LEARNING AND CONTROL OF NETWORK PHENOMENA

Mikhail Hayhoe

A DISSERTATION

in

Electrical and Systems Engineering

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2022

Supervisor of Dissertation

Victor M. Preciado, Associate Professor of Electrical and Systems Engineering

Graduate Group Chairperson

Troy Olsson, Associate Professor of Electrical and Systems Engineering

Dissertation Committee

Alejandro Ribeiro, Professor of Electrical and Systems Engineering
Hamed Hassani, Assistant Professor of Electrical and Systems Engineering
Sanjeev Khanna, Henry Salvatori Professor of Computer and Information Science
Brett Hemenway Falk, Research Professor of Computer and Information Science

LEARNING AND CONTROL OF NETWORK PHENOMENA

To my family,

without whom none of this would have been possible.

ACKNOWLEDGEMENT

First and foremost, I want to thank my advisor Prof. Victor Preciado. I have enjoyed the opportunity to learn from him and explore research alongside him, and have appreciated his guidance and support. I am sorry that we were not able to spend more time together throughout my time at Penn, but am happy to see his health improving with each passing day.

I am very thankful to my committee members Prof. Alejandro Ribeiro, Prof. Hamed Hassani, Prof. Sanjeev Khanna, and Prof. Brett Hemenway Falk. Alejandro's guidance and support in the last year of my thesis have been invaluable, and I am eternally grateful for his insights and advice. Hamed helped me early in my doctorate, both in and out of the classroom, and his calm and friendly demeanor was always appreciated. Sanjeev and Brett's willingness to meet and discuss research while Victor was recovering was very helpful, and helped keep me on track in many ways; I am very grateful for their open ears, kindness, and support.

I want to further thank all my teachers in life, from elementary school through my time at Penn. Phil Moorlag and Chris Edwards helped engender and develop my interest in math and computer science, and supported my initial explorations into these spaces through their teaching and trust. I especially want to thank Prof. Bahman Gharesifard and Prof. Fady Alajaji, whose initial support during my Bachelor's degree continued as they advised me during my Master's to establish a strong foundation in research that has enabled my continued success. I would not be half the researcher I am today without their guidance.

I have had the pleasure to meet and work alongside many brilliant, kind, and overall outstanding people at Penn: Alp, Paco, Shaoru, Jacob, Mahyar, Cassiano, Ximing, Alex, Harshat, Hans, Luana, Mohammad, Kendall, Juan, Landon, and many others. I am lucky enough to say that I have too many other great friends to list here, but especially want to thank Drew, Ethan, Kate, Mark, Tim, Rachel, Alex, Matt, Caitlin, Ashley, Kevin, and

Adie. I also want to thank Dr. Allan Goldberg, whose care and guidance has helped me in more ways than I can count. You have all helped me feel and be comfortable, happy, and successful, and I appreciate all of you.

Finally, I cannot express enough my deepest gratitude to my family: Zach, Carylin, Mark, Linnea, Keith, Caryl, and Monica. I also want to thank my constant companion and forever best friend Remy and his brother Arlo. I am the person that I am today due to all of your unconditional love and support. I love all of you, and dedicate this thesis to you.

Mikhail Hayhoe, Philadelphia, December 2022

ABSTRACT

LEARNING AND CONTROL OF NETWORK PHENOMENA

Mikhail Hayhoe

Victor M. Preciado

The intersection of dynamical systems and networks are used to model a huge variety of phenomena. From social networks, to traffic routes and self-driving cars, to swarms of robots and multiagent systems, to individuals moving about in a geographical area, networks can represent an enormous variety of interacting systems. These networks are typically represented via *graphs*, which are mathematical objects that encode the information of both the entities and relationships within a network. These graphs in turn can be represented by *graph matrices*, which encode the relationships between entities in the network.

The specific problems that this thesis considers span the learning and control of network phenomena via graph matrices: learning structural correspondences across correlated networks; identification of unknown networked dynamical systems with known control inputs; the generalization performance of machine learning algorithms for graphs and hypergraphs; and machine learning and data-driven control of an epidemic spreading across a network. The key intuition throughout is that many properties of networks and networked dynamical systems can be understood by examining the eigenvalue spectrum of an associated graph matrix. Following this intuition, we propose an algorithm for network alignment using spectral information called `SPECTRE` that exhibits state-of-the-art performance for aligning networks that are only moderately correlated. Next, we present a method for learning the spectra of a graph matrix using only the sparse output measurements of a networked dynamical system. We further propose a new architecture for signal processing on higher-order graphs, along with a new generalization bound on the performance of graph neural networks via spectral similarity. This generalization result is valid for arbitrary graphs regardless of their structure, engendering the first bound on the generalization performance of a machine

learning approach for higher-order graphs. Finally, we present a data-driven framework for multi-task learning and non-linear control of epidemics.

TABLE OF CONTENTS

ACKNOWLEDGEMENT iv

ABSTRACT vi

LIST OF TABLES x

LIST OF ILLUSTRATIONS xi

CHAPTER 1 : Introduction 1

1.1 Spectral identification [21] 2

1.2 Network Alignment [17] 4

1.3 Graph Learning [20] 6

1.4 Modeling and data-driven control of epidemics [19] 8

CHAPTER 2 : Spectral Identification 14

2.1 Background 16

2.2 Spectral Estimation for Discrete-Time Dynamics 19

2.3 Continuous-Time Dynamics 26

2.4 Simulations 29

CHAPTER 3 : Network Alignment 34

3.1 Related Work 37

3.2 Preliminaries 39

3.3 Algorithms 41

3.4 Numerical Experiments 48

CHAPTER 4 : Transferability of hyper-graph neural networks 53

4.1 Preliminaries 55

4.2 Transferability via spectral similarity 68
4.3 Transferability of Hyper-Graph Neural Networks 70
4.4 Transferability between random graphs 72
4.5 Stability of Graph Neural Networks via spectral similarity 78
4.6 Simulations 82

CHAPTER 5 : Data-driven modeling and control of epidemics 84
5.1 Background and Notation 88
5.2 Model 90
5.3 Parameter estimation via multitask learning 95
5.4 Optimal control using geometric programming 100

CHAPTER 6 : Conclusion and Future Work 110

APPENDICES 114

LIST OF TABLES

TABLE 2.1 Notation for spectral identification 16

TABLE 3.1 Notation for network alignment 40

TABLE 4.1 Notation for a hypergraph with n nodes and m hyperedges 58
TABLE 4.2 Comparison of hypergraph signal processing approaches which use graph representations, including their connections with higher-order spectral theory, the nonlinear hypergraph Laplacian (4.7), simplicial complices (SCs), and whether they can process both node and hyperedge signals. 64
TABLE 4.3 Cross-validated comparison & ablation test. 83

TABLE 5.1 Summary of parameters in epidemic model. We assume $\rho_{IR} + \rho_{IH} < 1$, $\rho_{EI} < 1$, $\rho_{AR} < 1$, and $\rho_{HR} < 1$, since these parameters capture the fraction of individuals in each compartment that transition to other compartments, which must be less than one for the model to be meaningful. We also assume $\gamma_A \in [0, 1]$, reflecting the fact that asymptomatic individuals are less likely to spread the disease. 92

TABLE B.1 Properties of all networks used in numerical experiments. 142
TABLE B.2 Space and time complexity of minimal-cost GP (5.13) and minimal-deaths GP (5.11) instances solved herein, over multiple time horizons T_c. Space complexity is measured in terms of the number of decision variables and constraints of the GP in posynomial form and in the equivalent convex form, the latter of which the solver uses to find the optimal solution. Time complexity measures the running time for the solver to instantiate and solve the program. 147

LIST OF ILLUSTRATIONS

FIGURE 2.1 10-agent preferential attachment network in discrete-time, generated according to [67]. Edge thickness in (a) corresponds to edge weight, and red edges have negative weights. Weights are randomly generated according to Uniform$[-1, 1]$, and initial condition is randomly generated as $\mathbf{x}_0 \sim \text{Uniform}[0, 1]^n$. There are 8 observable (non-zero) eigenvalues of S in this case, which are recovered via our estimation approach. 30

FIGURE 2.2 8-agent single integrator ring network in continuous-time, with sampling rate $\tau = 1$, random initial condition $\mathbf{x}_0 \sim \text{Uniform}[0, 1]^n$, and edges weighted as in Figure 2.1. Even though the system is unstable, all 8 eigenvalues of S are are recovered with high accuracy via our estimation approach. 31

FIGURE 2.3 10-agent preferential attachment network in discrete-time, generated according to [67]. The dynamics here follow the more general case of (2.14) from Section 2.2.1, where each node has a 3-dimensional state. The initial condition is generated according to $\mathbf{x}_0 \sim \text{Uniform}[0, 1]^n$, the vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are generated according to Uniform$[0, 1]^3$, and the entries of the symmetric matrix A are generated as $a_{ij} \sim \text{Uniform}[0, 1]$, $i \geq j$. In this case S has 10 eigenvalues, which are all recovered via our estimation approach. . . 32

FIGURE 3.1 Correct correspondence of some nodes in two correlated networks, illustrated by dashed lines. A network alignment algorithm should find these correspondences. 34

FIGURE 3.2 Example of `EstimateSeeds`. The size of each node denotes its relative spectral centrality score in the network, and the ground-truth correspondence is illustrated by the position of the nodes. Here with $k = 3$ and $w = 1$, u_1 is matched with v_1 and v_2; u_2 is matched with v_1, v_2 and v_3; and u_3 is matched with v_2 and v_3. Of these pairs, the seeds (u_2, v_1) and (u_3, v_3) are correct. 43

FIGURE 3.3 Example of `SafeExpand`, with matching threshold $r = 3$ and noisy seed set estimate $\mathcal{S} = \{(1, 1'), (3, 3'), (2, 1')\}$. White pairs have score zero, yellow have positive score (with border thickness denoting total number), green are matched, and red are removed (one or both nodes already matched in another pair). Arrows describe the direction in which scores are incremented. 45

FIGURE 3.4 Example of `LooseExpand` rebuilding the set $\mathcal{S}$. We take all unmatched neighboring pairs of previously matched nodes (i.e., those pairs with score exactly one) and add them to $\mathcal{S}$. White pairs have score zero, yellow have positive score, and green are matched. Arrows describe the direction in which scores are incremented. 47

FIGURE 3.5 Performance of network alignment methods with varying edge correlation levels. SPECTRE (in blue) achieves consistently higher performance than its competitors, increasingly so as the correlation between the networks grows smaller. 50
FIGURE 3.6 Performance of SPECTRE on *C. jejuni* and *E. coli* PPI networks. . . 52

FIGURE 4.1 Example hypergraph and several of its graph representations, none of which faithfully represent the original hypergraph on their own. To illustrate the connection to the original hypergraph (a) and dual hypergraph (b), edges are grouped and coloured in subfigures (c)-(f). The clique expansion in (d) cannot distinguish the real hyperedge e_2 from the clique $\{v_4, v_6, v_7\}$. The line graph in (e) cannot distinguish the dual hyperedge of v_4 from the clique $\{e_1, e_2, e_3\}$. The bipartite expansion in (f) has heterogeneous nodes and a different notion of signal processing. Finally, the star expansion in (c) adds many new nodes for each hyperedge intersection, which is impractical for dense hypergraphs and removes the interpretability of node convolutions. 61

FIGURE 5.1 Illustration of the epidemic model under consideration. 90
FIGURE 5.2 The learning pipeline for a given region. Global parameters, learned using all available data, are shown in magenta; initial conditions, learned for each region, are shown in blue; mobility mapping parameters, again different across regions, are shown in red. 95
FIGURE 5.3 Example of predicted cumulative deaths for several US counties, trained on 90 days of mobility and death count data, and tested on 30 days of the same. Predictions are shown in blue, and the rolling seven-day averages of real deaths data are shown as dashed lines. The area in white denotes training data, and the area in red denotes test data. 99
FIGURE 5.4 Minimal-cost control strategy $\mathbf{u}^\star(t)$ for Philadelphia County, PA, obtained by solving the geometric program (5.13). For comparison, we plot the true mobility data $\mathbf{m}(t)$ over the same time period. As shown in Lemma 5.2.1, number of hospitalized individuals in the linear model, $H(t)$, upper-bounds the number of hospitalized individuals in the nonlinear model, $\tilde{H}(t)$. We also include the hospitalizations predicted based on the true mobility data $\mathbf{m}(t)$ as a baseline. Parameters of the models used herein were learned as described in Section 5.3.3. We obtain the budget $\mathcal{B}^\star$ to use in geometric program (5.11) by taking the total cost of the minimal-cost control strategy, i.e., $\mathcal{B}^\star = \sum_{t=0}^{T_c-1} C_t(\mathbf{u}^\star(t))$. Control actions in Figure (a) for categories except transit stations are similar, and are overlaid. 107

FIGURE 5.5 Optimal minimal-death control strategy $\mathbf{u}^\star(t)$ for Philadelphia County, PA, obtained by solving the minimal-death GP (5.11), from August 1st to November 30th, 2020. In Figure (c) we compare the cumulative deaths predicted using the optimal control strategy $\mathbf{u}^\star(t)$ in the nonlinear model, $\tilde{D}(t)$ (shown in blue), cumulative deaths predicted in the linear model, $D(t)$ (shown in red), and the cumulative deaths predicted based on the true mobility data $\mathbf{m}(t)$ as a prediction baseline (shown in green). Parameters of the models used herein were learned as described in Section 5.3.3, with the budget $\mathcal{B}^\star$ taken from the solution to the minimal-cost GP (5.13). Control actions in Figure (a) for categories except transit stations are similar, and are overlaid. 109

FIGURE B.1 Effect of changing k and w for GEMSEC-Artists network with 60% correlation. 140

FIGURE B.2 Runtime of `SPECTRE` on multiple networks with varying numbers of edges. 142

FIGURE B.3 Exploration of mobility pattern data in Philadelphia county. Analysis of pairwise correlation suggests two correlated groupings, which is supported by two principal components explaining over 98% of observed variance in the data. 144

FIGURE B.4 Example optimal minimal-death control strategy $\mathbf{u}^\star(t)$ with lower-dimensional mobility data for Philadelphia County, PA, obtained by solving the minimal-death GP (5.11), from September 1st to November 30th, 2020. In (c) and (d), we plot the prediction using the linear model with the strategy $\mathbf{u}^\star(t)$, $H(t)$ and $D(t)$, predictions using the nonlinear model with the strategy $\mathbf{u}^\star(t)$, $\tilde{H}(t)$ and $\tilde{D}(t)$, and the predictions using the true mobility data $\mathbf{m}(t)$ as a baseline for no intervention, $\hat{H}(t)$ and $\hat{D}(t)$. Parameters of the models used herein were learned as described in Section 5.3.3 using the principal component projections of mobility data $\tilde{\mathbf{m}}(t)$, with the budget $\mathcal{B}^\star$ taken from the solution to the minimal-cost GP (5.13). 146

FIGURE B.5 Difference between final hospitalizations (resp. cumulative deaths) in the linear model, $H(T_c)$ (resp. $D(T_c)$), and in the nonlinear model, $\tilde{H}(T_c)$ (resp. $\tilde{D}(T_c)$) as a function of the hospitalization threshold τ_H and the budget $\mathcal{B}(\tau_H)$. For each value of τ_H, we solve the minimal-cost GP (5.13) to find the budget $\mathcal{B}(\tau_H)$, then take multiples of this budget from $1\times$ to $10\times$. Trends appear consistent across hospitalization thresholds τ_H, but the difference between the models quickly shrinks as the budget $\mathcal{B}(\tau_H)$ grows. 147

CHAPTER 1

Introduction

Networks and dynamics are ubiquitous. Their intersection, namely networked dynamical systems, are used to model a huge variety of phenomena. From social networks, to group chats and messages, to traffic routes and self-driving cars, swarms of robots and multiagent systems, to individuals moving about in a geographical area, networks can represent an enormous variety of interacting systems [1]. These networks are typically represented via *graphs*, which are mathematical objects that encode the information of both the entities, called *nodes*, and relationships, called *edges*, within a network. These graphs in turn can be represented in a variety of ways, but by far the most common is via matrices. These matrix representations of graphs, which we call *graph matrices*, encode the relationships between entities in the network, which is collectively referred to as the *network topology* or *structure*. In cases where the entities or relationships themselves contain additional information beyond their structure, especially in the case of dynamical systems on networks, we may store this data in *node or edge signals*. While we will consider these signals and the wealth of information they contain about the networked dynamics, this thesis will mainly focus on the topology of networks via matrix representations of graphs.

