64	0.578125	11	26	18	7	2
psychology	401	7 641	0.052480	1 632	2 229	1 971	1 115	694
psychology.meta	62	557	0.111311	199	237	90	25	6
pt.meta.stackoverflow	140	2 986	0.046885	703	1 081	775	362	65
pt.stackoverflow	2 936	152 483	0.019255	28 143	50 055	42 386	21 287	10 612
puzzling	209	24 985	0.008365	6 912	9 471	5 731	2 020	851
puzzling.meta	98	1 365	0.071795	351	582	309	97	26
quant	693	20 283	0.034167	3 329	5 345	5 392	3 556	2 661
quant.meta	47	252	0.186508	95	115	37	3	2
quantumcomputing	306	7 823	0.039115	1 124	2 585	2 475	1 105	534
quantumcomputing.meta	50	187	0.267380	50	73	43	18	3
raspberrypi	598	35 872	0.016670	7 901	11 252	9 351	4 765	2 603
raspberrypi.meta	61	451	0.135255	213	169	58	8	3
retrocomputing	546	4 976	0.109727	925	1 694	1 366	692	299
retrocomputing.meta	70	304	0.230263	30	188	56	27	3
reverseengineering	347	8 754	0.039639	1 878	2 693	2 172	1 249	762
reverseengineering.meta	37	150	0.246667	56	62	28	3	1
robotics	276	6 261	0.044082	1 528	1 850	1 519	806	558
robotics.meta	39	159	0.245283	52	71	28	8	0
rpg	1 247	46 635	0.026740	4 236	12 463	15 431	9 542	4 963
rpg.meta	150	2 627	0.057099	310	986	844	379	108
ru.meta.stackoverflow	242	4 613	0.052460	445	1 312	1 574	979	303
rus	390	20 999	0.018572	12 276	5 131	2 341	840	411
rus.meta	30	214	0.140187	92	81	37	4	0
russian	166	4 516	0.036758	2 407	1 337	552	180	40
russian.meta	37	176	0.210227	80	61	25	7	3
salesforce	2 085	124 492	0.016748	22 537	37 977	33 635	19 220	11 123
salesforce.meta	79	795	0.099371	412	246	118	18	1
scicomp	346	10 381	0.033330	1 905	3 156	2 883	1 566	871
scicomp.meta	48	215	0.223256	75	90	42	8	0
scifi	3 693	69 344	0.053256	17 338	26 498	17 146	6 584	1 778
scifi.meta	149	3 265	0.045636	506	1 560	889	266	44
security	1 253	65 817	0.019038	11 950	19 799	18 266	9 809	5 993
security.meta	101	1 124	0.089858	311	507	242	52	12
serverfault	3 864	314 342	0.012292	40 967	83 417	92 763	60 560	36 635
sharepoint	1 722	99 911	0.017235	16 092	27 312	28 073	17 305	11 129
sharepoint.meta	78	581	0.134251	206	233	127	14	1
sitecore	362	11 395	0.031768	5 106	4 265	1 611	342	71
sitecore.meta	24	202	0.118812	40	60	99	3	0
skeptics	682	10 700	0.063738	2 227	4 165	2 952	1 042	314
skeptics.meta	100	1 529	0.065402	528	605	310	77	9
softwareengineering	1 674	61 392	0.027267	8 950	17 773	17 580	10 572	6 517
softwareengineering.meta	165	2 611	0.063194	421	1 023	776	310	81
softwarerecs	962	21 792	0.044145	3 090	6 533	6 199	3 723	2 247
softwarerecs.meta	85	654	0.129969	86	297	189	66	16
sound	1 224	9 786	0.125077	2 122	2 717	2 330	1 624	993
sound.meta	42	160	0.262500	65	66	25	1	3
space	1 203	17 392	0.069170	1 672	4 012	4 924	3 712	3 072
space.meta	74	682	0.108504	205	237	150	63	27
spanish	274	8 592	0.031890	2 276	2 722	2 140	1 010	444
spanish.meta	84	498	0.168675	94	216	135	42	11

Table 5.7: Basic statistics of hypergraphs derived from StackExchange sites (Part 7). n is the number of nodes, m is the number of edges, and columns labeled $i \in [5]$ count edges of cardinality i.

	n	m	n/m	1	2	3	4	5
sports	261	5 730	0.045550	926	2 371	1 637	609	187
sports.meta	57	350	0.162857	76	170	82	21	1
sqa	462	11 242	0.041096	2 263	3 250	2 881	1 705	1 143
sqa.meta	41	211	0.194313	115	71	17	7	1
stackapps	210	2 756	0.076197	277	858	883	514	224
stats	1 572	196 835	0.007986	19 622	47 967	57 502	41 443	30 301
stats.meta	132	1 685	0.078338	327	576	491	198	93
stellar	115	1 493	0.077026	585	438	298	109	63
stellar.meta	19	31	0.612903	9	14	8	0	0
substrate	512	1 814	0.282249	366	563	491	260	134
substrate.meta	40	44	0.909091	6	21	13	2	2
superuser	5 676	480 854	0.011804	64 273	127 561	135 549	91 137	62 334
sustainability	234	2 012	0.116302	431	713	536	235	97
sustainability.meta	37	151	0.245033	38	75	32	6	0
tex	2 035	237 763	0.008559	60 247	84 998	59 476	23 747	9 295
tex.meta	163	2 277	0.071585	389	921	671	235	61
tezos	210	1 828	0.114880	567	605	380	180	96
tezos.meta	18	32	0.562500	7	15	8	1	1
tor	218	5 636	0.038680	1 888	1 817	1 147	464	320
tor.meta	43	163	0.263804	57	76	25	4	1
travel	1 916	45 040	0.042540	2 985	8 914	13 809	11 528	7 804
travel.meta	99	1 379	0.071791	293	567	406	98	15
tridion	274	7 234	0.037877	1 471	2 758	1 915	818	272
tridion.meta	14	138	0.101449	93	39	6	0	0
ukrainian	124	2 094	0.059217	664	873	404	127	26
ukrainian.meta	33	104	0.317308	21	45	31	6	1
unix	2 777	220 644	0.012586	29 059	61 964	66 657	40 340	22 624
unix.meta	118	1 668	0.070743	367	727	407	144	23
ux	1 032	31 459	0.032805	4 660	8 934	8 823	5 530	3 512
ux.meta	94	899	0.104561	273	358	199	54	15
vegetarianism	115	677	0.169867	85	233	205	106	48
vegetarianism.meta	41	133	0.308271	26	62	32	13	0
vi	421	12 558	0.033524	4 494	4 802	2 358	694	210
vi.meta	35	201	0.174129	63	105	30	3	0
video	327	8 661	0.037755	2 705	2 693	1 831	882	550
video.meta	41	200	0.205000	63	96	32	8	1
webapps	951	33 202	0.028643	14 343	11 667	5 160	1 435	597
webapps.meta	106	937	0.113127	97	447	311	76	6
webmasters	1 078	36 840	0.029262	5 772	10 197	10 531	6 286	4 054
webmasters.meta	70	649	0.107858	202	258	135	45	9
windowsphone	287	3 440	0.083430	975	1 257	801	306	101
windowsphone.meta	44	148	0.297297	47	64	27	8	2
woodworking	244	3 739	0.065258	1 129	1 270	880	347	113
woodworking.meta	34	142	0.239437	69	46	25	2	0
wordpress	702	112 778	0.006225	27 669	37 039	28 491	13 228	6 351
wordpress.meta	82	866	0.094688	381	330	118	30	7
workplace	498	30 369	0.016398	6 371	9 325	8 103	4 221	2 349
workplace.meta	113	1 829	0.061782	506	699	447	150	27
worldbuilding	675	34 358	0.019646	2 958	8 284	10 839	7 267	5 010
worldbuilding.meta	120	2 032	0.059055	445	901	511	147	28
writing	391	11 699	0.033422	2 456	3 869	3 055	1 557	762
writing.meta	88	789	0.111534	145	415	173	49	7

SHA: SHAKESPEARE'S PLAYS

The sha collection is a subset of the HYPERBARD dataset recently introduced by Coupette, Vreeken, and Rieck [64], based on the TEI-encoded XML files of William Shakespeare's plays provided by Folger Digital Texts. Here, each hypergraph represents one of Shakespeare's plays, which are categorized into three types: comedy, history, and tragedy. In each hypergraph representing a play, nodes correspond to named characters in the play, and edges correspond to groups of characters simultaneously present on stage. These hypergraphs are documented extensively in the paper introducing the HYPERBARD dataset [64].

SYN-C, SYN-R, SYN-S: SYNTHETIC HYPERGRAPHS

To generate synthetic hypergraphs, we wrote hypergraph generators extending three well-known graph models to hypergraphs.

(i) For syn-c, we extended the configuration model, which, for undirected graphs, is specified by a degree sequence. Our hypergraph configuration model is specified by a node degree sequence and an edge cardinality sequence.

(ii) For syn-r, we extended the Erdős-Rényi random graph model, which, for undirected graphs, is specified by a number of nodes n and an edge existence probability p. Our Erdős-Rényi random hypergraph model is specified by a number of nodes n, a number of edges m, and the probability p of a one in any cell of the node-to-edge incidence matrix.

(iii) For syn-s, we extended the stochastic block model which, for undirected graphs, is specified by a vector of c community sizes and a $c \times c$ affinity matrix specifying affiliation probabilities between communities. Our hypergraph stochastic block model is specified by a vector of c_V node community sizes, a vector of c_E edge community sizes, and a $c_V \times c_E$ affinity matrix specifying affiliation probabilities between node communities and edge communities.

We used each of our generators to create 250 hypergraphs with identical node count n, edge count m, and density c/nm, where c is the number of filled cells in the node-to-edge incidence matrix.

CAVEATS. Our generators work by pairing node and edge indices, and duplicated (node, edge) index pairs are discarded to generate simple hypergraphs, which can lead to small deviations from the input specification in practice.

5.D Implementation Details

To simplify the computation of Wasserstein distances between adjacent nodes, we leverage the following fact about the relevant distances (i.e., transportation costs) between nodes.

Lemma 5.16. *Given a hypergraph* $H = (V, E)$ *and nodes* $i, j, k, \ell \in V$ *with* $i \sim j$ *as well as* $\mu_i(k) > 0$ *and* $\mu_j(\ell) > 0$, $d(k, \ell) \leqslant 3$.

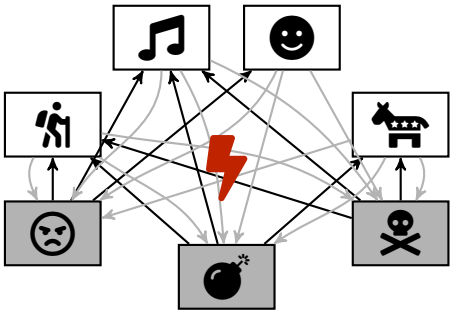
Proof. By the triangle inequality and the definition of our probability measures, we have $d(k, \ell) \leqslant d(k, i) + d(i, j) + d(j, \ell) = 3$. □

We also speed up the computation of Wasserstein distances by reducing each instance to its smallest equivalent instance, exploiting the following observation.

Lemma 5.17. *Given a hypergraph* $H = (V, E)$ *and nodes* $i, j \in V$ *with* $i \sim j$, *if* $\mu_i(k) = \mu_j(k)$ *for some node* $k \in V$, *then* $W_1(\mu_i, \mu_j) = W_1(\mu_i^{-k}, \mu_j^{-k})$, *where* μ_i^{-k} *is defined as*

$$\mu_i^{-k}(j) = \begin{cases} 0 & j = k \\ \mu_i(j) & j \neq k\,. \end{cases} \tag{5.28}$$

Proof. If $\mu_i(k) = \mu_j(k) = 0$, the claim holds trivially. Otherwise, $\mu_i(k) = \mu_j(k) = \beta > 0$. In this case, let C^* be an optimal coupling between μ_i and μ_j. If the probability mass allocated to k by μ_i does not get moved at all in C^*, it contributes 0 to $W_1(\mu_i, \mu_j)$, and we are done. Therefore, assume otherwise. Then there exist nodes $p, q \in V$ such that probability mass gets moved from p to k and from k to q in C^*. By the triangle inequality, $d(p, q) \leqslant d(p, k) + d(k, q)$, and as $d(k, k) = 0$, the cost of moving that mass directly from p to q and keeping all mass at k cannot be larger than the cost of moving the mass from p to k and from k to q. Hence, we can modify C^* such that the mass allocated to k by μ_i does not get moved at all without increasing the coupling cost. Thus, there always exists an optimal coupling in which all mass at k remains at k, and the claim follows. □



6

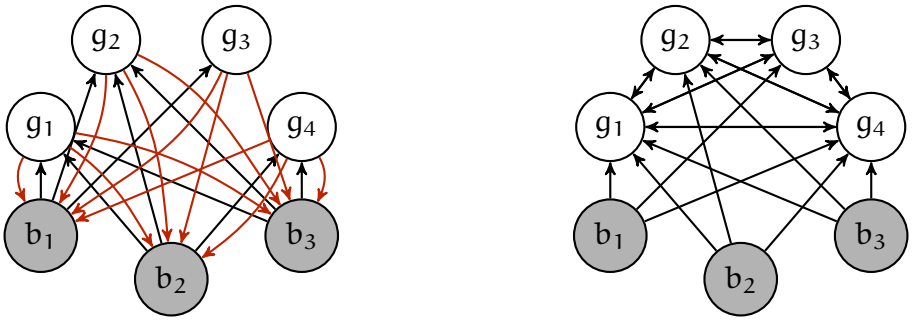
Responsibility: Gamine

In Chapter 4, we engaged with two types of *complexity*: the representational complexity of relational data, studied further during our *expressivity* explorations in Chapter 5, and the social complexity of the context in which our research is embedded. While in Chapter 4, the primary social context under consideration was our *research community*, in this chapter, we broaden our notion of context to include *society at large*. Here, one topic of increasing concern is how algorithms shape our perception of the world, especially by mediating access to information on digital platforms. This motivates us to delve into the *responsibility* dimension of Graphland, prompting us to ask:

> *How can we take* responsibility *for the risks created by graph-based algorithms?*

6.1 Introduction

Recommendation algorithms mediate access to content on digital platforms, and as such, they critically influence how individuals and societies perceive the world and form their opinions [91, 130, 211, 228, 243]. They are also a focal point of competing stakeholder interests: content creators seeking to express themselves and increase their reach, content consumers seeking to inform or entertain themselves according to their preferences, and platform operators, primarily seeking monetization. In recent years, platforms have come under increasing scrutiny from re-



(a) Minimizing *segregation* but not *exposure* **(b)** Minimizing *segregation* and *exposure*

Figure 6.1: 3-out-regular directed graphs with four *good* nodes (white) and three *bad* nodes (gray). Edges running from good nodes to bad nodes are drawn in red. The left graph minimizes the *segregation* objective from Fabbri et al. [87], but random walks oscillate between good nodes and bad nodes. In contrast, only the right graph minimizes our *exposure* objective.

searchers and regulators alike due to concerns and evidence that their recommendation algorithms create filter bubbles [50, 146, 161, 245] and fuel radicalization [123, 160, 216, 224, 269]. One of the main challenges in this context is dealing with content that is considered harmful [17, 54, 275]. To address this challenge while balancing the interests of creators, users, and platforms, rather than *block* harmful content altogether, one approach is to *minimize* the exposure to such content that is induced specifically by algorithmic recommendations.

In this spirit, and modeling media items and recommendations as a directed graph, we study the problem of reducing the exposure to harmful content via *edge rewiring*, i.e., replacing certain recommendations by others. This problem was recently introduced by Fabbri et al. [87], who proposed to address it by modeling harmfulness as a *binary* node label and minimizing the *maximum segregation*, defined as the largest expected number of steps taken by a random walk starting at a harmful node until it visits a benign node. However, while Fabbri et al. [87] posed a theoretically interesting and practically important problem, their approach has some crucial limitations.

First, treating harmfulness as dichotomous fails to capture the complexity of real-world harmfulness assessments: Regardless of the specific yardstick we use, some content will appear outright benign, and other content will appear outright bad—but plenty of content will fall somewhere in between, and not all problematic content will appear equally harmful. Second, the segregation objective ignores completely all random-walk continuations that return to harmful content after the first visit to a benign node. However, *benign nodes do not act as absorbing states* in practice, especially for users who already prefer harmful content. The consequences are illustrated in Fig. 6.1a. Here, the segregation objective by Fabbri et al. [87] judges that the graph provides minimal exposure to harmful content

(the hitting time from any harmful node to a benign node is 1), while long random walks, which model user behavior more faithfully, necessarily oscillate between harmful and benign content.

Contributions. In this chapter, we make three contributions. First, we remedy the above-mentioned limitations of the rewiring problem as formalized in prior work. That is, we model harmfulness as *real-valued* node costs, thus more nuancedly representing real-world scenarios, and we propose a novel minimization objective, the *expected total exposure*, defined as the sum of the costs of absorbing random walks starting at any node. Notably, in our model, no node is an absorbing state, but any node can lead to absorption, which represents more realistically how users cease to interact with a platform. Our objective truly minimizes the exposure to harmful content. For example, it correctly identifies the graph in Fig. 6.1b as significantly less harmful than that in Fig. 6.1a, while for the segregation objective by Fabbri et al. [87], the two graphs are indistinguishable.

Second, on the algorithmic side, we show that although minimizing the expected total exposure is NP-hard and NP-hard to approximate to within an additive error, its maximization version is equivalent to a submodular maximization problem under the assumption that the input graph contains a small number of *safe* nodes, i.e., nodes that cannot reach nodes with non-zero costs. If these safe nodes are present—which holds in 80% of the real-world graphs used in our experiments—the greedy method yields a $(1 - 1/e)$-approximation.

Third, based on our theoretical insights, we introduce Gamine, a fast greedy algorithm for reducing exposure to harmful content via edge rewiring. Gamine leverages provably effective strategies for pruning unpromising rewiring candidates, and it works both with and without quality constraints on recommendations. With just 100 rewirings on YouTube graphs containing hundred thousands of edges, Gamine reduces the exposure by 50%, while ensuring that its recommendations are at least 95% as relevant as the original recommendations.

Structure. We introduce our problems, REM and QREM, in Section 6.2, and analyze their hardness, approximability, and solution structure in Section 6.3. Building on our theoretical analyses, we develop Gamine as an efficient greedy algorithm for tackling these problems in Section 6.4, before discussing related work in Section 6.5. We demonstrate the performance of Gamine through extensive experiments on synthetic and real-world data in Section 6.6, and conclude with a discussion in Section 6.7. We make all code, datasets, and results publicly available,[1] providing further supplementary material in Sections 6.A to 6.E.

[1] 10.5281/zenodo.7936816

6.2 Problems

We consider a directed graph $G = (V, E)$ of content items (V) and what-to-consume-next recommendations (E), with $n = |V|$ nodes and $m = |E|$ edges. Since we can typically make a fixed number of recommendations for a given content item, such *recommendation graphs* are often d-out-regular, i.e., all nodes have $d = m/n$ out-neighbors, but we do not restrict ourselves to this setting. Rather, each node i has an out-degree $\delta^+(i) = |\Gamma^+(i)|$, where $\Gamma^+(i)$ is the set of out-neighbors of i, and a cost $c_i \in [0, 1]$, which quantifies the harmfulness of content item i, ranging from 0 (not harmful at all) to 1 (maximally harmful). For convenience, we define $\Delta^+ = \max\{\delta^+(i) \mid i \in V\}$ and collect all costs into a vector $\mathbf{c} \in [0, 1]^n$.

We model user behavior as a random-walk process on the recommendation graph G. Each edge (i, j) in the recommendation graph is associated with a transition probability p_{ij} such that $\sum_{j \in \Gamma^+(i)} p_{ij} = 1 - \alpha_i$, where α_i is the *absorption probability* of a random walk at node i (i.e., the probability that the walk ends at i). Intuitively, we can interpret α_i as the probability that a user stops interacting with a media platform after consuming content i. For simplicity, we assume $\alpha_i = \alpha \in (0, 1]$ for all $i \in V$. Thus, we can represent the random-walk process on G by the transition matrix $\mathbf{P} \in [0, 1 - \alpha]^{n \times n}$, where

$$\mathbf{P}[i, j] = \begin{cases} p_{ij} & \text{if } (i, j) \in E\,, \\ 0 & \text{otherwise}\,. \end{cases} \tag{6.1}$$

This is an absorbing Markov chain, and the expected number of visits from a node i to a node j before absorption is given by the entry (i, j) of the *fundamental matrix* $\mathbf{F} \in \mathbb{R}_{\geqslant 0}^{n \times n}$, defined as

$$\mathbf{F} = \sum_{i=0}^{\infty} \mathbf{P}^i = (\mathbf{I} - \mathbf{P})^{-1}\,, \tag{6.2}$$

where $\mathbf{I}$ is the $n \times n$-dimensional identity matrix, and the series converges since $\|\mathbf{P}\|_\infty = \max_i \sum_{j=0}^{n} \mathbf{P}[i, j] = 1 - \alpha < 1$. A derivation of the latter statement and of Eq. (6.2) can be found, e.g., in the classic textbook by Doyle and Snell [79]. Denoting the i-th unit vector as $\boldsymbol{e}_i$, observe that the row vector $\boldsymbol{e}_i^\top \mathbf{F}$ gives the expected number of visits, before absorption, from i to any node, and the column vector $\mathbf{F}\boldsymbol{e}_i$ gives the expected number of visits from any node to i. Hence, $\boldsymbol{e}_i^\top \mathbf{F}\mathbf{c} = \sum_{j \in V} \mathbf{F}[i, j]\mathbf{c}_j$ gives the expected exposure to harmful content of users starting their random walk at node i, referred to as the *exposure* of i. The *expected to-*

tal exposure to harm in the graph G, then, is given by the non-negative function

$$f(G) = \mathbf{1}^\top \mathbf{F}\mathbf{c} \,, \tag{6.3}$$

where $\mathbf{1}$ is the vector with each entry equal to 1.

We would like to *minimize* the exposure function given in Eq. (6.3) by making r edits to the graph G—i.e., we seek an effective *post-processing strategy* for harm reduction. For ease of exposition, and in line with our motivating application, we restrict edits to *edge rewirings* denoted as (i, j, k), in which we replace an edge $(i, j) \in E$ by an edge $(i, k) \notin E$ with $i \neq k$, setting $p_{ik} = p_{ij}$ (other edits are discussed in Section 6.E). Seeking edge rewirings to minimize the expected total exposure yields the following problem definition.

Problem 6.1 (r-Rewiring Exposure Minimization [REM])**.** *Given a graph* G*, its random-walk transition matrix* $\mathbf{P}$*, a node cost vector* $\mathbf{c}$*, and a budget* r*, minimize* $f(G_r)$*, where* G_r *is* G *after* r *rewirings.*

Equivalently, we can *maximize* the *reduction* in the expected total exposure to harmful content,

$$f_\Delta(G, G_r) = f(G) - f(G_r) \,. \tag{6.4}$$

Note that while any set of rewirings minimizing $f(G_r)$ also maximizes $f_\Delta(G, G_r)$, the approximabilities of f and f_Δ can differ widely.

As Problem 6.1 does not impose any constraints on the rewiring operations, the optimal solution might contain rewirings (i, j, k) such that node k is totally unrelated to i (and hence, potentially irrelevant for users of recommender systems). To guarantee a certain recommendation quality with our rewirings, we need additional information on the relevance of individual nodes in the context of other nodes. We assume that this information is given as a *relevance matrix* $\mathbf{R} \in \mathbb{R}_{\geqslant 0}^{n \times n}$, where $\mathbf{R}[i, j]$ denotes the relevance of node j in the context of node i. Given such relevance information, and assuming that the out-neighbors of a node i are ordered as $\mathbf{r}_i \in V^{\delta^+(i)}$, we can define a *relevance function* θ with range $[0, 1]$ to judge the quality of the recommendation sequence at node i, depending on the relevance and ordering of recommended nodes, and demand that any rewiring retain $\theta(\mathbf{r}_i) \geqslant q$ for all $i \in V$ and some *quality threshold* $q \in [0, 1]$. One potential choice for θ is the normalized discounted cumulative gain (NDCG), a popular ranking quality measure. Given $\mathbf{R}$, and denoting as $\mathrm{idx}_i(j)$ the relevance rank of j

for i, we define the *discounted cumulative gain* (DCG) [135] of node i as

$$\mathrm{DCG}(i) = \sum_{j \in \Gamma^+(i)} \frac{\mathbf{R}[i,j]}{\log_2(1 + \mathrm{idx}_i(j))} \,. \tag{6.5}$$

Denoting the $\delta^+(i)$ most relevant nodes for node i as $\mathsf{T}_{\delta^+}(i) = \{j \mid \mathrm{idx}_i(j) \leqslant \delta^+(i)\}$, we obtain the *normalized* discounted cumulative gain as

$$\textsc{n}\mathrm{DCG}(i) = \frac{\mathrm{DCG}(i)}{\textsc{i}\mathrm{DCG}(i)} \,, \tag{6.6}$$

where

$$\textsc{i}\mathrm{DCG}(i) = \sum_{j \in \mathsf{T}_{\delta^+}(i)} \frac{\mathbf{R}[i,j]}{\log_2(1 + \mathrm{idx}_i(j))} \tag{6.7}$$

is the *ideal* discounted cumulative gain. Asserting that $\textsc{n}\mathrm{DCG}(i) \geqslant q$ in the original graph G (which holds when a system simply recommends the top-ranked items, such that before rewiring, $\textsc{n}\mathrm{DCG}(i) = \textsc{i}\mathrm{DCG}(i)$ for all $i \in V$), we can demand that all rewirings (i, j, k) maintain $\textsc{n}\mathrm{DCG}(i) \geqslant q$ for $i \in V$.

Regardless of its particular instantiation, introducing θ allows us to consider a variant of REM with relevance constraints.

Problem 6.2 (q-Relevant r-Rewiring Exposure Minimization [QREM])**.** *Given a graph* G*, its random-walk transition matrix* $\mathbf{P}$*, a node cost vector* $\mathbf{c}$*, a budget* r*, a relevance matrix* $\mathbf{R}$*, a relevance function* θ*, and a quality threshold* q*, minimize* $f(G_r)$ *under the condition that* $\theta(\mathbf{r}_i) \geqslant q$ *for all* $i \in V$*.*

For $q = 0$, QREM is equivalent to REM. Collecting our notation in Table 6.7, we now seek to address both problems.

6.3 Theory

To start with, we establish some theoretical properties of our problems, the functions f and f_Δ, and potential solution approaches. In particular, we prove several hardness and approximability results, we gauge the potential of the greedy approach, and we elucidate useful structure in our REM objective.

For the development of our results, the concept of *node safety* will be critical. Hence, we start by calling a node *safe* if $\boldsymbol{e}_i^\top \mathbf{F}\mathbf{c} = 0$, i.e., no node j with $c_j > 0$ is reachable from i, and *unsafe* otherwise. Note that the existence of a safe node in a graph G containing at least one unsafe node (i.e., $c_i > 0$ for some $i \in V$) implies that G is not strongly connected. The node safety property partitions V into two sets of safe resp. unsafe nodes, $S = \{i \in V \mid \boldsymbol{e}_i^\top \mathbf{F}\mathbf{c} = 0\}$ and $U = \{i \in V \mid \boldsymbol{e}_i^\top \mathbf{F}\mathbf{c} > 0\}$.

6.3.1 Hardness

NP-Hardness of REM and QREM. Having introduced our node-safety terminology, we now show that the r-rewiring exposure minimization problem (and hence, also the q-relevant r-rewiring exposure minimization) is NP-hard.

Theorem 6.3 (NP-Hardness of REM)**.** *The r-rewiring exposure minimization problem is NP-hard, even on 3-out-regular input graphs with binary costs* $\mathbf{c} \in \{0,1\}^n$.

Proof. We reduce from minimum vertex cover for undirected cubic, i.e., 3-regular graphs (MVC-3), which is known to be NP-hard [107]. To this end, we transform an instance of MVC-3 into an instance of REM with a directed, 3-out-regular input graph (REM-3 instance) as follows. From a cubic undirected graph $G' = (V', E')$ with $n' = |V'|$ and $m' = |E'| = {}^{3n'}/_{2}$, we construct our directed REM-3 instance by defining a graph $G = (V, E)$ with $n = |V| = 2n' + 4$ nodes and $m = |E| = 6n' + 12$ edges such that

$$V = V' \cup \overline{V'} \cup S \text{ , for } \overline{V'} = \{b_i \mid i \in V'\} \text{ , } S = \{g_1, g_2, g_3, g_4\} \text{ , } \tag{6.8}$$

$$\begin{aligned} E = {} & \{(i, b_i) \mid i \in V'\} \cup \{(b_i, j) \mid \{i, j\} \in E'\} \\ & \cup \{(g_i, g_j) \in S \times S \mid i \neq j\} \cup \{(i, g_x) \mid i \in V', x \in \{1, 2\}\} \text{ ,} \end{aligned} \tag{6.9}$$

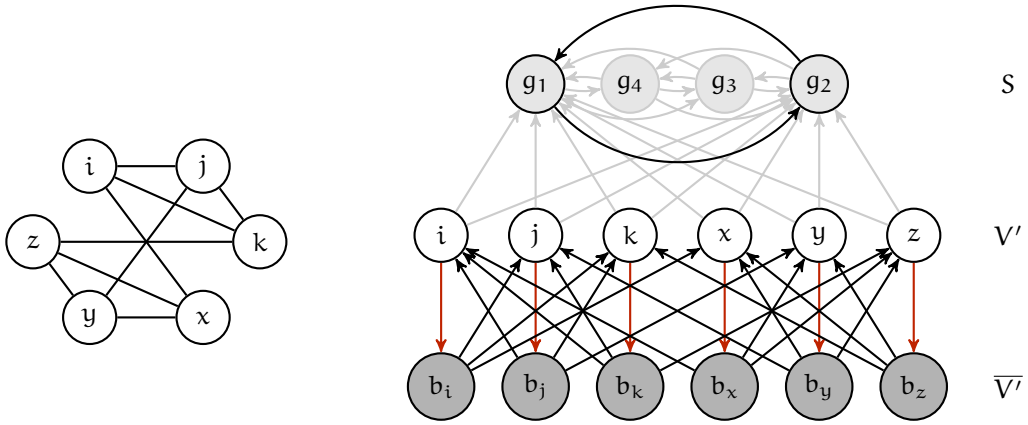
$$\mathbf{P}[x, y] = \frac{1-\alpha}{\delta^+(x)} = \frac{1-\alpha}{3} \text{ , and} \tag{6.10}$$

$$c_x = \begin{cases} 1 & x \in \overline{V'} \\ 0 & \text{otherwise .} \end{cases} \tag{6.11}$$

That is, for each node $i \in V'$, we introduce a node $i \in V$ with $c_i = 0$, a node $b_i \in V$ with $c_{b_i} = 1$, and an edge $(i, b_i) \in E$ in G. We then encode the original edge set implicitly by defining two edges $(b_i, j) \in E$ and $(b_j, i) \in E$ for each edge $\{i, j\} \in E'$. Finally, we add a complete 3-out-regular graph of zero-cost nodes and connect each node representing a node from V' to the first two nodes of that graph.

Intuitively, the edges $\{(i, b_i) \mid i \in V'\}$ will be our prime candidates for rewiring, and rewiring an edge (i, b_i) in REM-3 will correspond to selecting node i into the vertex cover of the original MVC-3 instance. The implicit encoding of the original edge set introduces the asymmetry necessary to tell from the value of our objective function if an optimal r-rewiring of G corresponds to a vertex cover of cardinality r in G'. Adding a complete 3-out-regular graph of zero-cost nodes gives us a strongly connected component S of safe nodes as rewiring targets, and it ensures that G is 3-out-regular. The entire transformation is visualized in Fig. 6.2.

In the graph G thus constructed, the only nodes ever exposed to harm are the n' nodes in V' and the n' nodes in $\overline{V'}$. As illustrated in Fig. 6.3, random walks



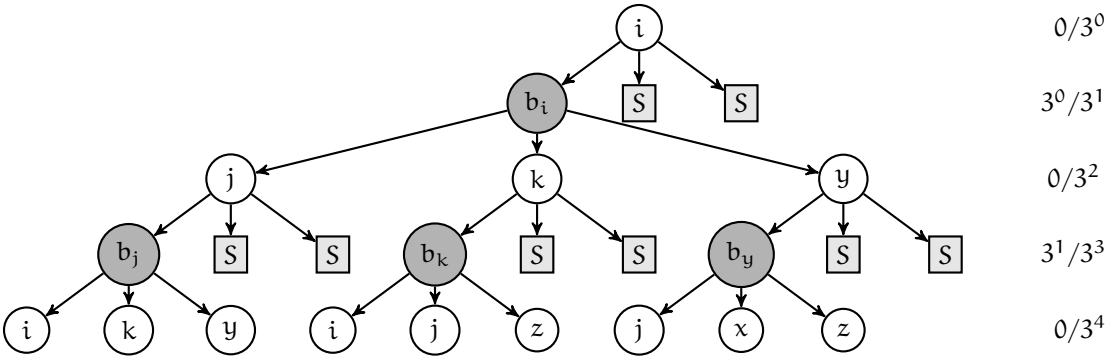
(a) Toy MVC-3 instance $G' = (V', E')$ **(b)** REM-3 instance constructed from G'

Figure 6.2: Reduction setup for Theorem 6.3. Fig. 6.2a depicts a toy MVC-3 instance, which we transform into a REM-3 instance as shown in Fig. 6.2b. In Fig. 6.2b, white nodes represent V', gray nodes represent $\overline{V'}$, silver nodes represent S, and edges (a, b) with $c_a = 0$ and $c_b = 1$ are drawn in red. Silver edges and nodes with silver boundaries are needed to ensure that G is 3-out-regular, and all edges are traversed with probability $\frac{(1-\alpha)}{3}$.