The key intuition behind this thesis is that many properties of networked dynamical systems can be understood by examining the eigenvalue spectrum of an associated graph matrix, which is called *spectral graph theory* [2]. This tool allows us to reason about both the topological structure of networks and the properties of dynamical systems thereon, and has thus seen a wide range of applications [3]. Google's search engine is famously based off of a spectral graph-theoretic measurement called PageRank [4]. When modeling epidemics spreading on networks, the largest eigenvalue of the adjacency matrix $A \in \mathbb{R}^{n \times n}$ can explain whether the epidemic dies out or survives indefinitely [5]. In studying the dynamical evolution of opinions over a network, the spectral gap of the Laplacian matrix $L \in \mathbb{R}^{n \times n}$ (i.e., the difference between the largest and second-largest eigenvalues) is related to the rate of

convergence to a dynamical equilibrium [6]. In a later chapter, we show how the spectra of graph matrices relate to the robustness of machine learning architectures that use graphs, generalizing earlier results that make assumptions on the structure of graph matrices [7]–[11]. There enormous list of applications also includes multiagent coordination [12], [13], synchronization of oscillators [14], [15], community detection [16], and many others.

The specific problems that this thesis considers span the learning and control of network phenomena: learning structural correspondences across correlated networks [17]; identification of unknown networked dynamical systems with known control inputs [18], [19]; learning and data-driven control of an epidemic spreading across a network [19]; and the learning and generalization performance of machine learning algorithms for graphs and hypergraphs [20]. We introduce each of these problems, and our solutions, below.[1]

1.1. Spectral identification [21]

As mentioned earlier, the spectrum of graph matrices are useful for understanding properties and problems in networked dynamical systems, including multi-agent coordination [12], [13], synchronization of oscillators [14], [15], community detection [16], and many others. Unfortunately, it is often computationally and practically expensive to compute the spectrum of such systems from output measurements alone. Hence, we aim to estimate the observable eigenvalues of an unknown graph matrix governing the dynamics of a networked dynamical system using as few measurements as possible.

As a practical example, consider a network of small sellers as they set a price for an item or service. These sellers may be brick-and-mortar stores, larger businesses, or even individuals listing items on eBay or Craigslist. No seller wants to charge too much, since they will lose customers, nor do they wish to charge too little and lose money. Thus at the beginning of each day, each seller checks the price charged by their closest competitors yesterday, and averages them to come up with a baseline current price. The seller may then wish to add

[1]This work was supported by NSF TRIPODS-1934876, NSF CAREER-ECCS-1651433, NSF III-2008456, and the Rockefeller Foundation.

or subtract some amount from this baseline price, based on their own business strategy. Individual sellers are not privy to whom each other seller considers their direct competitors, nor may they be aware of historical pricing information. Instead, they can observe some aggregate sale price based on the pricing information which they have access to. Thus, our algorithm allows an individual seller in this unknown financial interaction network to learn the properties of the components of the network which they are able to observe with as few measurements as possible. This information can then inform pricing strategy such as how quickly the price may change in the future, or by how much, or how quickly the prices of competitors will agree.

We now state the problem formally. Consider a system[2] on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \{1, \ldots, n\}$, with state, control input, and output at a discrete time k denoted by $\mathbf{x}[k] \in \mathbb{R}^n$, $\mathbf{u}[k] \in \mathbb{R}^n$, and $y[k] \in \mathbb{R}$, respectively, governed by the dynamics

$$\begin{aligned} \mathbf{x}[k+1] &= S\mathbf{x}[k] + \mathbf{u}[k], \quad \mathbf{x}[0] = \mathbf{x}_0, \\ y[k] &= \mathbf{c}^\mathsf{T}\mathbf{x}[k], \end{aligned} \tag{1.1}$$

where $\mathbf{c} \in \mathbb{R}^n$ is the observation vector, and $\mathbf{x}_0 \in \mathbb{R}^n$ is the initial condition of the system. $S \in \mathbb{R}^{n \times n}$ is a graph matrix, such as the adjacency or Laplacian, but may have arbitrary weights and need not be symmetric, i.e., the underlying graph may be directed. The parameters of the dynamical system of interest are unknown a priori; even still, our technique allows us to use the sequence of output measurements $y[0], y[1], \ldots, y[2n-1]$ to estimate the observable eigenvalues of the system, using the following Hankel matrix (reproduced from Chapter 2):

$$H \coloneqq \begin{bmatrix} \Delta y[n] & \cdots & \Delta y[2n-1] \\ \vdots & \ddots & \vdots \\ \Delta y[2n-1] & \cdots & \Delta y[3n-2] \end{bmatrix}. \tag{2.13}$$

[2]While we consider a discrete-time system here, the results apply similarly to continuous-time systems.

If $\mathbf{x}_0$ is random, then almost surely the observable eigenvalues correspond to the observable eigenmodes of $(S, \mathbf{c}^\intercal)$ according to the Popov-Belevitch-Hautus (PBH) test [21]. Moreover, while all existing techniques assume the system is autonomous, our technique can accommodate the presence of arbitrary periodic control inputs $\mathbf{u}[k] \in \mathbb{R}^n$. The key technique discussed in Chapter 2 is summarized in Theorem 2.2.1, which we state below.

Theorem 2.2.1. *Given the sequence of observations* $(y[k])_{k=0}^{3n-1}$ *from the discrete-time system in* (2.4)*, consider the matrix* H *from* (2.13) *and denote its rank by* r*. For periodic control inputs* $\mathbf{u}[k]$ *with period* r *and the differences of measurements* $\Delta y[r+k] = y[r+k] - y[k]$*, the observable eigenvalues are roots of the polynomial*

$$p_S(x) = x^r + \alpha_{r-1}x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

where the coefficients $\alpha_0, \ldots, \alpha_{r-1}$ *are given by*

$$\begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{r-1} \end{bmatrix} = - \begin{bmatrix} \Delta y[r] & \cdots & \Delta y[2r-1] \\ \vdots & \ddots & \vdots \\ \Delta y[2r-1] & \cdots & \Delta y[3r-2]] \end{bmatrix}^{-1} \begin{bmatrix} \Delta y[2r] \\ \vdots \\ \Delta y[3r-1] \end{bmatrix}.$$

Moreover, $\lambda_i \in \mathcal{S}_S$ *is a root of* $p_S(x)$ *with multiplicity* $\tilde{m}_i$*.*

Informally, the value $\tilde{m}_i$ (discussed in Chapter 2) is related to "how observable" the eigenmode corresponding to λ_i can be. In practice, to use as few output measurements as possible, we can apply no control input and collect measurements to build the Hankel matrix of observed differences of measurements iteratively. When it stops growing in rank, i.e., after $2r$ measurements, we can then estimate all $(S, \mathbf{c}^\intercal)$-observable eigenvalues almost surely.

1.2. Network Alignment [17]

Typically, network phenomena are assumed to occur over individual graphs. However, in many practical applications, phenomena of interest may occur across networks rather than within them. Money laundering, for example, typically involves many intermediate trans-

actions which can be in different currencies, through different banks, or different cryptocurrencies. Thus, learning a matching between individuals across networks would allow for the study of phenomena across these same types of networks which would otherwise go unnoticed. *Network alignment*, or *graph matching*, is the problem of finding such a structure-preserving matching of entities across separate but correlated networks[3]. For example, we may match the user accounts of the same individuals across Facebook and Twitter based on their friend/following networks, or match accounts owned by the same individuals/organizations across different financial and cryptocurrency transaction networks.

Unfortunately, solving the network alignment problem exactly is very difficult; indeed, determining whether a subgraph of one graph is isomorphic to another graph is an NP-complete problem [22]. As such, the best we can hope is to align the two graphs via heuristics. Our approach, `SPECTRE`, is a scalable algorithm able to accurately and robustly align pairs of networks exhibiting moderate correlation using only spectral graph theory and no prior information. While many approaches for network alignment require additional information in the form of a *seed set* of ground-truth node pairs, `SPECTRE` uses only the structure of the two graphs. Instead, a *noisy seed set* is created by pairing nodes using spectral centrality measures. Using this noisy seed set, the two graphs are aligned using bootstrap percolation techniques. The main idea is to percolate matching scores across the product graph[4] until a threshold r has been met for a node pair from the original graphs, at which point we consider them matched, remove every other pair containing either of them, and then increment the scores of all neighboring pairs in the product graph. While this product graph in principle has $O(n^2)$ nodes, in practice we only store pairs with positive matching score and remove $O(n)$ nodes from the product graph whenever we match a pair, so the memory requirements are much closer to $O(n \log n)$. Indeed, we can align networks with tens of thousands of nodes and hundreds of thousands of edges in minutes, while many other algorithms cannot handle such networks at all. Furthermore, `SPECTRE` shows improvements in accuracy by as much as

[3]We consider aligning two graphs, although in general we may try to align multiple.

[4]For two graphs $\mathcal{G}_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{V}_2, \mathcal{E}_2)$, the product graph is $\mathcal{H} = (\tilde{\mathcal{V}}, \tilde{\mathcal{E}})$, where $\tilde{\mathcal{V}} = \mathcal{V}_1 \times \mathcal{V}_2$ and $\tilde{\mathcal{E}} = \{((u_1, u_2), (v_1, v_2)) \in \tilde{\mathcal{V}} \mid u_1, v_1 \in \mathcal{V}_1,\ u_2, v_2 \in \mathcal{V}_2,\ \text{either } (u_1, v_1) \in \mathcal{E}_1 \text{ and/or } (u_2, v_2) \in \mathcal{E}_2\}$

300% over competitors when the networks to align exhibit low correlation.

1.3. Graph Learning [20]

In many real applications, data for learning is relational in nature. Indeed, graphs have become a ubiquitous tool for modeling and analyzing relational data. However, inherently a graph may only faithfully represent relations among pairs of individuals. In many applications, such a drawback may be restrictive, because relations may exist between arbitrarily sized groups of individuals. From communications face-to-face, over email, or on social networks, to transactions in cryptocurrencies, to co-authorship of academic papers, there are many instances in which a graph with relations of an order that can be higher than two may be useful. Thus, this proposal will explore ways in which to do learning on higher-order graphs, which we call hypergraphs throughout. To this end, we introduce Hyper-graph Expansion Neural Networks (HENNs) below, which build hypergraph convolutions in a two-phase approach using representations that are graphs.

Definition 4.1.4. *A Hyper-graph Expansion Neural Network (HENN) is a signal processing architecture of the form* $\Phi(\cdot; S_c, \Theta_c, S_l, \Theta_l)$*, which is composed of cascades of clique expansion layers*

$$X_v^{l+1} = \sigma\left(\sum_{k=0}^{K}\left(D_v^{-1/2}BWB^\intercal D_v^{-1/2}\right)^k X_v^l H_v^{k,l}\right), \tag{4.11}$$

and line graph layers

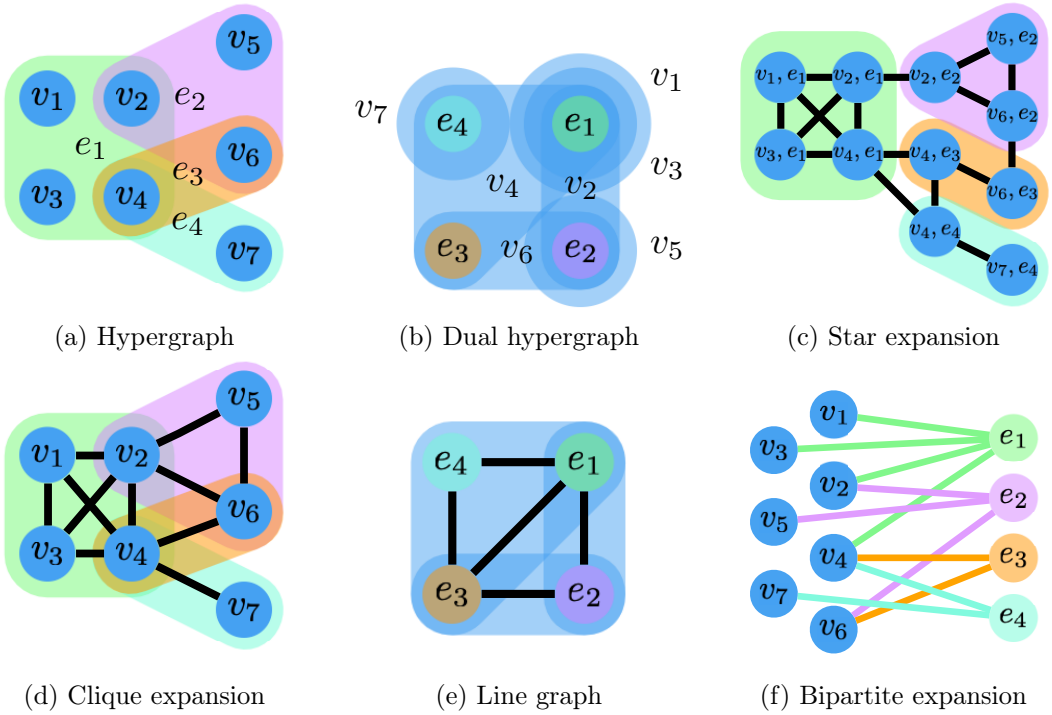
$$X_e^{l+1} = \sigma\left(\sum_{k=0}^{K}\left(D_{ee}^{-1/2}B^\intercal BWD_{ee}^{-1/2}\right)^k X_e^l H_e^{k,l}\right), \tag{4.12}$$

where D_{ee} *is the hyperedge degree matrix from the line graph. We pool between these types of layers using max-pooling according to node-hyperedge membership, i.e., based on the entries of* B*, following the definition of the nonlinear hypergraph Laplacian in* (4.7)*.*

HENN is designed to take advantage of Hodge theory and the corresponding the spectral

interpretation of hypergraphs via simplicial complices, it includes nonlinear interactions between nodes and hyperedges, and can process both node and hyperedge signals.

Informally, transferability is a type of generalization property for graph signal processing architectures which says that, given two graphs that describe the same or similar phenomenon, the graphs should process signals in a similar manner [23]. There have been three main approaches for codifying the similarity of graphs, i.e., that graphs model similar phenomena: comparing the original graph to a mildly perturbed version [7], [9], [10]; assuming a latent space model by which similar graphs are created [23], [24]; and obtaining graphs from a graphon [25], which can be understood as the continuous limit objects of sequences of graphs [11], [26]. Unfortunately, these assumptions on the origins of graphs may not hold in practice. The core concept is that GNNs are spectral operators and, hence, should be transferable across graphs with similar spectra. To this end, we provide the first GNN transferability bound directly between two arbitrary graphs of the same size, which can be



(a) Hypergraph (b) Dual hypergraph (c) Star expansion

(d) Clique expansion (e) Line graph (f) Bipartite expansion

Figure 4.1: Example hypergraph and graph representations (repeated from page 61).

computed before any training has occurred. The key tool in our analysis is to explicitly measure the *spectral similarity* [27], [28] of the graphs for which the transferability bound is desired, via computing and comparing their spectra directly.

Using our approach, we provide what is to the best of our knowledge the first bound on the transferability performance of neural networks that perform convolutions with signals supported on arbitrary hypergraphs. The key herein is to consider graph representations of hypergraphs (examples of which are illustrated in Figure 4.1), which may have arbitrary structure. Thus, we may apply our results by simply measuring the spectral similarity of the graph expansions of the hypergraphs of interest, before any training has occurred. We also explore transferability without direct computation of the eigenvalue spectrum in the context of small perturbations, random graphs, and graphs sampled from graphons to show that transferability bounds in each of these regimes may be recovered using our methods.

1.4. Modeling and data-driven control of epidemics [19]

One interesting class of networked dynamics is spreading processes, which can model the evolution of opinion, the spread of a disease or computer virus, the dissemination of ideas and marketing, and much more. In this thesis, the spreading process we consider is that of an epidemic spreading throughout a population, but the tools herein are broadly applicable.

In order to be able to predict and eventually control an epidemic, it is important to have a model which can accurately describe its evolution. Many techniques exist for modeling epidemics (see, e.g., [29]), and many works recently learn the parameters of such models using large-scale disease data. In a departure from the other chapters in this thesis, herein we will focus more on the dynamical process rather than the network on which it occurs. Specifically, we use a compartmental epidemic model, shown in Figure 5.1, which describes the progression of the disease in a population using a sequence of compartments. All parameters and the governing dynamics are described in Chapter 5. Our goal is to create such a model that can incorporate large-scale mobility data to describe the impact of non-pharmaceutical interventions, i.e., mobility restrictions, on the evolution of an epidemic.

Moreover, we construct our model in such a way that it may be used in our downstream control tasks, by enforcing that the learned mappings $f(\cdot)$ from mobility to infectivity can be used in a *geometric programming* framework. Geometric programs (GPs) [30] are a special class of optimization problems described using posynomials which are exactly transformable to convex programs. GPs are desirable because while they initially appear non-convex, it is possible to solve them *exactly and efficiently* using traditional convex optimization techniques. Indeed, while something like a semi-definite program will fail when the numbers of variables and constraints are in the order of tens of thousands, we are able to solve GPs (*not* relaxations, but exact programs) with hundreds of thousands of variables and constraints in a matter of minutes.

In particular, the mobility data mappings are chosen to be posynomials, i.e., functions $f : \mathbb{R}^n_{>0} \to \mathbb{R}$ of the form

$$f(x_1, x_2, \dots x_n) = \sum_{k=1}^{K} c_k x_1^{a_{k,1}} x_2^{a_{k,2}} \cdots x_n^{a_{k,n}}, \qquad c_k > 0,\ a_{k,i} \in \mathbb{R}\ \forall i, k \tag{1.2}$$

Since posynomials admit arbitrary real numbers as exponents but only positive coordinates and coefficients, they are neither a subset nor a superset of polynomials.

1.4.1. Data-driven modeling

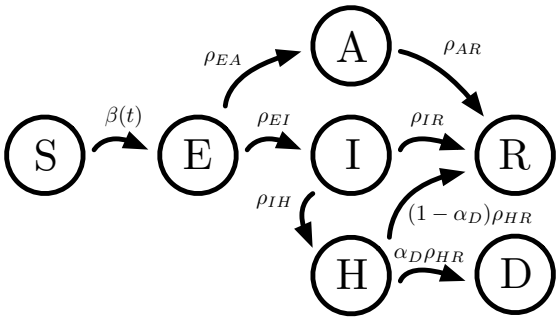


Figure 5.1: Illustration of the epidemic model under consideration. The compartments are Susceptible (i.e., healthy), Exposed, Asymptomatic, Infected (and symptomatic), Hospitalized, Recovered, and Dead. The parameters of the model describe the rates at which individuals within the population progress from one compartment to the next (repeated from page 90).

Data on COVID-19 cases and deaths, as well as cell-phone location and mobility data, is available at sufficient granularity to be able to learn the parameters of models for each of the over $3,000$ counties across the entire United States. However, to learn the parameters of these models independently is to ignore the facets of the given disease which are consistent across geographical regions. While mobility patterns and the subsequent infectivity of the disease may vary geographically (between dense urban populations and sparse rural ones, for example), the progression of the disease itself among individuals is consistent. For example, the number of days between when an individual is exposed and first shows symptoms (the disease's latency period) depends not on the geographical location of the individual but more on the disease itself. As such, we use a multitask learning framework in which learning the parameters of a model for each region is its own individual task, while the parameters intrinsic to the disease are trained using all available data, regardless of region, in one centralized task. The learning pipeline, from data to the predictions, is illustrated in Figure 5.2. With this model in place, we use the auto-differentiation package `autograd` [31] to compute numerical gradients and learn all parameters, both globally and locally for each region.

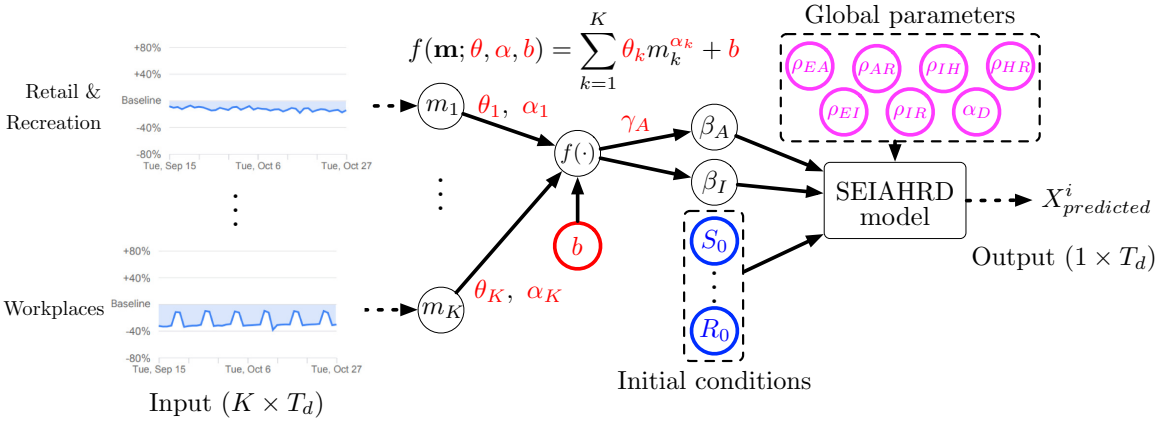


Figure 5.2: The learning pipeline for a given region. Global parameters, learned using all available data, are shown in magenta; initial conditions, learned for each region, are shown in blue; mobility mapping parameters, again different across regions, are shown in red (repeated from page 95).

1.4.2. Control via geometric programming

Given a model that can accurately predict the evolution of an epidemic, the natural question to ask is how a decision maker may intervene to save lives. Particular examples of control interventions include dispersing medication, vaccinating individuals, as well as non-pharmaceutical interventions (NPIs) such as enforcing mobility restrictions and quarantines. As the initial wave of COVID-19 made clear, when a new disease arrives the only effective tool to fight its spread are NPIs until specific medical interventions (such as vaccines) can be developed and tested. However, these NPIs are not implementable for free; indeed, we have seen that they result in financial harm to individuals, businesses, and the economy as a whole. As such, it is of broad societal interest to understand how we may control an epidemic using non-pharmaceutical interventions in a way that saves lives while also understanding the impact to the economy.

While many approaches for control of epidemics using compartmental models exist, working directly with the dynamics tends to be difficult. As such, many authors cast the problem of epidemic control as altering a related metric, such as lowering the spectral radius of the interaction network's adjacency matrix. Indeed, if the edge weights correspond to interaction frequency or strength, quarantining can be understood as edge removal and mobility restrictions can be modeled as lowering the edge weights. However, this spectral property relates to the asymptotic behavior of the epidemic, and does nothing to control its transient behavior. Hence, with a tractable model for the dynamics, it is much more practical to control the transient behavior of the epidemic. Since the model described earlier was built to be tractable for control, we can work with the dynamics of the epidemic itself. As a reasonable assumption we set $S(t) = S_0 \ \forall$, i.e., we assume the number of healthy individuals in a region over a short time window will be approximately constant relative to the overall population. This assumption renders the dynamics amenable to geometric programming, and it is easy to show that every state in this model is an upper bound to the corresponding state of the true dynamics. Hence, reducing deaths in the simplified model will result in

reducing deaths of the original modeled dynamics (see Chapter 5 and [19] for full details). We point out that this simplification is not necessary, and we could instead cast our problem as a signomial program, which are a superset of geometric programs wherein the functions may have arbitrary (non-positive) coefficients. Unfortunately, signomial programs are much more computationally challenging to solve.

We present our main approach for controlling epidemics using geometric programming below. Given some finite control horizon T_c, posynomial cost function $C_t(\cdot)$, which may vary in time, some prescribed budget $\mathcal{B}$ on the total cost of the implemented NPIs (either direct financial cost or lost potential revenue, taxes, etc.), some limit on the total number of hospitalized individuals τ_H, and some discount factor $\gamma_D \in (0, 1]$ which accounts for the fact that immediate deaths are more of a concern than those in the future (with $\gamma_\infty = (\gamma_D)^{T_c}/(1-\gamma_D)$ representing the infinite-horizon factor), we summarize our problem below.

Theorem 5.4.1. *If $C_t(\mathbf{u}(t))$ is a posynomial cost function for all t, and the set of admissible control actions $\mathcal{U}$ is described by posynomial inequalities and monomial equalities, then the following is a geometric program:*

$$\begin{aligned} \underset{\mathbf{u}(0),\mathbf{u}(1),\ldots,\mathbf{u}(T_c-1)}{\text{minimize}} \quad & \sum_{t=1}^{T_c-1} \gamma_D^t D(t) + \gamma_\infty D(T_c) \\ \text{subject to} \quad & \sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t)) \leq \mathcal{B} \\ & H(t) \leq \tau_H, && t = 1, \ldots, T_c, \\ & \mathbf{u}(t) \in \mathcal{U}, && t = 0, \ldots, T_c-1, \end{aligned} \tag{5.11}$$

where $\mathbf{u}(t)$ are the NPI control actions, and $H(t)$ and $D(t)$ are the number of individuals who are hospitalized and who die at time t, respectively.

Upon solving the above problem with some prescribed cost function and budget, a decision-maker is provided the best strategy for implementing non-pharmaceutical interventions while observing the limits of available hospital beds and without causing more harm to the econ-

omy than the desired budget.

Exactly how to set the budget $\mathcal{B}$ may not be clear a priori. However, it is well-known that exceeding the number of available hospital beds can lead to disastrous outcomes as patients cannot be treated appropriately, which leads to excess deaths. Thus, we present the following method for finding a budget by designing a control strategy that keeps the number of hospitalized individuals below the hospitalization threshold τ_H while minimizing the cost of the non-pharmaceutical interventions.

Theorem 5.4.2. *If $C_t(\mathbf{u}(t))$ is a posynomial cost function for all t, and the set of admissible control actions $\mathcal{U}$ is described by posynomial inequalities and monomial equalities, then the minimal budget required to keep hospitalizations below a given threshold τ_H is given by*

$$\mathcal{B}^\star = \sum_{t=0}^{T_c-1} C_t(\mathbf{u}^\star(t)), \tag{5.12}$$

where $\mathbf{u}^\star$ is the solution to the geometric program

$$\begin{aligned} \underset{\mathbf{u}(0),\ldots,\mathbf{u}(T_c-1)}{\text{minimize}} \quad & \sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t)) \\ \text{subject to} \quad & H(t) \leq \tau_H, \quad t = 1,\ldots,T_c, \\ & \mathbf{u}(t) \in \mathcal{U}, \quad t = 0,\ldots,T_c-1. \end{aligned} \tag{5.13}$$

where $\mathbf{u}(t)$ are the NPI control actions, and $H(t)$ is the number of individuals who are hospitalized at time t.

CHAPTER 2

Spectral Identification

M. Hayhoe, F. Barreras, and V. M. Preciado, "Sparse estimation of Laplacian eigenvalues in multiagent networks," in IFAC 2020 World Congress*, IFAC, 2020, pp. 1043–1048.* DOI*: https://doi.org/10.1016/j.ifacol.2020.12.1288*

M. Hayhoe, F. Barreras, and V. M. Preciado, "A dynamical approach to efficient eigenvalue estimation in general multiagent networks," Automatica*, vol. 140, 2022.* DOI*: https://doi.org/10.1016/j.automatica.2022.110234*

The spectra of graph matrices which represent networks of interacting dynamical agents provide a wealth of global information about the network structure and function; see, e.g., [32]–[36]. However, in many applications we may not know or have access to the network topology; instead, the only available information is in the form of output measurements from a networked dynamical system. Hence, it is of great practical interest to be able to estimate this unknown network structure efficiently using output measurements alone.

A widely-studied graph matrix whose spectrum we may wish to estimate is the graph Laplacian, which finds applications in many fields (e.g., [12], [13], [15], [37]), including graph-theoretical problems, as illustrated in [16], [38], [39], among others. Indeed, the eigenvalues of the normalized Laplacian are relevant in the analysis of diffusion processes, random walks over graphs, or discrete-time consensus dynamics [2]. Moreover, we study the relationship between the spectrum of the normalized Laplacian and the generalization of graph neural networks in Chapter 4. Beyond the Laplacian, the spectrum of the adjacency matrix of a network is relevant in the analysis of, for example, epidemic processes [40].

Owing to the practical importance of graph matrix spectra, numerous methods have been proposed to estimate the eigenvalues of graph matrices that represent networks of dynamical agents. Kempe and McSherry [41] proposed a distributed algorithm based on orthogonal

iteration (see [42]) for computing higher-dimensional invariant subspaces. Franceschelli, Gasparri, Giua, and Seatzu [43] estimated the Laplacian spectrum by imposing specific dynamics on the agents in the network, which is often unrealistic. The authors of [44]–[47] proposed distributed algorithms for estimating and computing bounds on algebraic connectivity (i.e., the second smallest Laplacian eigenvalue). Leonardos, Preciado, and Daniilidis [48] proposed a method to compute the largest (or smallest) eigenvalues and eigenvectors of any graph. Approaches by Tran and Kibangou [49] and Gusrialdi and Qu [50] deduce the spectrum of the Laplacian by running consensus algorithms on the network separately from the dynamics. Using the Koopman operator, [51] and [52] recover the spectrum of the Laplacian using sparse local measurements; unfortunately, these methods require full observability or for the system to reset to known initial conditions multiple times.

We find in the literature several works closely related to the techniques used in this chapter. Yuan, Stan, Shi, Barahona, and Goncalves [53] and Charalambous, Rabbat, Johansson, and Hadjicostis [54] use a Hankel matrix to compute roots of a polynomial corresponding to the eigenvalues of the system, but consider only autonomous dynamics without the presence of a control input. *Prony's method* can be used to estimate the parameters of a uniformly sampled superposition of complex exponentials, which can be used for spectral estimation (see [55]), among other problems (see [56], [57]). However, Prony's method only applies to symmetric matrices; hence, it can only be applied for the spectral reconstruction of undirected networks. The *Newton-Girard equations* (see, e.g., [58]) allow us to recover eigenvalues by analyzing symmetric polynomials of the traces of powers of the matrix. However, computing the traces of powers of matrices is computationally expensive and requires a large amount of (centralized) data, which may not be feasible to collect in many applications. Preciado and Jadbabaie [59] computed the traces of powers of a graph matrix to derive bounds on spectral properties such as the spectral radius. Related methods analyze the spectrum of a graph by counting walks in graphs, as in [60]–[62].

In this chapter, we present an approach to efficiently estimate the eigenvalues of *any* graph

matrix, such as the Laplacian, corresponding to an unknown network of multiagent systems, using only a single temporal sequence of output measurements. In contrast to other works, our approach allows for the presence of arbitrary periodic control inputs, which often arise in industrial applications and circuit design (see, e.g., [63], [64], and references thereafter). The network structure may be weighted or directed and may include multi-edges or self-loops. The temporal sequence of measurements can correspond to the output signal of a single agent or to any weighted linear combination of outputs from a collection of agents. Notably, our method requires no knowledge of which agents contribute to the measurements, nor does it require prior knowledge of the network topology or initial condition. Our approach allows for the estimation of all complex eigenvalues associated with observable network modes, regardless of the (unknown) network structure. Moreover, the length of the sequence of measurements required is, at most, the period of the control input plus twice the number of agents in the network; however, in practice, it may be as low as twice the number of observable network modes. The proposed approach requires no parameter tuning, allows for arbitrary periodic control inputs, and may be applied in both discrete- and continuous-time to general (potentially unstable) multiagent systems.

2.1. Background

Symbol	Meaning
$\mathbb{R}$	set of real numbers
$\mathbb{N}$	set of natural numbers
$\mathbf{e}_i$	i-th vector in the canonical basis of $\mathbb{R}^n$
$\otimes$	Kronecker product
$\oplus$	Direct sum
$\sigma(X) \coloneqq \{\lambda_i\}_{i=1}^n$	eigenvalue spectrum of matrix X
$\mathrm{rk}(X)$	rank of matrix X
$A(\mathcal{G})$	adjacency matrix, $[A]_{ij} \neq 0 \Rightarrow (i,j) \in \mathcal{E}$
$D(\mathcal{G})$	degree matrix, $[D]_{ii} = \sum_{j=1}^n [A]_{ij}$

Table 2.1: Notation for spectral identification

Throughout this chapter lower-case letters denote scalars, lower-case bold letters denote vectors, and upper-case letters denote matrices. A *directed* graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ has node set $\mathcal{V} = \{1, \ldots, n\}$ and edge set $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where $(i,j) \in \mathcal{E}$ means there is an edge oriented

from node i toward node j. The graph $\mathcal{G}$ may have self-loops or (possibly negative) edge weights, and may contain multi-edges.