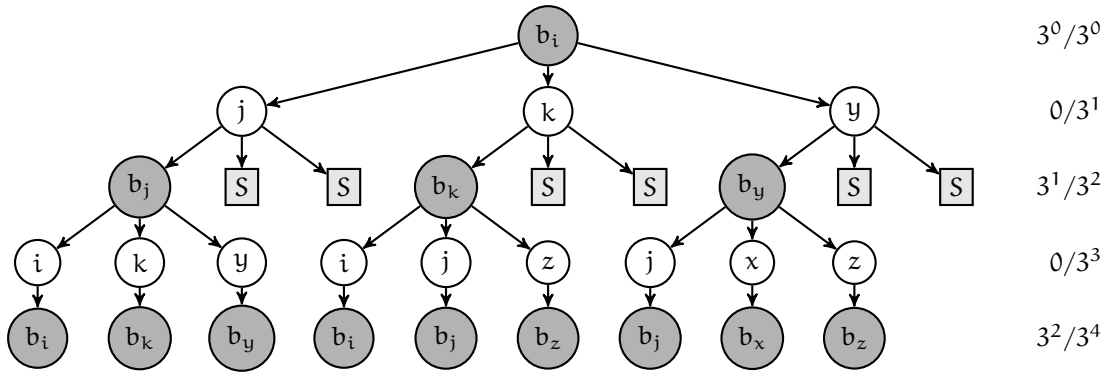
starting from a node in $\overline{V'}$ only see nodes with cost 1 after an *even* number of steps, random walks starting from a node in V' only see nodes with cost 1 after an *odd* number of steps, and as G is 3-out-regular, *all* random walks have a branching factor of 3, such that they see exactly 3^t (not necessarily distinct) nodes at t steps from their origin. Each node in V' has three out-neighbors, and before the first rewiring, exactly one of them is a node with cost 1. Thus if the random walks do not get absorbed, the regularity in our construction implies that the probability of encountering a node with cost 1 after 2 steps from a node in $\overline{V'}$ is $\frac{3^1}{3^2} = \frac{1}{3}$, just like the probability of encountering a node with cost 1 after 3 steps from a node in V' is $\frac{3^1}{3^3} = \frac{1}{9}$. Therefore, the starting value of our objective function can be written succinctly as

$$f(G) = \underbrace{n' \sum_{t=0}^{\infty} 3^{-t}(1-\alpha)^{2t}}_{\text{Contributions from } \overline{V'}} + \underbrace{n' \sum_{t=0}^{\infty} 3^{-t-1}(1-\alpha)^{2t+1}}_{\text{Contributions from } V'} . \tag{6.12}$$

Since S contains four safe nodes and G is 3-out-regular, we can always rewire edges with unsafe targets to safe targets without creating multi-edges. Therefore, as long as $r \leqslant n'$, an optimal rewiring X will contain triples of shape (i, b_i, g_x), where g_x is any node that is safe *after* the r rewirings have been performed (this includes the nodes in S but can also include other zero-cost nodes i for which $(i, b_i, g_{x'})$ is part of the rewiring for some other, safe node $g_{x'}$). Now, each indi-



(a) Walks starting at node $i \in V'$



(b) Walks starting at node $b_i \in \overline{V'}$ (safe nodes at step 4 not shown)

Figure 6.3: Random walks in 3-out-regular directed graphs $G = (V, E)$ constructed from undirected MVC-3 instances as depicted in Fig. 6.2. All edges are traversed with probability $\frac{1-\alpha}{3}$, nodes in $\overline{V'}$ are drawn in gray, and branches leading into the safe component S are collated into silver square boxes labeled S. The annotations at level t to the right of each random-walk tree indicate the fraction of nodes with cost 1 of all nodes encountered after taking exactly t steps.

vidual rewiring reduces the objective by

$$\underbrace{\frac{1}{3}(1-\alpha)}_{(1)} + \underbrace{\frac{3}{9}(1-\alpha)^2}_{(2)} + \underbrace{\frac{2\gamma'}{27}(1-\alpha)^3}_{(3)} + \varepsilon' \,, \tag{6.13}$$

where ε' is a term summarizing all contributions from walks longer than 3 steps, and γ' is the number of edges that are *newly* covered in G' by selecting the source node i of our rewiring in G into the vertex cover C of G', i.e.,

$$\gamma' = |\{e \in E' \mid C \cap e = \emptyset\}| \;\text{ for }\; C = \{i \in V' \mid (i, b_i) \in E_X\} \,, \tag{6.14}$$

where E_X is the set of previously rewired edges in G.

More elaborately, in Eq. (6.13), the component marked (1) is the exposure of i to b_i at distance 1 via the walk (i, b_i), and the component marked (2) is the exposure of b_j to b_i at distance 2 via the walk (b_j, i, b_i), for the three nodes j such that $\{i, j\} \in E'$, where $|\{\{x, y\} \in E' \mid y = j\}| = 3$ because G' is 3-regular. The component marked (3) is the sum of (i) the exposure of i to nodes b_j with $\{i, j\} \in E'$ at distance 3 via the walk (i, b_i, j, b_j), and (ii) the exposure of nodes j with $\{i, j\} \in E'$ to node i at distance 3 via the walk (j, b_j, i, b_i), each of which is

$$\frac{1}{27}(1-\alpha)^3 \cdot (1 - |E_X \cap \{(j, b_j)\}|) = \begin{cases} \frac{1}{27}(1-\alpha)^3 & \text{if } (j, b_j) \notin E_X \\ 0 & \text{otherwise .} \end{cases} \tag{6.15}$$

Hence, the objective function reduces by $\frac{2}{27}(1-\alpha)^3$ for each edge $\{i, j\} \in E'$ that is covered for the *first* time when we select i into the vertex cover of G', and because G' is 3-regular, each rewiring can cover at most 3 new edges, such that $\gamma' \leqslant 3$.

Thus, an optimal r-rewiring of G reduces the objective function in Eq. (6.12) by

$$f_\Delta(G, G_r) = \frac{r}{3}(1-\alpha) + \frac{3r}{9}(1-\alpha)^2 + \frac{2\gamma}{27}(1-\alpha)^3 + \varepsilon \,, \tag{6.16}$$

where $\gamma \geqslant \frac{3}{2}r$ is the number of edges in G' that are covered by the source nodes of our rewirings, and ε is the sum of the small terms ε' associated with each rewiring. Therefore, G' has a minimum vertex cover of size at most r if and only if an optimal r-rewiring of G reduces our objective by

$$f_\Delta(G, G_r) = \frac{r}{3}(1-\alpha) + \frac{3r}{9}(1-\alpha)^2 + \frac{2m'}{27}(1-\alpha)^3 + \varepsilon \,, \tag{6.17}$$

i.e., G' has a minimum vertex cover of size at most r if and only if

$$\begin{aligned} f(G_r) &= f(G) - f_\Delta(G, G_r) \\ &= n' + \frac{n'-r}{3}(1-\alpha) + \frac{3(n'-r)}{9}(1-\alpha)^2 + \frac{3n'-2m'}{27}(1-\alpha)^3 + (\xi - \varepsilon) \\ &= n' + \frac{n'-r}{3}(1-\alpha) + \frac{3(n'-r)}{9}(1-\alpha)^2 + \frac{3n'-2\frac{3n'}{2}}{27}(1-\alpha)^3 + (\xi - \varepsilon) \\ &= n' + \frac{n'-r}{3}(1-\alpha) + \frac{3(n'-r)}{9}(1-\alpha)^2 + (\xi - \varepsilon) \,, \end{aligned} \tag{6.18}$$

where $\xi \geqslant \varepsilon$ is the *entire* exposure of random walks in G due to nodes encountered after four or more steps, i.e.,

$$\xi = \underbrace{n' \sum_{t=0}^{\infty} 3^{-t}(1-\alpha)^{2t}}_{\textit{All} \text{ contributions from } \overline{V'}} + \underbrace{n' \sum_{t=0}^{\infty} 3^{-t-1}(1-\alpha)^{2t+1}}_{\textit{All} \text{ contributions from } V'} \tag{6.19}$$
$$- \underbrace{n' \sum_{t=0}^{1} 3^{-t}(1-\alpha)^{2t}}_{\text{At most 3 steps from } \overline{V'}} - \underbrace{n' \sum_{t=0}^{1} 3^{-t-1}(1-\alpha)^{2t+1}}_{\text{At most 3 steps from } V'} .$$

As we do not know ε exactly, we cannot check Eq. (6.18) directly to decide whether G' has a vertex cover of size at most r. Instead, we would like to check if

$$f_{\Delta}(G, G_r) \geqslant \frac{r}{3}(1-\alpha) + \frac{3r}{9}(1-\alpha)^2 + \frac{2m'}{27}(1-\alpha)^3 , \tag{6.20}$$

that is, for the purposes of our decision, we would like to ignore ε. Observe that as $\varepsilon \leqslant \xi$, we can safely do this if

$$\xi < \frac{2}{27}(1-\alpha)^3 , \tag{6.21}$$

as in this case, the *entire* exposure of random walks due to nodes encountered after four or more steps in G is smaller than the change of the objective function we obtain by covering a *single* new edge in the original MVC-3 instance G'. In Lemma 6.4, we prove that if we choose $\alpha \geqslant \frac{1}{2}$, then Eq. (6.21) is guaranteed. Hence, G' has a minimum vertex cover of size at most r if and only if Eq. (6.20) holds, and by setting

$$C = \{i \in V' \mid (i, b_i) \in E_X\} , \tag{6.22}$$

we obtain the vertex cover C of G' . □

Lemma 6.4. *If in the setting of Theorem 6.3, we set the random-walk absorption probability to $\alpha \geqslant \frac{1}{2}$, then $\xi < \frac{2}{27}(1-\alpha)^3$.*

Proof. Recall the definition of ξ from Eq. (6.19), and observe that the infinite series involved have closed-form solutions

$$\sum_{t=0}^{\infty} 3^{-t}(1-\alpha)^{2t} = \sum_{t=0}^{\infty} \frac{(1-\alpha)^{2t}}{3^t} = \sum_{t=0}^{\infty} \left(\frac{(1-\alpha)^2}{3}\right)^t = \frac{1}{1-\frac{(1-\alpha)^2}{3}} , \text{ and} \tag{6.23}$$

$$\sum_{t=0}^{\infty} 3^{-t-1}(1-\alpha)^{2t+1} = \sum_{t=0}^{\infty} \frac{(1-\alpha)^{2t+1}}{3^{t+1}} = \sum_{t=0}^{\infty} \frac{1-\alpha}{3}\left(\frac{(1-\alpha)^2}{3}\right)^t = \frac{\frac{1-\alpha}{3}}{1-\frac{(1-\alpha)^2}{3}}\,, \tag{6.24}$$

and that the partial sums evaluate to

$$\sum_{t=0}^{1} 3^{-t}(1-\alpha)^{2t} = 1 + \frac{1}{3}(1-\alpha)^2\,, \text{ and} \tag{6.25}$$

$$\sum_{t=0}^{1} 3^{-t-1}(1-\alpha)^{2t+1} = \frac{1}{3}(1-\alpha) + \frac{1}{9}(1-\alpha)^3\,. \tag{6.26}$$

Using these equalities to rewrite Eq. (6.19) for ξ, and setting $x = 1-\alpha$ intermittently, where for $\alpha \geqslant \frac{1}{2}$, we have $x \leqslant \frac{1}{2}$, we obtain

$$\begin{aligned}
\xi =& \frac{1+\frac{(1-\alpha)}{3}}{1-\frac{(1-\alpha)^2}{3}} - 1 - \frac{(1-\alpha)^2}{3} - \frac{(1-\alpha)}{3} - \frac{(1-\alpha)^3}{9} = \frac{1+\frac{x}{3}}{1-\frac{x^2}{3}} - 1 - \frac{x^2}{3} - \frac{x}{3} - \frac{x^3}{9} \qquad (6.27)\\
=& \frac{1+\frac{x}{3}}{1-\frac{x^2}{3}} - \frac{1-\frac{x^2}{3}}{1-\frac{x^2}{3}} - \frac{\frac{x^2}{3} - \frac{x^2}{3}\frac{x^2}{3}}{1-\frac{x^2}{3}} - \frac{\frac{x}{3} - \frac{x}{3}\frac{x^2}{3}}{1-\frac{x^2}{3}} - \frac{\frac{x^3}{9} - \frac{x^3}{9}\frac{x^2}{3}}{1-\frac{x^2}{3}}\\
=& \frac{1+\frac{x}{3} - 1 + \frac{x^2}{3} - \frac{x^2}{3} + \frac{x^2}{3}\frac{x^2}{3} - \frac{x}{3} + \frac{x}{3}\frac{x^2}{3} - \frac{x^3}{9} + \frac{x^3}{9}\frac{x^2}{3}}{1-\frac{x^2}{3}}\\
=& \frac{\frac{x^4}{9} + \frac{x^5}{27}}{1-\frac{x^2}{3}} = \frac{\frac{x}{9} + \frac{x^2}{27}}{1-\frac{x^2}{3}} x^3\\
\leqslant& \frac{\frac{1}{18} + \frac{1}{108}}{1-\frac{1}{12}} x^3 = \frac{108+18}{18\cdot 108}\cdot\frac{12}{11}x^3 = \frac{126}{3\cdot 54\cdot 11}x^3 = \frac{126}{1\,749}x^3 = \frac{3\,402}{1\,749\cdot 27}x^3\\
<& \frac{3\,498}{1\,749\cdot 27}x^3 = \frac{2}{27}x^3 = \frac{2}{27}(1-\alpha)^3\,,
\end{aligned}$$

as required. □

Note that the choice of $\alpha \geqslant \frac{1}{2}$ in Lemma 6.4 is almost tight, as when setting $\alpha = 1 - \frac{1}{10}(\sqrt{201} - 9) \approx 0.48$, we obtain

$$\frac{\frac{1-\alpha}{9} + \frac{(1-\alpha)^2}{27}}{1-\frac{(1-\alpha)^2}{3}} = \frac{2}{27}\,. \tag{6.28}$$

We show the slightly looser bound as it suffices to prove Theorem 6.3 and simplifies the presentation.

Hardness of Approximation for REM and QREM. Having established the NP-hardness of REM and QREM, we now show that REM (and again, consequently QREM) is also hard to approximate under the Unique Games Conjecture (UGC) [143], an influential conjecture in hardness-of-approximation theory.

Theorem 6.5. *Assuming the UGC, REM is hard to approximate to within an additive error of both $\Theta(n)$ and $\Theta(r)$.*

Proof. Under the UGC, MVC is hard to approximate to within a factor of $(2-\varepsilon)$ [144], and it is generally hardest to approximate on regular graphs [89]. Therefore, consider again the reduction construction from the proof of Theorem 6.3 with an original MVC-3 graph $G' = (V', E')$ as well as a transformed REM graph $G = (V, E)$, and assume that $\alpha = \frac{1}{2}$, satisfying Lemma 6.4.

A solution to REM on a graph derived from an MVC-3 instance that has a minimum vertex cover of size r which approximates the optimum to within an additive error of

$$\frac{2r}{27}(1-\alpha)^3 - \frac{2\frac{r-\varepsilon r}{2}}{27}(1-\alpha)^3 = \frac{(1+\varepsilon)r}{27}(1-\alpha)^3 = \frac{(1+\varepsilon)r}{27\cdot 8} \tag{6.29}$$

would rewire edges such that $\frac{r-\varepsilon r}{2}$ edges in the MVC-3 instance remain uncovered. In this case, taking both endpoints of all uncovered edges yields a vertex cover of size $r + 2\frac{r-\varepsilon r}{2} = (2-\varepsilon)r$. Thus, if there existed an algorithm $\mathcal{A}$ discovering the stated approximate solution to REM in polynomial time, we could obtain a $(2-\varepsilon)$-approximation to MVC-3 in polynomial time by transforming the MVC-3 instance into a REM-3 instance, running $\mathcal{A}$ for all integers $r \in \{\frac{n'}{4}, \ldots, \frac{2n'}{3}\}$, where $\frac{n'}{4}$ and $\frac{2n'}{3}$ are the minimum resp. maximum cardinality of an MVC on a 3-regular undirected graph with n' nodes, reconstructing the vertex cover solutions, and finally picking the solution with the smallest cardinality. This would contradict the UGC. Observing that the cardinality of an MVC in 3-regular undirected graphs with n' nodes is in $r \in \Theta(n')$, that $n' \in \Theta(n)$, and that $\frac{(1+\varepsilon)}{27\cdot 8} \in \Theta(1)$, the claim follows. □

6.3.2 Approximability

Although we cannot approximate f directly, we *can* approximate f_Δ *with guarantees* under mild assumptions, detailed below. To formulate this result and its assumptions, observe that the node safety property not only partitions the nodes V into two sets of safe nodes S resp. unsafe nodes U, but it also partitions the edges E into four sets, E_{SS}, E_{SU}, E_{US}, and E_{UU}, where $E_{AB} = \{(i,j) \in E \mid i \in A, j \in B\}$, and $E_{SU} = \emptyset$ by construction. Further, observe that if $S \neq \emptyset$, then f is minimized, and f_Δ is maximized, once $E_{UU} = \emptyset$. This allows us to state the following result.

Lemma 6.6. *If there exists a safe node in* G *and we allow multi-edges, maximizing* f_Δ *is equivalent to maximizing a monotone, submodular set function over* E_{uu}.

Proof. By assumption, there exists a safe node in G. Therefore, fix a safe node s, and observe that s is an optimal rewiring target because $\boldsymbol{e}_s^\top \mathbf{F}\mathbf{c} = 0$. Hence, there exists an optimal strategy for maximizing f_Δ that selects only rewirings (i, j, s) with $(i, j) \in E_{uu}$. Now denote the set of rewirings as X, and the set of rewired edges as $E_X = \{(i, j) \mid (i, j, k) \in X\}$. Knowing that there exists an optimal rewiring for which $E_X \subseteq E_{uu}$, we can define a set function $\hat{f}_\Delta$ over the set E_{uu} that is equivalent to f_Δ as

$$\hat{f}_\Delta(E_X) = f(G) - f(G_{E_X}) \,. \tag{6.30}$$

The function $\hat{f}_\Delta$ is *monotone* because we only perform rewirings from E_{uu} to s, and no such rewiring can decrease $\hat{f}_\Delta$. To see that $\hat{f}_\Delta$ is also *submodular*, fix $E_X \subseteq E_{uu}$, and consider $x_1 \neq x_2 \in E_{uu} \setminus E_X$. Observe that x_1 and x_2 consist of unsafe nodes, which cannot be reachable from s—otherwise, $\boldsymbol{e}_s^\top \mathbf{F}\mathbf{c} > 0$, and s would not be safe. Hence, there is no exposure to harm that is *only* removed when both x_1 *and* x_2 are rewired, and we have

$$f(G_{E_X}) - f(G_{E_X \cup \{x1, x_2\}}) \leqslant \big(f(G_{E_X}) - f(G_{E_X \cup \{x_1\}})\big) + \big(f(G_{E_X}) - f(G_{E_X \cup \{x_2\}})\big) \,. \tag{6.31}$$

Using the definition from Eq. (6.30), we get

$$f(G_{E_X}) - f(G_{E_X \cup \{x1, x_2\}}) = f(G_{E_X}) - f(G) + \hat{f}_\Delta(E_X \cup \{x_1, x_2\}) \,, \tag{6.32}$$

$$f(G_{E_X}) - f(G_{E_X \cup \{x_1\}}) = f(G_{E_X}) - f(G) + \hat{f}_\Delta(E_X \cup \{x_1\}) \,, \text{ and} \tag{6.33}$$

$$f(G_{E_X}) - f(G_{E_X \cup \{x_2\}}) = f(G_{E_X}) - f(G) + \hat{f}_\Delta(E_X \cup \{x_2\}) \,, \tag{6.34}$$

for the three parts of Eq. (6.31). Putting things together, we obtain

$$\begin{aligned} f(G_{E_X}) - f(G) + \hat{f}_\Delta(E_X \cup \{x_1, x_2\}) &\leqslant f(G_{E_X}) - f(G) + \hat{f}_\Delta(E_X \cup \{x_2\}) \\ &\quad + \; f(G_{E_X}) - f(G) + \hat{f}_\Delta(E_X \cup \{x_1\}) \\ \Leftrightarrow f(G) - f(G_{E_X}) + \hat{f}_\Delta(E_X \cup \{x_1, x_2\}) &\leqslant \hat{f}_\Delta(E_X \cup \{x_1\}) + \hat{f}_\Delta(E_X \cup \{x_2\}) \\ \Leftrightarrow \hat{f}_\Delta(E_X) + \hat{f}_\Delta(E_X \cup \{x_1, x_2\}) &\leqslant \hat{f}_\Delta(E_X \cup \{x_1\}) + \hat{f}_\Delta(E_X \cup \{x_2\}) \,, \end{aligned} \tag{6.35}$$

which is the definition of submodularity. □

Our motivating application, however, ideally prevents multi-edges. To get a similar result without multi-edges, denote by

$$\Lambda^+ = \max \{\delta^+(i) \mid i \in U\} \tag{6.36}$$

the maximum out-degree of any *unsafe* node in G, and assume that $|S| \geqslant \Lambda^+$. Now, we obtain the following.

Theorem 6.7. *If $|S| \geqslant \Lambda^+$, then maximizing f_Δ is equivalent to maximizing a monotone and submodular set function over E_{UU}.*

Proof. Following the reasoning provided for Lemma 6.6, with the modification that we need $|S| \geqslant \Lambda^+$ to ensure that safe targets are always available for rewiring without creating multi-edges. □

Observe that the larger the number of zero-cost nodes, the smaller the number of edges, or the more homophilous the linking, the higher the probability that safe nodes exist in a graph. Notably, the precondition of Theorem 6.7 holds for the graph constructed to prove Theorem 6.3 (Fig. 6.2) as well as for most of the real-world graphs used in our experiments (Fig. 6.25). However, Theorem 6.7 only applies to the maximization version of REM (Eq. (6.42)) and not to the maximization version of QREM, since in the quality-constrained setting, some safe nodes might not be available as rewiring targets for edges emanating from unsafe nodes.

Still, for the maximization version of REM, due to Theorem 6.7, using a greedy approach to optimize f_Δ provides an approximation guarantee with respect to the optimal solution [200].

Corollary 6.8. *If the precondition of Theorem 6.7 holds, then the greedy algorithm, which always picks the rewiring (i, j, k) that maximizes $f_\Delta(G, G_1)$ for the current G, yields a $(1 - 1/e)$-approximation for f_Δ.*

Note that Corollary 6.8 does not provide any approximation guarantee for the minimization of f: Although f is necessarily supermodular when f_Δ is submodular, supermodular minimization is much less well-behaved than submodular maximization, such that approximation guarantees obtained in the latter setting do not generally carry over to the former [132, 280]. Further, observe that Corollary 6.8, which follows from Lemma 6.6 and Theorem 6.7, does not contradict Theorem 6.5. For the graphs used in Theorem 6.3 and Theorem 6.5, which satisfy the precondition of Theorem 6.7, the value of f_Δ stated in Eq. (6.17) is

$$f_\Delta(G, G_r) = \frac{r}{3}(1-\alpha) + \frac{3r}{9}(1-\alpha)^2 + \frac{2m'}{27}(1-\alpha)^3 + \varepsilon\,. \tag{6.37}$$

Table 6.1: Summary of an edge rewiring (i, j, k) in a graph $G = (V, E)$ with random-walk transition matrix $\mathbf{P}$ and fundamental matrix $\mathbf{F} = (\mathbf{I} - \mathbf{P})^{-1}$.

$G' = (V, E')$, for $E' = (E \setminus \{(i,j)\}) \cup \{(i,k)\}$, $(i,j) \in E$, $(i,k) \notin E$
$\mathbf{P}'[x,y] = \begin{cases} 0 & \text{if } x = i \text{ and } y = j\,, \\ \mathbf{P}[i,j] & \text{if } x = i \text{ and } y = k\,, \\ \mathbf{P}[x,y] & \text{otherwise}\,. \end{cases}$
$\mathbf{F}' = \mathbf{F} - \frac{\mathbf{F}\boldsymbol{u}\boldsymbol{v}^\top\mathbf{F}}{1+\boldsymbol{v}^\top\mathbf{F}\boldsymbol{u}}$, with $\boldsymbol{u} = p_{ij}\boldsymbol{e}_i$, $\boldsymbol{v} = \boldsymbol{e}_j - \boldsymbol{e}_k$, cf. Eq. (6.40)

Therefore, the $(1-1/e)$-approximation of f_Δ guaranteed by Corollary 6.8 still loses an additive term of $\Theta(n)$ and $\Theta(r)$, as required by Theorem 6.5.

6.3.3 Greedy Rewiring

Given the quality assurance of a greedy approach at least for REM, we seek to design an efficient greedy algorithm to tackle both REM and QREM. To this end, we analyze the mechanics of individual rewirings to understand how we can identify and perform greedily optimal rewirings efficiently. As each greedy step constitutes a rank-one update of the transition matrix $\mathbf{P}$, we can express the new transition matrix $\mathbf{P}'$ as

$$\mathbf{P}' = \mathbf{P} + \boldsymbol{u}(-\boldsymbol{v})^\top\,, \tag{6.38}$$

where $\boldsymbol{u} = p_{ij}\boldsymbol{e}_i$ and $\boldsymbol{v} = \boldsymbol{e}_j - \boldsymbol{e}_k$, and we omit the dependence on i, j, and k for notational conciseness. This corresponds to a rank-one update of $\mathbf{F}$, such that we obtain the new fundamental matrix $\mathbf{F}'$ as

$$\mathbf{F}' = (\mathbf{I} - (\mathbf{P} + \boldsymbol{u}(-\boldsymbol{v})^\top))^{-1} = (\mathbf{I} - \mathbf{P} + \boldsymbol{u}\boldsymbol{v}^\top)^{-1}\,. \tag{6.39}$$

The rank-one update allows us to use the Sherman-Morrison formula [239] to compute the updated fundamental matrix as

$$\mathbf{F}' = \mathbf{F} - \frac{\mathbf{F}\boldsymbol{u}\boldsymbol{v}^\top\mathbf{F}}{1 + \boldsymbol{v}^\top\mathbf{F}\boldsymbol{u}}\,. \tag{6.40}$$

The mechanics of an individual edge rewiring are summarized in Table 6.1. They will help us *perform* greedy updates efficiently.

To also *identify* greedily optimal rewirings efficiently, leveraging Eq. (6.40), we assess the impact of a rewiring on the value of our objective function, which will help us prune weak rewiring candidates. For a rewiring (i, j, k) represented by $\boldsymbol{u}$

and $\boldsymbol{v}$, the value of the exposure function f for the new graph G' is

$$\begin{aligned} f(G') &= \mathbf{1}^\top \mathbf{F}' \mathbf{c} = \mathbf{1}^\top \left(\mathbf{F} - \frac{\mathbf{F}\mathbf{u}\boldsymbol{v}^\top \mathbf{F}}{1 + \boldsymbol{v}^\top \mathbf{F}\mathbf{u}} \right) \mathbf{c} = \mathbf{1}^\top \mathbf{F}\mathbf{c} - \mathbf{1}^\top \left(\frac{\mathbf{F}\mathbf{u}\boldsymbol{v}^\top \mathbf{F}}{1 + \boldsymbol{v}^\top \mathbf{F}\mathbf{u}} \right) \mathbf{c} \\ &= f(G) - \frac{(\mathbf{1}^\top \mathbf{F}\mathbf{u})(\boldsymbol{v}^\top \mathbf{F}\mathbf{c})}{1 + \boldsymbol{v}^\top \mathbf{F}\mathbf{u}} = f(G) - \frac{\sigma\tau}{\rho} = f(G) - \Delta \,, \end{aligned} \tag{6.41}$$

with $\sigma = \mathbf{1}^\top \mathbf{F}\mathbf{u}$, $\tau = \boldsymbol{v}^\top \mathbf{F}\mathbf{c}$, $\rho = 1 + \boldsymbol{v}^\top \mathbf{F}\mathbf{u}$, and

$$\Delta = f_\Delta(G, G') = \frac{\sigma\tau}{\rho} = \frac{(\mathbf{1}^\top \mathbf{F}\mathbf{u})(\boldsymbol{v}^\top \mathbf{F}\mathbf{c})}{1 + \boldsymbol{v}^\top \mathbf{F}\mathbf{u}} \,. \tag{6.42}$$

The interpretation of the above quantities is as follows: σ is the p_{ij}-scaled i-th column sum of $\mathbf{F}$ (expected number of visits to i), τ is the cost-scaled sum of the differences between the j-th row and the k-th row of $\mathbf{F}$ (expected number of visits from j resp. k), and ρ is a normalization factor scaling the update by 1 plus the p_{ij}-scaled difference in the expected number of visits from j to i and from k to i, ensuring that $\mathbf{F}'\mathbf{1} = \mathbf{F}\mathbf{1}$. Scrutinizing Eq. (6.42), we observe the following.

Lemma 6.9. *For a rewiring (i, j, k) represented by $\mathbf{u}$ and $\boldsymbol{v}$, (i) ρ is always positive, (ii) σ is always positive, and (iii) τ can have any sign.*

Proof. We obtain this result by analyzing the definitions of ρ, σ, and τ.

First, for ρ, we have

$$\rho = 1 + \boldsymbol{v}^\top \mathbf{F}\mathbf{u} = 1 + \boldsymbol{v}^\top p_{ij} \mathbf{F}[:, i] = 1 + p_{ij} \mathbf{F}[j, i] - p_{ij} \mathbf{F}[k, i] \,. \tag{6.43}$$

For a node x, $p_{ij}\mathbf{F}[x, i]$ is the expected number of times we traverse the edge (i, j) in a random walk starting at x. Now, the probability that we reach j from $k \notin \{i, j\}$ is at most $(1 - \alpha)$, and the probability that we traverse (i, j) from k without first visiting j is at most $(1 - \alpha)p_{ij}$. Since $\alpha > 0$ and $p_{ij} \leqslant 1 - \alpha$, therefore, we have

$$p_{ij}\mathbf{F}[k, i] \leqslant (1 - \alpha)p_{ij} + (1 - \alpha)p_{ij}\mathbf{F}[j, i] < 1 + p_{ij}\mathbf{F}[j, i] \,, \tag{6.44}$$

and hence, $\rho > 0$.

Second, for σ, we have

$$\sigma = \mathbf{1}^\top \mathbf{F}\mathbf{u} = p_{ij} \sum_x \mathbf{F}[x, i] \,, \tag{6.45}$$

which is positive as all row sums of $\mathbf{F}$ are positive.

And third, for τ, we have

$$\tau = \boldsymbol{\nu}^\top \mathbf{F}\mathbf{c} = \boldsymbol{e}_j^\top \mathbf{F}\mathbf{c} - \boldsymbol{e}_k^\top \mathbf{F}\mathbf{c}\,, \tag{6.46}$$

which is *positive* (resp. *negative*) if j is *more* (resp. *less*) exposed to harm than k, and *zero* if both nodes are *equally* exposed to harm. □

To express when we can safely prune rewiring candidates, we call a rewiring (i, j, k) *greedily permissible* if $\Delta > 0$, i.e., if it reduces our objective, and *greedily optimal* if it maximizes Δ. For QREM, we further call a rewiring (i, j, k) *greedily* q*-permissible* if it ensures that $\theta(\mathbf{r}_i) \geqslant q$ under the given relevance function θ. With this terminology, we can confirm our intuition about rewirings as a corollary of Eqs. (6.41) and (6.42), combined with Lemma 6.9.

Corollary 6.10. *A rewiring* (i, j, k) *is greedily permissible if and only if* $\tau > 0$*, i.e., if* j *is more exposed to harm than* k*.*

For the greedily optimal rewiring, that is, to maximize Δ, we would like $\sigma\tau$ to be as *large* as possible, and ρ to be as *small* as possible. Inspecting Eq. (6.42), we find that to accomplish this objective, it helps if (in expectation) i is visited more often (from σ), j is more exposed and k is less exposed to harm (from τ), and i is harder to reach from j and easier to reach from k (from ρ).

In the next section, we leverage these insights to guide our efficient implementation of the greedy method for REM and QREM.

6.4 Algorithm

In the previous section, we identified useful structure in the fundamental matrix $\mathbf{F}$, the exposure function f, and our maximization objective f_Δ. Now, we leverage this structure to design an efficient greedy algorithm for REM and QREM. We develop this algorithm in three steps, focusing on REM in the first two steps, and integrating the capability to handle QREM in the third step.

6.4.1 Naïve Implementation

Given a graph G, its transition matrix $\mathbf{P}$, a cost vector $\mathbf{c}$, and a budget r, a naïve greedy implementation for REM, stated as Algorithm 6.1, computes the fundamental matrix and gradually fills up an initially empty set of rewirings by performing r greedy steps before returning the selected rewirings. In each greedy step, we identify the triple (i, j, k) that maximizes Eq. (6.42) by going through all edges $(i, j) \in E$ and computing Δ for rewirings to all potential targets k. We then update E, $\mathbf{P}$, and $\mathbf{F}$ to reflect a rewiring replacing (i, j) by (i, k) (cf. Table 6.1), and

Algorithm 6.1: Naïve greedy REM

Input: Graph $G = (V, E)$, transition matrix $\mathbf{P}$, costs $\mathbf{c}$, budget r
Output: Set of r rewirings X of shape (i, j, k)

$\mathbf{F} \leftarrow (\mathbf{I} - \mathbf{P})^{-1}$ ▷ Eq. (6.2)
$X \leftarrow \emptyset$
for $i \in \mathbb{N}_{\leqslant r}$ **do**
  1-REM ( )
return X
Function 1-REM ( )
  $\Delta, i', j', k' \leftarrow 0, \bot, \bot, \bot$
  for $(i, j) \in E$ **do**
    for $k \in V \setminus (\Gamma^+(i) \cup \{i\})$ **do**
      $\boldsymbol{u} \leftarrow \mathbf{P}[i, j]\boldsymbol{e}_i$
      $\boldsymbol{v} \leftarrow \boldsymbol{e}_j - \boldsymbol{e}_k$
      $\Delta_{ijk} \leftarrow \frac{(\mathbf{1}^\top \mathbf{F}\boldsymbol{u})(\boldsymbol{v}^\top \mathbf{F}\mathbf{c})}{1 + \boldsymbol{v}^\top \mathbf{F}\boldsymbol{u}}$ ▷ Eq. (6.42)
      if $\Delta_{ijk} > \Delta$ **then**
        $\Delta, i', j', k' \leftarrow \Delta_{ijk}, i, j, k$
  $E \leftarrow (E \setminus \{(i', j')\}) \cup \{(i', k')\}$ ▷ Table 6.1
  $\mathbf{P}[i', k'] \leftarrow \mathbf{P}[i', j']$
  $\mathbf{P}[i', j'] \leftarrow 0$
  $\mathbf{F} \leftarrow \mathbf{F} - \frac{\mathbf{F}\boldsymbol{u}\boldsymbol{v}^\top \mathbf{F}}{1 + \boldsymbol{v}^\top \mathbf{F}\boldsymbol{u}}$
  $X \leftarrow X \cup \{(i', j', k')\}$

add the triple (i, j, k) to our set of rewirings. Computing the fundamental matrix naïvely takes time $\mathcal{O}(n^3)$, computing Δ takes time $\mathcal{O}(n)$ and is done $\mathcal{O}(mn)$ times, and updating $\mathbf{F}$ takes time $\mathcal{O}(n^2)$. Hence, we arrive at a time complexity of $\mathcal{O}(rn^2(n + m))$. But we can do better.