2.1.1. Discrete-Time Laplacian Dynamics

We begin our exposition with a well-studied example: a simple, undirected network of single integrators following discrete-time (DT) Laplacian dynamics. In this context, we introduce a methodology to estimate the eigenvalues of the Laplacian matrix from a finite sequence of output measurements; for full details of this case, see [18]. In the following sections, we will extend this result to arbitrary directed networks of discrete-time agents with periodic control inputs.

Consider the following autonomous DT dynamics:

$$\begin{aligned} \mathbf{x}[k+1] &= \mathcal{L}\mathbf{x}[k],\ \mathbf{x}[0] = \mathbf{x}_0, \\ y[k] &= \mathbf{c}^\intercal \mathbf{x}[k], \end{aligned} \tag{2.1}$$

where $\mathcal{L} \coloneqq D(\mathcal{G})^{-1}A(\mathcal{G})$ is the normalized Laplacian matrix[5] of an unknown undirected graph $\mathcal{G}$, $k \in \mathbb{N}$, and $\mathbf{c}, \mathbf{x}_0$ are arbitrary (possibly unknown) vectors in $\mathbb{R}^n$. The normalized Laplacian $\mathcal{L}$ of an undirected graph is always diagonalizable with real eigenvalues $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$ (see [2]). Denoting by $\mathbf{u}_i$ and $\mathbf{w}_i$ the (unknown) right and left eigenvectors corresponding to the eigenvalue λ_i, we have that $\mathcal{L} = U\Lambda W$, where $\Lambda \coloneqq \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$, $U \coloneqq [\mathbf{u}_1, \ldots, \mathbf{u}_n]$, and $W \coloneqq [\mathbf{w}_1^\intercal; \cdots; \mathbf{w}_n^\intercal] = U^{-1}$; hence,

$$y[k] = \mathbf{c}^\intercal \mathcal{L}^k \mathbf{x}_0 = (\mathbf{c}^\intercal U)\,\Lambda^k\,(W\mathbf{x}_0) = \sum_{i=1}^{n} \omega_i \lambda_i^k, \tag{2.2}$$

where the weights are given by $\omega_i \coloneqq [\mathbf{c}^\intercal U]_i\,[W\mathbf{x}_0]_i = \mathbf{c}^\intercal \mathbf{u}_i \mathbf{w}_i^\intercal \mathbf{x}_0$. Notice $\omega_i = 0$ when $\mathbf{c}^\intercal \mathbf{u}_i = 0$ or $\mathbf{w}_i^\intercal \mathbf{x}_0 = 0$. If $\omega_i = 0$ for some index i, then the i-th eigenvalue λ_i does not influence the output $y[k]$ in (2.2); consequently, we cannot estimate λ_i from this sequence of outputs. However, if $\mathbf{x}_0$ is random, then almost surely $\mathbf{w}_i^\intercal \mathbf{x}_0 \neq 0$; hence, almost surely $\omega_i = 0$

[5] The symmetric normalized Laplacian may also be defined as $\mathcal{L} \coloneqq I - D(\mathcal{G})^{-1/2}A(\mathcal{G})D(\mathcal{G})^{-1/2}$; results hold similarly.

only if $\mathbf{c}^\intercal \mathbf{u}_i = 0$. According to the Popov-Belevitch-Hautus (PBH) test [65], almost surely those eigenvalues corresponding to unobservable eigenmodes of the Laplacian dynamics will be those for which $\omega_i = 0$ and it will be impossible to recover them from our observations. Furthermore, repeated eigenvalues will not impact the output whenever $\sum_{j\colon \lambda_j = \lambda_i} \omega_j = 0$. Defining $\mathbf{w}_j^\intercal \mathbf{x}_0 = \alpha_j$ (which is different than zero almost surely), this condition is equivalent to

$$\sum_{j\colon \lambda_j = \lambda_i} \omega_j = \mathbf{c}^\intercal \sum_{j\colon \lambda_j = \lambda_i} \alpha_j \mathbf{u}_j = \mathbf{c}^\intercal \mathbf{u}^{(\lambda_i)} = 0$$

where $\mathbf{u}^{(\lambda_i)} \coloneqq \sum_{j\colon \lambda_j = \lambda_i} \alpha_j \mathbf{u}_j$. Clearly $\mathbf{u}^{(\lambda_i)}$ is in the eigenspace of the eigenvalue λ_i; hence, according to the PBH test, $\mathbf{c}^\intercal \mathbf{u}^{(\lambda_i)} = 0$ implies that λ_i corresponds to an unobservable eigenmode (almost surely). We will denote by $\mathcal{S}_{\mathcal{L}}$ the set of eigenvalues of $\mathcal{L}$ corresponding to observable eigenmodes of the pair $(\mathcal{L}, \mathbf{c}^\intercal)$.

Below (cf. [18]) we describe a methodology to efficiently reconstruct the observable eigenvalues $\lambda_i \in \mathcal{S}_{\mathcal{L}}$ from a finite sequence of output observations.

Theorem 2.1.1. *Given the sequence of observations $(y[k])_{k=0}^{2n-1}$ from the system in (2.1), define the following Hankel matrix*

$$Y \coloneqq \begin{bmatrix} y[0] & \cdots & y[n-1] \\ \vdots & \ddots & \vdots \\ y[n-1] & \cdots & y[2n-2] \end{bmatrix}. \tag{2.3}$$

The rank of Y satisfies $r \coloneqq rk(Y) = |\mathcal{S}_{\mathcal{L}}| \leq n$. The observable eigenvalues of $\mathcal{L}$ are roots of the polynomial

$$p_{\mathcal{L}}(x) = x^r + \alpha_{r-1} x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

where the coefficients $\alpha_0, \ldots, \alpha_{r-1}$ *are given by*

$$\begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{r-1} \end{bmatrix} = - \begin{bmatrix} y[0] & \cdots & y[r-1] \\ \vdots & \vdots & \ddots & \vdots \\ y[r-1] & \cdots & y[2r-2] \end{bmatrix}^{-1} \begin{bmatrix} y[r] \\ \vdots \\ y[2r-1] \end{bmatrix}.$$

In what follows, we extend this result to arbitrary (possibly weighted or directed) networks, in both discrete– and continuous-time, as well as non-autonomous dynamics by considering an arbitrary periodic control input.

2.2. Spectral Estimation for Discrete-Time Dynamics

Let $S \in \mathbb{R}^{n \times n}$ be any graph matrix whose sparsity pattern describes the connections of an arbitrary (unknown) graph $\mathcal{G}$ with n nodes. For example, S may correspond to the adjacency matrix A, the normalized Laplacian $\mathcal{L}$, or any other graph matrix. The graph $\mathcal{G}$ may be directed, have self-loops, or be weighted. Consider the following discrete-time dynamics:

$$\begin{aligned} \mathbf{x}\left[k+1\right] &= S\mathbf{x}\left[k\right] + \mathbf{u}[k],\ \mathbf{x}\left[0\right] = \mathbf{x}_0, \\ y\left[k\right] &= \mathbf{c}^\intercal \mathbf{x}\left[k\right], \end{aligned} \tag{2.4}$$

where $\mathbf{u}[k] \in \mathbb{R}^n$ is a control input, $k \in \mathbb{N}$, and $\mathbf{c}, \mathbf{x}_0$ are arbitrary (possibly unknown) vectors in $\mathbb{R}^n$. Hence, for any graph matrix S, the observations from our system (2.4) can be written as

$$y\left[k\right] = \mathbf{c}^\intercal S^k \mathbf{x}_0 + \sum_{l=0}^{k-1} \mathbf{c}^\intercal S^{k-l-1} \mathbf{u}[l] \tag{2.5}$$

In this work, we aim to recover the eigenvalues of the graph matrix S using as few measurements as possible. If $\mathbf{y}[k] = C^\intercal \mathbf{x}[k]$, where $C \in \mathbb{R}^{m \times n}$ and $m > 1$, the problem would become easier, since we would obtain m measurements at each step. Since our focus is on *sparse* measurements, we limit ourselves to the hardest case of $m = 1$. For simplicity, we study the case of periodic control inputs (see [64]) with period r, so $\mathbf{u}[r+k] = \mathbf{u}[k]$. Note that the case of no control input is also captured in this framework by requiring $\mathbf{u}[r+k] = \mathbf{u}[k] = 0$ for all

$r, k \in \mathbb{N}$. Hence, we define the differences of output measurements $\Delta y[r+k] \coloneqq y[r+k]-y[k]$, and analyze them below.

Lemma 2.2.1. *For periodic control inputs* $\mathbf{u}[k]$ *with period* r*, the differences of output measurements satisfy*

$$\Delta y[r+k] \coloneqq y[r+k] - y[k] = \mathbf{c}^\intercal S^k(\mathbf{x}[r] - \mathbf{x}_0). \tag{2.6}$$

Proof. See Appendix A.1.1. □

We may view our approach as a decentralized estimation problem when $\mathbf{c} = \mathbf{e}_i$, wherein agent i is attempting to estimate the eigenvalues of S by observing its own output. More generally, we may observe the weighted sum of the states of a subset $\mathcal{S} \subseteq \mathcal{V}$ of agents; hence, $\mathbf{c} = \sum_{i\in\mathcal{S}} \beta_i \mathbf{e}_i$, which corresponds to a group of agents collectively estimating the spectrum of S using a weighted linear combination of their outputs using (possibly unknown) weights $\{\beta_i\}_{i\in\mathcal{S}}$.

To extend the result in Section 2.1.1 to more general (possibly weighted or directed) dynamics, we start by defining the Jordan decomposition of S as

$$S = VJV^{-1} = V\,\mathrm{diag}\,(J_1, \ldots, J_d)\,V^{-1},$$

where $J_i,\ i \in \{1, \ldots, d\}$, is the $m_i \times m_i$ Jordan block associated with the i-th eigenvalue λ_i. Note that there may be multiple Jordan blocks associated with a single eigenvalue; hence, it may be that $\lambda_i = \lambda_j$ for some $i, j \in \{1, \ldots, d\}$. Thus, we also define the largest block size for each distinct eigenvalue λ_i as $\hat{m}_i \coloneqq \max_{j:\lambda_j=\lambda_i} m_j$. The powers of the matrix S satisfy $S^k = \left(VJV^{-1}\right)^k = VJ^kV^{-1}$, where the $m_i \times m_i$ Jordan block raised to the power k, J_i^k, is

the upper-triangular matrix

$$J_i^k = \begin{bmatrix} \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \cdots & \binom{k}{m_i-1}\lambda_i^{k-(m_i-1)} \\ & \ddots & \ddots & \vdots \\ & & \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} \\ & & & \lambda_i^k \end{bmatrix}. \tag{2.7}$$

Now, for $s \in \{0, \ldots, m_i - 1\}$, define the weights $\omega_i^{(s)}$ as

$$\omega_i^{(s)} \coloneqq \sum_{q=s+1}^{m_i} [\mathbf{c}^\intercal V]_{i,q-s} \left[V^{-1}(\mathbf{x}[r] - \mathbf{x}_0)\right]_{i,q}, \tag{2.8}$$

where $[\mathbf{c}^\intercal V]_{i,q}$ and $\left[V^{-1}(\mathbf{x}[r] - \mathbf{x}_0)\right]_{i,q}$, $q \in \{1, \ldots, m_i\}$ are the q-th components of the m_i-dimensional i-th blocks of $\mathbf{c}^\intercal V$ and $V^{-1}(\mathbf{x}[r] - \mathbf{x}_0)$, respectively, associated with Jordan block J_i. Finally, define the total weights corresponding to each distinct eigenvalue as

$$\bar{\omega}_i^{(s)} \coloneqq \sum_{j:\lambda_j=\lambda_i} \omega_j^{(s)}. \tag{2.9}$$

In general it is possible that $\bar{\omega}_i^{(s)} = 0$, which may make it impossible to recover λ_i. According to the PBH test, if $\mathbf{x}_0$ is random then almost surely these eigenvalues correspond to unobservable eigenmodes of the pair $(S, \mathbf{c}^\intercal)$. We denote the set of observable eigenvalues by

$$\mathcal{S}_S \coloneqq \left\{\lambda_i \in \sigma(S) : \exists s \text{ s.t. } \bar{\omega}_i^{(s)} \neq 0\right\}. \tag{2.10}$$

For an eigenvalue $\lambda_i \in \mathcal{S}_S$, we define

$$\tilde{m}_i \coloneqq 1 + \max\left\{s = 0, \ldots, \hat{m}_i - 1 : \bar{\omega}_i^{(s)} \neq 0\right\}, \tag{2.11}$$

and denote the set of indices corresponding to distinct observable eigenvalues as $\mathcal{I} \coloneqq \{i \in \{1, \ldots, n\} : \lambda_i \in \mathcal{S}_S\}$. Denoting the i-th blocks (corresponding to J_i) of $\mathbf{c}^\intercal V$ and

$V^{-1}(\mathbf{x}[r]-\mathbf{x}_0)$ as $[\mathbf{c}^\intercal V]_i$ and $[V^{-1}(\mathbf{x}[r]-\mathbf{x}_0)]_i$, respectively, from Lemma 2.2.1 we have

$$\Delta y[r+k]=\sum_{i=1}^{d}[\mathbf{c}^\intercal V]_i J_i^k[V^{-1}(\mathbf{x}[r]-\mathbf{x}_0)]_i=\sum_{i\in\mathcal{I}}\sum_{s=0}^{\hat{m}_i-1}\bar{\omega}_i^{(s)}\binom{k}{s}\lambda_i^{k-s}. \tag{2.12}$$

In what follows, we will propose a computationally efficient methodology to recover the eigenvalues in $\mathcal{S}_S$ using the output sequence $(y[k])_{k=0}^{3n-1}$; however, as we will show, a shorter sequence suffices in practice. Towards that goal, we define the Hankel matrix of differences of observations

$$H\coloneqq\begin{bmatrix}\Delta y[n] & \cdots & \Delta y[2n-1]\\ \vdots & \ddots & \vdots\\ \Delta y[2n-1] & \cdots & \Delta y[3n-2]\end{bmatrix}. \tag{2.13}$$

The following result relates the rank of this matrix to the largest observable Jordan blocks of S.

Lemma 2.2.2. *The rank of H in* (2.13) *satisfies*

$$rk(H)=\sum_{i\in\mathcal{I}}\tilde{m}_i,$$

where $\tilde{m}_i$ is defined in (2.11).

Proof. See Appendix A.1.2. □

With this lemma in hand, we present our main result on estimating the observable eigenvalues of the pair $(S,\mathbf{c}^\intercal)$.