6.4.2 Forgoing Matrix Inversion

When identifying the greedy rewiring, we never need access to $\mathbf{F}$ directly. Rather, in Eq. (6.42), we work with $\mathbf{1}^\top \mathbf{F}$, corresponding to the column sums of $\mathbf{F}$, and with $\mathbf{F}\mathbf{c}$, corresponding to the cost-scaled row sums of $\mathbf{F}$. We can approximate both via power iteration:

$$\mathbf{1}^\top \mathbf{F} = \mathbf{1}^\top \sum_{i=0}^{\infty} \mathbf{P}^i = \mathbf{1}^\top + \mathbf{1}^\top \mathbf{P} + (\mathbf{1}^\top \mathbf{P})\mathbf{P} + ((\mathbf{1}^\top \mathbf{P})\mathbf{P})\mathbf{P} + \dots \tag{6.47}$$

$$\mathbf{F}\mathbf{c} = (\sum_{i=0}^{\infty} \mathbf{P}^i)\mathbf{c} = \mathbf{c} + \mathbf{P}\mathbf{c} + \mathbf{P}(\mathbf{P}\mathbf{c}) + \mathbf{P}(\mathbf{P}(\mathbf{P}\mathbf{c})) + \dots \tag{6.48}$$

Algorithm 6.2: Exact greedy REM

Input: Graph $G = (V, E)$, transition matrix $\mathbf{P}$, costs $\mathbf{c}$, budget r
Output: Set of r rewirings X of shape (i, j, k)

$X \leftarrow \emptyset$
for $i \in \mathbb{N}_{\leqslant r}$ **do**
  Precompute $\mathbf{1}^\top \mathbf{F}$ and $\mathbf{F}\mathbf{c}$ ▷ $\mathcal{O}(\kappa m)$
  Precompute $\mathbf{1}^\top \mathbf{F}\mathbf{u}$ for $(i, j) \in E$ ▷ $\mathcal{O}(m)$
  Precompute $\boldsymbol{\nu}^\top \mathbf{F}\mathbf{c}$ for $j \neq k \in V$ ▷ $\mathcal{O}(n^2)$
  Precompute $\mathbf{F}\mathbf{u}$ for $(i, j) \in E$ ▷ $\mathcal{O}(\kappa n^2)$
  1-REM ( )
return X
Function 1-REM ( )
  $\Delta, i', j', k' \leftarrow 0, \bot, \bot, \bot$
  for $(i, j) \in E$ **do** ▷ $\mathcal{O}(m)$
    for $k \in V \setminus (\Gamma^+(i) \cup \{i\})$ **do** ▷ $\mathcal{O}(n)$
      $\mathbf{u} \leftarrow \mathbf{P}[i, j]\boldsymbol{e}_i$
      $\boldsymbol{\nu} \leftarrow \boldsymbol{e}_j - \boldsymbol{e}_k$
      $\Delta_{ijk} \leftarrow \frac{(\mathbf{1}^\top \mathbf{F}\mathbf{u})(\boldsymbol{\nu}^\top \mathbf{F}\mathbf{c})}{1 + \boldsymbol{\nu}^\top \mathbf{F}\mathbf{u}}$ ▷ $\mathcal{O}(1)$
      if $\Delta_{ijk} > \Delta$ **then**
        $\Delta, i', j', k' \leftarrow \Delta_{ijk}, i, j, k$
  $E \leftarrow (E \setminus \{(i', j')\}) \cup \{(i', k')\}$
  $\mathbf{P}[i', k'] \leftarrow \mathbf{P}[i', j']$
  $\mathbf{P}[i', j'] \leftarrow 0$
  $X \leftarrow X \cup \{(i', j', k')\}$

For each term in these sums, we need to perform $\mathcal{O}(m)$ multiplications, such that we can compute $\mathbf{1}^\top \mathbf{F}$ and $\mathbf{F}\mathbf{c}$ in time $\mathcal{O}(\kappa m)$, where κ is the number of power iterations. This allows us to compute $\mathbf{1}^\top \mathbf{F}\mathbf{u}$ for all $(i, j) \in E$ in time $\mathcal{O}(m)$ and $\boldsymbol{\nu}^\top \mathbf{F}\mathbf{c}$ for all $j \neq k \in V$ in time $\mathcal{O}(n^2)$. To compute Δ in time $\mathcal{O}(1)$, as $\mathbf{F}$ is now unknown, we need to compute $\mathbf{F}\mathbf{u}$ for all $(i, j) \in E$ via power iteration, which is doable in time $\mathcal{O}(\kappa n^2)$. This changes the running time from $\mathcal{O}(r n^2 (n + m))$ to $\mathcal{O}(r \kappa n (n + m))$, and we provide the resulting algorithm as Algorithm 6.2. But we can do better.

6.4.3 Reducing the Number of Candidate Rewirings

Observe that to further improve the time complexity of our algorithm, we need to reduce the number of rewiring candidates considered. To this end, note that the quantity τ is maximized for the nodes j and k with the largest difference in cost-scaled row sums (cf. Eq. (6.46)). How exactly we leverage this fact depends on our problem.

If we solve REM, instead of considering all possible rewiring targets, we focus on the $\Delta^+ + 2$ candidate targets K with the smallest exposure, which we can identify in time $\mathcal{O}(n)$ without sorting $\mathbf{Fc}$. This ensures that for each $(i,j) \in E$, there is at least one $k \in K$ such that $k \neq i$ and $k \neq j$, which ascertains that despite restricting to K, for each $i \in V$, we still consider the rewiring (i,j,k) maximizing τ. With this modification, we reduce the number of candidate targets from $\mathcal{O}(n)$ to $\mathcal{O}(\Delta^+)$, and the time to compute all relevant $\boldsymbol{\nu}^T\mathbf{Fc}$ values from $\mathcal{O}(n^2)$ to $\mathcal{O}(\Delta^+ n)$.

To obtain a subquadratic complexity, however, we still need to eliminate the computation of $\mathbf{Fu}$ for all $(i,j) \in E$. This also means that we can no longer afford to compute ρ for each of the now $\mathcal{O}(m\Delta^+)$ rewiring candidates under consideration, as this can only be done in constant time if $\mathbf{Fu}$ is already precomputed for the relevant edge (i,j). However, ρ is driven by the difference between two *entries* of $\mathbf{F}$, whereas τ is driven by the difference between two *row sums* of $\mathbf{F}$, and σ is driven by a single *column sum* of $\mathbf{F}$. Thus, although $\sigma\tau > \sigma\tau'$ does not generally imply ${}^{\sigma\tau}/_{\rho} > {}^{\sigma\tau'}/_{\rho'}$, the variation in $\sigma\tau$ is typically much larger than that in ρ, and large $\sigma\tau$ values mostly dominate small values of ρ. Consequently, as demonstrated in Section 6.D.2, the correlation between $\Delta = {}^{\sigma\tau}/_{\rho}$ and

$$\hat{\Delta} = \Delta\rho = \sigma\tau \tag{6.49}$$

is almost perfect. Thus, instead of Δ, we opt to compute $\hat{\Delta}$ as a heuristic, and we further hedge against small fluctuations without increasing the time complexity of our algorithm by computing Δ for the rewirings associated with the $\mathcal{O}(1)$ *largest* values of $\hat{\Delta}$, rather than selecting the rewiring with the *best* $\hat{\Delta}$ value directly. Using $\hat{\Delta}$ instead of Δ, we obtain a running time of $\mathcal{O}(r\kappa\Delta^+(n+m))$ when solving REM.

When solving QREM, we are given a relevance matrix $\mathbf{R}$, a relevance function θ, and a quality threshold q as additional inputs. Instead of considering the $\Delta^+ + 2$ nodes K with the smallest exposure as candidate targets for *all* edges, for *each* edge (i,j), we first identify the set of rewiring candidates (i,j,k) such that (i,j,k) is q-permissible, i.e., $\theta(\mathbf{r}_i) \geqslant q$ after replacing (i,j) by (i,k), and then select the node k_{ij} with the smallest exposure to construct our most promising rewiring candidate (i,j,k_{ij}) for edge (i,j). This ensures that we can still identify the rewiring (i,j,k) that maximizes $\sigma\tau$ *and* satisfies our quality constraints, and it leaves us to consider $\mathcal{O}(m)$ rewiring candidates. Again using $\hat{\Delta}$ instead of Δ, we can now solve QREM in time $\mathcal{O}(r\kappa\ell g m + h)$, where ℓ is the maximum number of targets k such that (i,j,k) is q-permissible, g is the complexity of evaluating θ, and h is the complexity of determining the initial set Q of q-permissible rewirings.

Algorithm 6.3: Heuristic greedy REM with Gamine

Input: Graph $G = (V, E)$, transition matrix $\mathbf{P}$, costs $\mathbf{c}$, budget r
Output: Set of r rewirings X of shape (i, j, k)

$X \leftarrow \emptyset$
for $i \in \mathbb{N}_{\leqslant r}$ **do**
  Compute $\mathbf{1}^\top \mathbf{F}$ and $\mathbf{Fc}$ ▷ $\mathcal{O}(\kappa m)$
  Compute $\mathbf{1}^\top \mathbf{Fu}$ for all $(i, j) \in E$ ▷ $\mathcal{O}(m)$
  $K \leftarrow \{k \mid \mathbf{Fc}[k] \in \{(\Delta^+ + 2) \text{ smallest } \mathbf{Fc}\text{-values}\}\}$ ▷ $\mathcal{O}(n)$
  Compute $\boldsymbol{v}^\top \mathbf{Fc}$ for $j \in V$ and $k \in K$ ▷ $\mathcal{O}(\Delta^+ n)$
  1-REM ( )
return X

Function 1-REM ( )
  $\widehat{\Delta}, i', j', k' \leftarrow 0, \bot, \bot, \bot$
  for $(i, j) \in E$ **do** ▷ $\mathcal{O}(m)$
    for $k_{ij} \in K \setminus (\Gamma^+(i) \cup \{i\})$ **do** ▷ $\mathcal{O}(\Delta^+)$
      $\mathbf{u} \leftarrow \mathbf{P}[i, j]\boldsymbol{e}_i$
      $\boldsymbol{v} \leftarrow \boldsymbol{e}_j - \boldsymbol{e}_{k_{ij}}$
      $\widehat{\Delta}_{ijk} \leftarrow (\mathbf{1}^\top \mathbf{Fu})(\boldsymbol{v}^\top \mathbf{Fc})$ ▷ $\mathcal{O}(1)$
      if $\widehat{\Delta}_{ijk} > \widehat{\Delta}$ **then**
        $\widehat{\Delta}, i', j', k' \leftarrow \widehat{\Delta}_{ijk}, i, j, k_{ij}$
  $E \leftarrow (E \setminus \{(i', j')\}) \cup \{(i', k')\}$
  $\mathbf{P}[i', k'] \leftarrow \mathbf{P}[i', j']$
  $\mathbf{P}[i', j'] \leftarrow 0$
  $X \leftarrow X \cup \{(i', j', k')\}$

Thus, we have arrived at our efficient greedy algorithm, which we call Gamine (Greedy approximate minimization of exposure), stated as Algorithm 6.3 (REM) and as Algorithm 6.4 (QREM). Gamine solves REM in time $\mathcal{O}(r\kappa\Delta^+(n + m))$ and QREM in time $\mathcal{O}(r\kappa\ell gm + h)$, and its inner loops are parallelizable. In realistic recommendation settings, the graph G is d-out-regular for $d \in \mathcal{O}(1)$, and hence, $\Delta^+ \in \mathcal{O}(1)$ and $m = dn \in \mathcal{O}(n)$. Further, for QREM, we can expect that θ is evaluable in time $\mathcal{O}(1)$, and that only the $\mathcal{O}(1)$ nodes most relevant for i will be considered as potential rewiring targets of any edge (i, j), such that $\ell \in \mathcal{O}(1)$ and $h \in \mathcal{O}(m) = \mathcal{O}(n)$. As we can also safely work with a number of power iterations $\kappa \in \mathcal{O}(1)$ (Section 6.D.1), in realistic settings, Gamine solves both REM and QREM in time $\mathcal{O}(rn)$, which, for $r \in \mathcal{O}(1)$, is linear in the order of the input graph G.

6.5 Related Work

Our work methodically relates to research on *graph edits* with distinct goals, such as improving robustness, reducing distances, or increasing centralities [48, 191,

Algorithm 6.4: Heuristic greedy QREM with GAMINE

Input: Graph $G = (V, E)$, transition matrix $\mathbf{P}$, costs $\mathbf{c}$, budget r, relevance matrix $\mathbf{R}$, relevance function θ, quality threshold q
Output: Set of r rewirings X of shape (i, j, k)

$X \leftarrow \emptyset$
$Q \leftarrow \{(i,j,k) \mid (i,j) \in E, (i,k) \notin E, (i,j,k) \text{ is q-permissible}\}$ ▷ $\mathcal{O}(h)$
for $i \in \mathbb{N}_{\leqslant r}$ **do**
  Compute $\mathbf{1}^\top \mathbf{F}$ and $\mathbf{F}\mathbf{c}$ ▷ $\mathcal{O}(\kappa m)$
  Compute $\mathbf{1}^\top \mathbf{F}\mathbf{u}$ for all $(i,j) \in E$ ▷ $\mathcal{O}(m)$
  for $(i,j) \in E$ **do** ▷ $\mathcal{O}(m)$
    $k_{ij} \leftarrow \operatorname{argmin}\{\boldsymbol{e}_k^\top \mathbf{F}\mathbf{c} \mid (i,j,k) \in Q\}$ ▷ $\mathcal{O}(\ell)$
    Compute $\boldsymbol{v}^\top \mathbf{F}\mathbf{c}$ for j and k_{ij} ▷ $\mathcal{O}(1)$
  1-REM ( )
return X
Function 1-REM ( )
  $\hat{\Delta}, i', j', k' \leftarrow 0, \perp, \perp, \perp$
  for $(i,j) \in E$ **do** ▷ $\mathcal{O}(m)$
    $\mathbf{u} \leftarrow \mathbf{P}[i,j]\boldsymbol{e}_i$
    $\boldsymbol{v} \leftarrow \boldsymbol{e}_j - \boldsymbol{e}_{k_{ij}}$
    $\hat{\Delta}_{ijk} \leftarrow (\mathbf{1}^\top \mathbf{F}\mathbf{u})(\boldsymbol{v}^\top \mathbf{F}\mathbf{c})$ ▷ $\mathcal{O}(1)$
    if $\hat{\Delta}_{ijk} > \hat{\Delta}$ **then**
      $\hat{\Delta}, i', j', k' \leftarrow \hat{\Delta}_{ijk}, i, j, k_{ij}$
  $E \leftarrow (E \setminus \{(i',j')\}) \cup \{(i',k')\}$
  $\mathbf{P}[i',k'] \leftarrow \mathbf{P}[i',j']$
  $\mathbf{P}[i',j'] \leftarrow 0$
  $X \leftarrow X \cup \{(i',j',k')\}$
  Update Q ▷ $\mathcal{O}(\ell g)$

212], and research leveraging *random walks* to rank nodes [188, 206, 264] or recommend links [213, 276]. The agenda of our work, however, aligns most closely with the literature studying harm reduction, bias mitigation, and conflict prevention in graphs. Here, the large body of research on shaping opinions or mitigating negative phenomena in *graphs of user interactions* (especially on social media) [2, 8, 74, 97, 98, 103, 192, 256, 259, 282, 283] pursues goals *similar* to ours in graphs capturing *different* digital contexts.

As our research is motivated by recent work demonstrating how recommendations on digital media platforms like YouTube can fuel radicalization [184, 224], the comparatively scarce literature on harm reduction in *graphs of content items* is even more closely related. Our contribution is inspired by Fabbri et al. [87], who study how edge rewiring can reduce *radicalization pathways* in recommendation

graphs. Fabbri et al. [87] encode harmfulness in binary node labels, model benign nodes as absorbing states, and aim to minimize the *maximum segregation* of any node, defined as the largest expected length of a random walk starting at a harmful node before it visits a benign node. In contrast, we encode harmfulness in more nuanced, real-valued node attributes, use an absorbing Markov chain model that more naturally reflects user behavior (where benign nodes are also transient states), and aim to minimize the expected *total exposure* to harm in random walks starting at any node. Thus, our work not only eliminates several limitations of the work by Fabbri et al. [87], but it also provides a different perspective on harm mitigation in recommendation graphs.

While Fabbri et al. [87], like us, consider *recommendation graphs*, Haddadan et al. [114] focus on polarization mitigation in the *Web graph* via *edge insertion*. They define the so-called *polarized bubble radius*, defined as the expected length of a random walk starting at a given page to a page of a different *opinion*, and ask how edge *insertions* can minimize the *sum* of the polarized bubble radii of *parochial* nodes, i.e., nodes with a relatively large polarized bubble radius. The setting from Haddadan et al. [114] was recently reconsidered by Adriaens, Wang, and Gionis [3], who tackle the minimization objective directly instead of using the maximization objective as a proxy, providing approximation bounds as well as speed-ups for the standard greedy method. Both Fabbri et al. [87] and the works on edge insertion employ random-walk objectives that—unlike our exposure function—do not depend on random walks starting at *all* nodes. In our experiments, we compare with the algorithm introduced by Fabbri et al. [87], which we call MMS. We refrain from comparing with edge insertion strategies because they consider a different graph edit operation and are already outperformed by MMS.

6.6 Experiments

Having developed Gamine as a theoretically well-founded algorithm to address REM and QREM in Section 6.4, we would now like to ensure that it also works well in practice. Therefore, in our experiments, we seek to answer five questions:

Q1 Impact of modeling and parameter choices. How do our modeling and parameter choices impact the performance of Gamine?

Q2 Performance comparisons. How well does Gamine reduce the exposure to harm compared to existing methods and baselines?

Q3 Empirical scalability. How does Gamine scale *in practice*?

Q4 Data complexity. What features of the input data make reducing exposure to harm easier resp. harder on different datasets?

Table 6.2: Overview of the datasets used in our experiments. For each dataset of graphs with identical basic statistics, we report the number of graphs N in the collection, the regular out-degree d, the number of nodes n, and the number of edges m, as well as the range of the expected exposure $f(G)/n$ under our various cost functions, edge wirings, and edge transition probabilities.

Dataset	N	d	n	m	$f(G)/n$
SU, SH	$2 \cdot 4 \cdot 36$	5	10^i for $i \in \{2,3,4,5\}$	5×10^i	[1.291, 15.231]
YT-100K	$3 \cdot 6$	5	40 415	202 075	[0.900, 8.475]
		10		404 150	[0.938, 8.701]
		20		808 300	[0.989, 9.444]
YT-10K	$3 \cdot 6$	5	150 572	752 860	[0.806, 5.785]
		10		1 505 720	[0.883, 7.576]
		20		3 011 440	[0.949, 8.987]
NF-JAN06	$3 \cdot 6$	5	11 931	59 655	[4.217, 9.533]
		10		119 310	[4.248, 9.567]
		20		238 620	[4.217, 9.533]
NF-Cov19	$3 \cdot 6$	5	57 447	287 235	[4.609, 11.068]
		10		574 470	[4.392, 10.769]
		20		1 148 940	[4.329, 10.741]
NF-ALL	$3 \cdot 6$	5	93 455	467 275	[5.565, 11.896]
		10		934 550	[5.315, 11.660]
		20		1 869 100	[5.138, 11.517]

Q5 General guidelines. What general guidelines for reducing exposure to harm in recommendation graphs under budget constraints can we derive from our results?

6.6.1 Setup

Before answering the questions laid out above, we give an overview of our setup, especially regarding our datasets and our competitors.

Datasets

To achieve our experimental goals, we work with both synthetic and real-world data, as summarized in Table 6.2. Below, we briefly introduce these datasets. Further details, including on data generation and preprocessing, are provided in Section 6.C.

Synthetic Data. As our synthetic data, we generate a total of 288 synthetic graphs of four different sizes using two different edge placement models and various

parametrizations. The first model, SU, chooses out-edges *uniformly* at random, similar to a directed Erdős-Rényi model [85]. In contrast, the second model, SH, chooses edges *preferentially* to favor small distances between the costs of the source and the target node, implementing the concept of *homophily* [190]. We use these graphs primarily to analyze the behavior of our objective function, and to understand the impact of using $\hat{\Delta}$ instead of Δ to select the greedily optimal rewiring.

Real-world Data. We work with real-world data from two domains, video recommendations (YT) and news feeds (NF). For our *video application*, we use the YouTube data by Ribeiro et al. [184, 224]. This data was originally collected to study radicalization phenomena on the YouTube platform, and it contains identifiers and "Up Next"-recommendations for videos from selected channels categorized by the authors to reflect different degrees and directions of radicalization. For our *news application*, we use subsets of the NELA-GT-2021 dataset [111], which contains 1.8 million news articles published in 2021 from 367 outlets, along with veracity labels from Media Bias/Fact Check. Prior versions of both datasets are used in the experiments reported by Fabbri et al. [87], whose method will be our main competitor.

Parametrizations. To comprehensively assess the effect of modeling assumptions regarding the input graph and its associated random-walk process on our measure of exposure as well as on the performance of Gamine and its competitors, we experiment with a variety of parametrizations expressing these assumptions. For all datasets, we distinguish three random-walk absorption probabilities $\alpha \in \{0.05, 0.1, 0.2\}$ and two probability shapes $\chi \in \{\mathbf{U}, \mathbf{S}\}$ over the out-edges of each node (**U**niform and **S**kewed). For our synthetic datasets, we further experiment with three fractions of latently harmful nodes $\beta \in \{0.3, 0.5, 0.7\}$ and two cost functions $c \in \{c_B, c_R\}$, one binary and one real-valued. Lastly, for our real-world datasets, we distinguish three regular out-degrees $d \in \{5, 10, 20\}$, five quality thresholds $q \in \{0.0, 0.5, 0.9, 0.95, 0.99\}$, as well as four cost functions, two binary (c_{B1}, c_{B2}) and two real-valued (c_{R1}, c_{R2}). We derive these cost functions from the YouTube channels resp. news outlets and the labels provided with the original datasets, as detailed below.

Cost Functions for YouTube Datasets (YT). Our cost functions for the YT datasets map the channel categories provided with the original YouTube dataset to costs based on different mapping rules. Reflecting the original purpose of the dataset to study radicalization phenomena, these categories are (i) political directions based on a US-American yardstick (left, left-center, center, right-center, and right), (ii) terms used by commentators to characterize right-wing movements ex-

Table 6.3: Number of videos |V|, number of channels |C|, and costs of videos from each channel under our four different cost functions, for each of our YouTube datasets.

Category	YT-100K |V|	YT-10K |V|	YT-100K |C|	YT-10K |C|	c_{B1}	c_{B2}	c_{R1}	c_{R2}
Alt-lite	8 908	31 483	90	106	1.0	1.0	0.8	0.8
Alt-right	658	4 685	41	71	1.0	1.0	1.0	1.0
Incel	44	322	13	28	0.0	1.0	0.4	0.6
IDW	6 720	19 146	79	85	1.0	1.0	0.6	0.2
MGTOW	431	6 863	49	71	0.0	1.0	0.4	0.6
MRA	167	1 522	17	27	0.0	1.0	0.2	0.4
NONE	2 477	6 590	21	30	0.0	0.0	0.0	0.0
PUA	4 414	14 209	87	119	0.0	1.0	0.2	0.4
center	2 503	10 117	16	16	0.0	0.0	0.0	0.0
left	4 433	14 705	16	16	0.0	0.0	0.0	0.0
left-center	8 587	33 617	24	24	0.0	0.0	0.0	0.0
right	370	3 253	6	6	0.0	0.0	0.0	0.0
right-center	703	4 060	5	5	0.0	0.0	0.0	0.0

hibiting different degrees of extremism (IDW [Intellectual Dark Web], Alt-lite, Alt-right), (iii) names of various anti-feminist communities (Incel [Involuntary Celibate], MGTOW [Men Get Their Own Way], MRA [Men's Rights Association], PUA [Pickup Artists]), and (iv) content not falling into the previous categories (NONE).

In Table 6.3, we provide the number of videos and the number of channels per category in our YT-100K and YT-10K datasets, as well as the assignment of costs to video channels under our four different cost functions. As a complement, Table 6.4 lists the average expected initial exposure of nodes in each of our YouTube recommendation graphs.

Table 6.4: Average expected initial exposure $f(G)/n$ of nodes in G for the YouTube datasets.

	d	α	χ	c_{B1}	c_{B2}	c_{R1}	c_{R2}
YT-100K	5	0.05	**S**	6.318	8.129	4.335	2.518
			U	6.506	8.475	4.486	2.637
		0.10	**S**	3.251	4.245	2.303	1.491
			U	3.357	4.412	2.377	1.530
		0.20	**S**	1.694	2.234	1.245	0.900
			U	1.737	2.297	1.272	0.908
	10	0.05	**S**	6.387	8.316	4.440	2.688
			U	6.605	8.701	4.623	2.842
		0.10	**S**	3.355	4.417	2.395	1.584
			U	3.466	4.590	2.482	1.647
		0.20	**S**	1.750	2.317	1.290	0.938
			U	1.796	2.382	1.324	0.961
	20	0.05	**S**	6.983	9.026	4.880	3.014
			U	7.372	9.444	5.153	3.195
		0.10	**S**	3.606	4.716	2.582	1.722
			U	3.749	4.874	2.687	1.801
		0.20	**S**	1.844	2.429	1.359	0.989
			U	1.894	2.486	1.398	1.022
YT-10K	5	0.05	**S**	4.198	5.597	2.992	1.926
			U	4.173	5.785	3.040	2.066
		0.10	**S**	2.330	3.217	1.734	1.244
			U	2.401	3.369	1.801	1.315
		0.20	**S**	1.309	1.854	1.019	0.806
			U	1.353	1.922	1.053	0.834
	10	0.05	**S**	5.093	6.729	3.641	2.377
			U	5.712	7.576	4.101	2.704
		0.10	**S**	2.729	3.743	2.027	1.448
			U	2.958	4.063	2.203	1.584
		0.20	**S**	1.450	2.046	1.125	0.883
			U	1.525	2.152	1.185	0.932
	20	0.05	**S**	6.185	8.186	4.405	2.820
			U	6.741	8.987	4.819	3.094
		0.10	**S**	3.120	4.285	2.310	1.625
			U	3.306	4.569	2.460	1.741
		0.20	**S**	1.577	2.228	1.222	0.949
			U	1.638	2.324	1.275	0.996

Table 6.5: Number of news outlets |C| and number of videos in each of our NELA-GT-2021 datasets for each unique combination of assigned costs.

c_{B1}	c_{B2}	c_{R1}	c_{R2}	$\lvert C\rvert$	NF-Jan6 $\lvert V\rvert$	NF-Cov19 $\lvert V\rvert$	NF-All $\lvert V\rvert$
0	0	0.0	0.0	4	147	631	993
0	0	0.2	0.0	69	3 188	17 794	29 021
0	0	0.4	0.0	18	1 463	3 986	6 549
0	1	0.6	0.0	1	0	0	0
0	1	0.6	1.0	40	3 920	19 398	32 303
1	1	0.6	0.5	73	1 533	8 804	15 549
1	1	0.8	0.5	112	1 497	6 436	14 337
1	1	1.0	0.5	24	237	983	1 987

Cost Functions for NELA-GT Datasets (NF). The costs we assign to nodes in our NF datasets are based on the *Media Bias/Fact Check scores* and the *questionable source* and *conspiracy/pseudoscience* flags of news outlets provided with the original dataset. As the number of news outlets covered by this dataset is too large to detail their individual cost assignments, here, we instead state how we transform the labels provided with the dataset into cost assignments under our four different cost functions. We define

$$c_{B1} = \begin{cases} 1 & \text{if } \texttt{questionable\_source} = 1 \\ & \text{or } \texttt{conspiracy\_pseudoscience} = 1 \\ 0 & \text{otherwise,} \end{cases} \tag{6.50}$$

$$c_{B2} = \begin{cases} 1 & \text{if } \texttt{factuality} \leqslant 2 \\ 0 & \text{otherwise,} \end{cases} \tag{6.51}$$

$$c_{R1} = 1 - \frac{\texttt{factuality}}{5}\text{, and} \tag{6.52}$$

$$c_{R2} = 1 - \frac{\texttt{label}}{2}\text{,} \tag{6.53}$$

where `typewritten variables` are the names of the corresponding columns in the original data, $\texttt{questionable\_source} \in \{0, 1\}$, $\texttt{conspiracy\_pseudoscience} \in \{0, 1\}$, $\texttt{factuality} \in \{0, 1, 2, 3, 4, 5\}$, and $\texttt{label} \in \{0, 1, 2\}$ is an aggregate label combining the other scores. An overview of the resulting cost assignments in each of our NF datasets is given in Table 6.5. In Table 6.6, we additionally state the expected total exposure as well as the total segregation and the maximum segregation from Fabbri et al. [87] for all NF datasets with $\alpha = 0.05$ and $\chi = \mathbf{U}$.

Table 6.6: Initial expected total exposure to harm $f(G)$, as well as total initial segregation and maximum initial segregation from Fabbri et al. [87], on each of our NF graphs with $\alpha = 0.05$, $\chi = \mathbf{U}$, and $\mu = 1.0$ for segregation computations under c_{R1} and c_{R2}.

	d	c	f(G)	Total Segregation	Maximum Segregation
NF-Jan06	5	c_{B1}	51 940	6 896	19.99
		c_{B2}	118 506	25 038	19.99
		c_{R1}	104 948	259	2.04
		c_{R2}	92 536	8 555	19.99
	10	c_{B1}	50 682	5 342	5.79
		c_{B2}	114 141	18 151	19.99
		c_{R1}	103 482	250	1.47
		c_{R2}	88 800	6 172	10.33
	20	c_{B1}	50 319	4 889	4.05
		c_{B2}	113 738	16 194	6.50
		c_{R1}	103 445	246	1.21
		c_{R2}	88 579	5 782	2.93
NF-Cov19	5	c_{B1}	264 781	54 699	19.99
		c_{B2}	635 867	201 579	19.99
		c_{R1}	520 763	1 075	2.19
		c_{R2}	503 477	81 903	19.99
	10	c_{B1}	252 294	44 313	19.99
		c_{B2}	618 645	157 105	19.99
		c_{R1}	513 982	1 038	1.86
		c_{R2}	492 498	65 199	19.99
	20	c_{B1}	248 708	38 597	19.99
		c_{B2}	617 014	134 998	19.99
		c_{R1}	513 113	1 020	1.48
		c_{R2}	492 660	56 696	19.05
NF-All	5	c_{B1}	520 092	128 388	19.99
		c_{B2}	1 111 742	383 640	19.99
		c_{R1}	890 383	3 013	19.99
		c_{R2}	851 697	136 356	19.99
	10	c_{B1}	496 667	103 825	19.99
		c_{B2}	1 089 690	307 444	19.99
		c_{R1}	880 536	2 666	15.10
		c_{R2}	841 357	111 298	19.99
	20	c_{B1}	480 186	88 983	19.99
		c_{B2}	1 076 287	260 998	19.99
		c_{R1}	874 785	2 187	6.90
		c_{R2}	836 194	96 895	19.99

Algorithms

We compare Gamine, our algorithm for REM and QREM, with four baselines (BL1–BL4) and the algorithm by Fabbri et al. [87] for minimizing the maximum segregation, which we call MMS. In all experiments with relevance information, we use the normalized discounted cumulative gain (nDCG, defined in Eq. (6.6)), which is also used by MMS, as a relevance function θ, and consider the 100 most relevant nodes as potential rewiring targets. Note that we can compute the nDCG in time $\mathcal{O}(\Delta^+)$, assuming that lookup operations for matrix elements and relevance ranks take constant time. As $\Delta^+ \in \mathcal{O}(1)$ for $\mathcal{O}(1)$-out-regular graphs, this entails that in our experiments, we can evaluate $\theta \equiv$ nDCG in constant time—in line with our theoretical analysis.

As MMS can only handle binary costs, we transform nonbinary costs c into binary costs c' by thresholding to ensure $c_i \geqslant \mu \Leftrightarrow c'_i = 1$ for some rounding threshold $\mu \in (0, 1]$. In our experiments, we use the binarization thresholds 1.0, 0.6, and 0.4, which are chosen to ensure that they yield *different* binarized costs given our original real-valued costs as inputs. Note, however, that the resulting problem instances differ from the original instances, and as such, it is hardly possible to fairly compare Gamine with MMS in the real-valued setting. Hence, we focus our performance comparisons on the binary setting. Since MMS requires access to relevance information, we further restrict our comparisons with MMS to data where relevance information is available.

Our baselines BL1–BL4 are ablations of Gamine, such that outperforming them establishes that each component of our approach is beneficial. We order these baselines by the competition strength we expect from them, from no competition (BL1) to strong competition (BL4). Intuitively, BL1 does not consider our objective at all, BL2 is a heuristic focusing on the τ component of our objective, BL3 is a heuristic focusing on the σ component of our objective, and BL4 eliminates the iterative element of our approach. The first three baselines run in r rounds. In each round, BL1 randomly selects a permissible rewiring via rejection sampling. BL2 selects the rewiring (i, j, k) with the node j maximizing $\boldsymbol{e}_j^\top \mathbf{F}\boldsymbol{c}$ as its old target, the node i with $j \in \Gamma^+(i)$ maximizing $\mathbf{1}^\top \mathbf{F}\boldsymbol{e}_i$ as its source, and the available node k minimizing $\boldsymbol{e}_k^\top \mathbf{F}\boldsymbol{c}$ as its new target. In contrast, BL3 selects the rewiring (i, j, k) with the node i maximizing $\mathbf{1}^\top \mathbf{F}\boldsymbol{e}_i$ as its source, the node j with $j \in \Gamma^+(i)$ maximizing $\boldsymbol{e}_j^\top \mathbf{F}\boldsymbol{c}$ as its old target, and the available node k minimizing $\boldsymbol{e}_k^\top \mathbf{F}\boldsymbol{c}$ as its new target. The fourth baseline, BL4, runs in one round only, and it selects the r rewirings with the largest value of $\widehat{\Delta}$, while ensuring each edge is rewired at most once.

Implementation and Reproducibility

All algorithms, including Gamine, the baselines, and MMS, are implemented in Python 3.10. We run our experiments on a 2.9 GHz 6-Core Intel Core i9 with 32 GB RAM, report wall-clock time, and make all code, datasets, and results publicly available.[2]

6.6.2 Results

Q1 Impact of Modeling Choices

To understand the impact of a particular modeling choice on the performance of Gamine and its competitors, we analyze groups of experimental settings that vary only the parameter of interest while keeping the other parameters constant, focusing on the YT-100k datasets. We primarily report the evolution of the ratio

$$\frac{f(G_r)}{f(G)} = \frac{f(G) - f_\Delta(G, G_r)}{f(G)}\,, \tag{6.54}$$

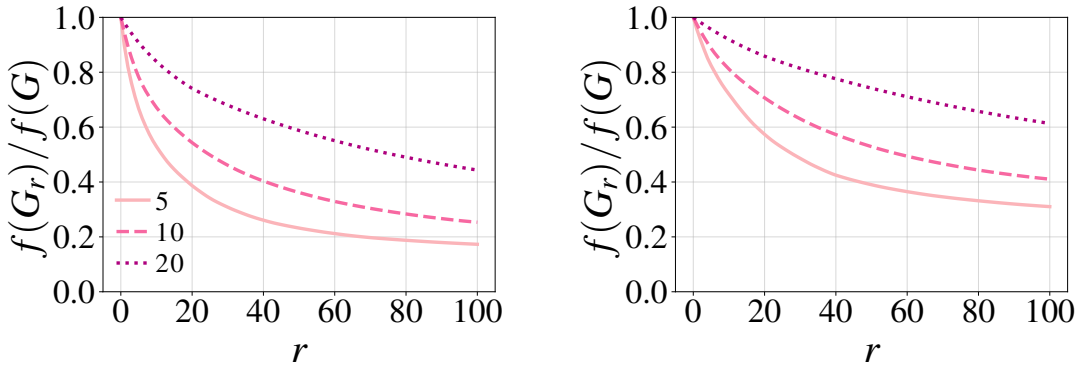
which indicates what fraction of the initial expected total exposure is left after r rewirings, and hence is comparable across REM instances with different starting values. Overall, we observe that Gamine robustly reduces the expected total exposure to harm, and that it changes its behavior predictably under parameter variations.

Impact of Regular Out-Degree d. Since the impact of individual edges on the objective function decreases as d increases, for a given budget r, we expect Gamine to reduce our objective more strongly for smaller values of d. This is exactly what we find, as illustrated in Fig. 6.4, and the pattern persists across absorption probabilities α, probability shapes χ, quality thresholds q, and cost functions c.

Impact of Absorption Probability α and Out-Edge Probability Distribution Shape χ. For smaller random-walk absorption probabilities α, we obtain longer random walks and thus higher exposure to harmful content, and for $\chi = \mathbf{S}$, some edges are traversed particularly often. Thus, given a constant budget r, we expect Gamine to achieve a *larger* decrease of f for smaller α, and an initially *faster* decrease on graphs with skewed out-edge probability distributions. Again, this is what we find, as depicted in Fig. 6.5.

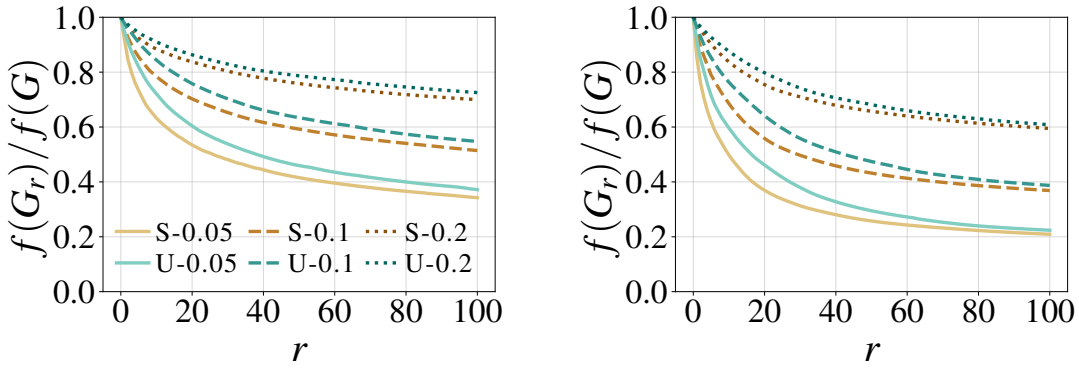
Impact of Cost Function c. As the binary cost function c_{B1} (used also in [87] on a prior version of the data from [224]) labels only videos from Alt-Right, Alt-Lite, and Intellectual Dark Web (IDW) channels as harmful ($c_{B1} = 1$) and all other

[2] 10.5281/zenodo.7936816



(a) Absorption probability $\alpha = 0.05$ **(b)** Absorption probability $\alpha = 0.1$

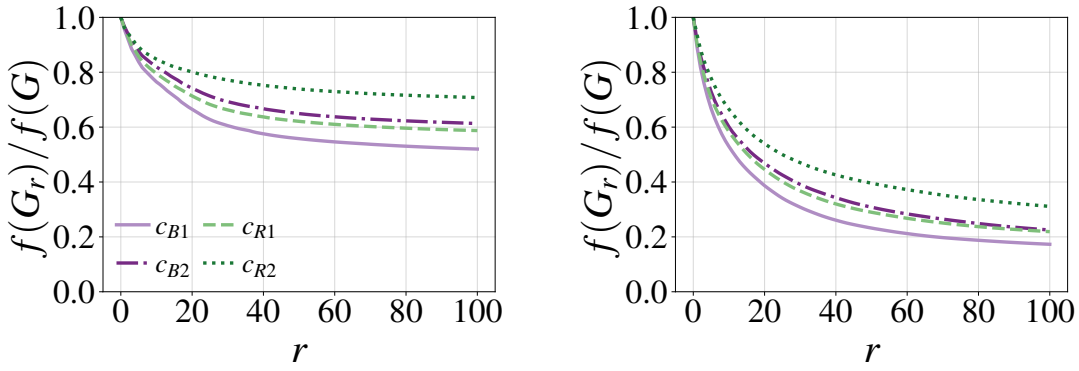
Figure 6.4: Performance of GAMINE for out-regular degrees $d \in \{5, 10, 20\}$, run with $q = 0.0$ under c_{B1} on YT-100K with $\chi = \mathbf{U}$. The smaller the out-degree, the stronger GAMINE.



(a) Cost function c_{R1} **(b)** Cost function c_{B1}

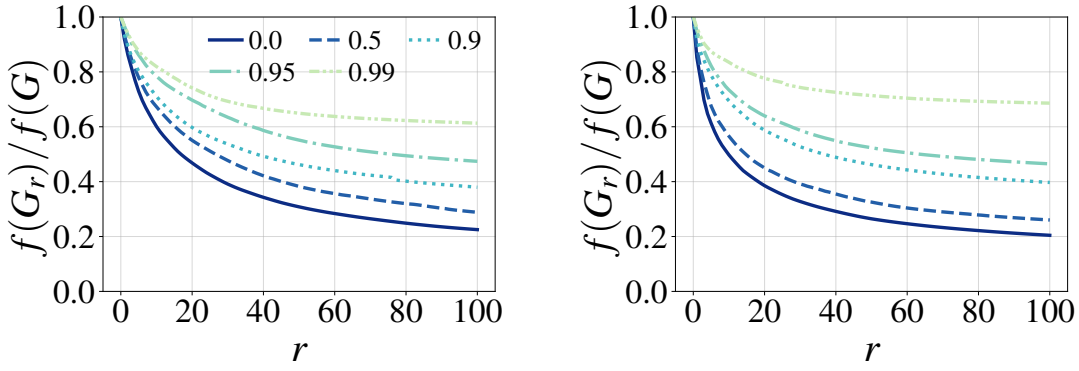
Figure 6.5: Performance of GAMINE for absorption probabilities $\alpha \in \{0.05, 0.1, 0.2\}$ and out-edge probability distribution shapes $\chi \in \{\mathbf{U}, \mathbf{S}\}$, run with $q = 0.5$ on YT-100K with $d = 5$. The smaller the absorption probability, the stronger the performance of GAMINE, and our objective function drops faster when the out-edge probability distribution is skewed.

videos as benign ($c_{B1} = 0$), whereas all other cost functions also assign positive cost to videos from anti-feminist channels (Incel, MGTOW, MRA, and PUA) (cf. Table 6.3), we expect GAMINE to perform strongest under c_{B1}. As exemplified in Fig. 6.6, this is exactly what we observe, and the pattern persists across regular out-degrees d, absorption probabilities α, distribution shapes χ, and quality thresholds q. Interestingly, we also consistently observe that GAMINE is roughly equally strong under the binary cost function c_{B2} and the real-valued cost function c_{R1}, and weakest under the real-valued cost function c_{R2}. As c_{R1} and c_{R2} differ only in how they assign costs to videos from IDW and anti-feminist channels, with c_{R1} (c_{R2}) placing IDW to the *right* (*left*) of anti-feminist channels, this means that reducing the exposure to harm is *harder* when we consider the IDW more benign than anti-feminist communities, *even though* there are more IDW videos in YT-100K than videos from all anti-feminist communities combined.



(a) Quality threshold $q = 0.99$ **(b)** Quality threshold $q = 0.0$

Figure 6.6: Performance of Gamine under cost functions $c \in \{c_{B1}, c_{B2}, c_{R1}, c_{R2}\}$, run on YT-100k with $d = 5$, $\alpha = 0.05$, and $\chi = \mathbf{U}$. Gamine is strongest under the binary cost function c_{B1}, weakest under the real-valued cost function c_{R2}, and roughly equally strong under the binary cB2 and the real-valued cR1.



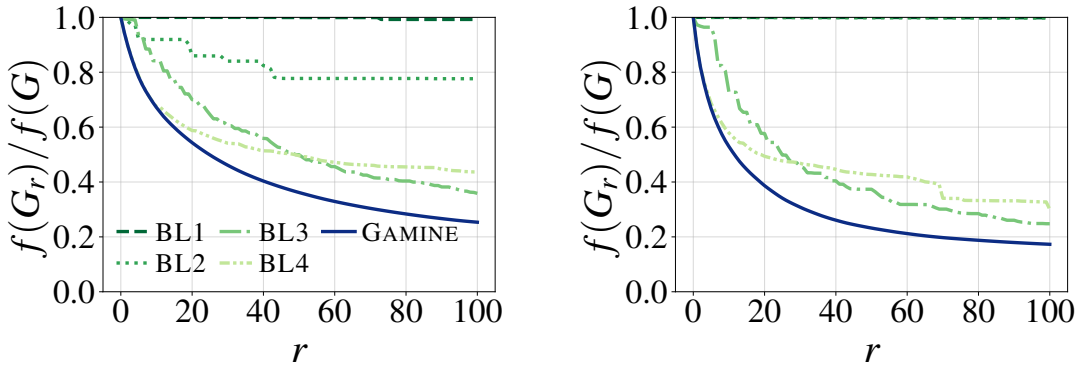
(a) Probability shape $\chi = \mathbf{U}$ **(b)** Probability shape $\chi = \mathbf{S}$

Figure 6.7: Performance of Gamine for quality thresholds $q \in \{0.0, 0.5, 0.9, 0.95, 0.99\}$ as measured by c_{B2}, run on YT-100k with $d = 5$ and $\alpha = 0.05$. Gamine can ensure $q = 0.5$ with little loss in performance, and it can reduce our objective considerably even under a strict $q = 0.95$.

Impact of Quality Threshold q. The higher the quality threshold q, the more constrained our rewiring options. Thus, under a given budget r, we expect Gamine to reduce our objective more strongly for smaller q. As illustrated in Fig. 6.7, our experiments confirm this intuition, and the effect is more pronounced if the out-edge probability distribution is skewed. We further observe that Gamine can guarantee $q = 0.5$ with little performance impact, and it can strongly reduce the exposure to harm even under a strict $q = 0.95$: With just 100 edge rewirings, it reduces the expected total exposure to harm by 50%, while ensuring that its recommendations are at most 5% less relevant than the original recommendations.

Q2 Performance Comparisons

Having ensured that Gamine robustly and predictably reduces the total exposure across the entire spectrum of modeling choices, we now compare it with its



(a) Regular out-degree $d = 10$ **(b)** Regular out-degree $d = 5$

Figure 6.8: Performance of GAMINE with $q = 0.0$, compared with the four baselines BL1, BL2, BL3, and BL4 under c_{B1}, run on YT-100K with $\alpha = 0.05$ and $\chi = \mathbf{U}$. As BL4 is roundless, we apply its rewirings in decreasing order of Δ to depict its performance as a function of r. GAMINE outcompetes all baselines, but BL3 and BL4 also show strong performance.

competitors. Overall, we find that GAMINE offers more reliable performance and achieves stronger harm reduction than its contenders.

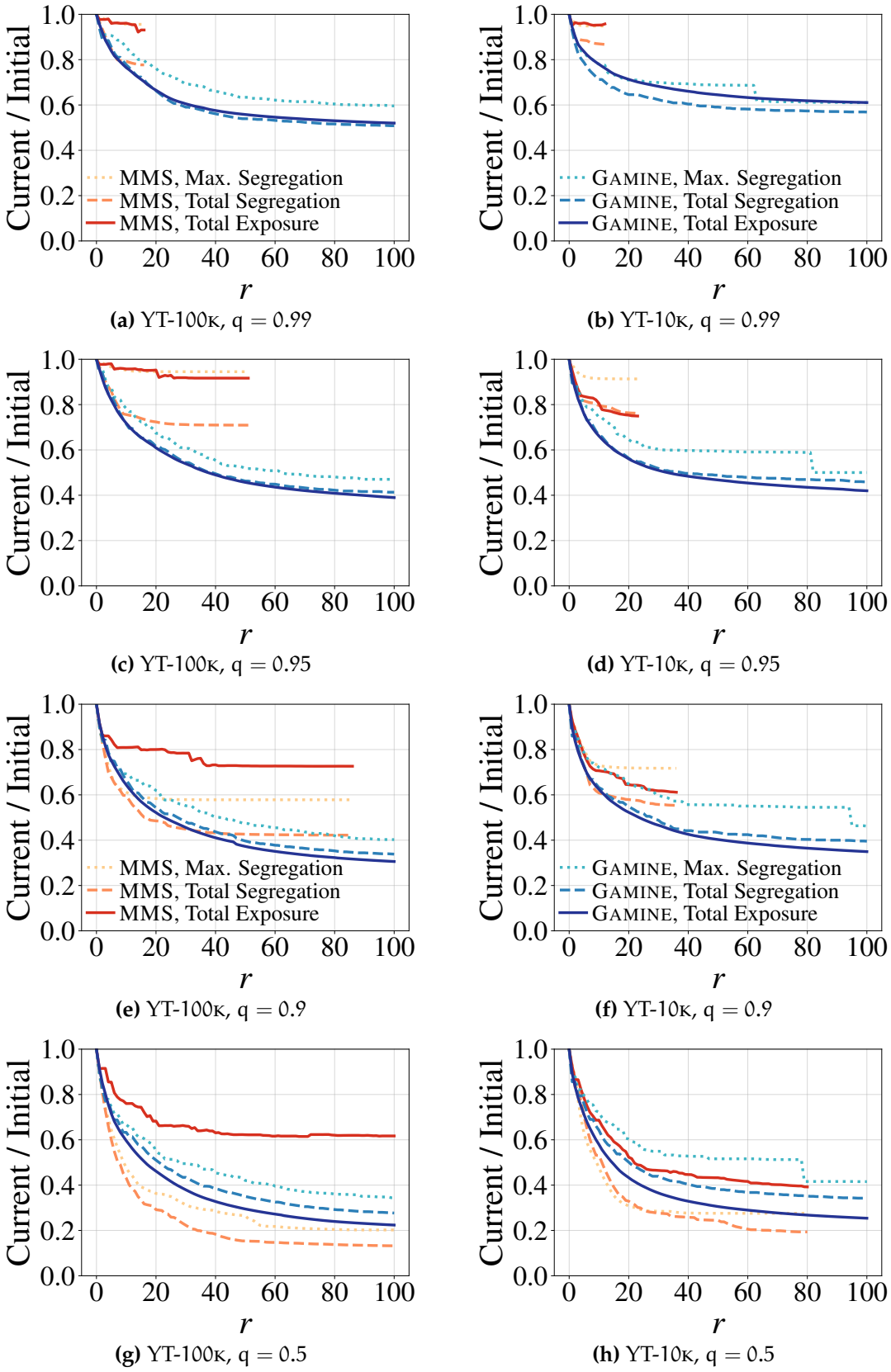
COMPARISON WITH BASELINES BL1–BL4. First, we compare GAMINE with our four baselines, each representing a different ablation of our algorithm. As depicted in Fig. 6.8, the general pattern we observe matches our performance expectations (from weak performance of BL1 to strong performance of BL4), but we are struck by the strong performance of BL3 (selecting based on σ), especially in contrast to the weak performance of BL2 (selecting based on τ). This suggests that whereas the most *exposed* node does not necessarily have a highly visited node as an in-neighbor, the most *visited* node tends to have a highly exposed node as an out-neighbor. In other words, for some highly *prominent* videos, the YouTube algorithm problematically appears to recommend highly *harm-inducing* content to watch next. Despite the competitive performance of BL3 and BL4, GAMINE consistently outperforms these baselines, too, and unlike the baselines, it *smoothly* reduces the exposure function. This lends additional support to our reliance on $\sigma\tau$ (rewiring a highly visited i away from a highly exposed j) as an *iteratively* evaluated heuristic.

COMPARISON WITH MMS. Having established that all components of GAMINE are needed to achieve its performance, we now compare our algorithm with MMS, the method proposed by Fabbri et al. [87]. To this end, on our real-world datasets, we run both GAMINE and MMS using their respective objective functions, i.e., the expected total exposure to harm of random walks starting at any node (*total exposure*, GAMINE) and the maximum expected number of random-walk steps from a harmful node to a benign node (*maximum segregation*, MMS). To ensure fair com-

parisons, in the following, we report the performance of Gamine and MMS under the objectives of *both* algorithms as well as the *total segregation*, which sums the segregation scores of all harmful nodes.

YouTube Datasets (YT). For the YT datasets, as illustrated in Fig. 6.9, we find that under strict quality control ($q \in \{0.9, 0.95, 0.99\}$), Gamine outperforms MMS on *all* objectives, and MMS stops early as it can no longer reduce its objective function. For $q = 0.5$, MMS outperforms Gamine on the segregation-based objectives, but Gamine still outperforms MMS on our exposure-based objective, sometimes at twice the margin (Fig. 6.9g). Further, while Gamine delivers consistent and predictable performance that is strong on exposure-based and segregation-based objectives, we observe much less consistency in the performance of MMS. For example, it is counterintuitive that MMS identifies 100 rewirings on the smaller YT-100k data but stops early on the larger YT-10k data. Moreover, MMS delivers the results shown in Fig. 6.9 under c_{B1}, but it cannot decrease its objective at all on the same data under c_{B2}, which differs from c_{B1} only in that it also assigns harm to anti-feminist content (cf. Table 6.3). We attribute this brittleness to the reliance on the *maximum*-based *segregation* objective, which, by design, is less robust than our *sum*-based *exposure* objective.

NELA-GT Datasets (NF). While on the YT datasets, 100 rewirings identified by Gamine reduce the expected total exposure to harm by 50% while guaranteeing recommendations still 95% as relevant as the original recommendations (cf. Fig. 6.7), the reduction we achieve on the NF datasets is more moderate. Our best result here, depicted in Fig. 6.10, is a reduction of the expected total exposure to harm by about 30%, again under a 95% quality guarantee. Notably, changing the quality threshold q has a smaller impact on the NF than on the YT datasets, and sometimes it has no performance impact at all. In fact, for the NF-Jan06 graphs involved in Fig. 6.10, $q = 0.95 \equiv 0.9 \equiv 0.5$ (hence, we only draw the line for $q = 0.95$). This indicates that unlike on the YT datasets, on the NF datasets, Gamine is actually affected by the restriction of rewirings to the 100 most relevant candidates, which we implement for all real-world datasets (cf. Section 6.6.1). This is likely a consequence of the different criteria used to (re)construct the recommendation graphs and the relevance scores from the original datasets, and it underscores the importance of data preprocessing decisions also in the context of method evaluation.



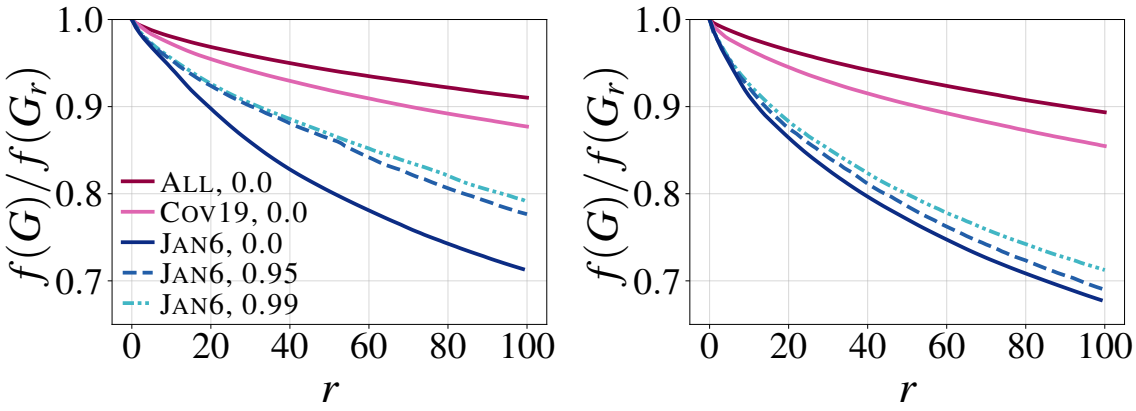
(a) YT-100K, q = 0.99 **(b)** YT-10K, q = 0.99

(c) YT-100K, q = 0.95 **(d)** YT-10K, q = 0.95

(e) YT-100K, q = 0.9 **(f)** YT-10K, q = 0.9

(g) YT-100K, q = 0.5 **(h)** YT-10K, q = 0.5

Figure 6.9: Performance of GAMINE and MMS when measured under c_{B1} by the maximum segregation or the total segregation from Fabbri et al. [87], or by the total exposure as defined in Eq. (6.3), run on YT-100K (left) and YT-10K (right) with $d = 5$, $\alpha = 0.05$, and $\chi = \mathbf{u}$. For all but $q = 0.5$, GAMINE outperforms MMS on *all* objectives, and MMS stops early because it can no longer reduce the maximum segregation.



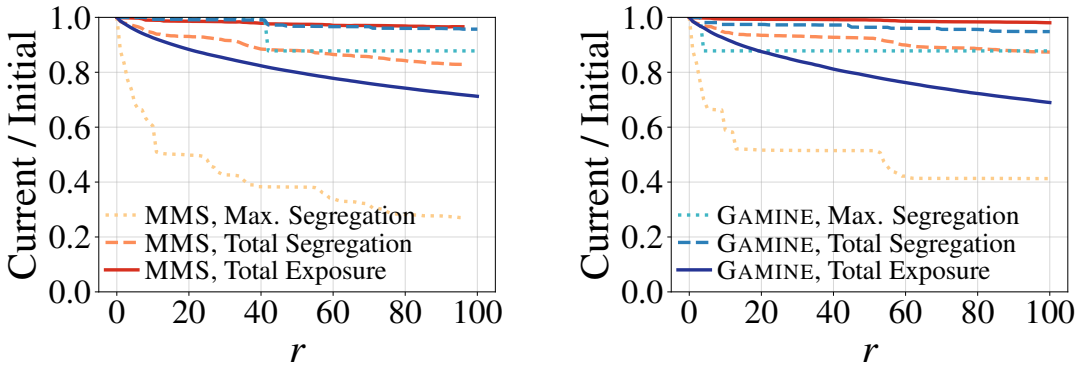
(a) Cost function c_{R2}

(b) Cost function c_{B1}

Figure 6.10: Performance of Gamine on our NF datasets with $d = 5$, $\alpha = 0.05$, and $\chi = \mathbf{U}$. Varying q on NF-Jan06 has a much smaller performance impact than what we observed on YT-100k, and the relative reduction in the expected total exposure after 100 rewirings is at the level of what we achieve after 10 rewirings on YT-100k (cf. Fig. 6.7).

As illustrated in Fig. 6.11, on the NF-Jan06 dataset under c_{B1}, Gamine still outperforms MMS on our exposure objective, but MMS achieves a much stronger relative reduction of its segregation objective. However, MMS counterintuitively reduces its objective function more strongly under a *stricter* quality threshold—a behavior we never observe with Gamine under our exposure objective. As given the same recommendation sequence $\mathbf{r}_i$ at node i, a rewiring (i, j, k) that is q-permissible under $q = 0.99$ is also q-permissible under $q = 0.5$, this suggests that MMS is highly dependent on its trajectory and sometimes requires greedily suboptimal choices to obtain the best possible result after r rewirings.

Moreover, the promising performance we observe for MMS on NF-Jan06 under c_{B1} does not carry over to NF-Cov19 and NF-All, or even to other cost functions on NF-Jan06: On NF-Cov19 and NF-All under c_{B1}, and on NF-Jan06 under c_{B2} or c_{R2} with binarization threshold $\mu = 1.0$, MMS cannot reduce its segregation objective at all, *even though* the starting value of the maximum segregation is exactly the same as for NF-Jan06 under c_{B1} (cf. Table 6.6). On NF-Jan06 under c_{R2} with binarization threshold $\mu = 1.0$, MMS stops after four rewirings with a reduction of 25%. However, the maximum segregation is already minuscule from the start—a result of the interplay between our cost assignment under c_{R2}, our binarization threshold, and our definition of edges from the original data. Thus, overall, our experiments on the NF data confirm our impression from the YT data that MMS is less robust than Gamine.



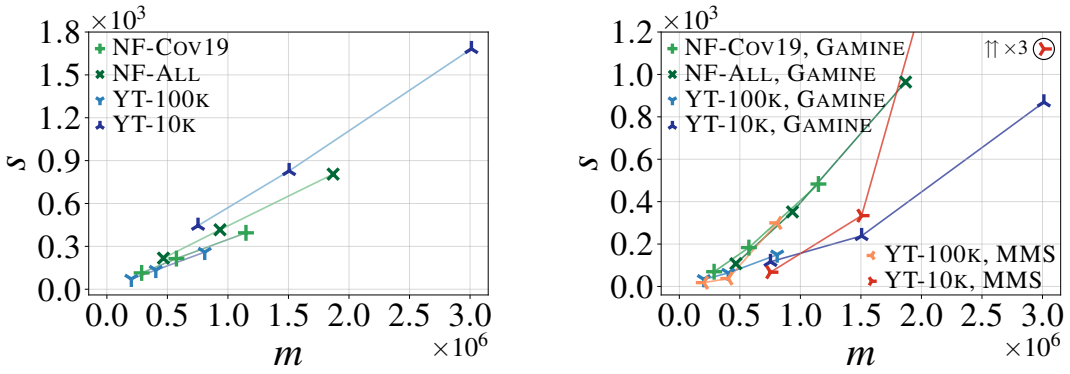
(a) Quality threshold $q = 0.99$ **(b)** Quality threshold $q = 0.5$

Figure 6.11: Performance of GAMINE and MMS when measured under c_{B1} by the maximum segregation or the total segregation from Fabbri et al. [87], or by the total exposure as defined in Eq. (6.3), run on NF-JAN06 with $d = 5$, $\alpha = 0.05$, and $\chi = \mathbf{U}$. GAMINE outperforms MMS on the exposure objective, but MMS reduces its segregation objective more strongly. Counterintuitively, MMS achieves a *stronger* reduction of its segregation objective under a *stricter* quality threshold.

Q3 EMPIRICAL SCALABILITY OF GAMINE

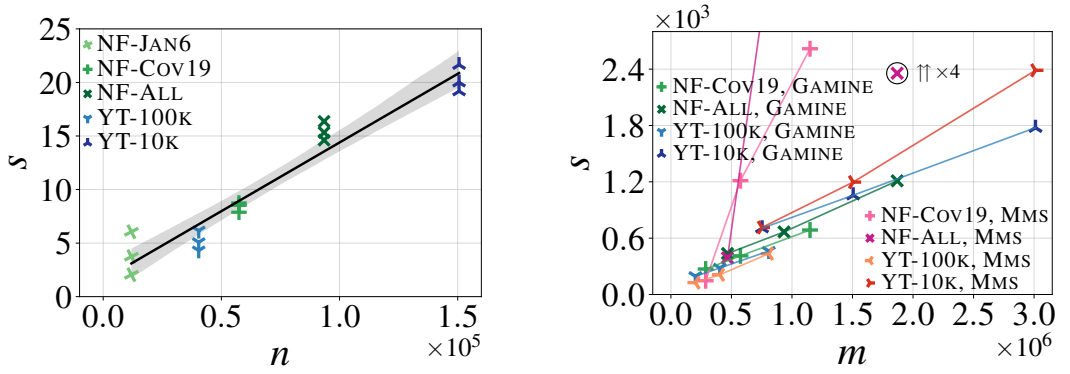
In our previous experiments, we found that GAMINE robustly and reliably reduces the expected total exposure to harm. Now, we seek to ascertain that its practical scaling behavior matches our theoretical predictions, i.e., that under realistic assumptions on the input, GAMINE scales linearly in n and m. We are also interested in comparing GAMINE's scalability to that of MMS. To this end, we measure the time taken to compute a single rewiring and report, in Fig. 6.12, the *average* over ten rewirings for each of our datasets. This corresponds to the time taken by 1-REM in GAMINE and by 1-REWIRING in MMS, which drives the overall scaling behavior of both algorithms. We find that GAMINE scales approximately linearly, whereas MMS scales approximately quadratically (contrasting with the empirical time complexity of $\mathcal{O}(n \log n)$ claimed in [87]). This is because our implementation of MMS follows the original, whose evaluation of the segregation objective takes time $\mathcal{O}(n)$ and is performed $\mathcal{O}(m)$ times. The speed of precomputations depends on the problem variant (REM vs. QREM).

As illustrated in Fig. 6.13, in our experiments, precomputations add approximately linear overhead for GAMINE and somewhat unpredictable, at times quadratic overhead for MMS. The volatility in the overhead of MMS could be due to two factors. First, the relevance precomputations for MMS are slightly more complicated than for GAMINE. Second, one part of MMS's precomputations not present in GAMINE is a matrix inverse approximation via power iteration. This computation is quadratic in the number of *harmful* nodes, as MMS considers only these nodes as transient states.



(a) Gamine on REM **(b)** Gamine and MMS on QREM

Figure 6.12: Scaling of Gamine and MMS on our four largest real-world datasets, under c_{B1} with $\alpha = 0.05$, $\chi = \mathfrak{U}$, and $q = 0.0$ (REM) resp. 0.99 (QREM). We report the seconds s to compute a single rewiring as a function of $m = dn$ (MMS does not identify any rewirings on NF-Cov19 and NF-All). Gamine scales more favorably than MMS.



(a) Gamine on REM **(b)** Gamine and MMS on QREM

Figure 6.13: Empirical scaling of precomputations for Gamine and MMS, computed with $\alpha = 0.05$, $\chi = \mathfrak{U}$, c_{B1}, and $q = 0.0$ (REM) resp. 0.99 (QREM). We depict scaling for REM as a function of n, with a linear regression fitted across datasets, and scaling for QREM as a function of m, where we connect the observations stemming from different regular out-degrees of the same dataset for NF-Cov19, NF-All, and YT-10k. Precomputations for REM are much faster than for QREM, and while Gamine scales approximately linearly, MMS scales somewhat unpredictably.

To conclude our scalability investigations, in addition to Gamine's scaling behavior as a function of n and m, for QREM, we would like to understand how the scaling behavior of our method depends on the quality threshold q. To this end, we run Gamine on each of our YT-100k datasets with $q \in \{x/100 \mid 0 < x < 100, x \bmod 5 = 0\} \cup \{0.99\}$. Since increasing q eliminates rewiring candidates, we hope to see the running time decrease as q increases, and we expect a larger acceleration on graphs with higher (regular) out-degrees. As reported in Fig. 6.14, this is precisely what we find—and the dependence on q is particularly small for our sparser YT-100k datasets.