Theorem 2.2.1. *Given the sequence of observations $(y[k])_{k=0}^{3n-1}$ from the discrete-time system in* (2.4)*, consider the matrix H from* (2.13) *and denote its rank by r. For periodic control inputs $\mathbf{u}[k]$ with period r and the differences of measurements $\Delta y[r+k]=y[r+k]-y[k]$,*

the observable eigenvalues are roots of the polynomial

$$p_S(x) = x^r + \alpha_{r-1}x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

where the coefficients $\alpha_0, \ldots, \alpha_{r-1}$ *are given by*

$$\begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{r-1} \end{bmatrix} = - \begin{bmatrix} \Delta y[r] & \cdots & \Delta y[2r-1] \\ \vdots & \ddots & \vdots \\ \Delta y[2r-1] & \cdots & \Delta y[3r-2]] \end{bmatrix}^{-1} \begin{bmatrix} \Delta y[2r] \\ \vdots \\ \Delta y[3r-1] \end{bmatrix}.$$

Moreover, $\lambda_i \in \mathcal{S}_S$ *is a root of* $p_S(x)$ *with multiplicity* $\tilde{m}_i$*.*

Proof. See Appendix A.1.3. □

Remark 2.2.1 (Sparse measurements)**.** *While in Theorem 2.2.1 we make use of* $3n$ *observations* $(y[k])_{k=0}^{3n-1}$*, in practice, fewer observations may be required. Consider an online setting where the output measurements are taken sequentially, one at a time. For a control input with period* s*, we can build a* $k \times k$ *Hankel matrix using the first* $2k-1$ *differences of observations from the system,* $\Delta y[s], \Delta y[s+1], \ldots, \Delta y[s+2k-2]$*, and check its rank. If it is full rank we continue taking measurements. By the structure of the Hankel matrix, if the rank does not grow after including measurement* $y[2k+s]$ *then it must be that there exists some non-trivial* $\alpha_0, \ldots, \alpha_{k-1}$ *such that* $\Delta y[k+s] = \alpha_0 \Delta y[s] + \cdots \alpha_{k-1}\Delta y[k-1+s]$*; hence,*

$$\begin{aligned} \mathbf{c}^\intercal V J^k V^{-1}(\mathbf{x}[s] - \mathbf{x}_0) &= \sum_{l=0}^{k-1} \alpha_l \mathbf{c}^\intercal V J^l V^{-1}(\mathbf{x}[s] - \mathbf{x}_0) \\ &= \mathbf{c}^\intercal V \left(\sum_{l=0}^{k-1} \alpha_l J^l \right) V^{-1}(\mathbf{x}[s] - \mathbf{x}_0). \end{aligned}$$

If all eigenmodes are observable and the initial condition $\mathbf{x}_0$ *is random, then by Cayley-Hamilton theorem we must have* $k = n$ *(almost surely). If* l *of the eigenmodes are unobservable (including multiplicities), then necessarily* $k \geq n - l$*, since at most* l *entries could be*

zeroed by the vector $\mathbf{c}^\intercal V$*. Thus, the rank of the Hankel matrix will stop growing once* $k = r$ *(almost surely), i.e., after we collect enough measurements to recover all distinct eigenvalues corresponding to observable eigenmodes.*

Remark 2.2.2 (Periodic control inputs)**.** *The estimation procedure in Theorem 2.2.1 requires that the control inputs be applied with a period* r*. However, in practice any known period* s *will allow us to recover the observable eigenvalues so long as we take our measured differences to be* $\Delta y[s+k] = y[s+k] - y[k]$*. The larger the period, the more freedom we have in the control inputs, but the more measurements are required to estimate the observable eigenvalues. In particular, to use the fewest possible measurements, we may apply no control input, define* $\Delta y[k] = y[k]$*, and recover the observable eigenvalues using* $2r$ *measurements, where* r *is the number of observable network modes.*

2.2.1. Network of Identical Discrete-Time Agents

In many applications, the network of interest will consist of agents with more general dynamics beyond single integrators. With this in mind, we consider a network of n agents where each agent follows the dynamics $\mathbf{x}_i[k+1] = A\mathbf{x}_i[k] + \mathbf{u}_i[k]$, where $\mathbf{x}_i$ is a d-dimensional vector of states, A is a known $d \times d$ state transition matrix, and $\mathbf{u}_i[k]$ is an input consisting of a linear combination of the states of the neighboring agents of i. Assuming that all agents start with an arbitrary initial condition $\boldsymbol{\beta}$ weighted by x_{0i}, and the output of agent i is $\boldsymbol{\gamma}^\intercal \mathbf{x}_i[k]$ weighted by c_i, we obtain the following network dynamics:

$$\begin{aligned} \mathbf{x}_i[k+1] &= A\mathbf{x}_i[k] + \sum_{j=1}^{n} s_{ij}\mathbf{x}_j[k], \ \mathbf{x}_i[0] = x_{0i}\boldsymbol{\beta}, \\ y[k] &= \sum_{i=1}^{n} c_i \boldsymbol{\gamma}^\intercal \mathbf{x}_i[k], \end{aligned} \tag{2.14}$$

where $s_{ij} = [S]_{ij}$, $c_i = [\mathbf{c}]_i$, $x_{0i} = [\mathbf{x}_0]_i$. Stacking the vectors of states in a large vector $\mathbf{x} = (\mathbf{x}_1^\intercal, \ldots, \mathbf{x}_n^\intercal)^\intercal$, the dynamics can be written compactly as

$$\begin{aligned} \mathbf{x}[k+1] &= (I_n \otimes A + S \otimes I_d)\,\mathbf{x}[k]\,,\ \mathbf{x}[0] = \mathbf{x}_0 \otimes \boldsymbol{\beta}, \\ y[k] &= (\mathbf{c} \otimes \boldsymbol{\gamma})^\intercal\,\mathbf{x}[k]\,. \end{aligned}$$

We assume the state matrix A as well as the vectors of individual initial condition $\boldsymbol{\beta}$ and observation $\boldsymbol{\gamma}$ are known, but the graph matrix S and weighting vectors for initial conditions $\mathbf{x}_0$ and observations $\mathbf{c}$ are unknown. Our aim is to estimate the observable eigenvalues of S from a finite sequence of outputs. This result is stated in the following theorem.

Theorem 2.2.2. *Given the sequence of observations $(y[k])_{k=0}^{2n-1}$ from the system in (2.14), consider the Hankel matrix H defined in (2.13) with $\Delta y[k] = y[k]$ and denote its rank by r. The weighted sums of eigenvalues $\sigma_k \coloneqq \sum_{i=1}^{d}\sum_{s=0}^{m_i-1}\omega_i^{(s)}\binom{k}{s}\lambda_i^{k-s}$ satisfy the following equality:*

$$\begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_{2r-1} \end{bmatrix} = \begin{bmatrix} b_0^0\nu_0 & 0 & \cdots & 0 \\ b_0^1\nu_1 & b_1^1\nu_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_0^{2r-1}\nu_{2r-1} & b_1^{2r-1}\nu_{2r-2} & \cdots & b_{2r-1}^{2r-1}\nu_0 \end{bmatrix}^{-1} \begin{bmatrix} y[0] \\ y[1] \\ \vdots \\ y[2r-1] \end{bmatrix}$$

where $\nu_{k-s} \coloneqq \boldsymbol{\gamma}^\intercal A^{k-s}\boldsymbol{\beta}$, $b_s^k \coloneqq \binom{k}{s}$, and the matrix is invertible when $\boldsymbol{\gamma}^\intercal\boldsymbol{\beta} \neq 0$. Then, the observable eigenvalues of S are roots of the polynomial

$$p_S(x) = x^r + \alpha_{r-1}x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

where the coefficients $\alpha_0, \ldots, \alpha_{r-1}$ *satisfy*

$$\begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{r-1} \end{bmatrix} = - \begin{bmatrix} \sigma_0 & \cdots & \sigma_{r-1} \\ \vdots & \ddots & \vdots \\ \sigma_{r-1} & \cdots & \sigma_{2r-2} \end{bmatrix}^{-1} \begin{bmatrix} \sigma_r \\ \vdots \\ \sigma_{2r-1} \end{bmatrix}.$$

Proof. See Appendix A.1.4. □

Remark 2.2.3 (Computational complexity)**.** *Theorem 2.2.2 provides a methodology for the reconstruction of the observable spectrum of the unknown graph matrix* S *from* $2n$ *output observations. From a computational point of view, this method involves the inversion of a lower triangular* $2r \times 2r$ *matrix, the inversion of an* $r \times r$ *Hankel matrix, and finding the roots of a degree-*r *polynomial, where* r *is the rank of* H*. This leads to an overall worst-case complexity of* $O(r^3)$ *due to the matrix inversion.*

2.3. Continuous-Time Dynamics

In the case of continuous-time (CT) dynamics, there are subtle but important differences to that of discrete-time. Fortunately, similar results can still be derived in this domain, as we describe in the following subsections.

2.3.1. Network of simple agents

We begin our exposition by considering the case of a network of agents obeying the following CT dynamics:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= S\mathbf{x}(t) + \mathbf{u}(t),\ \mathbf{x}(0) = \mathbf{x}_0, \\ y(t) &= \mathbf{c}^\intercal \mathbf{x}(t), \end{aligned} \tag{2.15}$$

where $\mathbf{u}(t) \in \mathbb{R}^n$ is a periodic control input so that $u(t + r\tau) = u(t)$ for some period $r\tau$, $S \in \mathbb{R}^{n \times n}$ is a graph matrix whose connectivity structure matches that of a potentially weighted or directed graph $\mathcal{G}$. We thus have $\mathbf{x}(t) = e^{St}\mathbf{x}_0 + \int_0^t e^{S(t-s)}\mathbf{u}(s)ds$. In practice, we

consider discrete samples y_k of the output with an arbitrary period[6] $\tau > 0$, i.e., $y_k \coloneqq y(k\tau)$ for $k \in \mathbb{N}$. Using the Jordan decomposition $S = VJV^{-1}$, we have

$$y_k = \mathbf{c}^\intercal V e^{Jk\tau} V^{-1}\mathbf{x}_0 + \int_0^{k\tau} \mathbf{c}^\intercal V e^{J(k\tau - s)} V^{-1}\mathbf{u}(s)ds.$$

In contrast to the discrete-time case, here the observations are comprised of exponentiated Jordan matrices, where the $m_i \times m_i$ exponentiated Jordan block $e^{Jk\tau}$ is the upper-triangular matrix

$$e^{Jk\tau} = \begin{bmatrix} e^{\lambda_i k\tau} & k\tau e^{\lambda_i k\tau} & \cdots & \frac{(k\tau)^{(m_i-1)}}{(m_i-1)!}e^{\lambda_i k\tau} \\ & e^{\lambda_i k\tau} & \cdots & \frac{(k\tau)^{(m_i-2)}}{(m_i-2)!}e^{\lambda_i k\tau} \\ & & \ddots & \vdots \\ & & & e^{\lambda_i k\tau} \end{bmatrix}. \tag{2.16}$$

With this in mind, we define the differences of output measurements $\Delta y_{r+k} \coloneqq y_{r+k} - y_k$. Then, similarly to Lemma 2.2.1, we can show (see Appendix A.1.5) that

$$\begin{aligned} \Delta y_{r+k} &= \mathbf{c}^\intercal V e^{Jk\tau} V^{-1}\left(\mathbf{x}(r\tau) - \mathbf{x}_0\right) \\ &= \sum_{i\in\mathcal{I}} \sum_{s=0}^{\tilde{m}_i - 1} \bar{\omega}_i^{(s)} \frac{(k\tau)^s}{s!} \left(e^{\lambda_i \tau}\right)^k, \end{aligned} \tag{2.17}$$

where, for these continuous-time dynamics, in an analogous form to (2.8) and (2.9) we define

$$\bar{\omega}_i^{(s)} \coloneqq \sum_{j:\lambda_j=\lambda_i} \sum_{q=s+1}^{m_i} [\mathbf{c}^\intercal V]_{i,q-s} [V^{-1}(\mathbf{x}(r\tau) - \mathbf{x}_0)]_{i,q}.$$

In the absence of a control input, we have that $r = 0$ and $\Delta y_k = y_k$; hence, $y_k = \mathbf{c}^\intercal V e^{Jk\tau} V^{-1}\mathbf{x}_0$, and the following results apply accordingly. Similarly to the discrete-time case, the set of observable eigenvalues is $\mathcal{S}_S \coloneqq \left\{\lambda_i \in \sigma(S) \colon \exists s \text{ s.t. } \bar{\omega}_i^{(s)} \neq 0\right\}$ which, with an arbitrary random initial condition $\mathbf{x}_0$, is almost surely the set of observable eigenmodes

[6] Practically speaking, τ should be large enough to allow multiple sequential measurements of the system output and to prevent numerical issues.

of the pair $(S, \mathbf{c}^\intercal)$ according to the PBH test. Fortunately, we may apply analogous results to those in Section 2.2 in order to estimate the eigenvalues corresponding to observable eigenmodes. This notion is formalized in the corollary below.

Corollary 2.3.1. *Given the sequence of observations* $(y_k)_{k=0}^{3n-1}$ *from the continuous-time system in* (2.15) *with fixed sampling rate* $\tau > 0$ *and control inputs with period* $r\tau$*, the observable eigenvalues of the graph matrix* S *may be obtained via*

$$\lambda_i = \log(\eta_i)/\tau,$$

where η_i *are the roots of the polynomial*

$$p_S(x) = x^r + \alpha_{r-1}x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

whose coefficients $\alpha_0, \ldots, \alpha_{r-1}$ *are obtained from the observations* $(y_k)_{k=0}^{3n-1}$ *as in Theorem 2.2.1.*

Proof. See Appendix A.1.5. □

2.3.2. Network of Identical Continuous-Time Agents

Similarly to the setting in Section 2.2.1, we consider the dynamics of a network of continuous-time agents beyond single integrators. Assume that each agent is a linear system with state matrix A, whose input is a linear combination of the state of its neighbors, its initial state is proportional to a vector $\boldsymbol{\beta}$, and the measured output is the linear combination $y(t) = \sum_i c_i \boldsymbol{\gamma}^\intercal \mathbf{x}_i(t)$. Hence, the global dynamics of the network can be described (in a compact form) analogously to the discrete-time case as

$$\begin{aligned} \dot{\mathbf{x}}(t) &= (I_n \otimes A + S \otimes I_d)\,\mathbf{x}(t),\ \mathbf{x}(0) = \mathbf{x}_0 \otimes \boldsymbol{\beta}, \\ y(t) &= (\mathbf{c} \otimes \boldsymbol{\gamma})^\intercal\, \mathbf{x}(t). \end{aligned} \tag{2.18}$$

Hence, considering a sampling period $\tau > 0$, we have

$$\begin{aligned} y_k \coloneqq y(k\tau) &= (\mathbf{c} \otimes \boldsymbol{\gamma})^{\intercal} e^{(I_n \otimes A + S \otimes I_d)k\tau} (\mathbf{x}_0 \otimes \boldsymbol{\beta}) \\ &= (\mathbf{c} \otimes \boldsymbol{\gamma})^{\intercal} \left(e^{Sk\tau} \otimes e^{Ak\tau} \right) (\mathbf{x}_0 \otimes \boldsymbol{\beta}), \end{aligned}$$

Where the last equality follows by commutativity of the identity matrix and properties of the Kronecker product [66]. Thus,

$$y_k = \left(\mathbf{c}^{\intercal} V e^{Jk\tau} V^{-1} \mathbf{x}_0 \right) \left(\boldsymbol{\gamma}^{\intercal} e^{Ak\tau} \boldsymbol{\beta} \right) = \nu_k \sum_{i=1}^{d} \sum_{s=0}^{m_i - 1} \omega_i^{(s)} \frac{(k\tau)^s}{s!} (e^{\lambda_i \tau})^k,$$

where $\nu_k \coloneqq \boldsymbol{\gamma}^{\intercal} e^{Ak\tau} \boldsymbol{\beta}$ and $\omega_i^{(s)}$ is defined in (2.8). The following corollary describes how we may estimate the observable eigenvalues of this system from a sequence of output measurements.

Corollary 2.3.2. *Given the sequence of observations $(y_k)_{k=0}^{2n-1}$ from the continuous-time system in* (2.18) *with fixed sampling rate $\tau > 0$, the observable eigenvalues of the graph matrix S may be obtained via $\lambda_i = \log(\eta_i)/\tau$, where η_i are the roots of the polynomial*

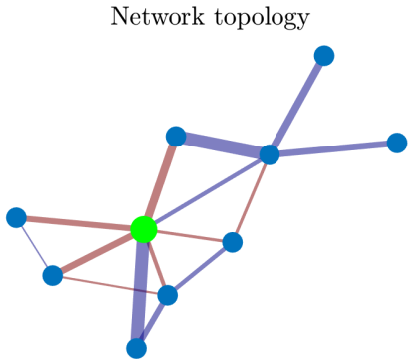
$$p_S(x) = x^r + \alpha_{r-1} x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

whose coefficients $\alpha_0, \ldots, \alpha_{r-1}$ are obtained from $\sigma_k \coloneqq \sum_{i=1}^{d} \sum_{s=0}^{m_i - 1} \omega_i^{(s)} \frac{(k\tau)^s}{s!} (e^{\lambda_i \tau})^k$ computed from the observations $(y_k)_{k=0}^{2n-1}$, as in Theorem 2.2.2.