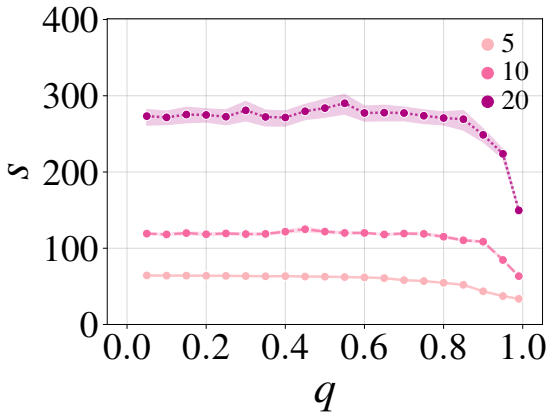
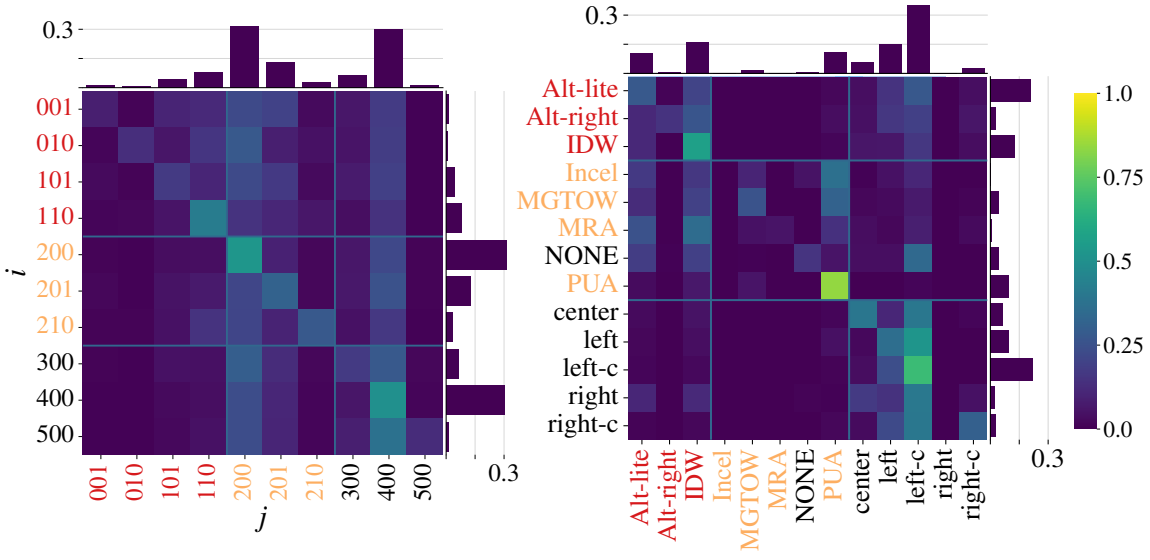


Figure 6.14: Empirical scaling of GAMINE as a function of the quality threshold q, run on YT-100K with $\alpha = 0.05$, $\chi = \mathbf{U}$, and c_{B1}, for $d \in \{5, 10, 20\}$. The larger q, the faster GAMINE.

Q4 DATA COMPLEXITY

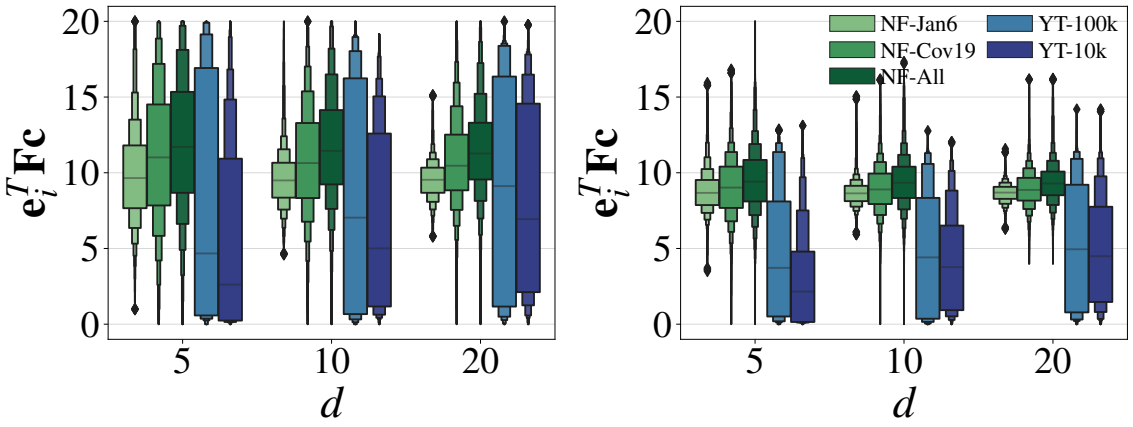
Given that GAMINE strongly reduces the expected total exposure to harm with few rewirings on the YouTube data, as evidenced in Figs. 6.7 to 6.9, one might be surprised that its performance seems cosiderably weaker on the NELA-GT data (cf. Figs. 6.10 and 6.11): While GAMINE still reduces the expected total exposure and outperforms MMS (which struggles to reduce its objective at all on the NF data), the impact of individual rewirings is much smaller than on the YouTube datasets, and the value of the quality threshold q barely makes a difference. This motivates us to investigate how *data complexity* impacts our ability to reduce the expected total exposure to harm via edge rewiring: Could reducing exposure to harm be *intrinsically* harder on NF data than on YT data?

The answer is yes. First, the in-degree distributions of the YT graphs are an order of magnitude more skewed than those of the NF graphs (Section 6.C.2, Fig. 6.23). This is unsurprising given the different origins of their edges (user interactions vs. cosine similarities), but it creates opportunities for high-impact rewirings involving highly prominent nodes in YT graphs (which GAMINE seizes in practice, see below). Second, as depicted in Fig. 6.15, harmful and benign nodes are much more strongly interwoven in the NF data than in the YT data. This means that harmful content is less siloed in the NF graphs, but it also impedes strong reductions of the expected total exposure. Third, as a result of the two previous properties, as illustrated in Fig. 6.16, the initial node exposures are much more concentrated in the NF graphs than in the YT graphs, with a median sometimes twice as large as the median of the identically parametrized YT graphs, and a much higher average exposure (cf. $f(G)/n$ in Table 6.2). Finally, the relevance scores are much more skewed in the YT data than in the NF data (Section 6.C.2,



(a) NF-All **(b)** YT-10k

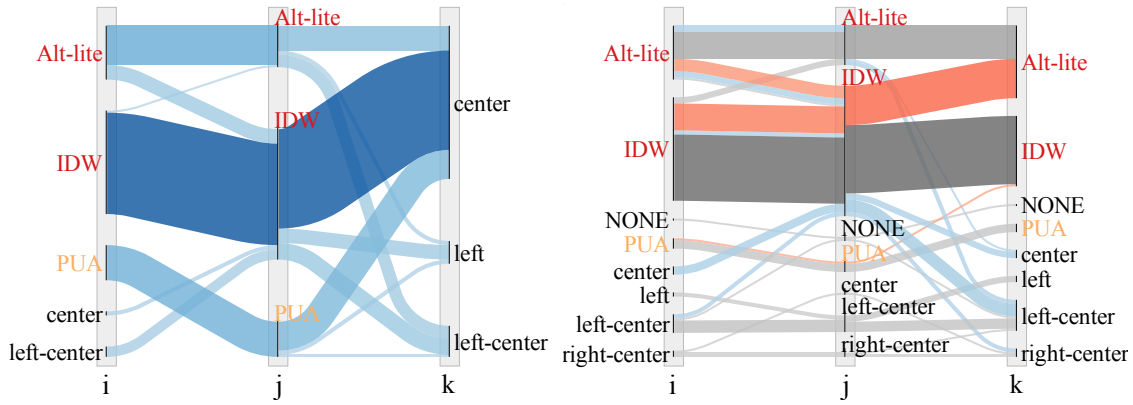
Figure 6.15: Fractions of edges running between news outlet resp. video channel categories for real-world graphs with $d = 5$, with marginals indicating the fraction of sources (right) resp. targets (top) in each category. News outlet categories are denoted as triples (veracity score, conspiracy-pseudoscience flag, questionable-source flag); for video channel categories, {left, right}-center is abbreviated as {left, right}-c; and label colors are coarse indicators of harm. In NF-All, harmful and benign nodes are more interconnected than in YT-10k.



(a) Cost function c_{B2}, probability shape $\chi = S$ **(b)** Cost function c_{R1}, probability shape $\chi = U$

Figure 6.16: Distributions of initial node exposures $e_i^T Fc$ in our real-world datasets, computed with $\alpha = 0.05$. Note that cost functions sharing a name are defined differently for the YT and NF datasets (based on their semantics). The NF datasets generally exhibit more concentrated exposure distributions than the YT datasets and higher median exposures.

Fig. 6.24). Hence, while we are strongly constrained by q on the YT data even when considering only the 100 highest-ranked nodes as potential rewiring targets, we are almost unconstrained in the same setting on the NF data, which explains the comparative irrelevance of q on the NF data. Thus, the performance differences we observe between our datasets are due to intrinsic dataset properties: REM and QREM are simply more complex on the news data than on the video data.



(a) Quality threshold $q = 0.0$

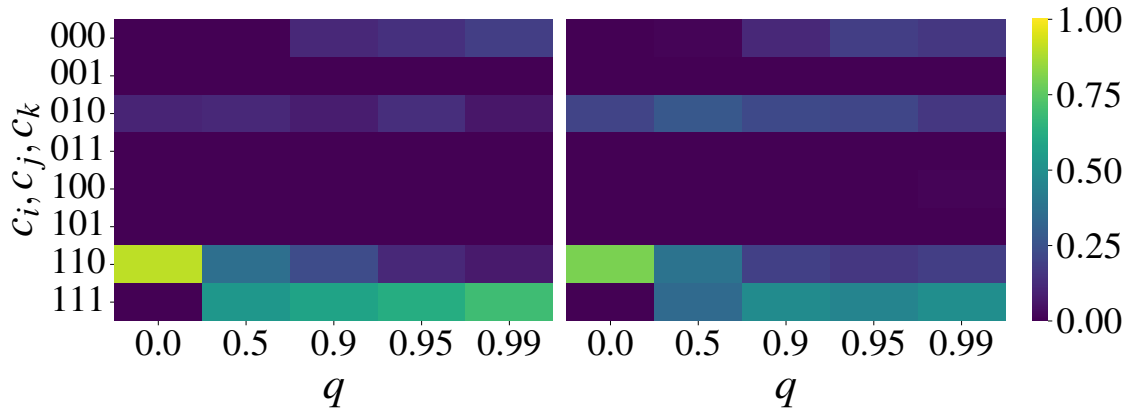
(b) Quality threshold $q = 0.99$

Figure 6.17: Channel class of videos in rewirings (i, j, k) on YT-100K with $d = 5$, $\alpha = 0.05$, and $\chi = \mathbf{U}$, computed using c_{R1}, for different quality thresholds. Rewirings between classes are color-scaled by their count, using blues if $c_{R1}(k) < c_{R1}(j)$, reds if $c_{R1}(k) > c_{R1}(j)$, and grays otherwise. For $q = 0.0$, we only replace costly targets j by less costly targets k, as expected, but for $q = 0.99$, we see many rewirings with $c_{R1}(k) \geqslant c_{R1}(j)$.

Q5 General Guidelines

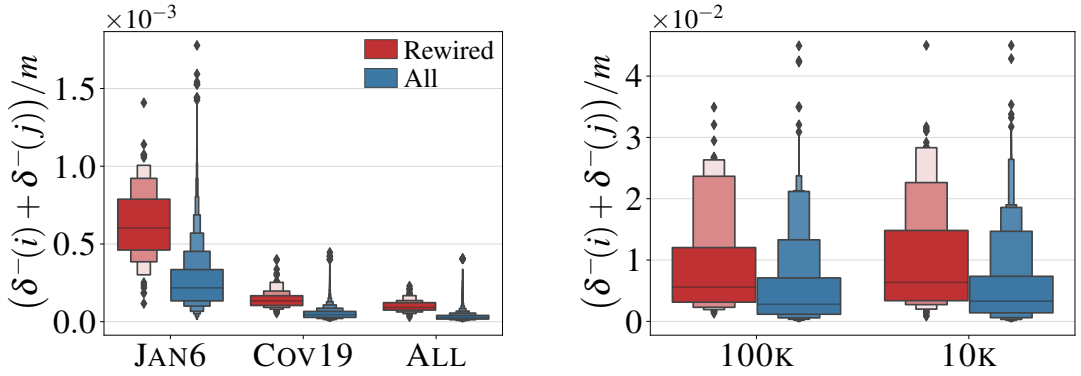
Finally, we would like to abstract the findings from our experiments into general guidelines for reducing exposure to harm in recommendation graphs, especially under quality constraints. To this end, we analyze the metadata associated with our rewirings. In particular, for each set of rewirings (i, j, k) obtained in our experiments, we are interested in the channel resp. news outlet classes involved, as well as in the distributions of cost triples (c_i, c_j, c_k) and in-degree tuples $(\delta^-(i), \delta^-(j))$. As exemplified in Fig. 6.17, while we consistently rewire edges from harmful to benign targets in the quality-unconstrained setting ($q = 0.0$), under strict quality control ($q = 0.99$), we frequently see rewirings from harmful to equally or more harmful targets.

More generally, as illustrated in Fig. 6.18, the larger the threshold q, the more we rewire *among* harmful, resp. benign, nodes ($c_i = c_j = c_k = 1$, resp. 0)—which MMS does not even allow. Furthermore, the edges we rewire typically connect nodes with large in-degrees: As illustrated in Fig. 6.19, the distribution of in-degree sums for edges rewired by Gamine has a higher median than the distribution of in-degree sums for all edges, and the former is generally shifted toward higher in-degree sums as compared to the latter. We conclude that a simplified strategy for reducing exposure to harm under quality constraints is to identify edges that connect high-cost nodes with large in-degrees, and rewire them to the node with the lowest exposure among all nodes meeting the quality constraints.



(a) YT-100K **(b)** YT-10K

Figure 6.18: Mapping the nodes in each rewiring (i, j, k) to their costs (c_i, c_j, c_k), we report the fraction of rewirings in each cost class under c_{B1} and $q \in \{0.0, 0.5, 0.9, 0.95, 0.99\}$, for YT graphs with $d = 5$, $\alpha = 0.05$, and $\chi = \mathbf{U}$. While most intuitively suboptimal classes occur rarely (e.g., 001, 011, 101), under quality constraints, we often rewire *among* harmful nodes.



(a) NF, $q = 0.0$ **(b)** YT, $q = 0.5$

Figure 6.19: Distributions of normalized in-degree sums $\frac{\delta^-(i)+\delta^-(j)}{m}$ for edges rewired by Gamine under c_{B1} vs. all edges, on real-world graphs with $d = 5$, $\alpha = 0.05$, and $\chi = \mathbf{U}$. Gamine tends to rewire edges with larger in-degree sums.

6.7 Conclusion

We studied the problem of reducing the exposure to harmful content in recommendation graphs by edge rewiring. Modeling this exposure via absorbing random walks, we introduced QREM and REM as formalizations of the problem with and without quality constraints on recommendations. We proved that both problems are NP-hard and NP-hard to approximate to within an additive error, but that under mild assumptions, the greedy method provides a $(1 - 1/e)$-approximation for the REM problem. Hence, we introduced Gamine, a greedy algorithm for REM and QREM running in linear time under realistic assumptions on the input, and we confirmed its effectiveness, robustness, and efficiency through ex-

tensive experiments on synthetic data as well as on real-world data from video recommendation and news feed applications.

Our work improves over the state of the art (MMS by Fabbri et al. [87]) in terms of performance, and it eliminates several limitations of prior work. In particular, while Fabbri et al. [87] model benign nodes as *absorbing* states and consider a brittle *max*-objective that is minimized even by highly harm-exposing bipartite recommendation graphs, we model benign nodes as *transient* states and consider a robust *sum*-objective that captures the overall consumption of harmful content by users starting at any node in the graph. Whereas MMS can only handle *binary* node labels, GAMINE natively works with *real-valued* node attributes, which permits a more nuanced encoding of harmfulness. Furthermore, our problem formulation exposes the modeling choices involved in algorithm design for harm reduction, which not only allows us to rigorously assess the impact of these choices on the performance of our algorithm but also enables users to make these choices consciously, flexibly, and transparently.

We see potential for future work in several directions. First, several amendments to our objective are worth exploring. For example, it would be interesting to adapt our objective to mitigate *polarization*, i.e., the separation of content with opposing views, with positions modeled as positive and negative node costs. Moreover, we currently assume that all nodes are equally likely as starting points of random walks, which is unrealistic in many applications. Removing this limitation requires replacing the all-ones vector in our objective by a vector of starting probabilities, which could change the analysis and lead to a different heuristic. Second, like Fabbri et al. [87], we regard the recommendation graph as given, and translating our algorithm to an online setting, especially with regret guarantees, provides intriguing challenges in theory and practice. Finally, we observe that harm reduction in recommendation graphs, in which edge rewirings are the most natural edits, has largely been studied in separation from harm reduction in other graphs representing consumption phenomena, such as user interaction graphs, where other edits (such as edge insertions or edge deletions) are viable alternatives. A framework for optimizing functions under budget constraints that includes edge rewirings, insertions, and deletions could unify these research lines and facilitate future progress.

Appendices

6.A Ethics Statement

In this work, we introduce Gamine, a method to reduce the exposure to harm induced by recommendation algorithms on digital media platforms via edge rewiring, i.e., replacing certain recommendations by others. While removing harm-inducing recommendations constitutes a milder intervention than censoring content directly, it still steers attention away from certain content to other content, which, if pushed to the extreme, can have censorship-like effects. Although in its intended usage, Gamine primarily counteracts the tendency of recommendation algorithms to overexpose harmful content as similar to other harmful content, when fed with a contrived cost function, it could also be used to discriminate against content considered undesirable for problematic reasons (e.g., due to political biases or stereotypes against minorities). However, as the changes to recommendations suggested by Gamine could also be made by amending recommendation algorithms directly, the risk of *intentional* abuse is no greater than that inherent in the recommendation algorithms themselves, and *unintentional* abuse can be prevented by rigorous impact assessments and cost function audits before and during deployment. Thus, we are confident that overall, Gamine can contribute to the health of digital platforms.

6.B Notation

For easy reference, we collect the notation used in this chapter in Table 6.7.

6.C Dataset Details

In this section, we provide more information on the data used in our experiments.

6.C.1 Synthetic Data

Preprocessing

Since the viewports of popular electronic devices typically fit around five recommendations, as our synthetic data, we generate synthetic 5-out-regular graphs. We experiment with four graph sizes using three absorption probabilities, two shapes of probability distributions over out-edges, three fractions of latently harmful nodes, and two cost functions, one binary and one real-valued based on a mixture of two beta distributions (cf. Fig. 6.20), to assign costs to nodes. We state the details on these parameters in Table 6.8. For each of the resulting 144 configurations, we place edges using two different edge-placement models, SU and SH, for a total of 288 graphs. For each node i, SU chooses d distinct nodes $j \neq i$ as tar-

Table 6.7: Most important notation used in this chapter. (Table continued on next page.)

Symbol	Definition	Description
		GRAPH NOTATION
G	$= (V, E)$	Graph
n	$= \lvert V \rvert$	Number of nodes
m	$= \lvert E \rvert$	Number of edges
$\delta^-(i)$	$= \lvert\{j \mid (j,i) \in E\}\rvert$	In-degree of node i
$\Gamma^+(i)$	$= \{j \mid (i,j) \in E\}$	Set of out-neighbors of node i
$\delta^+(i)$	$= \lvert\Gamma^+(i)\rvert$	Out-degree of node i
d		Regular out-degree of an out-regular graph
Δ^+	$= \max\{\delta^+(i) \mid i \in V\}$	Maximum out-degree
S	$= \{i \in V \mid \boldsymbol{e}_i^\top \boldsymbol{F}\boldsymbol{c} = 0\}$	Set of safe nodes
U	$= \{i \in V \mid \boldsymbol{e}_i^\top \boldsymbol{F}\boldsymbol{c} > 0\}$	Set of unsafe nodes
Λ^+	$= \max\{\delta^+(i) \mid i \in U\}$	Maximum out-degree of an unsafe node
		MATRIX NOTATION
$\boldsymbol{M}[i,j]$		Element in row i, column j of $\boldsymbol{M}$
$\boldsymbol{M}[i,:]$		Row i of $\boldsymbol{M}$
$\boldsymbol{M}[:,j]$		Column j of $\boldsymbol{M}$
$\boldsymbol{e}_i$		i-th unit vector
$\boldsymbol{1}$		All-ones vector
$\boldsymbol{I}$		Identity matrix
$\lVert\boldsymbol{M}\rVert_\infty$	$= \max_i \sum_{j=0}^{n} \boldsymbol{M}[i,j]$	Infinity norm

gets for its edges uniformly at random, whereas SH chooses d distinct targets by sampling each j with probability $\frac{1-|c(i)-c(j)|}{\sum_{j\in V}(1-|c(i)-c(j)|)}$. Hence, in SH, edges are drawn preferentially between nodes of similar costs, implementing homophily, whereas in SU, edges are drawn uniformly at random.

STATISTICS

In Fig. 6.21, we show the distributions of initial exposures for nodes in our SU and SH graphs. For SU, we observe that the range of initial exposures is small and the cost function choice barely makes a difference, which is expected as edges are placed uniformly at random. In contrast, for SH, we observe the maximum range of initial exposures under c_B, as homophilous sampling under binary costs effectively splits the graph into two components consisting of harmful and benign nodes, respectively. Under c_R, we still observe a range of initial exposures that is twice to thrice as large as in SU graphs, and the probability shape $\chi \in \{\mathbf{U}, \mathbf{S}\}$ strongly influences the distribution of initial exposures. These are again effects of homophilous sampling.

Table 6.7: Most important notation used in this chapter. (Table continued from previous page.)

Symbol	Definition	Description
		Notation for REM and QREM
(i, j, k)		Rewiring replacing $(i, j) \in E$ by $(i, k) \notin E$ with $p_{ik} = p_{ij}$, cf. Table 6.1
$r \in \mathbb{N}$		Rewiring budget
$\alpha \in (0, 1]$		Random-walk absorption probability
$p_{ij} \in (0, 1 - \alpha]$		Probability of traversing (i, j) from i
$\mathbf{P} \in [0, 1 - \alpha]^{n \times n}$		Random-walk transition matrix
$\mathbf{F} = \sum_{i=0}^{\infty} \mathbf{P}^i = (\mathbf{I} - \mathbf{P})^{-1}$		Fundamental matrix
c		Cost function with range $[0, 1]$
$c_i \in [0, 1]$		Cost associated with node i
$\mathbf{c} \in [0, 1]^n$		Vector of node costs
$\kappa \in \mathbb{N}$		Number of power iterations
$\chi \in \{\mathbf{U}, \mathbf{S}\}$		Shape of probability distribution over the out-edges of a node
		Notation for QREM Only
$\mathbf{R} \in \mathbb{R}_{\geqslant 0}^{n \times n}$		Relevance matrix
θ		Relevance function with range $[0, 1]$
$q \in [0, 1]$		Quality threshold
$\mathbf{r}_i \in V^{\delta^+(i)}$		Relevance-ordered targets of out-edges of i
$\mathrm{idx}_i(j)$		Relevance rank of node j for node i
$T_{\delta^+}(i) = \{j \mid \mathrm{idx}_i(j) \leqslant \delta^+(i)\}$		Set of the $\delta^+(i)$ nodes most relevant for node i
$\mathrm{DCG} = \sum_{j \in \Gamma^+(i)} \frac{\mathbf{R}[i,j]}{\log_2(1 + \mathrm{idx}_i(j))}$		Discounted Cumulative Gain
$\mathrm{iDCG} = \sum_{j \in T_{\delta^+}(i)} \frac{\mathbf{R}[i,j]}{\log_2(1 + \mathrm{idx}_i(j))}$		Ideal Discounted Cumulative Gain
$\mathrm{nDCG} = \frac{\mathrm{DCG}(i)}{\mathrm{iDCG}(i)}$		Normalized Discounted Cumulative Gain
		Notation Related to the Exposure Function f and its Analysis
$f(G) = \mathbf{1}^\top \mathbf{F} \mathbf{c}$		Exposure function (minimization objective)
$f_\Delta(G, G_r) = f(G) - f(G_r)$		Reduction-in-exposure function (equivalent maximization objective)
G', $\mathbf{P}'$, $\mathbf{F}'$		Graph G, transition matrix $\mathbf{P}$, and fundamental matrix $\mathbf{F}$, as updated by rewiring (i, j, k), cf. Table 6.1
$\mathbf{u} = p_{ij} \boldsymbol{e}_i$		Vector capturing the source i of a rewiring (i, j, k) and the traversal probability of (i, j)
$\boldsymbol{v} = \boldsymbol{e}_j - \boldsymbol{e}_k$		Vector capturing the old target j and the new target k of a rewiring (i, j, k)
$\sigma = \mathbf{1}^\top \mathbf{F} \mathbf{u}$		p_{ij}-scaled i-th column sum
$\tau = \boldsymbol{v}^\top \mathbf{F} \mathbf{c}$		$\mathbf{c}$-scaled sum of differences between the j-th row sum and the k-th row sum
$\rho = 1 + \boldsymbol{v}^\top \mathbf{F} \mathbf{u}$		Normalization factor ensuring that $\mathbf{F}'\mathbf{1} = \mathbf{F}\mathbf{1}$
$\Delta = f_\Delta(G, G') = \sigma\tau/\rho$		Reduction of f obtained by a single rewiring (i, j, k)
$\widehat{\Delta} = \Delta\rho = \sigma\tau$		Heuristic for Δ

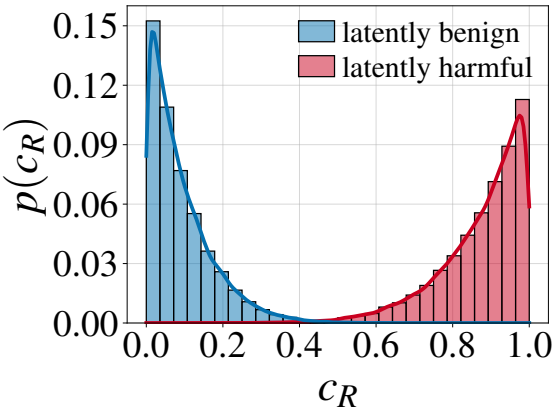
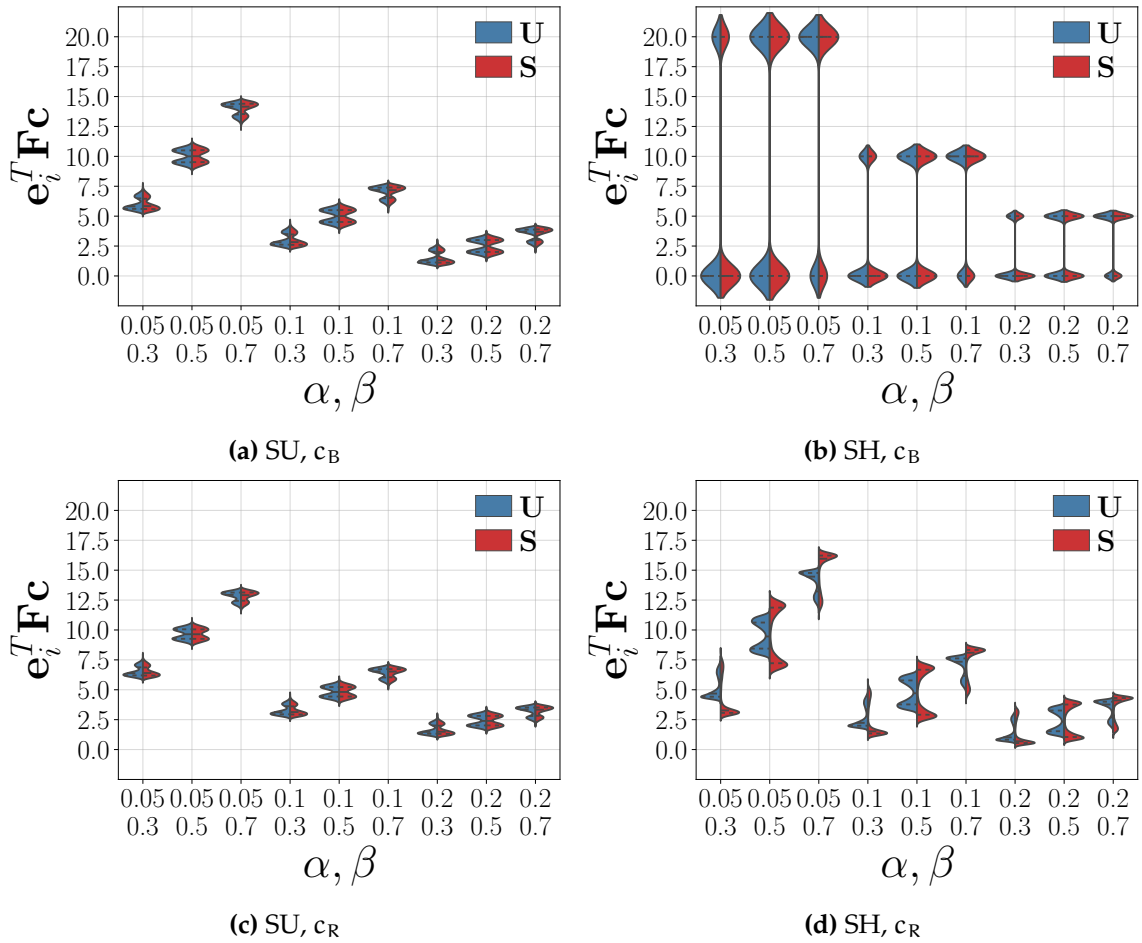


Figure 6.20: Histograms and kernel density estimates of 50 000 draws from the beta distributions used to assign costs to nodes in SU and SH under c_R. The cost of latently benign nodes is drawn from $\mathrm{Beta}(1, 10)$, whereas the cost of latently harmful nodes is drawn from $\mathrm{Beta}(7, 1)$.

Table 6.8: Parameters used to generate SU and SH graphs.

Parameter	Meaning
$d = 5$	Regular out-degree
$n \in \{10^i \mid i \in \{2, 3, 4, 5\}\}$	Number of nodes in G
$\alpha \in \{0.05, 0.1, 0.2\}$	Random-walk absorption probability
$\chi \in \{\mathbf{U}, \mathbf{S}\}$	Probability shape over a node's out-edges: – **U**niform $\left(p_{ij} = \frac{1-\alpha}{d}\right.$ for all $\left.(i, j) \in E\right)$; – **S**kewed $\left(\frac{1-\alpha}{d} \cdot p\right.$ for $\left.p \in \langle 0.35, 0.25, 0.20, 0.15, 0.05\rangle\right)$
$\beta \in \{0.3, 0.5, 0.7\}$	Fraction of latently harmful nodes – SU: fraction of nodes i with $c_i = 1$ – SH: fraction of nodes drawn from the beta distribution with parameters $\alpha = 7$, $\beta = 1$
$c \in \{c_B, c_R\}$	Cost functions $- c_B(i) = \begin{cases} 1 & \text{if } i \text{ is latently harmful} \\ 0 & \text{otherwise} \end{cases}$ $- c_R(i) = \begin{cases} \mathrm{Beta}(7, 1) & \text{if } i \text{ is latently harmful} \\ \mathrm{Beta}(1, 10) & \text{otherwise} \end{cases}$



(a) SU, c_B **(b)** SH, c_B

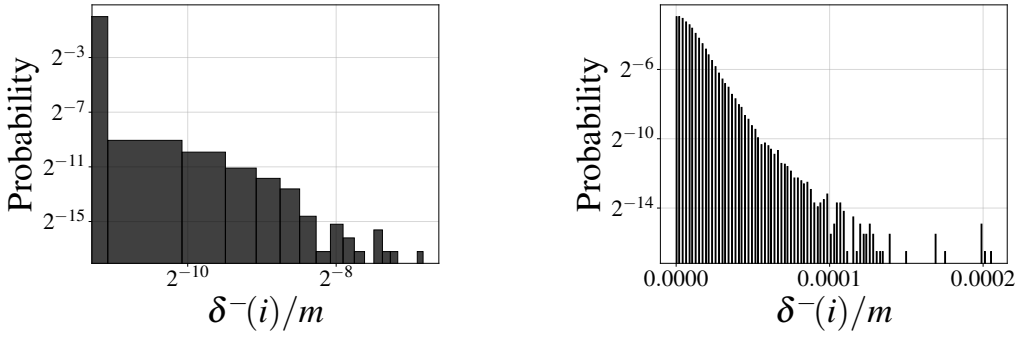
(c) SU, c_R **(d)** SH, c_R

Figure 6.21: Distributions of initial exposures $\boldsymbol{e}_i^T \mathbf{F} \mathbf{c}$ for nodes in SU and SH graphs with 100 000 nodes, for all combinations of absorption probabilities $\alpha \in \{0.05, 0.1, 0.2\}$ (x-axis label, first row), fractions of latently harmful nodes $\beta \in \{0.3, 0.5, 0.7\}$ (x-axis label, second row), and out-edge probability distributions $\chi \in \{\mathbf{U}, \mathbf{S}\}$ (color). While the choice of the cost function barely makes a difference under random edge placements (SU, left), it has a tremendous impact under homophilous edge placements (SH, right).

6.C.2 Real-World Data

Preprocessing

YouTube Datasets (YT). For our YouTube datasets, like Fabbri et al. [87], who experiment with a prior (not uniquely identified) version of this dataset, we generate d-regular recommendation graphs for $d \in \{5, 10, 20\}$ that contain only videos with at least 100 000, resp. 10 000, views as nodes (YT-100k, resp. YT-10k). Similar to Fabbri et al. [87] we treat the observed recommendations as implicit feedback interactions, eliminate sinks in the observed recommendation graph, use alternating least squares to generate relevance scores [125], and then take the nodes with the top scores as targets of out-edges in our reconstructed recommendation graphs. We additionally distinguish three absorption probabilities $\alpha \in \{0.05, 0.1, 0.2\}$ and two shapes $\chi \in \{\mathbf{U}, \mathbf{S}\}$ of probability distributions over



(a) YT-10K **(b)** NF-ALL

Figure 6.22: Normalized in-degree distributions of our two largest real-world datasets for $d = 20$. The in-degree distribution of the YT-10K graph is considerably more skewed than the in-degree distribution of the NF-ALL graph.

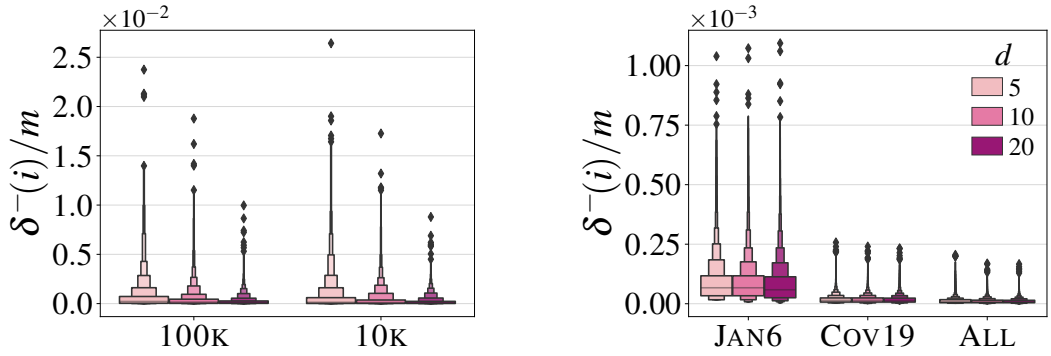
out-edges in our random-walk model, which leaves us with 36 transition matrices from 6 underlying graph structures.

NELA-GT DATASETS (NF). To create our NELA-GT datasets, we restrict ourselves to news items of *at least* 140 *characters*[3] that were published in *January* 2021 by one of the 341 outlets for which all veracity labels are present, and consider the articles containing the authors' *January* 6 *keywords* (NF-JAN06), the articles containing the authors' *COVID-19 keywords* (NF-COV19), and the collection containing *all* articles (NF-ALL). After embedding all news items using the *all-MiniLM-L6-v2* model from the Sentence-Transformers library [223], we compute pairwise cosine similarities $\cos(i, j)$ between all articles from the respective collection, transform these similarities into relevance scores between 0 and 1 via min-max-normalization ($\mathbf{R}[i, j] = \frac{\cos(i,j)+1}{2}$), and take the news items with the highest scores as targets of out-edges in our initial news feed graphs.

STATISTICS

IN-DEGREE DISTRIBUTIONS. As our real-world graphs are d-out-regular by construction, their out-degree distributions are uniform. In contrast, the in-degree distributions of these graphs are highly skewed. In Fig. 6.22, we show the normalized in-degree distributions of our two largest real-world datasets, NF-ALL and YT-10K. Note that in-degrees, at least visually, appear to be *exponentially* distributed in the NF-ALL graph and *power-law* distributed in the YT-10K graph. Further, as illustrated in Fig. 6.23, even when considering only non-zero in-degrees and *all* real-world graphs, the NF graphs appear to be about an order of magnitude less skewed than the YT graphs.

[3] This excludes, for example: "Post was not sent - check your email addresses ! Email check failed , please try again \n Sorry , your @ @ @ @ @ by email ." (136 characters)



(a) YT **(b)** NF

Figure 6.23: Distributions of *nonzero* in-degrees $\delta^-(i)$, normalized by the number of edges m, for each of our real-world datasets. The NF datasets are an order of magnitude more balanced than the YT datasets, and within one collection, smaller graphs have more skewed in-degree distributions.

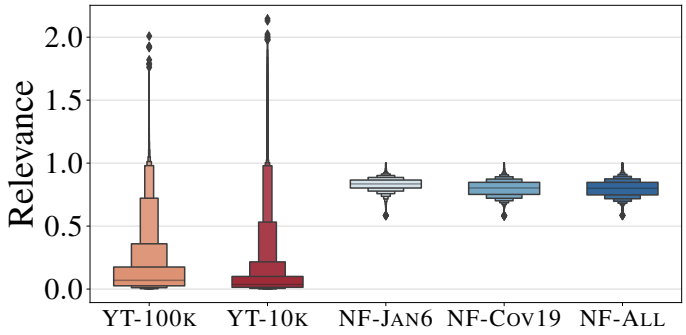
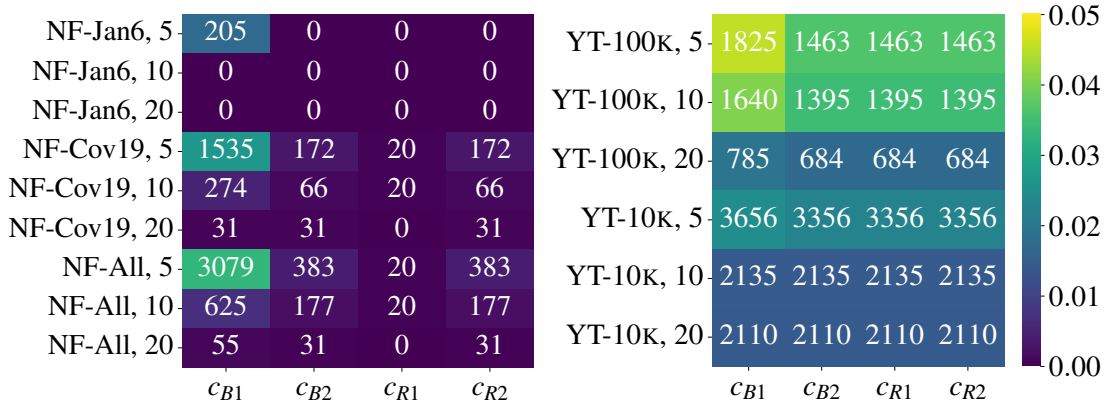


Figure 6.24: Distributions of relevance scores $\mathbf{R}[i, j]$ for each source node $i \in V$ and the top 100 nodes j considered as potential rewiring targets, for each of our real-world datasets in the QREM setting. The relevance distributions are much more skewed in the YT datasets than in the NF datasets.

Relevance-Score Distributions. Complementing the discussion in the main paper, in Fig. 6.24, we show the relevance score distribution for each of our real-world datasets. Note that the nDCG used in our QREM experiments does not expect relevance scores to lie within a particular range, and that the relevance scores obtained by preprocessing YT are not strictly bounded, but they are guaranteed to *mostly* lie between 0 and 1, whereas the relevance scores obtained by preprocessing NF are directly cosine similarities, rescaled to lie between 0 and 1. As the relevance scores of the NF datasets are very concentrated, the quality threshold q hardly constrains our rewiring options when solving QREM on NF graphs.

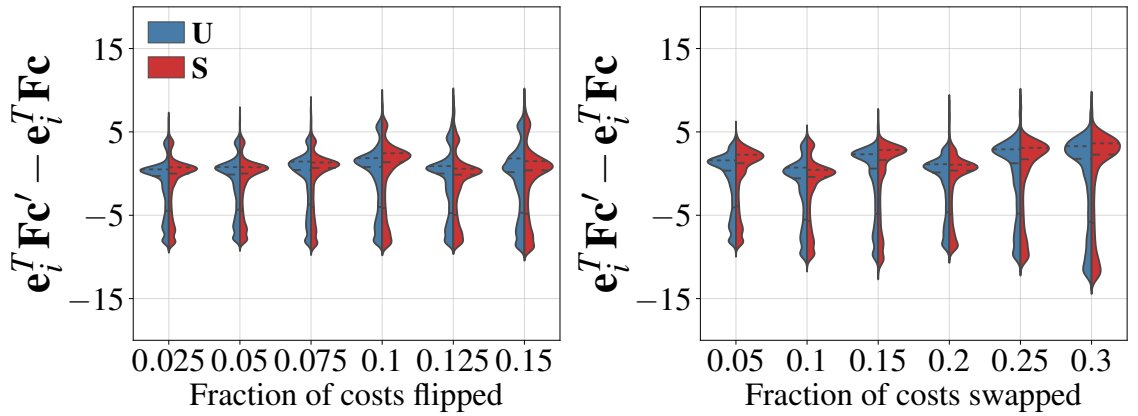


(a) NF **(b)** YT

Figure 6.25: Fraction (color) and number (annotation) of safe nodes in each of our real-world graphs with degrees $d \in \{5, 10, 20\}$, under our four cost functions $c \in \{c_{B1}, c_{B2}, c_{R1}, c_{R2}\}$. In all YT graphs and roughly two thirds of the NF graphs, the precondition of Theorem 6.7 holds, such that the greedy algorithm can approximate f_Δ up to a factor of $(1 - 1/e)$.

Presence of Safe Nodes. In Theorem 6.7 and Corollary 6.8, we established that if a graph G has at least Λ^+ safe nodes, where a node i is safe if its exposure $\boldsymbol{e}_i^\top \mathbf{F}\mathbf{c}$ is 0 and Λ^+ is the maximum degree of an unsafe node in G, then we can approximate f_Δ up to a factor of $(1 - 1/e)$. In Fig. 6.25, we demonstrate that under *all* our cost functions, this applies to *all* YT graphs and roughly *two thirds* of the NF graphs, with the notable exception of news articles on the topic of January 6 (i.e., content reporting on the Capitol riot). Hence, our theoretical approximation guarantee mostly holds also in practice.

Impact of Cost-Function Noise. To see how errors in harmfulness assessment might impact our ability to rewire edges effectively, we investigate the behavior of our exposure objective under noise in the cost function. In particular, we assess how the distribution of node exposures shifts when we change the original cost vector $\mathbf{c}$ to a cost vector $\mathbf{c}'$ by either swapping the cost of a randomly chosen harmful node with that of a randomly chosen benign node (*cost swaps*), or setting the cost c_i of a randomly chosen node i to its opposite, i.e., $1 - c_i$ (*cost flips*). Illustrating the results on the YT-100K dataset in Fig. 6.26, we observe that as expected—and *by construction*—, node exposures are generally sensitive to individual cost assignments. However, the median impact of moderate cost-function noise on node exposure levels is close to zero, and the most extreme cost fluctuations occur for nodes whose *observed* exposure decreases as compared to their *actual* exposure. The latter might lead Gamine to undervalue some highly exposed nodes in its rewiring considerations, but this risk is unavoidable when dealing with noisy data. In contrast to prior work, Gamine uses an exposure objective that



(a) Cost flips **(b)** Cost swaps

Figure 6.26: Distribution of differences between node exposures before and after the introduction of noise, shown for the YT-100k dataset with $d = 5$, $\alpha = 0.05$, and $\chi \in \{\mathbf{U}, \mathbf{S}\}$, as measured under c_{R1}. *Negative* values signal that adding noise *decreased* the exposure to harm of a particular node.

depends on the cost assignments of *all* nodes in the graph. Overall, our experiments with node-level cost-function noise demonstrate that this objective decays rather smoothly—not only in theory but also in practice.

6.D Implementation Details

In this section, we provide details on the choice of κ, the number of power iterations to use in our matrix computations, and on the effects of replacing our original maximization objective Δ by the heuristic $\widehat{\Delta}$.

6.D.1 Chosen Parameters

Hedging against Small Fluctuations. When developing Gamine, we state that we can hedge against small fluctuations in the relationship between Δ and $\widehat{\Delta}$ by computing Δ exactly for the rewiring candidates associated with the $\mathcal{O}(1)$ largest values of $\widehat{\Delta}$ before selecting the final rewiring. In our experiments, we compute Δ exactly for the top 100 rewiring candidates.

Error Bounds for Power Iteration. Recall that $\|\mathbf{M}^\kappa\| \leqslant \|\mathbf{M}\|^\kappa$ for any square matrix $\mathbf{M}$, associated matrix norm $\|\cdot\|$, and non-negative integer κ. Recall further that each row of $\mathbf{P}$ sums to $(1-\alpha)$, such that $\|\mathbf{P}\|_\infty = (1-\alpha)$ and

$$\|\mathbf{P}^\kappa\|_\infty = \|\mathbf{P}\|_\infty^\kappa = (1-\alpha)^\kappa \,. \tag{6.55}$$

Therefore, we can bound the approximation error in the infinity norm of our approximation of $\mathbf{F}$ as

$$\sum_{i=0}^{\infty} \mathbf{P}^i - \sum_{i=0}^{\kappa} \mathbf{P}^i = \frac{1}{1-(1-\alpha)} - \frac{1-(1-\alpha)^\kappa}{1-(1-\alpha)} = \frac{(1-\alpha)^\kappa}{\alpha} \,. \tag{6.56}$$

Thus, to obtain an approximation error on the row sums of at most ε, we need to set the number of iterations κ

$$\varepsilon \leqslant \frac{(1-\alpha)^\kappa}{\alpha} \Leftrightarrow \varepsilon\alpha \leqslant (1-\alpha)^\kappa \Leftrightarrow \kappa \geqslant \log_{1-\alpha} \varepsilon\alpha = \frac{\log \varepsilon\alpha}{\log 1-\alpha} \,. \tag{6.57}$$

For example, to obtain an absolute approximation error $\varepsilon \leqslant 0.01$ on the row sums, given an absorption probability of $\alpha = 0.05$, we need to set

$$\kappa \geqslant \frac{\log 0.005}{\log 0.95} = 148.18 \approx 150 \,, \tag{6.58}$$

independently of n. This justifies the assumption that $\kappa \in \mathcal{O}(1)$, and prompts us to set the number of iterations in all power iteration calculations to 150.

6.D.2 Impact of Using $\hat{\Delta}$ Instead of Δ

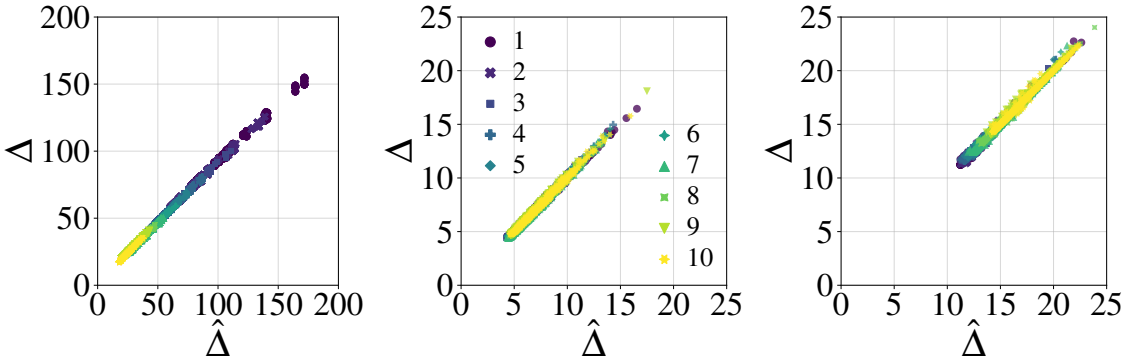
Having confirmed in the main text that Gamine scales linearly not only in theory but also in practice (cf. Fig. 6.12), we would like to ensure that moving from Δ to $\hat{\Delta}$, which enables this scalability, has little impact on the quality of our results. To this end, we investigate the relationship between Δ and $\hat{\Delta}$ on the smallest instances of our synthetic graphs, SU and SH. As illustrated in Fig. 6.27, Δ and $\hat{\Delta}$ are almost perfectly correlated, and Fig. 6.28 shows that this holds not only for the top-ranked candidates but for all candidates, under both product-moment correlation and, more importantly, rank correlation. Thus, we are confident that our reliance on $\hat{\Delta}$, rather than Δ, to select greedy rewirings hardly degrades our results.

6.E Other Graph Edit Operations

In this section, we define and analyze two other graph edits, which are less natural for recommendation graphs but potentially relevant in other applications: edge deletions and edge insertions.

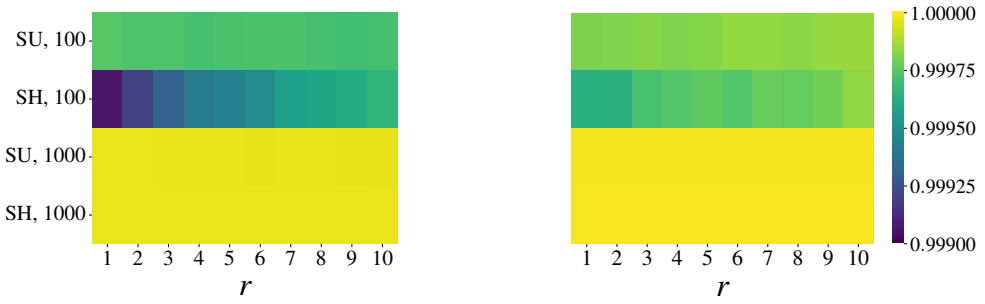
6.E.1 Edge Deletions

An edge deletion removes an edge (i, j) from G, redistributing the p_{ij} to the remaining edges outgoing from i. Assuming that we redistribute the freed probability mass evenly among the remaining out-neighbors of i, the necessary changes are summarized in Table 6.9. We require $\delta^+(i) > 1$, since otherwise, i would have



(a) SH, $n = 100$ **(b)** SU, $n = 100$ **(c)** SU, $n = 1\,000$

Figure 6.27: Correlation of the $\hat{\Delta}$ and Δ values for the 100 candidates (i, j, k) with the largest Δ, in 10 rewiring rounds on synthetic graphs with $\alpha = 0.05$, $\beta = 0.7$, and $\chi = \mathbf{U}$, under binary costs. Δ and $\hat{\Delta}$ are almost perfectly correlated.



(a) Pearson's Correlation **(b)** Spearman's Correlation

Figure 6.28: Pearson's product-moment correlation and Spearman's rank correlation between Δ and $\hat{\Delta}$, in 10 rewiring rounds on synthetic graphs with $\alpha = 0.05$, $\beta = 0.7$, and $\chi = \mathbf{U}$, under binary costs. Both correlations are almost perfect across all rounds, and they are even closer to 1 for the larger synthetic graphs than for the smaller ones.

no remaining neighbors among which to distribute the unused probability mass (and to exclude division by zero), which would effectively require us to create a new absorbing state.

What can we say about the components of $\Delta = {}^{\sigma\tau}/_{\rho}$? For ρ,

$$\rho = 1 + \boldsymbol{v}^{\mathsf{T}}\mathbf{F}\boldsymbol{u} = 1 + p_{ij}\mathbf{F}[j,i] - \frac{p_{ij}}{\delta^+(i) - 1} \sum_{k \in \Gamma^+(i) \setminus \{j\}} \mathbf{F}[k,i] \,, \tag{6.59}$$

which generalizes what we observed for edge rewirings. With the same reasoning as for edge rewiring, for each $k \in \Gamma^+(i)$, we have

$$\begin{aligned} p_{ij}\mathbf{F}[k,i] \leqslant (1-\alpha) + p_{ij}\mathbf{F}[j,i] < 1 + p_{ij}\mathbf{F}[j,i] \\ \Leftrightarrow \frac{p_{ij}}{\delta^+(i) - 1}\mathbf{F}[k,i] < \frac{1}{\delta^+(i) - 1} + \frac{p_{ij}}{\delta^+(i) - 1}\mathbf{F}[j,i] \,, \end{aligned} \tag{6.60}$$

Table 6.9: Summary of an *edge deletion* $-(i,j)$ in a graph $G = (V, E)$ with random-walk transition matrix $\mathbf{P}$ and fundamental matrix $\mathbf{F} = (\mathbf{I} - \mathbf{P})^{-1}$.

$G' = (V, E')$, for $E' = E \setminus \{(i,j)\}$, $(i,j) \in E$
$\mathbf{P}'[x,y] = \begin{cases} 0 & \text{if } x = i \text{ and } y = j \\ \mathbf{P}[i,y] + \frac{\mathbf{P}[i,j]}{\delta^+(i)-1} & \text{if } x = i \text{ and } y \in \Gamma^+(i) \setminus \{j\} \\ 0 & \text{otherwise .} \end{cases}$
$\mathbf{F}' = \mathbf{F} - \frac{\mathbf{F}\boldsymbol{u}\boldsymbol{v}^\top\mathbf{F}}{1+\boldsymbol{v}^\top\mathbf{F}\boldsymbol{u}}$, with $\boldsymbol{u} = p_{ij}\boldsymbol{e}_i$, $\boldsymbol{v} = \boldsymbol{e}_j - \frac{1}{\delta^+(i)-1}\sum_{k\in\Gamma^+(i)\setminus\{j\}} \boldsymbol{e}_k$

Table 6.10: Summary of an *edge insertion* $+(i,j)$ in a graph $G = (V, E)$ with random-walk transition matrix $\mathbf{P}$ and fundamental matrix $\mathbf{F} = (\mathbf{I} - \mathbf{P})^{-1}$.

$G' = (V, E')$, for $E' = E \cup \{(i,j)\}$, $(i,j) \notin E$
$\mathbf{P}'[x,y] = \begin{cases} p_{ij} & \text{if } x = i \text{ and } y = j \\ \mathbf{P}[i,y] - \frac{p_{ij}}{\delta^+(i)} & \text{if } x = i \text{ and } y \in \Gamma^+(i) \\ 0 & \text{otherwise ,} \end{cases}$ for $p_{ij} \leqslant 1 - \alpha$ chosen freely.
$\mathbf{F}' = \mathbf{F} - \frac{\mathbf{F}\boldsymbol{u}\boldsymbol{v}^\top\mathbf{F}}{1+\boldsymbol{v}^\top\mathbf{F}\boldsymbol{u}}$, with $\boldsymbol{u} = p_{ij}\boldsymbol{e}_i$, $\boldsymbol{v} = -\boldsymbol{e}_j + \frac{1}{\delta^+(i)}\sum_{k\in\Gamma^+(i)\setminus\{j\}} \boldsymbol{e}_k$

such that ρ again must be positive. For $\sigma = \mathbf{1}^\top\mathbf{F}\boldsymbol{u}$, as $\boldsymbol{u}$ is exactly the same as for edge rewirings, the analysis for σ under edge rewiring holds analogously. For τ, we get

$$\tau = \boldsymbol{v}^\top\mathbf{F}\boldsymbol{c} = \boldsymbol{e}_j\mathbf{F}\boldsymbol{c} - \frac{1}{\delta^+(i)-1}\sum_{k\in\Gamma^+(i)\setminus\{j\}} \boldsymbol{e}_k\mathbf{F}\boldsymbol{c}\,, \tag{6.61}$$

which can have any sign, and which we would like to be positive because ρ and σ are positive, too. Intuitively, this generalizes what we observed for edge rewirings: To maximize τ, we need to maximize the difference between the cost-scaled row sum of j and the *average* of the cost-scaled row sums of all other out-neighbors of i.

6.E.2 Edge Insertions

An edge insertion adds an edge (i,j) into G with a freely chosen $p_{ij} \leqslant 1 - \alpha$, reducing the probability masses associated with the other edges outgoing from i proportionally. Assuming that we subtract the required probability mass evenly from the original out-neighbors of i, the necessary changes are summarized in Table 6.10.

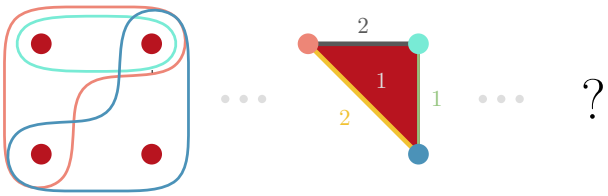
What can we say about the components of $\Delta = {}^{\sigma\tau}/_{\rho}$? For ρ,

$$\rho = 1 + \boldsymbol{\nu}^{\mathsf{T}}\mathbf{F}\mathbf{u} = 1 + \frac{1}{\delta^{+}(i)} \sum_{k \in \Gamma^{+}(i)} p_{ij}\mathbf{F}[k,i] - p_{ij}\mathbf{F}[j,i] \,, \tag{6.62}$$

which generalizes what we observed for edge rewirings. Unfortunately, as in this case, the edge (i,j) does not factor into the computation of $\mathbf{F}$ (as is the case for edge rewirings and edge deletions), we cannot guarantee that ρ is always positive. For $\sigma = \mathbf{1}^{\mathsf{T}}\mathbf{F}\mathbf{u}$, as $\mathbf{u}$ is exactly the same as for edge rewirings, the analysis for σ under edge rewiring holds analogously. For τ, we get

$$\tau = \boldsymbol{\nu}^{\mathsf{T}}\mathbf{F}\mathbf{c} = \frac{1}{\delta^{+}(i)} \sum_{k \in \Gamma^{+}(i)} \boldsymbol{e}_{k}\mathbf{F}\mathbf{c} - \boldsymbol{e}_{j}\mathbf{F}\mathbf{c} \,, \tag{6.63}$$

which can have any sign, and which we would like to be positive if ρ is positive, and negative if ρ is negative. Intuitively, this generalizes what we observed for edge rewirings: To maximize the τ, we need to maximize the difference between the *average* of the cost-scaled row sums of all out-neighbors of i and the cost-scaled row sum of j.



7

Outlook: Beyond Graphland

On our journey through Graphland, we explored graphs in five dimensions:

descriptivity, *multiplicity*, *complexity*, *expressivity*, and *responsibility*.

7.1 Looking Back

First, in the *descriptivity* dimension, we met Momo, which casts *graph similarity* as a *description* problem. Momo leverages the *Minimum Description Length* principle to descriptively compare undirected or directed graphs by capturing their similarity in a common model and their differences in transformations to individual models.

Second, in the *multiplicity* dimension, we encountered Gragra, which performs *graph group analysis*. Given a set of undirected or directed, unweighted or weighted graphs and a partition of this set into groups (e.g., by known covariates), Gragra describes the similarities and differences between graph groups by using *maximum-entropy modeling* along with *statistical testing* to discover a set of subgraphs with statistically significant associations to one or more graph groups.

Third, in the *complexity* dimension, we engaged with Hyperbard, a dataset of diverse relational data representations derived from Shakespeare's plays. Hyperbard's representations range from simple graphs capturing character co-occurrence to *temporal hypergraphs* encoding complex communication settings, highlighting the advantages and drawbacks of specific representations and underlining the impact of representation choice on graph mining results. Introduced

through a Shakespeare-style play, Hyperbard also criticizes insidious incentive structures and problematic patterns of practice in our research community, and it marks the author's emancipation from the concomitant culture.

Fourth, in the *expressivity* dimension, we cultivated Orchid, a unified framework for generalizing *Ollivier-Ricci curvature* to *hypergraphs*. Like the original Ollivier-Ricci curvature for graphs, Orchid grounds a concept bridging geometry and topology (curvature) in ideas from probability theory and optimal transport (comparing random walks via Wasserstein distances), yielding favorable theoretical properties as well as powerful and scalable features to support various hypergraph tasks in practice.

Fifth and finally, in the *responsibility* dimension, we designed Gamine, which mitigates exposure to harm in directed, node-weighted and edge-weighted (recommendation) graphs via *edge rewiring*. We formalized this task as the r-*rewiring exposure minimization* (REM) problem based on an exposure model using *absorbing random walks*, and established its NP-hardness in the general case, along with its approximability under mild assumptions. Leveraging structural insights into its objective function, Gamine efficiently and effectively addresses the problem, both without and with quality constraints on recommendations.

In Tables 7.1 to 7.3, we summarize our thesis in terms of its *datasets domains*, *graph settings*, and *mathematical foundations*: Across all dimensions, we worked with relational data from many different domains, introducing numerous novel datasets in the process (Table 7.1). We modeled our data using a variety of graph types, including directed, weighted, and temporal graphs as well as node-weighted and edge-weighted hypergraphs, often investigating several graphs simultaneously (Table 7.2). To analyze our problems and design our algorithms, we leveraged concepts and insights from several branches of mathematics, including graph theory, information theory, probability theory, geometry, and topology (Table 7.3). Nevertheless, there remains much more to be discovered.

7.2 Looking Ahead

Beyond the future work sketched in the chapters that introduce our contributions, we see particular potential for expanding, or perhaps going beyond, Graphland in five directions, which we describe in ascending order of generality: *topological data analysis for hypergraphs*, *rich relational data*, *causality in complex networks*, *theory of data*, and *community culture*.

Topological Data Analysis for Hypergraphs. Although Orchid draws on ideas from topology, the power of *topological approaches to hypergraph analysis* remains largely untapped. In particular, a hypergraph can be naturally associated with

Table 7.1: Dataset domains considered in this thesis. Abbreviations: *Econ*—Economics, *Info*—Information, *Infra*—Infrastructure, *Law*—Law, *Life*—Life Sciences, *Lit*—Literature, *Mus*—Music, *Sci*—Scientific interactions, *Syn*—Synthetic data. Icons: ★—original dataset(s) created by us, ⯪—dataset(s) reconstructed from prior work, ☆—dataset(s) reused from prior work.

	Dataset Domain								
	Econ	Info	Infra	Law	Life	Lit	Mus	Sci	Syn
Momo		☆	☆	★	☆			★	★
Gragra	★		★		⯪				★
Hyperbard						★			
Orchid		★			★	★	★	★	★
Gamine		⯪							★

Table 7.2: Graph settings considered in this thesis. Abbreviations: *nw*—node weights, *ew*—edge weights, *ed*—edge directions, *me*—multi-edges, *mg*—multiple graphs, *tg*—temporal graphs, *hg*: hypergraphs. Icons: ★—considered directly, ⯪—considered indirectly, ☆—simple extension.

	Graph Setting						
	nw	ew	ed	me	mg	tg	hg
Momo			★		★	☆	
Gragra		★	★		★	☆	
Hyperbard	★	★	★	★	★	★	★
Orchid	☆	☆	☆	★	★		★
Gamine	★	★	★	⯪			

Table 7.3: Mathematical foundations of this thesis. Abbreviations: *GT*—Graph Theory, *IT*—Information Theory, *PT*—Probability Theory, *GY*—Geometry, *TY*—Topology. Icons: ★—direct foundation, ⯪—indirect foundation, ☆—inspiration.

	Domain				
	GT	IT	PT	GY	TY
Momo	★	★	⯪		
Gragra	★	⯪	★		
Hyperbard	★	☆			☆
Orchid	★		⯪	★	★
Gamine	★		★		

a simplicial complex, called the *nerve* (or nerve complex) [46], that captures the structure of its edge intersections, and we can easily build nested sequences of simplicial complexes, called *filtrations* [49], from a hypergraph by exploiting that edges vary in cardinality. Combining these two concepts, and potentially adding *quantitative* information to the resulting *qualitative* objects (such as accounting for the sizes of edge intersections to create a *weighted nerve complex*), promises to yield

powerful new hypergraph descriptors, which could support hypergraph analysis, hypergraph mining, and hypergraph learning tasks.

Rich Relational Data. While we considered many different graph types, including the relatively understudied class of hypergraphs, several important variants of graphs and networks were comparatively underrepresented in our investigations: *attributed* graphs [218]; (partially) *ordered* graphs [80]; dynamic, temporal, or otherwise *evolving* graphs and networks [20, 117, 121, 153]; *multilayer* networks [147]; and *spatial* networks [21]. Many interesting real-world problems, such as understanding the dynamics of urban environments [189], however, involve rich relational data that is most faithfully modeled as (hyper)graphs which are (at least partially) attributed, directed, evolving, multilayer, ordered, spatial, *and* weighted. Developing methods to analyze such data, and compiling the *well-documented, open-source datasets* needed to rigorously evaluate these methods, thus constitutes an important and exciting area for future work.

Causality in Complex Networks. While the work presented in this thesis is predominantly *descriptive* (and, in the case of Gamine, somewhat *prescriptive*), domain scientists studying graph data often seek to understand the causal mechanisms that drive the complex systems underlying their data. In recent years, computer scientists have become more interested in causality [203, 214, 217], and domain scientists have made progress in developing methods to study causal questions in networked settings [9, 95, 247]. Nevertheless, many challenges surrounding *causal inference and causal effect estimation in complex networks* remain largely unsolved [36, 92]. Although understanding causal mechanisms is crucial for designing robust interventions into, or governance approaches for, complex networked systems, controlled experiments, which remain the gold standard for causal investigations, are often impossible or impractical [11]. Therefore, to make progress in this area, we need methods *extracting causal knowledge from observational network data*, and developing such methods represents a high-risk, high-reward avenue for further research.

Theory of Data. For a large part of this thesis, we took a *pragmatic* approach to data: Motivated by real-world phenomena, we defined our graph problem of interest, designed our method to solve this problem, and then used data in our experiments to demonstrate that our method performs well in practice. When selecting data for our experiments, we intuitively defined our graphs, based on readily available sources from domains we deemed interesting or familiar. While we tried to ensure some *diversity* in our data, e.g., with respect to their semantics

and basic statistics, we seldom worried, and much less reasoned formally, about the impact of our definitions and decisions on our experimental results.

At the same time, it is intuitively clear that how well our methods perform *also* depends on the input data: A task that is easy to solve on one dataset may be very hard to solve on another dataset. Moreover, how the features of a dataset impact the performance of an algorithm may further depend on the algorithm (and, of course, on the problem). This is particularly troublesome in graph mining and network analysis (as well as data mining and data analysis more broadly), where different works studying the same problem tend to use different datasets or evaluation criteria [112, 119, 251]. But it is also of concern for graph learning (and machine learning more broadly), where sweeping claims of superior performance are often based on experiments restricted to a small part of Graphland [172, 205, 210].

Hence, in light of its crucial importance in data analysis, data mining, and machine learning, *data* itself seems severely undertheorized. While some relevant work has been done in *database theory* [149] and *category theory* [244, Section 4.5], and some progress can be expected from the emerging field of *data-centric artificial intelligence* [134], questions regarding *data complexity*, *sampling from data spaces*, and *hardness in data* could also be fruitfully approached from a theoretical-computer-science perspective. Here, complexity notions like *descriptive complexity* [133] might provide starting points for further investigations, but developing a *theory of data* will likely require concepts that are yet to be invented.

Community Culture. Finally, mirroring the concerns voiced by Hyperbard, there is ample room for improvement regarding the practices prevailing in our research community. While the details could easily fill the pages of another book, one core insight here is that we can ameliorate our procedures by subjecting them to the same scientific scrutiny that we allocate to our papers. This approach has gained traction in the context of machine-learning *reproducibility* [25, 42, 113, 129, 220], but many other crucial aspects of our processes remain poorly understood. Although top conferences increasingly conduct experiments or distribute surveys among their participants [219, 226], to truly understand the shortcomings of our current workflows and develop viable alternatives, we need a more systematic approach. Furthermore, for some crucial steps in our processes, such as documenting data [51, 99, 120] and benchmarking methods [33, 252], more research into the requirements of different stakeholders and the ways to satisfy stakeholder needs is necessary to develop *best practices* that also fulfill their purpose *in practice*. Such research requires data mining and machine learning expertise, but it

will also benefit from engaging with other fields, such as requirements engineering, human-computer interaction, or the sociology of science.

* * *

Thus, we conclude our journey through Graphland—or rather, our guided tour, for we know that we shall return. As we continue to wander, and wonder, about our highly connected world, science expands in all directions. To us, it is what data are for graphs:

A Romance of Many Dimensions.

Bibliography

Where available, we provide DOIs for the referenced works. For papers which are available on arXiv but were published at conferences that do not assign DOIs, we use the DOIs allocated by arXiv.

[1] Edwin Abbott Abbott. *Flatland: A romance of many dimensions*. London: Seeley & Co., 1884 (page 1).