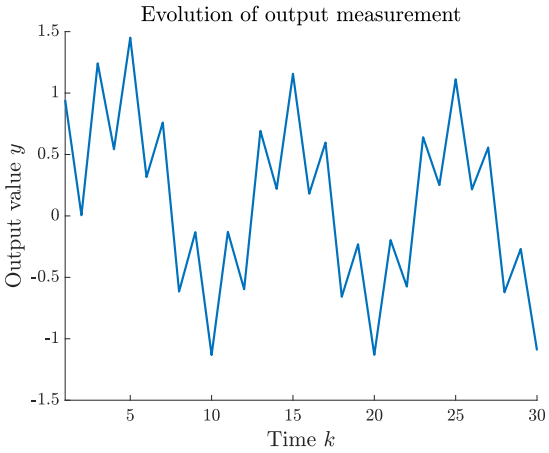
Proof. Follows similarly to Corollary 2.3.1, making use of Theorem 2.2.2. □

2.4. Simulations

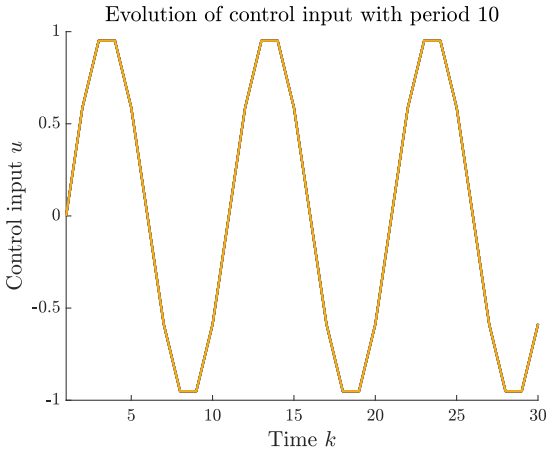
In this section we illustrate our results, in both discrete- and continuous-time, on networks where the underlying network structure is unknown to us. The evolution of the dynamics of these systems are simulated with an arbitrary random initial condition vector $\mathbf{x}_0$ and an observability vector $\mathbf{c}$. Both $\mathbf{x}_0$ and $\mathbf{c}$ are unknown to the algorithm. Then, we apply our



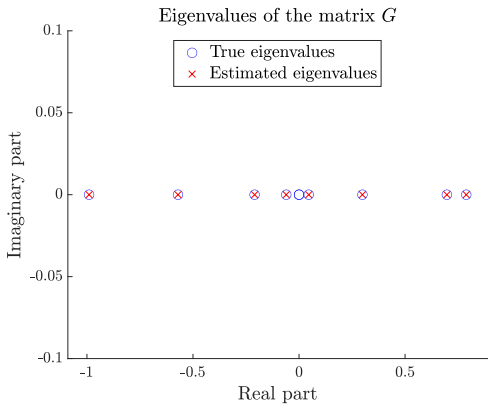
(a) Network topology, with single output agent highlighted.



(b) Output $y[k] = e_i^\intercal S^k x_0$, where we observe only agent i.



(c) Periodic control input $\mathbf{u}[k] = \sin(2k\pi/10)\mathbf{1}$ with period 10.

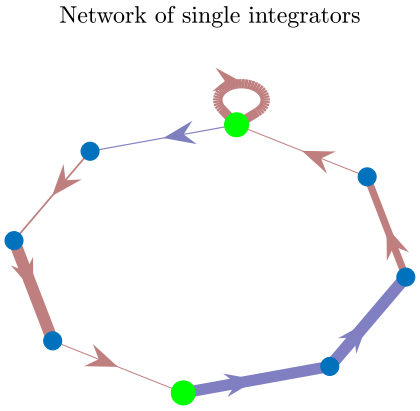


(d) Comparison of true and estimated eigenvalues; repeated values overlaid.

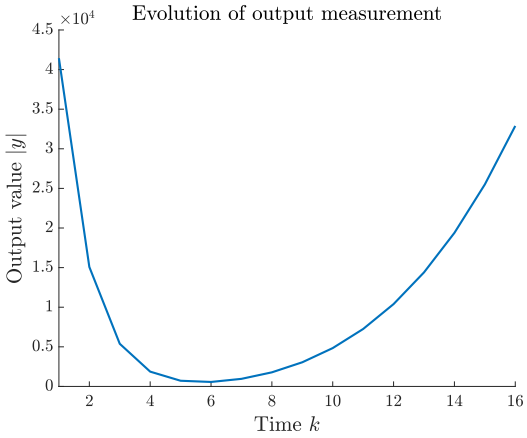
Figure 2.1: 10-agent preferential attachment network in discrete-time, generated according to [67]. Edge thickness in (a) corresponds to edge weight, and red edges have negative weights. Weights are randomly generated according to Uniform$[-1, 1]$, and initial condition is randomly generated as $\mathbf{x}_0 \sim \text{Uniform}[0, 1]^n$. There are 8 observable (non-zero) eigenvalues of S in this case, which are recovered via our estimation approach.

algorithms to estimate the eigenvalues of S from the sequence of observations $(y[k])_{k=0}^{3n-1}$ and compare our estimated eigenvalues against the true spectrum of the graph matrix S. Note that all $3n - 1$ observations may not be needed, as described in Remark 2.2.1.

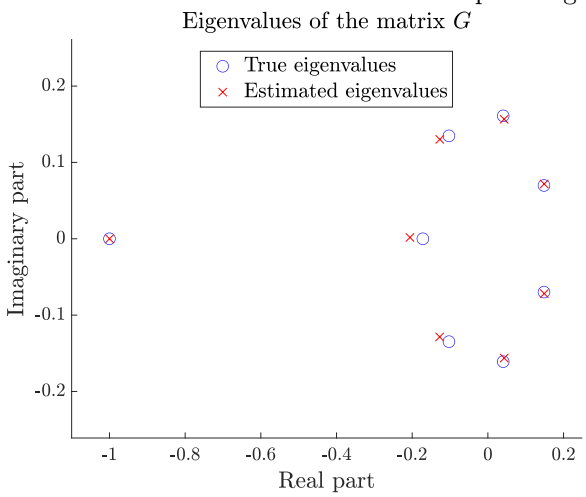
Figure 2.1 shows the application of Theorem 2.2.1 on the undirected, randomly generated 10-agent preferential attachment network shown in Figure 2.1(a) (see [67] for a full discussion of



(a) Network topology, with output agents highlighted.



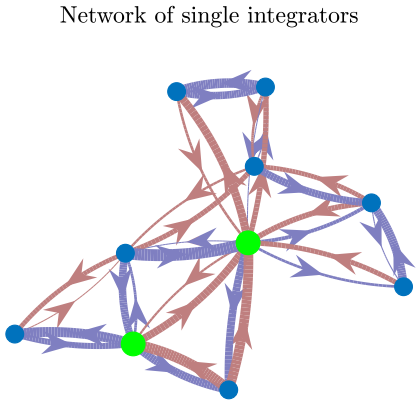
(b) Output $y_k = c^\intercal e^{-Sk\tau} x_0$; agents are observed with equal weight.



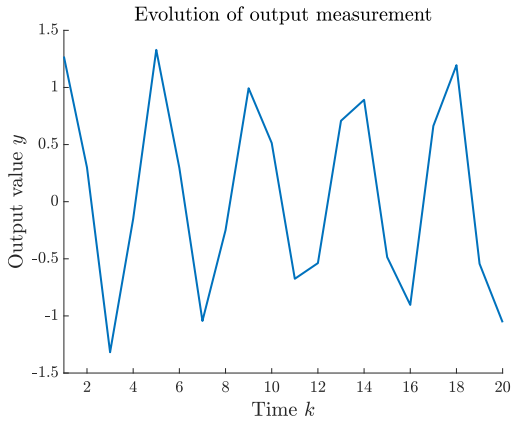
(c) Comparison of true and estimated eigenvalues.

Figure 2.2: 8-agent single integrator ring network in continuous-time, with sampling rate $\tau = 1$, random initial condition $\mathbf{x}_0 \sim \text{Uniform}[0,1]^n$, and edges weighted as in Figure 2.1. Even though the system is unstable, all 8 eigenvalues of S are are recovered with high accuracy via our estimation approach.

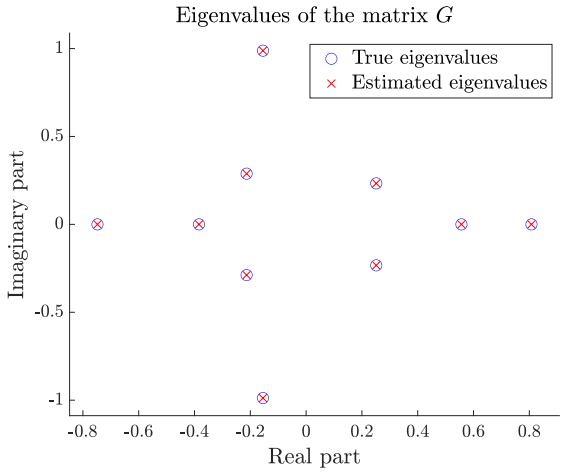
preferential attachment networks). The thickness of edges is proportional to their weight in the graph matrix S, with negative weights shown in red; the weights are generated according to a Uniform$[-1,1]$ distribution. We model each agent using the dynamics in (2.4), and each agent applies the control input in Figure 2.1(c). We assume that we only have access to the output of the agent indicated in Figure 2.1(a). In Figure 2.1(b), we show the evolution of the output signal; as only one agent's output is measured, this is a decentralized eigenvalue



(a) Network topology, with output agents highlighted in green.



(b) $y[k] = (\mathbf{c}\otimes\boldsymbol{\gamma})^{\intercal}(I_n\otimes A+S\otimes I_d)^k(\mathbf{x}_0\otimes\boldsymbol{\beta})$; agents are observed with equal weight.



(c) Comparison of true and estimated eigenvalues.

Figure 2.3: 10-agent preferential attachment network in discrete-time, generated according to [67]. The dynamics here follow the more general case of (2.14) from Section 2.2.1, where each node has a 3-dimensional state. The initial condition is generated according to $\mathbf{x}_0 \sim \text{Uniform}[0,1]^n$, the vectors $\boldsymbol{\beta}$ and $\boldsymbol{\gamma}$ are generated according to $\text{Uniform}[0,1]^3$, and the entries of the symmetric matrix A are generated as $a_{ij} \sim \text{Uniform}[0,1],\ i \geq j$. In this case S has 10 eigenvalues, which are all recovered via our estimation approach.

estimation problem. Figure 2.1(d) compares both the true and estimated eigenvalues of S. In this case there are 8 observable eigenvalues of S (the rest being identically zero), all of which are recovered using a sequence of 30 measurements retrieved from a single agent.

In Figure 2.2 we apply Corollary 2.3.1 on the 8-agent weighted and directed ring network shown in Figure 2.2(a), wherein the agents obey the continuous-time dynamics described in Section 2.3. In order to compare with other works in the literature, which consider only autonomous systems, we apply no control input. In this case the output is a linear combination of the states of the two agents highlighted in Figure 2.2(a). In this case the eigenvalues of the graph matrix S are complex, and the choice of S renders the system unstable, but we are still able to recover all eigenvalues with high accuracy.

In Figure 2.3 we estimate the eigenvalues of a network of 10 discrete-time identical agents. Figure 2.3(a) displays a randomly generated preferential attachment network over which the agents interact. In this case the edges are weighted according to a Uniform$[-1, 1]$ distribution, with thickness representing weight and negatively-weighted edges shown in red. The measurements we observe in Figure 2.3(b) are the sum of the outputs of the two agents located on the green nodes of the network. After applying Theorem 2.2.2, all 10 eigenvalues are recovered as shown in Figure 2.3(c).

Remark 2.4.1 (Numerical considerations)**.** *While Theorem 2.2.1 guarantees exact recovery of the observable eigenvalues of S, in practice there will be numerical errors due to computing the roots of a high-degree polynomial. Moreover, while we do not require the system to be stable to recover the eigenvalues, an unstable system will cause the quantities S^k and e^{St} to become large, introducing numerical issues when inverting the Hankel matrix. For example, our choice of S in Figure 2.2 renders the system unstable (as shown in Figure 2.2(b)); however, we still recover the entirety of the true spectrum of S with high accuracy as shown in Figure 2.2(c). The difference in accuracy from Figure 2.1 is due to the numerical sensitivity of root-finding techniques, since the outputs are large due to the system being unstable.*

CHAPTER 3

Network Alignment

M. Hayhoe, F. Barreras, H. Hassani, and V. M. Preciado, "SPECTRE: Seedless network alignment via spectral centralities," arXiv preprint arXiv:1811.01056*, 2018*

Network alignment is the problem of finding a structure-preserving correspondence between the nodes of two correlated, not necessarily identical, networks. An accurate solution to this problem would address central issues in different fields, varying from the deanonymization of social networks, to recognition tasks in computer vision, to the alignment of proteins in computational biology. An example of two correlated networks is given in Figure 3.1, with a correspondence that is indicated by the layout of the nodes.

An application of network alignment can be found in social network analysis, where it is possible to discover the identities of individuals in an anonymous network by aligning its structure with that of a correlated network in which nodes are identified [68]. From a marketing perspective, finding individuals which play similar roles across platforms allows advertisers to integrate information from different domains in order to target ads and product recommendations [69]. In computer vision, network alignment is used for tasks such as object recognition [70], image registration [71], or symmetry analysis [72]. In these problems nodes may represent salient points, lines, shapes, or other features in images, while edges are used

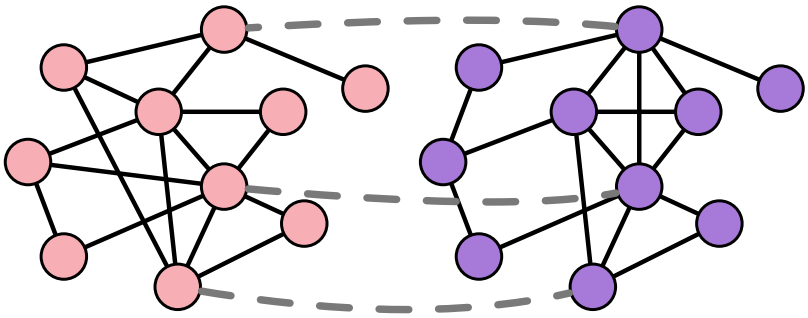


Figure 3.1: Correct correspondence of some nodes in two correlated networks, illustrated by dashed lines. A network alignment algorithm should find these correspondences.

to encode distances between them. A further application can be found in computational biology; in particular, the study of protein-protein interaction (PPI) networks [73]–[75]. PPI networks provide an understanding of the system-level functions of each protein, as well as insights into how biological motifs are conserved through evolution. However, due to mutations, these corresponding proteins often have different compositions [76] and, thus, the alignment of PPI networks needs to rely on their structural correlation. Another biological application of network alignment can be found in the problem of determining gene-disease causation [77], where the alignment of disease and PPI networks can be used to produce high-quality gene-disease candidates.

In a pioneering work, Narayanan and Shmatikov [68] succeeded in de-anonymizing a large-scale dataset from Netflix using publicly available auxiliary information on some users, which sparked controversy and contributed to a data privacy lawsuit [78]. Subsequent papers on network alignment assumed the availability of side information in the form of a seed set, i.e., a set of correctly-aligned nodes. This set might be used in a *seed-and-expand* strategy in which percolation techniques are used to "grow" a correct alignment throughout the nodes [79]. Alternatively, the alignment can be grown locally by using the Hungarian algorithm [80] or other relaxations of quadratic optimization problems [81]. The inclusion of prior information in the form of a seed set typically results in much higher precision (see, e.g., FINAL [82]); however, in many cases such a seed set is difficult, if not impossible, to obtain.

Another strategy in the literature involves constructing a node-specific signature which is then used to align the networks. This is done by matching pairs of nodes with similar signatures. Many such signatures have been proposed, ranging from simple neighborhood statistics [83], [84], to spectral signatures [81], [85], to distance to "important" nodes [86], to more complex networks embeddings [87]. However, the matching step in many of these algorithms has time complexity of $O(n^2)$ or higher, which quickly becomes infeasible for even moderately-sized networks. Algorithms like REGAL [87] and LowRankAlign [85] are exceptions, since they use low rank approximations to obtain scalable algorithms.

The *seed-and-expand* as well as the signature-based algorithms described above present a critical limitation: they only produce high quality alignments when the two networks to be aligned have very high edge correlation (e.g., see Figure 4 in [87]). This limitation poses a great challenge for the applicability of these algorithms, since the networks in many interesting real-world applications do not exhibit near-perfect correlation. This critical limitation in state-of-the-art algorithms can be intuitively understood as follows. In the seed-and-expand case, a percolation process will propagate incorrect alignments in a cascading fashion; in signature-based algorithms, it is very challenging to create node-specific signatures that are global and are robust to moderate structural correlations. To the best of our knowledge, there are no network alignment algorithms that obtain high accuracy and scalability without assuming near-perfect correlation of the networks or an initial set of correct pairs.