[2] Rediet Abebe et al. "Opinion dynamics optimization by varying susceptibility to persuasion via non-convex local search." In: *ACM Transactions on Knowledge Discovery from Data* 16.2 (2021), pp. 1–34. DOI: 10.1145/3466617 (page 161).

[3] Florian Adriaens, Honglian Wang, and Aristides Gionis. "Minimizing hitting time between disparate groups with shortcut edges." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2023, to appear (page 162).

[4] Charu C. Aggarwal and Haixun Wang, eds. *Managing and mining graph data*. Springer, 2010. DOI: 10.1007/978-1-4419-6045-0 (page 1).

[5] Tomoya Akamatsu. "A new transport distance and its associated Ricci curvature of hypergraphs." In: *Analysis and Geometry in Metric Spaces* 10.1 (2022), pp. 90–108. DOI: 10.1515/agms-2022-0135 (page 110).

[6] Sinan G. Aksoy, Cliff Joslyn, Carlos Ortiz Marrero, Brenda Praggastis, and Emilie Purvine. "Hypernetwork science via high-order hypergraph walks." In: *EPJ Data Science* 9.1 (2020), p. 16. DOI: 10.1140/epjds/s13688-020-00231-0 (pages 1, 62, 111, 122).

[7] Ilya Amburg, Nate Veldt, and Austin Benson. "Clustering in graphs and hypergraphs with categorical edge labels." In: *Proceedings of the ACM Web Conference*. 2020, pp. 706–717. DOI: 10.1145/3366423.3380152 (page 111).

[8] Victor Amelkin and Ambuj K. Singh. "Fighting opinion control in social networks via link recommendation." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2019, pp. 677–685. DOI: 10.1145/3292500.3330960 (page 161).

[9] Weihua An, Roberson Beauvile, and Benjamin Rosche. "Causal network analysis." In: *Annual Review of Sociology* 48 (2022), pp. 23–41. DOI: 10.1146/annurev-soc-030320-102100 (page 200).

[10] Shahab Asoodeh, Tingran Gao, and James Evans. "Curvature of hypergraphs via multi-marginal optimal transport." In: *IEEE Conference on Decision and Control (CDC)*. 2018, pp. 1180–1185. DOI: 10.1109/CDC.2018.8619706 (pages 98, 102, 104, 110).

[11] Susan Athey and Guido W. Imbens. "The state of applied econometrics: Causality and policy evaluation." In: *Journal of Economic Perspectives* 31.2 (2017), pp. 3–32. DOI: 10.1257/jep.31.2.3 (page 200).

[12] László Babai. "Group, graphs, algorithms: The graph isomorphism problem." In: *Proceedings of the International Congress of Mathematicians (ICM)*. Vol. 4. 2018, pp. 3319–3336. DOI: 10.1142/9789813272880_0183 (page 9).

[13] James P. Bagrow and Erik M. Bollt. "An information-theoretic, all-scales approach to comparing networks." In: *Applied Network Science* 4.1 (2019), p. 45. DOI: 10.1007/s41109-019-0156-x (pages 24, 31, 33).

[14] Lu Bai, Peng Ren, and Edwin R. Hancock. "A hypergraph kernel from isomorphism tests." In: *Proceedings of the International Conference on Pattern Recognition (ICPR)*. 2014, pp. 3880–3885. DOI: 10.1109/ICPR.2014.665 (page 111).

[15] Song Bai, Feihu Zhang, and Philip H.S. Torr. "Hypergraph convolution and hypergraph attention." In: *Pattern Recognition* 110 (2021), p. 107637. DOI: 10.1016/j.patcog.2020.107637 (page 62).

[16] Dominique Bakry and Michel Émery. "Diffusions hypercontractives." In: *Séminaire de probabilités de Strasbourg* 19 (1985), pp. 177–206. DOI: 10.1007/BFb0075847 (page 110).

[17] Jack Bandy. "Problematic machine behavior: A systematic literature review of algorithm audits." In: *Proceedings of the ACM on Human-Computer Interaction (CHI)*. 2021, pp. 1–34. DOI: 10.1145/3449148 (page 140).

[18] Anirban Banerjee. "On the spectrum of hypergraphs." In: *Linear Algebra and its Applications* 614 (2021), pp. 82–110. DOI: 10.1016/j.laa.2020.01.012 (pages 100, 102, 110).

[19] Albert-László Barabási. "Network science." In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1987 (2013), p. 20120375. DOI: 10.1098/rsta.2012.0375 (page 1).

[20] Alain Barrat, Marc Barthélemy, and Alessandro Vespignani. "Weighted evolving networks: Coupling topology and weight dynamics." In: *Physical Review Letters* 92.22 (2004), p. 228701. DOI: 10.1103/PhysRevLett.92.228701 (page 200).

[21] Marc Barthélemy. "Spatial networks." In: *Physics Reports* 499.1-3 (2011), pp. 1–101. DOI: 10.1016/j.physrep.2010.11.002 (page 200).

[22] Danielle S. Bassett and Olaf Sporns. "Network neuroscience." In: *Nature Neuroscience* 20.3 (2017), pp. 353–364. DOI: 10.1038/nn.4502 (page 49).

[23] Federico Battiston et al. "The physics of higher-order interactions in complex systems." In: *Nature Physics* 17.10 (2021), pp. 1093–1098. DOI: 10.1038/s41567-021-01371-4 (pages 1, 62).

[24] F. Bauer, F. Chung, Y. Lin, and Y. Liu. "Curvature aspects of graphs." In: *Proceedings of the American Mathematical Society* 145.5 (2017), pp. 2033–2042. DOI: 10.1090/proc/13145 (page 111).

[25] Samuel J. Bell, Onno Kampman, Jesse Dodge, and Neil Lawrence. "Modeling the machine learning multiverse." In: *Advances in Neural Information Processing Systems*. 2022, pp. 18416–18429. DOI: 10.48550/arXiv.2206.05985 (page 201).

[26] Claude Berge. *Graphes et hypergraphes*. Paris: Dunot, 1970 (page 1).

[27] Claude Berge. *Hypergraphs: Combinatorics of finite sets*. 45. Amsterdam: Elsevier, 1989 (page 62).

[28] Michele Berlingerio, Danai Koutra, Tina Eliassi-Rad, and Christos Faloutsos. "Network similarity via multiple social theories." In: *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2013, pp. 1439–1440. DOI: 10.1145/2492517.2492582 (page 24).

[29] Stefano Berto et al. "Association between resting-state functional brain connectivity and gene expression is altered in autism spectrum disorder." In: *Nature Communications* 13.1 (2022), p. 3328. DOI: 10.1038/s41467-022-31053-5 (page 38).

[30] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. "Size-invariant graph representations for graph classification extrapolations." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2021, pp. 837–851. DOI: 10.48550/arXiv.2103.05045 (page 39).

[31] Satyajit Bhadange, Akhil Arora, and Arnab Bhattacharya. "GARUDA: a system for large-scale mining of statistically significant connected subgraphs." In: *Proceedings of the VLDB Endowment* 9.13 (2016), pp. 1449–1452. DOI: 10.14778/3007263.3007281 (page 39).

[32] Christian Bick, Elizabeth Gross, Heather A. Harrington, and Michael T. Schaub. "What are higher order networks?" In: *SIAM Review* (2022). DOI: 10.48550/arXiv.2104.11329 (page 1).

[33] Bernd Bischl et al. "OpenML benchmarking suites." In: *Proceedings of the NeurIPS Datasets and Benchmarks Track*. 2021. DOI: 10.48550/arXiv.1708.03731 (page 201).

[34] Isabelle Bloch and Alain Bretto. "Mathematical morphology on hypergraphs, application to similarity and positive kernel." In: *Computer Vision and Image Understanding* 117.4 (2013), pp. 342–354. DOI: 10.1016/j.cviu.2012.10.013 (page 111).

[35] Stefano Boccaletti et al. "The structure and dynamics of multilayer networks." In: *Physics Reports* 544.1 (2014), pp. 1–122. DOI: 10.1016/j.physrep.2014.07.001 (page 1).

[36] Örjan Bodin et al. "Improving network approaches to the study of complex social–ecological interdependencies." In: *Nature Sustainability* 2.7 (2019), pp. 551–559. DOI: 10.1038/s41893-019-0308-0 (page 200).

[37] Cristian Bodnar et al. "Weisfeiler and Lehman go topological: Message passing simplicial networks." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2021, pp. 1026–1037. DOI: 10.48550/arXiv.2103.03212 (page 1).

[38] Carlo Bonferroni. "Teoria statistica delle classi e calcolo delle probabilita." In: *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze* 8 (1936), pp. 3–62 (page 121).

[39] Karsten Borgwardt, Elisabetta Ghisu, Felipe Llinares-López, Leslie O'Bray, and Bastian Rieck. "Graph kernels: State-of-the-art and future challenges." In: *Foundations and Trends in Machine Learning* 13.5–6 (2020), pp. 531–712. DOI: 10.1561/2200000076 (pages 1, 111).

[40] Karsten Borgwardt and Hans-Peter Kriegel. "Shortest-path kernels on graphs." In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2005, pp. 74–81. DOI: 10.1109/ICDM.2005.132 (page 24).

[41] David P. Bourne, David Cushing, Shiping Liu, Florentin Münch, and Norbert Peyerimhoff. "Ollivier-Ricci idleness functions of graphs." In: *SIAM Journal on Discrete Mathematics* 32.2 (2018), pp. 1408–1424. DOI: 10.1137/17M113446 (page 111).

[42] Xavier Bouthillier, César Laurent, and Pascal Vincent. "Unreproducible research is reproducible." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2019, pp. 725–734 (page 201).

[43] Ed Bullmore and Olaf Sporns. "Complex brain networks: Graph theoretical analysis of structural and functional systems." In: *Nature Reviews Neuroscience* 10.3 (2009), pp. 186–198. DOI: 10.1038/nrn2575 (page 49).

[44] Bureau of Transportation Statistics. *Data Bank 28DS - T-100 Domestic Segment Data (World Area Code)*. 2021. URL: https://www.bts.gov/browse-statistical-products-and-data/bts-publications/data-bank-28ds-t-100-domestic-segment-data (visited on 09/04/2021) (page 50).

[45] Chen Cai and Yusu Wang. *A simple yet effective baseline for non-attributed graph classification*. 2018. arXiv: 1811.03508 [cs.LG] (page 119).

[46] Gunnar Carlsson. "Topology and data." In: *Bulletin of the American Mathematical Society* 46.2 (2009), pp. 255–308 (page 199).

[47] Deepayan Chakrabarti and Christos Faloutsos. "Graph mining: Laws, generators, and algorithms." In: *ACM Computing Surveys (CSUR)* 38.1 (2006), pp. 1–69. DOI: 10.1145/1132952.1132954 (page 1).

[48] Hau Chan and Leman Akoglu. "Optimizing network robustness by edge rewiring: A general framework." In: *Data Mining and Knowledge Discovery* 30 (2016), pp. 1395–1425. DOI: 10.1007/s10618-015-0447-5 (page 160).

[49] Frédéric Chazal and Bertrand Michel. "An introduction to topological data analysis: Fundamental and practical aspects for data scientists." In: *Frontiers in Artificial Intelligence* 4 (2021), p. 667963. DOI: 10.3389/frai.2021.667963 (page 199).

[50] Uthsav Chitra and Christopher Musco. "Analyzing the impact of filter bubbles on social network polarization." In: *Proceedings of the ACM International Conference on Web Search and Data Mining*. 2020, pp. 115–123. DOI: 10.1145/3336191.3371825 (page 140).

[51] Kasia S. Chmielinski et al. *The dataset nutrition label (2nd Gen): Leveraging context to mitigate harms in artificial intelligence*. 2022. arXiv: 2201.03954 [cs.LG] (page 201).

[52] Diane J. Cook and Lawrence B. Holder. "Substructure discovery using Minimum Description Length and background knowledge." In: *Journal of Artificial Intelligence Research* 1 (1994), pp. 231–255. DOI: 10.1613/jair.43 (page 24).

[53] Cornell University. *arXiv Dataset, Version 18 (2020/11/22)*. 2020. URL: https://www.kaggle.com/Cornell-University/arxiv (pages 25, 26).

[54] Sasha Costanza-Chock, Inioluwa Deborah Raji, and Joy Buolamwini. "Who Audits the auditors? Recommendations from a field scan of the algorithmic auditing ecosystem." In: *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*. 2022, pp. 1571–1583. DOI: 10.1145/3531146.3533213 (page 140).

[55] Corinna Coupette. *Juristische Netzwerkforschung: Modellierung, Quantifizierung und Visualisierung relationaler Daten im Recht [Legal network science: Modeling, measuring, and mapping relational data in law]*. Tübingen: Mohr Siebeck, 2019. XVIII + 376. DOI: 10.1628/978-3-16-157012-4 (page 7).

[56] Corinna Coupette, Sebastian Dalleiger, and Bastian Rieck. "Ollivier-Ricci curvature for hypergraphs: A unified framework." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2023. DOI: 10.48550/arXiv.2210.12048 (page 5).

[57] Corinna Coupette and Dirk Hartung. "Rechtsstrukturvergleichung [Structural comparative law]." In: *RabelsZ–The Rabel Journal of Comparative and International Private Law* 86.4 (2022), pp. 935–975. DOI: 10.1628/rabelsz-2022-0082 (pages 6, 7).

[58] Corinna Coupette and Dirk Hartung. "Sharing and caring: Creating a culture of constructive criticism in computational legal studies." In: *MIT Computational Law Report* (2023), to appear. DOI: 10.48550/arXiv.2205.01071 (page 7).

[59] Corinna Coupette, Dirk Hartung, Janis Beckedorf, Maximilian Böther, and Daniel Martin Katz. "Law smells: Defining and detecting problematic patterns in legal drafting." In: *Artificial Intelligence and Law* 31 (2023), pp. 335–368. DOI: 10.1007/s10506-022-09315-w (pages 6, 7).

[60] Corinna Coupette and Christoph Lenzen. "A breezing proof of the KMW bound." In: *Symposium on Simplicity in Algorithms (SOSA)*. 2021, pp. 184–195. DOI: 10.1137/1.9781611976496.21 (pages 6, 7).

[61] Corinna Coupette, Stefan Neumann, and Aristides Gionis. "Reducing exposure to harmful content via graph rewiring." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2023, to appear. DOI: 10.1145/3580305.3599489 (page 6).

[62] Corinna Coupette, Jyotsna Singh, and Holger Spamann. "Simplify your law: Using information theory to deduplicate legal documents." In: *Proceedings of the International Conference on Data Mining Workshops (ICDMW)*. 2021, pp. 631–638. DOI: 10.1109/ICDMW53433.2021.00083 (pages 6, 7).

[63] Corinna Coupette and Jilles Vreeken. "Graph similarity description: How are these graphs similar?" In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2021, pp. 185–195. DOI: 10.1145/3447548.3467257 (pages 5, 46).

[64] Corinna Coupette, Jilles Vreeken, and Bastian Rieck. *All the world's a (hyper)graph: A data drama*. Submitted to Digital Scholarship in the Humanities (DSH). 2022. DOI: 10.48550/arXiv.2206.08225 (pages 5, 137).

[65] [Dataset] Corinna Coupette, Jilles Vreeken, and Bastian Rieck. *Hyperbard: (Hyper)graph representations of Shakespeare's plays*. Version 0.0.1. 2022. DOI: 10.5281/zenodo.6627159. URL: https://hyperbard.net (pages 59, 68).

[66] [Code] Corinna Coupette, Jilles Vreeken, and Bastian Rieck. *Hyperbard: (Hyper)graph representations of Shakespeare's plays*. Version 0.0.1. 2022. DOI: 10.5281/zenodo.6627161. URL: https://github.com/hyperbard/hyperbard (pages 59, 68).

[67] Corinna Coupette*, Janis Beckedorf*, Dirk Hartung, Michael Bommarito, and Daniel Martin Katz. "Measuring law over time: A network analytical framework with an application to statutes and regulations in the United States and Germany." In: *Frontiers in Physics* 9 (2021). DOI: 10.3389/fphy.2021.658463 (pages 6, 7, 25, 26).

[68] Corinna Coupette*, Sebastian Dalleiger*, and Jilles Vreeken. "Differentially describing groups of graphs." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2022, pp. 3959–3967. DOI: 10.1609/aaai.v36i4.20312 (page 5).

[69] Cameron Craddock et al. "The neuro bureau preprocessing initiative: Open sharing of preprocessed neuroimaging data and derivatives." In: *Frontiers in Neuroinformatics* 7 (2013). DOI: 10.3389/conf.fninf.2013.09.00041 (page 50).

[70] Imre Csiszár. "I-divergence geometry of probability distributions and minimization problems." In: *The Annals of Probability* (1975), pp. 146–158. DOI: 10.1214/aop/1176996454 (page 40).

[71] Marco Cuturi and Arnaud Doucet. "Fast computation of Wasserstein barycenters." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2014, pp. 685–693. DOI: 10.48550/arXiv.1310.4375 (page 105).

[72] Sebastian Dalleiger and Jilles Vreeken. "Explainable data decompositions." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2020, pp. 3709–3716. DOI: 10.1609/aaai.v34i04.5780 (pages 40, 46).

[73] Sebastian Dalleiger and Jilles Vreeken. "The relaxed maximum entropy fistribution and its application to pattern discovery." In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2020, pp. 978–983. DOI: 10.1109/ICDM50108.2020.00112 (pages 40, 46).

[74] Abhimanyu Das, Sreenivas Gollapudi, and Kamesh Munagala. "Modeling opinion dynamics in social networks." In: *Proceedings of the ACM International Conference on Web Search and Data Mining*. 2014, pp. 403–412. DOI: 10.1145/2556195.2559896 (page 161).

[75] Tijl de Bie. "Maximum entropy models and subjective interestingness: An application to tiles in binary databases." In: *Data Mining and Knowledge Discovery* 23 (2011), pp. 407–446. DOI: 10.1007/s10618-010-0209-3 (page 15).

[76] Henri de Plaen, Michael Fanuel, and Johan A.K. Suykens. "Wasserstein exponential kernels." In: *IEEE International Joint Conference on Neural Networks (IJCNN)* (2020), pp. 1–6. DOI: 10.48550/arXiv.2002.01878 (page 119).

[77] Reinhard Diestel, Alexander Schrijver, and Paul Seymour. "Graph theory." In: *Oberwolfach Reports* 7.1 (2010), pp. 521–580. DOI: 10.14760/OWR-2010-11 (page 1).

[78] Manh Tuan Do, Se-Eun Yoon, Bryan Hooi, and Kijung Shin. "Structural patterns and generative models of real-world hypergraphs." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2020, pp. 176–186. DOI: 10.1145/3394486.3403060 (page 111).

[79] Peter G. Doyle and J. Laurie Snell. *Random walks and electric networks*. Providence: American Mathematical Society, 1984. DOI: 10.5948/UP09781614440222 (page 142).

[80] Guillaume Ducoffe, Laurent Feuilloley, Michel Habib, and François Pitois. *Pattern detection in ordered graphs*. 2023. arXiv: 2302.11619 [cs.DS] (page 200).

[81] Daniele Durante, David B. Dunson, et al. "Bayesian inference and testing of group differences in brain networks." In: *Bayesian Analysis* 13.1 (2018), pp. 29–58. DOI: 10.1214/16-BA1030 (page 46).

[82] Daniele Durante, David B. Dunson, and Joshua T. Vogelstein. "Nonparametric Bayes modeling of populations of networks." In: *Journal of the American Statistical Association* 112.520 (2017), pp. 1516–1530. DOI: 0.1080/01621459.2016.1219260 (page 46).

[83] Marzieh Eidi and Jürgen Jost. "Ollivier–Ricci curvature of directed hypergraphs." In: *Scientific Reports* 10.1 (2020), pp. 1–14. DOI: 10.1038/s41598-020-68619-6 (pages 98, 110).

[84] Tina Eliassi-Rad, Vito Latora, Martin Rosvall, and Ingo Scholtes. "Higher-order graph models: From theoretical foundations to machine learning (Dagstuhl seminar 21352)." In: *Dagstuhl Reports*. Vol. 11. 7. 2021. DOI: 10.4230/DagRep.11.7.139 (page 1).

[85] Paul Erdős and Alfréd Rényi. "On random graphs I." In: *Publicationes Mathematicae* 6.1 (1959), pp. 290–297 (page 164).

[86] Leonhard Euler. "Solutio problematis ad geometriam situs pertinentis." In: *Commentarii Academiae Scientiarum Petropolitanae* 8 (1741), pp. 128–140 (page 1).

[87] Francesco Fabbri, Yanhao Wang, Francesco Bonchi, Carlos Castillo, and Michael Mathioudakis. "Rewiring what-to-watch-next recommendations to reduce radicalization pathways." In: *Proceedings of the ACM Web Conference*. 2022, pp. 2719–2728. DOI: 10.1145/3485447.3512143 (pages 140, 141, 161, 162, 164, 167–170, 173, 175, 177, 183, 188).

[88] Giorgio Fagiolo, Javier Reyes, and Stefano Schiavo. "The evolution of the world trade web: A weighted-network analysis." In: *Journal of Evolutionary Economics* 20.4 (2010), pp. 479–514. DOI: 10.1007/s00191-009-0160-x (page 53).

[89] Uriel Feige. *Vertex cover is hardest to approximate on regular graphs*. Technical Report MCS03–15. 2003 (page 151).

[90] Jing Feng, Xiao He, Nina Hubig, Christian Böhm, and Claudia Plant. "Compression-based graph mining exploiting structure primitives." In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2013, pp. 181–190. DOI: 10.1109/ICDM.2013.56 (page 24).

[91] Antonio Ferrara, Lisette Espín-Noboa, Fariba Karimi, and Claudia Wagner. "Link recommendations: Their impact on network structure and minorities." In: *Proceedings of the ACM Web Science Conference*. 2022, pp. 228–238. DOI: 10.1145/3501247.3531583 (page 139).

[92] Paul J. Ferraro, James N. Sanchirico, and Martin D. Smith. "Causal inference in coupled human and natural systems." In: *Proceedings of the National Academy of Sciences* 116.12 (2019), pp. 5311–5318. DOI: 10.1073/pnas.180556311 (page 200).

[93] Robin Forman. "Bochner's method for cell complexes and combinatorial Ricci curvature." In: *Discrete and Computational Geometry* 29.3 (2003), pp. 323–374. DOI: 10.1007/s00454-002-0743-x (page 110).

[94] Alex Fornito, Andrew Zalesky, and Michael Breakspear. "The connectomics of brain disorders." In: *Nature Reviews Neuroscience* 16.3 (2015), pp. 159–172. DOI: 10.1038/nrn3901 (page 49).

[95] Kenneth A. Frank and Ran Xu. "Causal inference for social network analysis." In: *The Oxford Handbook of Social Networks*. Oxford University Press, 2020, pp. 288–310. DOI: 10.1080/01621459.2022.2131557 (page 200).

[96] Tingran Gao, Shahab Asoodeh, Yi Huang, and James Evans. "Wasserstein soft label propagation on hypergraphs: Algorithm and generalization error bounds." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2019, pp. 3630–3637. DOI: 10.1609/aaai.v33i01.33013630 (page 111).

[97] Kiran Garimella, Gianmarco de Francisci Morales, Aristides Gionis, and Michael Mathioudakis. "Reducing controversy by connecting opposing views." In: *Proceedings of the ACM International Conference on Web Search and Data Mining*. 2017, pp. 81–90. DOI: 10.1145/3018661.3018703 (page 161).

[98] Kiran Garimella, Gianmarco de Francisci Morales, Aristides Gionis, and Michael Mathioudakis. "Quantifying controversy on social media." In: *ACM Transactions on Social Computing* 1.1 (2018), pp. 1–27. DOI: 10.1145/3140565 (page 161).

[99] Timnit Gebru et al. "Datasheets for datasets." In: *Communications of the ACM* 64.12 (2021), pp. 86–92. DOI: 10.1145/3458723 (pages 61, 70, 201).

[100] Debarghya Ghoshdastidar, Maurilio Gutzeit, Alexandra Carpentier, and Ulrike Von Luxburg. "Two-sample hypothesis testing for inhomogeneous random graphs." In: *The Annals of Statistics* 48.4 (2020), pp. 2208–2229. DOI: 10.1214/19-AOS1884 (page 46).

[101] Alison L. Gibbs and Francis Edward Su. "On choosing and bounding probability metrics." In: *International Statistical Review* 70.3 (2002), pp. 419–435. DOI: 10.2307/1403865 (page 106).

[102] Cedric E. Ginestet, Jun Li, Prakash Balachandran, Steven Rosenberg, Eric D. Kolaczyk, et al. "Hypothesis testing for network data in functional neuroimaging." In: *The Annals of Applied Statistics* 11.2 (2017), pp. 725–750. DOI: 10.1214/16-AOAS1015 (page 46).

[103] Aristides Gionis, Evimaria Terzi, and Panayiotis Tsaparas. "Opinion maximization in social networks." In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 2013, pp. 387–395. DOI: 10.1137/1.9781611972832.4 (page 161).

[104] Sebastian Goebl, Annika Tonch, Christian Böhm, and Claudia Plant. "MeGS: Partitioning meaningful subgraph structures using Minimum Description Length." In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2016, pp. 889–894. DOI: 10.1109/ICDM.2016.0108 (page 24).

[105] Guilherme Gomes, Vinayak Rao, and Jennifer Neville. "Multi-level hypothesis testing for populations of heterogeneous networks." In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2018, pp. 977–982. DOI: 10.1109/ICDM.2018.00121 (page 46).

[106] Adam Gosztolai and Alexis Arnaudon. "Unfolding the multiscale structure of networks with dynamical Ollivier–Ricci curvature." In: *Nature Communications* 12.1 (2021), p. 4561. DOI: 10.1038/s41467-021-24884-1 (page 98).

[107] Raymond Greenlaw and Rossella Petreschi. "Cubic graphs." In: *ACM Computing Surveys (CSUR)* 27.4 (1995), pp. 471–495. DOI: 10.1145/234782.234783 (page 145).

[108] Arthur Gretton, Karsten Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex Smola. "A kernel method for the two-sample-problem." In: *Advances in Neural Information Processing Systems*. 2006, pp. 513–520. DOI: 10.48550/arXiv.0805.2368 (page 119).

[109] Martin Grohe and Pascal Schweitzer. "The graph isomorphism problem." In: *Communications of the ACM* 63.11 (2020), pp. 128–134. DOI: 10.1145/3372123 (page 9).

[110] Peter Grünwald. *The Minimum Description Length principle*. Cambridge: MIT Press, 2007. DOI: 10.7551/mitpress/4643.001.0001 (page 12).

[111] Mauricio Gruppi, Benjamin D. Horne, and Sibel Adali. *NELA-GT-2021: A large multi-labelled news dataset for the study of misinformation in news articles*. 2021. arXiv: 2203.05659 [cs.CY] (page 164).

[112] Fabrice Guillet and Howard J. Hamilton, eds. *Quality measures in data mining*. Springer, 2007. DOI: 10.1007/978-3-540-44918-8 (page 201).

[113] Odd Erik Gundersen and Sigbjørn Kjensmo. "State of the art: Reproducibility in artificial intelligence." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2018, pp. 1644–1651. DOI: 10.1609/aaai.v32i1.11503 (page 201).

[114] Shahrzad Haddadan, Cristina Menghini, Matteo Riondato, and Eli Upfal. "Reducing polarization and increasing diverse navigability in graphs by inserting edges and swapping edge weights." In: *Data Mining and Knowledge Discovery* 36 (2022), pp. 2334–2378. DOI: 10.1007/s10618-022-00875-8 (page 162).

[115] Kathrin Hanauer, Monika Henzinger, and Christian Schulz. "Recent advances in fully dynamic graph algorithms – A quick reference guide." In: *ACM Journal of Experimental Algorithmics* 27 (2022), pp. 1–45. DOI: 10.1145/3555806 (page 1).

[116] Frank Harary. *Graph Theory*. New York: Addison-Wesley, 1969 (page 1).

[117] Frank Harary and Gopal Gupta. "Dynamic graph models." In: *Mathematical and Computer Modelling* 25.7 (1997), pp. 79–87. DOI: 10.1016/S0895-7177(97)00050-2 (pages 1, 200).

[118] Ye He, Lisa Byrge, and Daniel P. Kennedy. "Nonreplication of functional connectivity differences in autism spectrum disorder across multiple sites and denoising strategies." In: *Human Brain Mapping* 41.5 (2020), pp. 1334–1350. DOI: 10.1002/hbm.24879 (page 49).

[119] Lawrence B. Holder et al. "Current and future challenges in mining large networks: Report on the second SDM workshop on mining networks and graphs." In: *ACM SIGKDD Explorations Newsletter* 18.1 (2016), pp. 39–45. DOI: 10.1145/2980765.2980770 (page 201).

[120] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia S. Chmielinski. "The dataset nutrition label." In: *Data Protection and Privacy*. 2020, pp. 1–27 (page 201).

[121] Petter Holme and Jari Saramäki. "Temporal networks." In: *Physics Reports* 519.3 (2012), pp. 97–125. DOI: 10.1016/j.physrep.2012.03.001 (page 200).

[122] Seok-Jun Hong et al. "Toward neurosubtypes in autism." In: *Biological Psychiatry* 88.1 (2020), pp. 111–128. DOI: 10.1016/j.biopsych.2020.03.022 (page 50).

[123] Homa Hosseinmardi et al. "Examining the consumption of radical content on YouTube." In: *Proceedings of the National Academy of Sciences* 118.32 (2021), e2101967118. DOI: 10.1073/pnas.21019671 (page 140).

[124] Weihua Hu et al. *Open Graph Benchmark: Datasets for machine learning on graphs*. 2020. arXiv: 2005.00687 [cs.LG] (page 61).

[125] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets." In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2008, pp. 263–272. DOI: 10.1109/ICDM.2008.22 (page 188).

[126] Jing Huang and Jie Yang. "UniGNN: A unified framework for graph and hypergraph neural networks." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2021, pp. 2563–2569. DOI: 10.48550/arXiv.2105.00956 (page 111).

[127] Kexin Huang and Marinka Zitnik. "Graph meta learning via local subgraphs." In: *Advances in Neural Information Processing Systems*. 2020, pp. 5862–5874. DOI: 10.48550/arXiv.2006.07889 (page 9).

[128] Jocelyn V. Hull et al. "Resting-state functional connectivity in autism spectrum disorders: A review." In: *Frontiers in Psychiatry* 7 (2017), p. 205. DOI: 10.3389/fpsyt.2016.00205 (page 38).

[129] Jessica Hullman, Sayash Kapoor, Priyanka Nanayakkara, Andrew Gelman, and Arvind Narayanan. "The worst of both worlds: A comparative analysis of errors in learning from data in psychology and machine learning." In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 2022, pp. 335–348. DOI: 10.1145/3514094.3534196 (page 201).

[130] Eslam Hussein, Prerna Juneja, and Tanushree Mitra. "Measuring misinformation in video search platforms: An audit study on YouTube." In: *Proceedings of the ACM on Human-Computer Interaction (CHI)*. 2020, pp. 1–27. DOI: 10.1145/3392854 (page 139).

[131] Masahiro Ikeda, Yu Kitabeppu, Yuuki Takai, and Takato Uehara. *Coarse Ricci curvature of hypergraphs and its generalization*. 2021. arXiv: 2102.00698 [math.MG] (page 110).

[132] Victor P. Il'ev. "An approximation guarantee of the greedy descent algorithm for minimizing a supermodular set function." In: *Discrete Applied Mathematics* 114.1-3 (2001), pp. 131–146. DOI: 10.1016/S0166-218X(00)00366-8 (page 153).

[133] Neil Immerman. *Descriptive complexity*. New York: Springer Science & Business Media, 1998. DOI: 10.1007/978-1-4612-0539-5 (page 201).

[134] Mohammad Hossein Jarrahi, Ali Memariani, and Shion Guha. *The principles of data-centric AI (DCAI)*. 2022. arXiv: 2211.14611 [cs.LG] (page 201).

[135] Kalervo Järvelin and Jaana Kekäläinen. "Cumulated gain-based evaluation of IR techniques." In: *ACM Transactions on Information Systems (TOIS)* 20.4 (2002), pp. 422–446. DOI: 10.1145/582415.582418 (page 144).

[136] Edwin T. Jaynes. "On the rationale of maximum-entropy methods." In: *Proceedings of the IEEE* 70.9 (1982), pp. 939–952. DOI: 10.1109/PROC.1982.12425 (page 40).

[137] Jürgen Jost and Shiping Liu. "Ollivier's Ricci curvature, local clustering and curvature-dimension inequalities on graphs." In: *Discrete & Computational Geometry* 51.2 (2014), pp. 300–322. DOI: 10.1007/s00454-013-9558-1 (pages 100, 110).

[138] F. Kaden. "Graph distances and similarity." In: *Topics in Combinatorics and Graph Theory*. Springer, 1990, pp. 397–404. DOI: 10.1007/978-3-642-46908-4_45 (page 24).

[139] Sarang Kapoor, Dhish Kumar Saxena, and Matthijs van Leeuwen. "Online summarization of dynamic graphs using subjective interestingness for sequential data." In: *Data Mining and Knowledge Discovery* 35 (2020), pp. 1–39. DOI: 10.1007/s10618-020-00714-8 (page 24).

[140] Daniel Martin Katz, Corinna Coupette, Janis Beckedorf, and Dirk Hartung. "Complex societies and the growth of the law." In: *Scientific Reports* 10 (2020), p. 18737. DOI: 10.1038/s41598-020-73623-x (pages 6, 7).

[141] Xiangyu Ke, Arijit Khan, and Francesco Bonchi. "Multi-relation graph summarization." In: *ACM Transactions on Knowledge Discovery from Data* 16.5 (2022), pp. 1–30. DOI: 10.1145/3494561 (page 39).

[142] Mark Kempton, Gabor Lippner, and Florentin Münch. "Large scale Ricci curvature on graphs." In: *Calculus of Variations and Partial Differential Equations* 59.5 (2020), pp. 1–17. DOI: 10.1007/s00526-020-01829-y (page 110).

[143] Subhash Khot. "On the power of unique 2-prover 1-round games." In: *Proceedings of the Annual ACM Symposium on Theory of Computing (STOC)*. 2002, pp. 767–775. DOI: 10.1145/509907.510017 (page 151).