In this chapter we present `SPECTRE`, a scalable algorithm that uses spectral centrality measures together with bootstrap percolation techniques [88], [89] to align networks with high accuracy. Unlike most network alignment algorithms, `SPECTRE` requires no seeds (i.e., pairs of nodes identified beforehand) or side information. Instead, the algorithm is based on a *seed-and-expand* strategy; in the *seed* phase, `SPECTRE` generates an initial noisy estimate set via spectral centrality measures which is then used in the *expand* phase to robustly grow an alignment of the whole network. We show that while this seed estimate may contain a majority of incorrect pairs, this noise has little impact on the final alignment.

We present extensive numerical results describing the performance of our algorithm, including comparisons to existing algorithms in social and biological networks. As our results demonstrate, `SPECTRE` is able to align moderately correlated, large-scale networks with high accuracy. Moreover, `SPECTRE` shows a noticeable improvement over the state-of-the-art methods in aligning PPI networks. For example, on PPI networks of *C. jejuni* and *E. coli* bacteria, the best performance in the literature for two popular metrics, called *Edge Correctness* and *Induced Conserved Structure* (ICS) score [81], are 24% and 9%, respectively. However, by using `SPECTRE`, we obtain a 32% edge correctness and a 35% ICS score.

Our contributions can be summarized as follows:

- We introduce `SPECTRE`, a new algorithm based on iterated bootstrap percolation. To the best of our knowledge, this is the first scalable algorithm able to accurately and robustly align pairs of networks exhibiting moderate correlation using no prior information.

- We propose a method for generating an initial seed set estimate using eigenvector centrality to rank nodes. This noisy seed set may have a majority of incorrect pairs; however, `SPECTRE` can successfully screen these incorrect pairs using bootstrap percolation.

- Through extensive numerical experiments, we show that `SPECTRE` can recover high-accuracy alignments on both synthetic and real-world networks. Moreover, we compare `SPECTRE` to other algorithms in the literature, showing that it outperforms them when networks exhibit ground-truth correlation below 95%.

3.1. Related Work

As mentioned earlier, we can broadly categorize the algorithms found in the literature in two categories: (i) *seed-and-expand* type algorithms, and (ii) algorithms that match nodes according to their similarity given by some signature or embedding. While the *seed-and-expand* type of algorithms exhibit high quality performance in terms of precision and scalability, they make the critical assumption that the user has a seed set of initial pairs of nodes that are correctly aligned. In most real world applications such a seed set could be very costly to obtain, if possible at all. Although *signature-similarity* based algorithms can potentially overcome the need for a seed set, their performance is highly dependant on the type of node signature used and the construction of the similarity matrix is computationally demanding, rendering most of these algorithms unscalable.

The idea of using a *seed set* to align datasets can be traced back to Narayanan and Shmatikov's 2009 paper [68], where the authors used side information (in the form of an

attribute matrix) to de-anonymize large scale sparse datasets. In the context of networks, the work by Pedarsani and Grossglauser [90] was the first to give a theoretical treatment to the problem of network alignment, as well as the first to introduce the $G(n, p; s)$ network generation model, which is widely used to generate correlated networks on which to test algorithms. These pioneering papers gave theoretical grounds to many other algorithms which assume side information in the form of a *seed set* [79], [80], [91]. In particular, Yartseva and Grossglauser [79] introduce an algorithm that uses ideas from *bootstrap percolation* [92]: starting from a *seed set*, additional pairs are aligned if there are at least r aligned pairs that are "neighbors" of it (a precise definition of "neighboring pairs" will be provided later). Bootstrap percolation methods are both scalable and accurate, and some variations of them, for example [89], can considerably reduce the size of the *seed set* required for good performance.

Another family of algorithms attempts to solve the network alignment problem by designing node-level signatures and then aligning nodes with similar signatures. Many such signatures have been proposed, ranging from simple neighborhood statistics [83], [84], to spectral signatures [81], [85], to distance to "important" nodes [86], to more complex networks embeddings [87]. This approach has several scalability challenges since constructing a full similarity matrix for the nodes and obtaining a maximum weight matching cannot be solved (exactly) in linear time. Such computational considerations have motivated the development of multiple algorithms in the literature trying to combine a rich node signature with fast approximation algorithms for node matching, often as separate components. For example, the GRAAL family of algorithms uses a signature based on graphlet-degree distributions and matches nodes with a range of methods ranging from the Hungarian algorithm [83] to *seed and expand* [80] methods. Some of the most notable recent developments are signature-based algorithms like REGAL [87], FINAL [82] and gsaNA [86] which, making use of low-rank approximations and dimensionality reduction techniques, scale well to networks of hundreds of thousands of nodes. However, it is empirically observed that these algorithms only produce high-quality alignments when the two networks have near-perfect correlation. Indeed, many of these

signatures are built using spectral quantities related to eigenvectors, which are known to be unstable to graph perturbations [2], [93]. Hence, there is a need for a robust, scalable and *seedless* algorithm that produces high-quality alignments even on moderately correlated networks.

A critical part of the literature in network alignment deals with understanding and curbing error propagation in *seed-and-expand* algorithms [79], [89]. Most notably, the authors of [89] describe a bootstrap percolation strategy that is robust to the presence of incorrect pairs in the seed set and provably percolates to the whole network (on certain synthetic graphs). Simply put, this algorithm is more robust because it defers the matching of a pair of nodes until it accumulates enough "neighboring candidate pairs" (referred to as *tokens* herein). Our proposed algorithm leverages this idea, in conjunction with a boosting strategy, to overcome the dependance on a *seed set*, allowing us to obtain high-quality alignments even in moderately correlated networks.

Finally, we mention that in recent years there is growing literature regarding attributed network alignment and multiple network alignment. Most notably, the paper by Kazemi and Grosglausser [94] proposes the creation of a seed set in combination with a seed-and-expand strategy in the context of aligning multiple attributed networks. In contrast with the work in [94], the aim of this work is in improving the performance of state-of-the-art algorithms for the purely structural alignment problem.

3.2. Preliminaries

In this work we consider undirected graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of unweighted edges. We assume $\mathcal{G}$ is simple, i.e., it has no self-loops or multi-edges. The product graph $\mathcal{G}_{1\times 2}$ of two networks $\mathcal{G}_1$ and $\mathcal{G}_2$ is the graph with vertex set $\mathcal{V}_{1\times 2} = \mathcal{V}_1 \times \mathcal{V}_2$, and edge set $\big\{\{(i,j),(u,v)\} : \{i,u\} \in \mathcal{E}_1, \{j,v\} \in \mathcal{E}_2\big\} \subseteq \mathcal{V}_{1\times 2} \times \mathcal{V}_{1\times 2}$. The set of all neighbouring pairs of (i,j) in the product graph $\mathcal{G}_{1\times 2}$ are denoted by $\mathcal{N}_{(i,j)} = \big\{(u,v) \in \mathcal{V}_{1\times 2} : \{i,u\} \in \mathcal{E}_1, \{j,v\} \in \mathcal{E}_2\big\}$.

Symbol	Meaning
$\mathcal{G}$	Graph (undirected)
$\mathcal{G}_1,\ \mathcal{G}_2$	Correlated graphs
$\mathcal{G}_{1\times 2}$	Product graph
$\mathcal{V}$	Vertex set, i.e., $\{1,\dots,n\}$
$\mathcal{E}$	Edge Set, subset of $\mathcal{V}\times\mathcal{V}$
$i\sim j$	$\{i,j\}\in\mathcal{E}$
$\mathcal{N}_i(\mathcal{G})$	Neighbors of node i in graph $\mathcal{G}$
$\mathcal{N}_{(i,j)}$	Neighbors of pair (i,j) in product graph $\mathcal{G}_{1\times 2}$
$D(\mathcal{G})$	Degree matrix, i.e., $\mathrm{diag}\{\lvert\mathcal{N}_1(\mathcal{G})\rvert,\dots,\lvert\mathcal{N}_n(\mathcal{G})\rvert\}$
$A=A(\mathcal{G})$	Adjacency matrix, $[A]_{ij}=\mathbf{1}\{i\sim j\}$
$\lambda_i(M)$	ith eigenvalue of $M\in\mathbb{R}^{n\times n}$ (decreasing magnitude)
$\lambda_{max}(M)$	largest eigenvalue of M, i.e., $\lambda_1(M)$
$s(i,j)$	score of pair (i,j) from $\mathcal{G}_{1\times 2}$

Table 3.1: Notation for network alignment

Formally, the problem of network alignment on two graphs $\mathcal{G}_1=(\mathcal{V}_1,\mathcal{E}_1)$ and $\mathcal{G}_2=(\mathcal{V}_2,\mathcal{E}_2)$ is to find a matching set $\mathcal{M}\subset\mathcal{V}_1\times\mathcal{V}_2$, so that $(i,j)\in\mathcal{M}$ means $i\in\mathcal{V}_1$ corresponds to the same unique entity as $j\in\mathcal{V}_2$, which we write as $i\leftrightarrow j$. For example, in the context of social networks, we may imagine matching the account of an individual on Twitter to a Facebook account owned by the same individual. However, since the sets of users may be different in both networks, we may only hope to match pairs of nodes in $\mathcal{V}_1\cap\mathcal{V}_2$. Given a matching $\mathcal{M}$, we will define the correspondence $f_{\mathcal{M}}:\mathcal{V}_1\to\mathcal{V}_2$ as $f_{\mathcal{M}}(i)=j$ if $(i,j)\in\mathcal{M}$, and undefined otherwise. Similarly, we define the set of matched nodes in $\mathcal{V}_2$ as $\mathcal{M}(\mathcal{V}_1)=\{f_{\mathcal{M}}(i):i\in\mathcal{V}_1,\ i\text{ matched}\}$ and the set of matched edges as $\mathcal{M}(\mathcal{E}_1)=\{(f_{\mathcal{M}}(i),f_{\mathcal{M}}(j)):(i,j)\in\mathcal{E}_1;\ i,j\text{ matched}\}$, which may include edges not present in $\mathcal{E}_2$.

In order to measure the performance of our proposed algorithm, `SPECTRE`, it is necessary to have correlated networks where the true matching of nodes is available. In our numerical experiments, we create such networks from real-world data [95]. In order to symmetrically generate two networks while preserving access to the the true matching information, we use the following procedure. We start with an arbitrary graph $\mathcal{G}=(\mathcal{V},\mathcal{E})$ and generate a random graph $\tilde{\mathcal{G}}_1$ by independently subsampling each edge in $\mathcal{G}$ with a probability $1-s$. Hence, $\tilde{G}_1$ has the same node set as $\mathcal{G}$ and an expected number of edges $(1-s)|\mathcal{E}|$. We

repeat this procedure, independently, to obtain a second graph $\tilde{\mathcal{G}}_2$. Notice that, as a result of subsampling edges, $\tilde{\mathcal{G}}_1$ and/or $\tilde{\mathcal{G}}_2$ may become disconnected. To overcome this issue, we follow the iterative procedure described below. In a first step, we find the largest connected components of the two graphs, denoted by $\tilde{C}_1$ and $\tilde{C}_2$ respectively. We then look at the subgraphs of $\tilde{\mathcal{G}}_1$ and $\tilde{\mathcal{G}}_2$ induced by the nodes in $\tilde{C}_1 \cap \tilde{C}_2$. If these induced subgraphs are connected, we call them $\mathcal{G}_1$ and $\mathcal{G}_2$ and take them as the pair of networks to be aligned; if they are disconnected, we find their largest connected components and repeat these steps. This approach to generate correlated random graphs was introduced in [90] where the following edge similarity measure was also proposed:

$$Sim_e(\mathcal{G}_1, \mathcal{G}_2) = 2 \sum_{i,j \in \mathcal{V}_1} \frac{\mathbf{1}\big\{\{i,j\} \in \mathcal{E}_1, \{i,j\} \in \mathcal{E}_2\big\}}{|\mathcal{E}_1| + |\mathcal{E}_2|},$$

although there are other ways to measure the similarity of two graphs [96].

3.3. Algorithms

In this section we introduce `SPECTRE`, a scalable algorithm able to solve the network alignment problem with high accuracy in the absence of side information. `SPECTRE` uses spectral properties of $\mathcal{G}_1$ and $\mathcal{G}_2$ to create a noisy initial seed set $\mathcal{S}$, which will contain a number of correct pairs and many incorrect ones, as we will describe in Section 3.3.1. This initial set is not a proper matching, since the same node can be present in numerous pairs. $\mathcal{S}$ is then used to build a confident seed estimate $\mathcal{M}_0$, where nodes appear in at most one pair, following a strict percolation procedure described in Section 3.3.3. `SPECTRE` then performs a backtracking step, resetting the matching and using $\mathcal{M}_0$ as a new seed estimate. In Section 3.3.4, we propose a relaxed, looser, percolation which uses the confident seed estimate to percolate a matching $\mathcal{M}$ over the networks. Finally, if the percolation does not grow above a fraction f of the networks' size, we backtrack by using the final matching as input to the algorithm again as if it were a noisy seed set, and the process is repeated. Through this backtracking procedure `SPECTRE` is able to build a final matching that has significantly higher accuracy, even when the networks exhibit low correlation. Typically we choose $f = 3/4$, but this

parameter may be increased if a larger matching is desired. In Algorithm 1 below, we provide the general structure of `SPECTRE`, and in the following subsections we describe each subroutine in detail.

Algorithm 1 `SPECTRE`$(\mathcal{G}_1, \mathcal{G}_2, k, w, r)$

Input: graphs to align $\mathcal{G}_1$,$\mathcal{G}_2$; number of top seeds k; size of window w; token threshold r
Output: matching $\mathcal{M}$

```
C_1, C_2 ← eigenvector centralities of G_1 and G_2 (resp.)
S ← EstimateSeeds(k, w, C_1, C_2)
while |M| < f * min{|V_1|, |V_2|} do
    M_0 ← SafeExpand(S, r)          ▷ Confident percolation
    S ← M_0                         ▷ Backtracking
    M ← LooseExpand(S)              ▷ Relaxed percolation
    S ← M                           ▷ Backtracking
end while
return M
```

3.3.1. `EstimateSeeds` subroutine

`EstimateSeeds`, the first subroutine in `SPECTRE`, constructs a noisy seed set estimate which should contain some number of correct pairs, i.e. pairs of nodes that are correctly matched across the networks. To ensure that this occurs, nodes across networks should be matched using a procedure that is robust to perturbations in the network structure. In `SPECTRE`, this procedure is based on comparing the spectral centralities of nodes in different networks; an in-depth description of this choice is presented in Section 3.3.2. In particular, in order to create a noisy seed set estimate $\mathcal{S}$, we rank the nodes of $\mathcal{G}_1$ and $\mathcal{G}_2$ by their centrality scores and keep the top k most central nodes in each network. The rationale behind this choice of potential matches is that, for correlated graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, nodes with high centrality in $\mathcal{G}_1$ are likely to be aligned with nodes of high centrality in $\mathcal{G}_2$. Furthermore, the centrality ranking of matched nodes should be similar for the most central nodes. As a result, $\mathcal{S}$ contains $(2w+1)k - w(w+1)$ pairs, of which no more than k represent correct matches. While higher values of w increase the probability of finding k correct pairs, it also increases the number of incorrect pairs by $O(k)$, and thus we must be conservative with our choice of both parameters. Taking $k = O(\log n)$ and $w = 1, 2$ performs well in practice; typically we set $k = 10 \log n$ and $w = 1$.

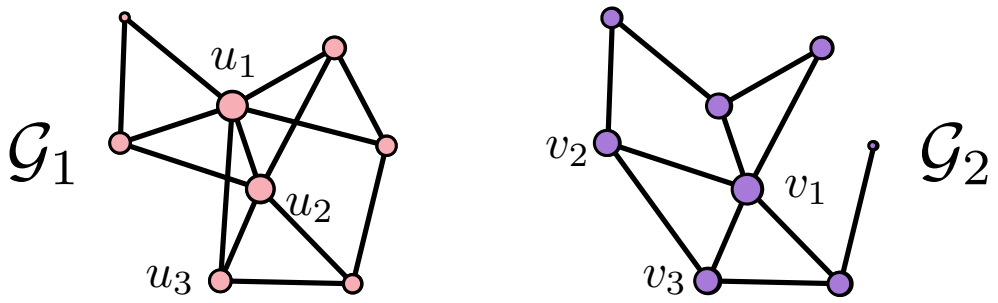


Figure 3.2: Example of `EstimateSeeds`. The size of each node denotes its relative spectral centrality score in the network, and the ground-truth correspondence is illustrated by the position of the nodes. Here with $k = 3$ and $w = 1$, u_1 is matched with v_1 and v_2; u_2 is matched with v_1, v_2 and v_3; and u_3 is matched with v_2 and v_3. Of these pairs, the seeds (u_2, v_1) and (u_3, v_3) are correct.