[144] Subhash Khot and Oded Regev. "Vertex cover might be hard to approximate to within $2-\varepsilon$." In: *Journal of Computer and System Sciences* 74.3 (2008), pp. 335–349. DOI: 10.1016/j.jcss.2007.06.019 (page 151).

[145] Jace B. King et al. "Generalizability and reproducibility of functional connectivity in autism." In: *Molecular Autism* 10.1 (2019), pp. 1–23. DOI: 10.1186/s13229-019-0273-5 (page 49).

[146] Baris Kirdemir and Nitin Agarwal. "Exploring bias and information bubbles in YouTube's video recommendation networks." In: *Proceedings of the International Conference on Complex Networks and Their Applications*. 2022, pp. 166–177. DOI: 10.1007/978-3-030-93413-2_15 (page 140).

[147] Mikko Kivelä et al. "Multilayer networks." In: *Journal of Complex Networks* 2.3 (2014), pp. 203–271. DOI: 10.1093/comnet/cnu016 (pages 1, 200).

[148] Donald E. Knuth. *The Stanford GraphBase: A platform for combinatorial computing*. New York: ACM Press, 1993. DOI: 10.1145/164984 (page 62).

[149] Phokion G Kolaitis, Jonathan Panttaja, and Wang-Chiew Tan. "The complexity of data exchange." In: *Proceedings of the ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*. 2006, pp. 30–39. DOI: 10.1145/1142351.1142357 (page 201).

[150] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. "Summarizing and understanding large graphs." In: *Statistical Analysis and Data Mining* 8.3 (2015), pp. 183–202. DOI: 10.1002/sam.11267 (pages 24–27, 39).

[151] Danai Koutra, Neil Shah, Joshua T. Vogelstein, Brian Gallagher, and Christos Faloutsos. "DeltaCon: Principled massive-graph similarity function with attribution." In: *ACM Transactions on Knowledge Discovery from Data* 10.3 (2016), pp. 1–43. DOI: 10.1145/2824443 (page 24).

[152] Nils M. Kriege, Fredrik D. Johansson, and Christopher Morris. "A survey on graph kernels." In: *Applied Network Science* 5.1 (2020), p. 6. DOI: 10.1007/s41109-019-0195-3 (page 111).

[153] Fabian Kuhn and Rotem Oshman. "Dynamic networks: Models and algorithms." In: *ACM SIGACT News* 42.1 (2011), pp. 82–96. DOI: 10.1145/1959045.1959064 (page 200).

[154] Suprateek Kundu, Jin Ming, Joe Nocera, and Keith M. McGregor. "Integrative learning for population of dynamic networks with covariates." In: *NeuroImage* 236 (2021), p. 118181. DOI: 10.1016/j.neuroimage.2021.118181 (page 46).

[155] Jérôme Kunegis. "KONECT: The Koblenz Network Collection." In: *Proceedings of the International Conference on World Wide Web*. 2013, pp. 1343–1350. DOI: 10.1145/2487788.2488173 (page 61).

[156] Tommaso Lanciano, Francesco Bonchi, and Aristides Gionis. "Explainable classification of brain networks via contrast subgraphs." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2020, pp. 3308–3318. DOI: 10.1145/3394486.3403383 (pages 46, 47, 50).

[157] Wilmer Leal, Marzieh Eidi, and Jürgen Jost. "Curvature-based analysis of directed hypernetworks." In: *Complex Networks* (2019), pp. 10–12 (page 110).

[158] Wilmer Leal, Marzieh Eidi, and Jürgen Jost. "Ricci curvature of random and empirical directed hypernetworks." In: *Applied Network Science* 5.1 (2020), p. 65. DOI: 10.1007/s41109-020-00309-8 (pages 98, 110).

[159] Wilmer Leal, Guillermo Restrepo, Peter F. Stadler, and Jürgen Jost. "Forman–Ricci curvature for hypergraphs." In: *Advances in Complex Systems* 24.1 (2021), 2150003:1–2150003:24. DOI: 10.1142/S021952592150003X (page 110).

[160] Mark Ledwich and Anna Zaitsev. "Algorithmic extremism: Examining YouTube's rabbit hole of radicalization." In: *First Monday* (2020). DOI: 10.5210/fm.v25i3.10419 (page 140).

[161] Mark Ledwich, Anna Zaitsev, and Anton Laukemper. "Radical bubbles on YouTube? Revisiting algorithmic extremism with personalised recommendations." In: *First Monday* (2022). DOI: 10.5210/fm.v27i12.12552 (page 140).

[162] Geon Lee, Jihoon Ko, and Kijung Shin. "Hypergraph motifs: Concepts, algorithms, and discoveries." In: *Proceedings of the VLDB Endowment* 13.12 (2020), pp. 2256–2269. DOI: 10.14778/3407790.3407823 (page 111).

[163] Geon Lee and Kijung Shin. "THyMe+: Temporal hypergraph motifs and fast algorithms for exact counting." In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2021, pp. 310–319. DOI: 10.1109/ICDM51629.2021.00042 (page 111).

[164] John Boaz Lee, Ryan Rossi, and Xiangnan Kong. "Graph classification using structural attention." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2018, pp. 1666–1674. DOI: 10.1145/3219819.3219980 (page 39).

[165] B.C.L. Lehmann, R.N. Henson, L. Geerligs, S.R. White, et al. "Characterising group-level brain connectivity: A framework using Bayesian exponential random graph models." In: *NeuroImage* 225 (2021), p. 117480. DOI: 10.1016/j.neuroimage.2020.117480 (page 46).

[166] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. "Graph evolution: Densification and shrinking diameters." In: *ACM Transactions on Knowledge Discovery from Data* 1.1 (2007), 2:1–2:41. DOI: 10.1145/1217299.1217301 (pages 25, 26).

[167] Jure Leskovec and Rok Sosič. "SNAP: A general-purpose network analysis and graph-mining library." In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 8.1 (2016), pp. 1–20. DOI: 10.1145/2898361 (pages 1, 61).

[168] M. Li and P. Vitányi. *An Introduction to Kolmogorov Complexity and its Applications*. New York: Springer, 1993. DOI: 10.1007/978-0-387-49820-1 (page 12).

[169] Ming Li, Xin Chen, Xin Li, Bin Ma, and P.M.B. Vitanyi. "The similarity metric." In: *IEEE Transactions on Information Technology* 50.12 (2004), pp. 3250–3264. DOI: 10.1109/TIT.2004.838101 (page 12).

[170] Yongsub Lim, U Kang, and Christos Faloutsos. "SlashBurn: Graph compression and mining beyond caveman communities." In: *IEEE Transactions on Knowledge and Data Engineering* 26.12 (2014), pp. 3077–3089. DOI: 10.1109/TKDE.2014.2320716 (pages 20, 24).

[171] Yong Lin, Linyuan Lu, and Shing-Tung Yau. "Ricci curvature of graphs." In: *Tohoku Mathematical Journal, Second Series* 63.4 (2011), pp. 605–627. DOI: 10.2748/tmj/1325886283 (pages 100, 110, 111).

[172] Renming Liu et al. *Towards a taxonomy of graph learning datasets*. 2021. arXiv: 2110.14809 [cs.LG] (page 201).

[173] Shiping Liu, Florentin Münch, Norbert Peyerimhoff, and Christian Rose. "Distance bounds for graphs with some negative Bakry-Émery curvature." In: *Analysis and Geometry in Metric Spaces* 7.1 (2019), pp. 1–14. DOI: 10.1515/agms-2019-0001 (page 110).

[174] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. "Graph summarization methods and applications: A survey." In: *ACM Computing Surveys (CSUR)* 51.3 (2018), pp. 1–34. DOI: 10.1145/3186727 (page 39).

[175] Felipe Llinares-López, Mahito Sugiyama, Laetitia Papaxanthos, and Karsten Borgwardt. "Fast and memory-efficient significant pattern mining via permutation testing." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2015, pp. 725–734. DOI: 10.1145/2783258.2783363 (page 46).

[176] Paulett Lloyd, Matthew C. Mahutga, and Jan de Leeuw. "Looking back and forging ahead: Thirty years of social network research on the world-system." In: *Journal of World-Systems Research* (2009), pp. 48–85. DOI: 10.5195/jwsr.2009.335 (page 53).

[177] Catherine Lord et al. "Autism spectrum disorder." In: *Nature Reviews Disease Primers* 6.1 (2020), pp. 1–23. DOI: 10.1038/s41572-019-0138-4 (page 38).

[178] Ilenia Lovato, Alessia Pini, Aymeric Stamm, Maxime Taquet, and Simone Vantini. "Multiscale null hypothesis testing for network-valued data: Analysis of brain networks of patients with autism." In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 70.2 (2021), pp. 372–397. DOI: 10.1111/rssc.12463 (page 46).

[179] Ilenia Lovato, Alessia Pini, Aymeric Stamm, and Simone Vantini. "Model-free two-sample test for network-valued data." In: *Computational Statistics and Data Analysis* 144 (2020), p. 106896. DOI: 10.1016/j.csda.2019.106896 (page 46).

[180] Joshua Lukemire, Suprateek Kundu, Giuseppe Pagnoni, and Ying Guo. "Bayesian joint modeling of multiple brain functional networks." In: *Journal of the American Statistical Association* (2020), pp. 1–13. DOI: 10.1080/01621459.2020.1796357 (page 46).

[181] Simón Lunagómez, Sofia C. Olhede, and Patrick J. Wolfe. "Modeling network populations via graph distances." In: *Journal of the American Statistical Association* (2020), pp. 1–18. DOI: 10.1080/01621459.2020.1763803 (page 46).

[182] Guixiang Ma, Nesreen K. Ahmed, Theodore L. Willke, and Philip S. Yu. "Deep graph similarity learning: A survey." In: *Data Mining and Knowledge Discovery* 35 (2021), pp. 688–725. DOI: 10.1007/s10618-020-00733-5 (page 24).

[183] Peter Macgregor and He Sun. "Finding bipartite components in hypergraphs." In: *Advances in Neural Information Processing Systems*. 2021, pp. 7912–7923. DOI: 10.48550/arXiv.2205.02771 (page 111).

[184] Robin Mamié, Manoel Horta Ribeiro, and Robert West. "Are anti-feminist communities gateways to the far right? Evidence from Reddit and YouTube." In: *Proceedings of the ACM Web Science Conference*. 2021, pp. 139–147. DOI: 10.1145/3447535.3462504 (pages 161, 164).

[185] Michael Mampaey, Jilles Vreeken, and Nikolaj Tatti. "Summarizing data succinctly with the most informative itemsets." In: *ACM Transactions on Knowledge Discovery from Data* 6.4 (2012), pp. 1–44. DOI: 10.1145/2382577.2382580 (page 40).

[186] Alessio Martino and Antonello Rizzi. "(Hyper)graph kernels over simplicial complexes." In: *Entropy* 22.10 (2020), p. 1155. DOI: 10.3390/e22101155 (page 111).

[187] P.-A.G. Maugis, Sofia C. Olhede, Carey E. Priebe, and Patrick J. Wolfe. "Testing for equivalence of network distribution using subgraph counts." In: *Journal of Computational and Graphical Statistics* 29.3 (2020), pp. 455–465. DOI: 10.1080/10618600.2020.1736085 (page 46).

[188] Charalampos Mavroforakis, Michael Mathioudakis, and Aristides Gionis. "Absorbing random-walk centrality: Theory and algorithms." In: *Proceedings of the IEEE International Conference on Data Mining (ICDM)*. 2015, pp. 901–906. DOI: 10.1109/ICDM.2015.103 (page 161).

[189] Timon McPhearson et al. "A social-ecological-technological systems framework for urban ecosystem services." In: *One Earth* 5.5 (2022), pp. 505–518. DOI: 10.1016/j.oneear.2022.04.007 (page 200).

[190] Miller McPherson, Lynn Smith-Lovin, and James M. Cook. "Birds of a feather: Homophily in social networks." In: *Annual Review of Sociology* (2001), pp. 415–444. DOI: 10.1146/annurev.soc.27.1.415 (page 164).

[191] Sourav Medya, Arlei Silva, Ambuj Singh, Prithwish Basu, and Ananthram Swami. "Group centrality maximization via network design." In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 2018, pp. 126–134. DOI: 10.1137/1.9781611975321.14 (page 160).

[192] Marco Minici, Federico Cinus, Corrado Monti, Francesco Bonchi, and Giuseppe Manco. "Cascade-based echo chamber detection." In: *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*. 2022, pp. 1511–1520. DOI: 10.1145/3511808.3557253 (page 161).

[193] Jacob Levy Moreno. *Who shall survive?: A new approach to the problem of human interrelations*. Washington: Nervous and Mental Disease Publishing Company, 1934 (page 1).

[194] Christopher Morris et al. "TUDataset: A collection of benchmark datasets for learning with graphs." In: *ICML Workshop on Graph Representation Learning and Beyond (GRL+)*. 2020. DOI: 10.48550/arXiv.2007.08663. URL: http://www.graphlearning.io (page 61).

[195] Barbara Mowat, Paul Werstine, Michael Poston, and Rebecca Niles, eds. *Shakespeare's plays, sonnets and poems*. The Folger Shakespeare Library. URL: https://shakespeare.folger.edu (visited on 05/29/2022) (pages 59, 64, 69, 95).

[196] Soumendu Sundar Mukherjee, Purnamrita Sarkar, and Lizhen Lin. "On clustering network-valued data." In: *Advances in Neural Information Processing Systems*. 2017, pp. 7074–7084. DOI: 10.48550/arXiv.1606.02401 (pages 39, 46).

[197] Florentin Münch and Christian Rose. "Spectrally positive Bakry-Émery Ricci curvature on graphs." In: *Journal de Mathématiques Pures et Appliquées* 143 (2020), pp. 334–344. DOI: 10.1016/j.matpur.2020.03.008 (page 110).

[198] Kevin A. Murgas, Emil Saucan, and Romeil Sandhu. "Hypergraph geometry reflects higher-order dynamics in protein interaction networks." In: *Scientific Reports* 12.1 (2022), p. 20879. DOI: 10.1038/s41598-022-24584-w (page 110).

[199] Sumner Byron Myers. "Riemannian manifolds with positive mean curvature." In: *Duke Mathematical Journal* 8.2 (1941), pp. 401–404. DOI: 10.1215/S0012-7094-41-00832-3 (page 107).

[200] George L. Nemhauser, Laurence A. Wolsey, and Marshall L. Fisher. "An analysis of approximations for maximizing submodular set functions—I." In: *Mathematical Programming* 14.1 (1978), pp. 265–294. DOI: 10.1007/BF01588971 (page 153).

[201] Mark E.J. Newman. "The structure and function of complex networks." In: *SIAM Review* 45.2 (2003), pp. 167–256. DOI: 10.1137/S003614450342480 (page 1).

[202] Giannis Nikolentzos, Polykarpos Meladianos, Stratis Limnios, and Michalis Vazirgiannis. "A degeneracy framework for graph similarity." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2018, pp. 2595–2601. DOI: 10.24963/ijcai.2018/360 (page 24).

[203] Ana Rita Nogueira, Andrea Pugnana, Salvatore Ruggieri, Dino Pedreschi, and Joao Gama. "Methods and tools for causal discovery and causal inference." In: *Wiley Interdisciplinary Reviews Data Mining and Knowledge Discovery* 12.2 (2022), e1449. DOI: 10.1002/widm.1449 (page 200).

[204] Jason S. Nomi and Lucina Q. Uddin. "Developmental changes in large-scale network connectivity in autism." In: *NeuroImage: Clinical* 7 (2015), pp. 732–741. DOI: 10.1016/j.nicl.2015.02.024 (page 50).

[205] Leslie O'Bray, Max Horn, Bastian Rieck, and Karsten Borgwardt. "Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2022. DOI: 10.48550/arXiv.2106.01098 (page 201).

[206] Lutz Oettershagen, Petra Mutzel, and Nils M. Kriege. "Temporal walk centrality: Ranking nodes in evolving networks." In: *Proceedings of the ACM Web Conference*. 2022, pp. 1640–1650. DOI: 10.1145/3485447.3512210 (page 161).

[207] Seongmin Ok. "A graph similarity for deep learning." In: *Advances in Neural Information Processing Systems*. 2020, pp. 1–12 (page 24).

[208] Yann Ollivier. "Ricci curvature of Markov chains on metric spaces." In: *Journal of Functional Analysis* 256.3 (2009), pp. 810–864. DOI: 10.1016/j.jfa.2008.11.001 (pages 100, 110).

[209] Yann Ollivier. "Ricci curvature of metric spaces." In: *Comptes Rendus Mathématique* 345.11 (2007), pp. 643–646. DOI: 10.1016/j.crma.2007.10.041 (pages 100, 110).

[210] John Palowitch, Anton Tsitsulin, Brandon Mayer, and Bryan Perozzi. "Graphworld: Fake graphs bring real insights for GNNs." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2022, pp. 3691–3701. DOI: 10.1145/3534678.3539203 (page 201).

[211] Kostantinos Papadamou et al. ""It is just a flu": Assessing the effect of watch history on YouTube's pseudoscientific video recommendations." In: *Proceedings of the International AAAI Conference on Web and Social Media*. 2022, pp. 723–734. DOI: 10.1609/icwsm.v16i1.19329 (page 139).

[212] Nikos Parotsidis, Evaggelia Pitoura, and Panayiotis Tsaparas. "Selecting shortcuts for a smaller world." In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 2015, pp. 28–36. DOI: 10.1137/1.9781611974010. (page 160).

[213] Bibek Paudel and Abraham Bernstein. "Random walks with erasure: Diversifying personalized recommendations on social and information networks." In: *Proceedings of the ACM Web Conference*. 2021, pp. 2046–2057. DOI: 10.1145/3442381.3449970 (page 161).

[214] Judea Pearl. *Causality*. Cambridge: Cambridge University Press, 2009. DOI: 10.1017/CBO9780511803161 (page 200).

[215] Tiago P. Peixoto. *The Netzschleuder network catalogue and repository*. 2020. DOI: 10.5281/zenodo.7839981. URL: https://networks.skewed.de/ (page 61).

[216] Niccolo Pescetelli, Daniel Barkoczi, and Manuel Cebrian. "Bots influence opinion dynamics without direct human-bot interaction: The mediating role of recommender systems." In: *Applied Network Science* 7.1 (2022), p. 46. DOI: 10.1007/s41109-022-00488-6 (page 140).

[217] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: Foundations and learning algorithms*. Cambridge: MIT Press, 2017 (page 200).

[218] Joseph J. Pfeiffer III, Sebastian Moreno, Timothy La Fond, Jennifer Neville, and Brian Gallagher. "Attributed graph models: Modeling network structure with correlated attributes." In: *Proceedings of the International Conference on World Wide Web*. 2014, pp. 831–842. DOI: 10.1145/2566486.2567993 (page 200).

[219] Joelle Pineau et al. "Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program)." In: *Journal of Machine Learning Research* 22.1 (2021), pp. 7459–7478. DOI: 10.48550/arXiv.2003.12206 (page 201).

[220] Edward Raff. "A step toward quantifying independently reproducible machine learning research." In: *Advances in Neural Information Processing Systems*. 2019, pp. 5486–5496. DOI: 10.48550/arXiv.1909.06674 (page 201).

[221] Sayan Ranu and Ambuj K. Singh. "Graphsig: A scalable approach to mining significant subgraphs in large graph databases." In: *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*. 2009, pp. 844–855. DOI: 10.1109/ICDE.2009.133 (page 39).

[222] John W. Raymond, Eleanor J. Gardiner, and Peter Willett. "Rascal: Calculation of graph similarity using maximum common edge subgraphs." In: *The Computer Journal* 45.6 (2002), pp. 631–644. DOI: 10.1093/comjnl/45.6.631 (page 24).

[223] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence embeddings using siamese BERT-networks." In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410 (page 189).

[224] Manoel Horta Ribeiro, Raphael Ottoni, Robert West, Virgílio A.F. Almeida, and Wagner Meira Jr. "Auditing radicalization pathways on YouTube." In: *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*. 2020, pp. 131–141. DOI: 10.1145/3351095.3372879 (pages 140, 161, 164, 170).

[225] Bastian Rieck. *On the expressivity of persistent homology in graph learning*. 2023. arXiv: 2302.09826 [cs.LG] (page 1).

[226] Bastian Rieck and Corinna Coupette. *Evaluating the "Learning on Graphs" conference experience*. 2023. arXiv: 2306.00586 [cs.LG] (pages 7, 201).

[227] Jorma Rissanen. "A universal prior for integers and estimation by Minimum Description Length." In: *The Annals of Statistics* 11.2 (1983), pp. 416–431. DOI: 10.1214/aos/1176346150 (page 15).

[228] Ronald E. Robertson et al. "Auditing partisan audience bias within google search." In: *Proceedings of the ACM on Human-Computer Interaction (CHI)*. 2018, pp. 1–22. DOI: 10.1145/3274417 (page 139).

[229] Edmund T. Rolls, Chu-Chung Huang, Ching-Po Lin, Jianfeng Feng, and Marc Joliot. "Automated anatomical labelling atlas 3." In: *NeuroImage* 206 (2020), p. 116189. DOI: 10.1016/j.neuroimage.2019.116189 (page 50).

[230] Ryan Rossi and Nesreen Ahmed. "The Network Data Repository with Interactive Graph Analytics and Visualization." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2015, pp. 4292–4293. DOI: 10.1609/aaai.v29i1.9277 (page 61).

[231] Indrava Roy, Sudharsan Vijayaraghavan, Sarath Jyotsna Ramaia, and Areejit Samal. "Forman–Ricci curvature and persistent homology of unweighted complex networks." In: *Chaos, Solitons & Fractals* 140 (2020), p. 110260. DOI: 10.1016/j.chaos.2020.110260 (page 110).

[232] Michael Alan Sacks, Marc J. Ventresca, and Brian Uzzi. "Global institutions and networks: Contingent change in the structure of world trade advantage, 1965–1980." In: *American Behavioral Scientist* 44.10 (2001), pp. 1579–1601. DOI: 10.1177/000276401219580 (page 53).

[233] Areejit Samal et al. "Comparative analysis of two discretizations of Ricci curvature for complex networks." In: *Scientific Reports* 8.1 (2018), pp. 1–16. DOI: 10.1038/s41598-018-27001-3 (pages 98, 111).

[234] Emil Saucan and Melanie Weber. "Forman's Ricci curvature – From networks to hypernetworks." In: *Proceedings of the International Conference on Complex Networks and Their Applications*. 2018, pp. 706–717. DOI: 10.1007/978-3-030-05411-3_56 (page 110).

[235] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Kernel principal component analysis." In: *Proceedings of the International Conference on Artificial Neural Networks (ICANN)* (1997), pp. 583–588. DOI: 10.1007/BFb0020217 (page 119).

[236] Gideon Schwarz. "Estimating the dimension of a model." In: *The Annals of Statistics* (1978), pp. 461–464. DOI: 10.1214/aos/1176344136 (page 41).

[237] Neil Shah, Danai Koutra, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. "Timecrunch: Interpretable dynamic graph summarization." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2015, pp. 1055–1064. DOI: 10.1145/2783258.2783321 (pages 24, 46).

[238] William Shakespeare. *The Complete Works of Shakespeare*. Ed. by W.J. Craig. Oxford: Oxford University Press, 1916 (page 95).

[239] Jack Sherman and Winifred J. Morrison. "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix." In: *The Annals of Mathematical Statistics* 21.1 (1950), pp. 124–127. DOI: 10.1214/aoms/1177729893 (page 154).

[240] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten Borgwardt. "Weisfeiler-Lehman graph kernels." In: *Journal of Machine Learning Research* 12.9 (2011), pp. 2539–2561 (page 39).

[241] Nino Shervashidze, S.V.N. Vishwanathan, Tobias Petri, Kurt Mehlhorn, and Karsten Borgwardt. "Efficient graphlet kernels for large graph comparison." In: *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2009, pp. 488–495 (page 24).

[242] Mirko Signorelli and Ernst C. Wit. "Model-based clustering for populations of networks." In: *Statistical Modelling* 20.1 (2020), pp. 9–29. DOI: 10.1177/1471082X1987112 (pages 39, 46).

[243] Larissa Spinelli and Mark Crovella. "Closed-loop opinion formation." In: *Proceedings of the ACM Web Science Conference*. 2017, pp. 73–82. DOI: 10.1109/TCNS.2021.3105616 (page 139).

[244] David I Spivak. *Category theory for the sciences*. Cambridge: MIT Press, 2014 (page 201).

[245] Ivan Srba et al. "Auditing YouTube's recommendation algorithm for misinformation filter bubbles." In: *ACM Transactions on Recommender Systems* 1.1 (2023), pp. 1–33. DOI: 10.1145/3568392 (page 140).

[246] Balasubramaniam Srinivasan, Da Zheng, and George Karypis. "Learning over families of sets—Hypergraph representation learning for higher order tasks." In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 2021, pp. 756–764. DOI: 10.1137/1.9781611976700.85 (page 62).

[247] Stavros K. Stavroglou, Athanasios A Pantelous, H Eugene Stanley, and Konstantin M Zuev. "Unveiling causal interactions in complex systems." In: *Proceedings of the National Academy of Sciences* 117.14 (2020), pp. 7599–7605. DOI: 10.1073/pnas.191826911 (page 200).

[248] Mahito Sugiyama, Felipe Llinares López, Niklas Kasenburg, and Karsten Borgwardt. "Significant subgraph mining with multiple testing correction." In: *Proceedings of the SIAM International Conference on Data Mining (SDM)*. 2015, pp. 37–45. DOI: 10.1137/1.9781611974010.5 (pages 39, 46, 47).

[249] Jimeng Sun, Christos Faloutsos, Spiros Papadimitriou, and Philip S. Yu. "GraphScope: parameter-free mining of large time-evolving graphs." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2007, pp. 687–696. DOI: 10.1145/1281192.1281266 (page 24).

[250] Kaustubh Supekar et al. "Brain hyperconnectivity in children with autism and its links to social deficits." In: *Cell Reports* 5.3 (2013), pp. 738–747. DOI: 10.1016/j.celrep.2013.10.001 (page 50).

[251] Nikolaj Tatti and Jilles Vreeken. "Comparing apples and oranges: Measuring differences between exploratory data mining results." In: *Data Mining and Knowledge Discovery* 25 (2012), pp. 173–207. DOI: 10.1007/s10618-012-0275-9 (page 201).

[252] Jeyan Thiyagalingam, Mallikarjun Shankar, Geoffrey Fox, and Tony Hey. "Scientific machine learning benchmarks." In: *Nature Reviews Physics* 4.6 (2022), pp. 413–420. DOI: 10.1038/s42254-022-00441-7 (page 201).

[253] Matteo Togninalli, Elisabetta Ghisu, Felipe Llinares-López, Bastian Rieck, and Karsten Borgwardt. "Wasserstein Weisfeiler-Lehman graph kernels." In: *Advances in Neural Information Processing Systems*. 2019, pp. 6439–6449. DOI: 10.48550/arXiv.1906.01277 (page 24).

[254] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. "Understanding over-squashing and bottlenecks on graphs via curvature." In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2022. DOI: 10.48550/arXiv.2111.14522 (pages 98, 110).

[255] Leo Torres, Ann S. Blevins, Danielle Bassett, and Tina Eliassi-Rad. "The why, how, and when of representations for complex systems." In: *SIAM Review* 63.3 (2021), pp. 435–485. DOI: 10.1137/20M1355896 (pages 1, 61).

[256] Sotiris Tsioutsiouliklis, Evaggelia Pitoura, Konstantinos Semertzidis, and Panayiotis Tsaparas. "Link recommendations for PageRank fairness." In: *Proceedings of the ACM Web Conference*. 2022, pp. 3541–3551. DOI: 10.1145/3485447.3512249 (page 161).

[257] Martijn P. van den Heuvel and Olaf Sporns. "A cross-disorder connectome landscape of brain dysconnectivity." In: *Nature Reviews Neuroscience* 20.7 (2019), pp. 435–446. DOI: 10.1038/s41583-019-0177-6 (page 49).

[258] Nate Veldt, Anthony Wirth, and David F. Gleich. "Parameterized correlation clustering in hypergraphs and bipartite graphs." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2020, pp. 1868–1876. DOI: 10.1145/3394486.3403238 (page 111).

[259] Antoine Vendeville, Anastasios Giovanidis, Effrosyni Papanastasiou, and Benjamin Guedj. "Opening up echo chambers via optimal content recommendation." In: *Proceedings of the International Conference on Complex Networks and Their Applications*. 2023, pp. 74–85. DOI: 10.1007/978-3-031-21127-0_7 (page 161).

[260] Joshua T. Vogelstein, William Gray Roncal, R. Jacob Vogelstein, and Carey E. Priebe. "Graph classification using signal-subgraphs: Applications in statistical connectomics." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.7 (2013), pp. 1539–1551. DOI: 10.1109/TPAMI.2012.235 (pages 46, 47).

[261] Quang H. Vuong. "Likelihood ratio tests for model selection and non-nested hypotheses." In: *Econometrica: Journal of the Econometric Society* (1989), pp. 307–333. DOI: 10.2307/1912557 (page 42).

[262] Gabriel Wachman and Roni Khardon. "Learning from interpretations: A rooted kernel for ordered hypergraphs." In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2007, pp. 943–950. DOI: 10.1145/1273496.1273615 (page 111).

[263] Lu Wang, Zhengwu Zhang, David Dunson, et al. "Common and individual structure of brain networks." In: *The Annals of Applied Statistics* 13.1 (2019), pp. 85–112. DOI: 10.1214/18-AOAS1193 (page 46).

[264] Tomasz Wąs, Talal Rahwan, and Oskar Skibski. "Random walk decay centrality." In: *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. 2019, pp. 2197–2204. DOI: 10.1609/aaai.v33i01.33012197 (page 161).

[265] Stanley Wasserman and Katherine Faust. *Social network analysis: Methods and applications*. Cambridge: Cambridge University Press, 1994 (page 1).

[266] Melanie Weber, Emil Saucan, and Jürgen Jost. "Characterizing complex networks with Forman-Ricci curvature and associated geometric flows." In: *Journal of Complex Networks* 5.4 (2017), pp. 527–550. DOI: 10.1093/comnet/cnw030 (page 111).

[267] Junjie Wee and Kelin Xia. "Forman persistent Ricci curvature (FPRC)-based machine learning models for protein-ligand binding affinity prediction." In: *Briefings in Bioinformatics* 22.6 (2021). DOI: 10.1093/bib/bbab136 (page 111).

[268] Junjie Wee and Kelin Xia. "Ollivier persistent Ricci curvature-based machine learning for the protein-ligand binding affinity prediction." In: *Journal of Chemical Information and Modeling* 61.4 (2021), pp. 1617–1626. DOI: 10.1021/acs.jcim.0c01415 (pages 111, 122).

[269] Joe Whittaker, Seán Looney, Alastair Reed, and Fabio Votta. "Recommender systems and the amplification of extremist content." In: *Internet Policy Review* 10.2 (2021), pp. 1–29. DOI: 10.14763/2021.2.1565 (page 140).

[270] World Bank. *World Integrated Trade Solution*. 2021. URL: https://wits.worldbank.org/Default.aspx?lang=en (visited on 09/04/2021) (page 51).

[271] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How powerful are graph neural networks?" In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2019. DOI: 10.48550/arXiv.1810.00826 (page 1).

[272] Yasharth Yadav, Areejit Samal, and Emil Saucan. "A poset-based approach to curvature of hypergraphs." In: *Symmetry* 14.2 (2022), p. 420. DOI: 10.3390/sym14020420 (page 110).

[273] Yujun Yan et al. "Groupinn: Grouping-based interpretable neural network for classification of limited, noisy brain data." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2019, pp. 772–782. DOI: 10.1145/3292500.3330921 (pages 46, 47).

[274] Pinar Yanardag and S.V.N. Vishwanathan. "Deep graph kernels." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2015, pp. 1365–1374. DOI: 10.1145/2783258.2783417 (page 24).

[275] Muhsin Yesilada and Stephan Lewandowsky. "Systematic review: YouTube recommendations and problematic content." In: *Internet Policy Review* 11.1 (2022), pp. 1–22. DOI: 10.14763/2022.1.1652 (page 140).

[276] Zhijun Yin, Manish Gupta, Tim Weninger, and Jiawei Han. "A unified framework for link recommendation using random walks." In: *Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. 2010, pp. 152–159. DOI: 10.1109/ASONAM.2010.27 (page 161).

[277] Jean-Gabriel Young, Alec Kirkley, and Mark E.J. Newman. "Clustering of heterogeneous populations of networks." In: *Physical Review E* 105.1 (2022), p. 014312. DOI: 10.1103/PhysRevE.105.014312 (page 39).

[278] Zhiping Zeng, Anthony K.H. Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou. "Comparing stars: On approximating graph edit distance." In: *Proceedings of the VLDB Endowment* 2.1 (2009), pp. 25–36. DOI: 10.14778/1687627.1687631 (page 24).

[279] Chuxu Zhang et al. "Few-shot learning on graphs." In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. 2022, pp. 5662–5669. DOI: 10.24963/ijcai.2022/789 (page 9).

[280] Xiaojuan Zhang, Qian Liu, Min Li, and Yang Zhou. "Fast algorithms for supermodular and non-supermodular minimization via bi-criteria strategy." In: *Journal of Combinatorial Optimization* 44.5 (2022), pp. 3549–3574. DOI: 10.1007/s10878-022-00914-6 (page 153).

[281] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. "Learning with hypergraphs: Clustering, classification, and embedding." In: *Advances in Neural Information Processing Systems*. 2006, pp. 1601–1608 (pages 1, 111).

[282] Liwang Zhu, Qi Bao, and Zhongzhi Zhang. "Minimizing polarization and disagreement in social networks via link recommendation." In: *Advances in Neural Information Processing Systems*. 2021, pp. 2072–2084 (page 161).

[283] Liwang Zhu and Zhongzhi Zhang. "A nearly-linear time algorithm for minimizing risk of conflict in social networks." In: *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 2022, pp. 2648–2656. DOI: 10.1145/3534678.3539469 (page 161).

[284] Marinka Zitnik and Jure Leskovec. "Predicting multicellular function through multi-layer tissue networks." In: *Bioinformatics* 33.14 (2017), pp. i190–i198. DOI: 10.1093/bioinformatics/btx252 (pages 25, 26).

Index