It is noteworthy to emphasize that the noisy seed set estimate $\mathcal{S}$, generated by the algorithm `EstimateSeeds`, typically contains a large fraction of incorrect pairs. However, since correct pairs increment scores of other correct pairs more effectively than incorrect pairs may increment scores among themselves, the algorithms `SafeExpand` and `LooseExpand` are able to robustly percolate and find a moderately accurate alignment, even in the presence of many incorrect pairs in $\mathcal{S}$. In some cases this initial iteration will in fact yield a highly accurate alignment, especially if the original network is dense. However, when the initial iteration is not enough, the backtracking step of `SPECTRE` is able to boost the moderately successful alignment in order to significantly increase both the size and accuracy of the final matching. As a result, provided that there are a sufficient amount of correct pairs in the noisy seed set estimate $\mathcal{S}$, then `SafeExpand` and `LooseExpand` will be able to overcome the presence of wrong pairs and percolate over the set of all the correct pairs.

Algorithm 2 `EstimateSeeds`(k, w, C_1, C_2)

Input: number of top seeds k; size of window w; centrality scores C_1, C_2
Output: noisy seed set $\mathcal{S}$

```
S ← ∅
for each of top k nodes i ∈ V_1 according to C_1 do
    add pairs (i, j) to S by selecting the correspondingly ranked node j ∈ V_2
    according to C_2, as well as w nodes before and after j in the ranking.
end for
return S
```

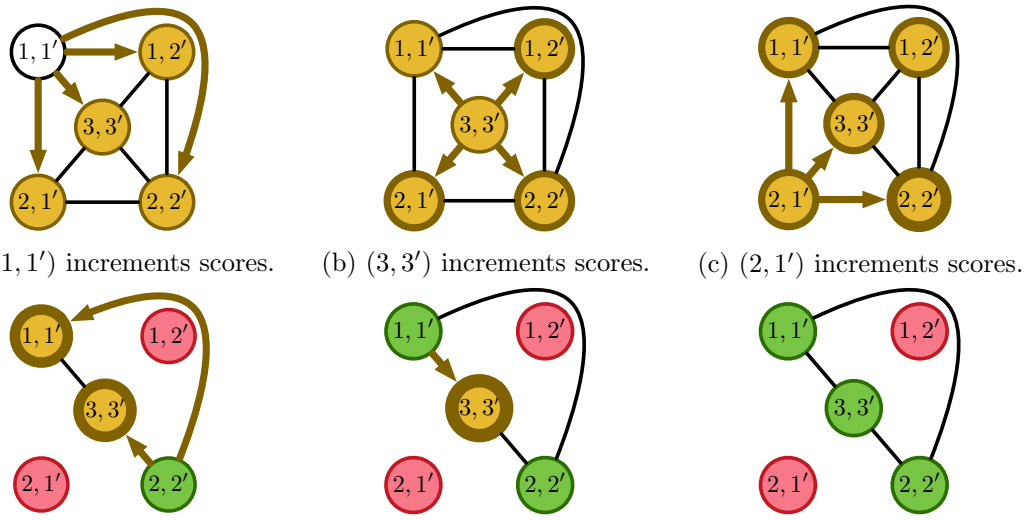
3.3.2. Centrality

As mentioned previously, there are many works which explore the creation of node signatures for use in network alignment problems [81], [83], [85], [87]. Of particular interest are those signatures which are robust to perturbations in network topology for the most central nodes. Specifically, we are interested in signatures which do not exhibit large changes in the relative rankings of the highest-scored nodes in the network when the nodes or edges are altered. For this reason, we choose a notion of *centrality* as such a nodal feature. Node centralities are commonly used to measure the importance of nodes, and can be used to estimate the influence of individuals in social networks [97], the importance of web pages [98], or the certainty of node measurements [99]. In `SPECTRE`, we use the *eigenvector* centrality ${C_{ev}}^7$, which is formally defined as

$$C_{ev}(i) = [v_1]_i, \tag{3.1}$$

where v_1 is the eigenvector of $A(\mathcal{G})$ for $\lambda_{max}(A)$.

This centrality measure is capable of being computed efficiently, even for large-scale networks. Indeed, modern algorithms allow the calculation in $O(m)$ time and storage, where $m = |\mathcal{E}|$, and the constants depend on $\lambda_1(A)$ and $\lambda_2(A)$ [100]. As mentioned earlier, eigenvectors are generally unstable to perturbations in the graph [2], [93] in that their individual entries may change dramatically. However, the eigenvectors corresponding to the largest eigenvalues are in fact the most robust to such perturbations. Indeed, in practice perturbations of the network topology do not dramatically change the *ranking* induced by this centrality measure for the nodes with the highest centrality. It is for this reason that eigenvector centrality is useful as a ranking metric for central nodes, but is ineffective when used as a node signature for all nodes in moderately correlated networks.



(a) $(1, 1')$ increments scores.

(b) $(3, 3')$ increments scores.

(c) $(2, 1')$ increments scores.

(d) Pair $(2, 2')$ matched and thus increments scores; $(1, 2')$, $(2, 1')$ are removed.

(e) Break tie at random, so $(1, 1')$ matched and then increments scores.

(f) Match $(3, 3')$. `SafeExpand` stops.

Figure 3.3: Example of `SafeExpand`, with matching threshold $r = 3$ and noisy seed set estimate $\mathcal{S} = \{(1, 1'), (3, 3'), (2, 1')\}$. White pairs have score zero, yellow have positive score (with border thickness denoting total number), green are matched, and red are removed (one or both nodes already matched in another pair). Arrows describe the direction in which scores are incremented.

3.3.3. `SafeExpand` subroutine

The subroutine `SafeExpand` uses the noisy seed set estimate $\mathcal{S}$ from `EstimateSeeds` to construct a confident seed estimate $\mathcal{M}_0$, which is a matching where each node is present in at most one pair (see Algorithm 3). For each possible pair (i, j) in $\mathcal{V}_{1\times 2}$, `SafeExpand` builds a confidence score $s(i, j)$ by allowing other pairs to increment these scores through the edges of the product graph $\mathcal{G}_{1\times 2}$. In practice, many of these scores will remain at zero since the networks we consider are not fully connected. As `SafeExpand` begins, each pair of nodes $(i, j) \in \mathcal{S}$ increases the score of all of its neighboring pairs in $\mathcal{G}_{1\times 2}$, i.e., all pairs in $\mathcal{N}_{(i,j)}$[8], by one. Notice that this set of neighboring pairs corresponds to all the pairs of nodes in $\mathcal{N}_i(\mathcal{G}_1) \times \mathcal{N}_j(\mathcal{G}_2)$. It is worth remarking that the originating pair (i, j) does not increment its own score. At the end of this spreading process, only pairs which are neighbors (in the

[7]Other centrality measures were tested, including PageRank, degree, betweenness, and closeness, but eigenvector performed best empirically.

[8]Recall $\mathcal{N}_{(i,j)} = \{(u, v) \in \mathcal{V}_{1\times 2} : \{i, u\} \in \mathcal{E}_1, \{j, v\} \in \mathcal{E}_2\}$.

product graph) of pairs in the noisy seed set estimate $\mathcal{S}$ will have a positive score.

In what follows, we sequentially grow the confident seed estimate $\mathcal{M}_0$ according to the following procedure, which repeats as long as some pair (i, j) has a score at least r, i.e., $s(i, j) \geq r$ for some $(i, j) \in \mathcal{V}_{1\times 2}$. Based on the percolation bounds established in [79], the value of r is typically chosen to be 4. First, we find the set of all pairs in $\mathcal{V}_{1\times 2}$ with the highest score, pick one of these pairs at random, and add it to the set $\mathcal{M}_0$. Next, this chosen pair increments the scores of its neighboring pairs (in the product graph) by one. According to this updated score, we pick the pair with the highest score (breaking ties at random), excluding any pair containing an already matched node (i.e., any node contained in any pair in $\mathcal{M}_0$). We then add the chosen pair to $\mathcal{M}_0$, increment the scores of its neighboring pairs, and repeat this procedure until the remaining unmatched pairs have less than r tokens. At the end of this procedure, we obtain the confident seed set estimate $\mathcal{M}_0$, representing a matching between nodes of $\mathcal{G}_1$ and $\mathcal{G}_2$. This matching is, in general, not perfect, since some nodes may be left unmatched.

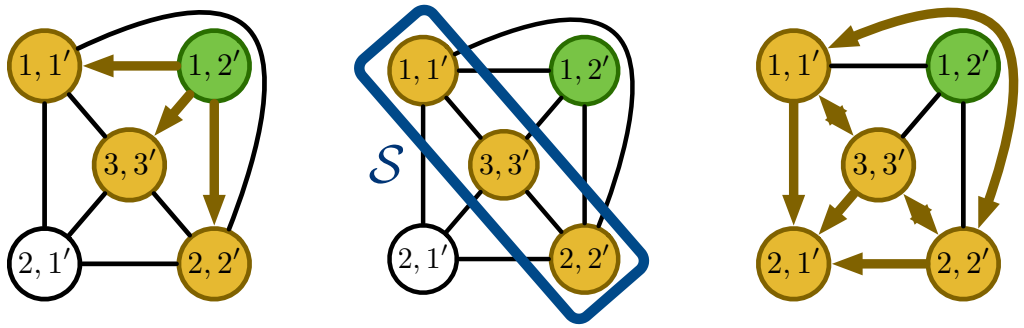
Algorithm 3 `SafeExpand`$(\mathcal{S}, r)$

Input: noisy seed set $\mathcal{S}$; token threshold r
Output: confident seed estimate $\mathcal{M}_0$

```
M_0 ← ∅
s(u, v) ← 0 for ever pair (u, v)                    ▷ Reset scores
for each pair (i, j) ∈ S do
    s(u, v) ← s(u, v) + 1, ∀(u, v) ∈ N_(i,j)        ▷ Increase score
end for
while any unmatched pair (i, j) has s(i, j) ≥ r do
    pick (i, j) randomly from highest-scoring pairs
    M_0 ← M_0 ∪ {(i, j)}                            ▷ Add pair to M_0
    s(u, v) ← s(u, v) + 1, ∀(u, v) ∈ N_(i,j)
end while
return M_0
```

3.3.4. `LooseExpand` subroutine

The last subroutine, `LooseExpand`, is similar to `SafeExpand`. While `SafeExpand` sets a high score threshold in order to be confident as it matches nodes, `LooseExpand` is more relaxed in its acceptance of matched pairs. However, it does take into account centrality measures when



(a) Pair $(1, 2')$ matched, increases scores, no pairs have score two or more.

(b) Rebuild noisy seed set $\mathcal{S}$ as $\{(1, 1'), (3, 3'), (2, 2')\}$.

(c) All pairs in $\mathcal{S}$ increment scores, if they have not already.

Figure 3.4: Example of `LooseExpand` rebuilding the set $\mathcal{S}$. We take all unmatched neighboring pairs of previously matched nodes (i.e., those pairs with score exactly one) and add them to $\mathcal{S}$. White pairs have score zero, yellow have positive score, and green are matched. Arrows describe the direction in which scores are incremented.

breaking ties, making `LooseExpand` more certain about the correctness of a pair relative to its competing pairs in terms of score. `LooseExpand` backtracks, starting a new matching $\mathcal{M}$ from scratch and repeatedly growing, taking the seed set estimate $\mathcal{S}$ (which after the backtracking step is in fact $\mathcal{M}_0$) as input; see Algorithm 4.

Similarly to `SafeExpand`, `LooseExpand` starts by having all pairs in $\mathcal{S}$ increase the scores of their neighboring pairs. Then, we find all the pairs composed of unmatched nodes with the highest score. Among those pairs, we select the pair whose constituent nodes have the lowest absolute difference in centrality measure and add it to $\mathcal{M}$. The selected pair then increments the score of its neighbors in the product graph by one, but only if the pair has not previously been used to increase scores. We use this procedure to iteratively add pairs to $\mathcal{M}$ until no pairs composed of unmatched nodes have score two or more. Then, we allow a relaxation of our percolation so our matching may spread further throughout the networks. In a rebuilding step, a new seed set $\mathcal{S}$ is created from scratch by taking all unmatched neighbors of matched pairs (i.e., all unmatched pairs with score exactly one), and the percolation process is repeated. This continues until no unmatched neighbors of matched pairs exist, and the final matching $\mathcal{M}$ is returned.

Finally, if the matching $\mathcal{M}$ has not grown above a fraction f of the smaller network's size, we perform a backtracking step. Using the final matching $\mathcal{M}$ as the noisy seed set $\mathcal{S}$, we repeat another iteration of `SafeExpand` and `LooseExpand`. This boosting procedure is critical in allowing `SPECTRE` to perform well on networks exhibiting lower correlations, since it allows a poor-quality matching to be iteratively updated until we obtain a high-quality final matching.

Algorithm 4 `LooseExpand`($\mathcal{S}$)

Input: confident seed estimate $\mathcal{S}$
Output: matching $\mathcal{M}$

$\mathcal{M}, \mathcal{U} \leftarrow \emptyset$ ▷ $\mathcal{U}$ is used pairs
$s(u,v) \leftarrow 0$ for ever pair (u,v) ▷ Reset scores
while $|\mathcal{S}| > 0$ **do**
  for each pair $(i,j) \in \mathcal{S}$ **do**
    $s(u,v) \leftarrow s(u,v) + 1,\ \forall (u,v) \in \mathcal{N}_{(i,j)}$
    $\mathcal{U} \leftarrow \mathcal{U} \cup \{(i,j)\}$ ▷ (i,j) used to increase scores
  end for
  while any unmatched pair (i,j) has $s(i,j) \geq 2$ **do**
    $\mathcal{X} \leftarrow \arg\max s(u,v) \cap (\mathcal{V}_1 \times \mathcal{V}_2 \setminus \mathcal{M})$
    $(i,j) \leftarrow \arg\min_{(u,v)\in\mathcal{X}} |C_1(u) - C_2(v)|$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{(i,j)\}$
    if $(i,j) \notin \mathcal{U}$ **then**
      $s(u,v) \leftarrow s(u,v) + 1,\ \forall (u,v) \in \mathcal{N}_{(i,j)}$
      $\mathcal{U} \leftarrow \mathcal{U} \cup \{(i,j)\}$
    end if
  end while
  $\mathcal{S} \leftarrow \{(i',j') \mid (i',j')$ neighbor of $(i,j) \in \mathcal{M}, (i',j') \notin \mathcal{U}, i'$ & j' unmatched $\}$
    ▷ Rebuild $\mathcal{S}$ from scratch
end while
return $\mathcal{M}$

3.4. Numerical Experiments

In order to verify the effectiveness of `SPECTRE`, we conducted extensive experiments on a variety of benchmark networks. Additional experimental results, including studies on parameter selection and runtime of the algorithms in this chapter, are presented in Appendix B.1.

We measure the performance of `SPECTRE` across four different metrics. The first is *Precision*, which measures the percentage of correct pairs in the final matching $\mathcal{M}$. The second is

Recall, or true positive rate, which is the fraction of possible correct pairs that are identified in $\mathcal{M}$. Algorithms may only align a subset of the nodes, thus these two metrics evaluate different things: precision provides a notion of accuracy of the matched nodes, while recall describes the proportion of the nodes which the algorithm was able to align. Since our algorithm may only hope to label nodes with degree at least 2 (due to how `LooseExpand` matches pairs), we measure the fraction of these nodes which are in $\mathcal{M}$. Formally, we can define the first two metrics as follows:

$$\begin{aligned}\text{Prec}(\mathcal{M}) &= \frac{|\{(i,j) \in \mathcal{M} : i \leftrightarrow j\}|}{|\mathcal{M}|}, \\ \text{Recall}(\mathcal{M}) &= \frac{|\{(i,j) \in \mathcal{M} : i \leftrightarrow j\}|}{|\{v \in \mathcal{V}_1 \cap \mathcal{V}_2 : d_1(v) \geq 2,\ d_2(v) \geq 2\}|},\end{aligned}$$

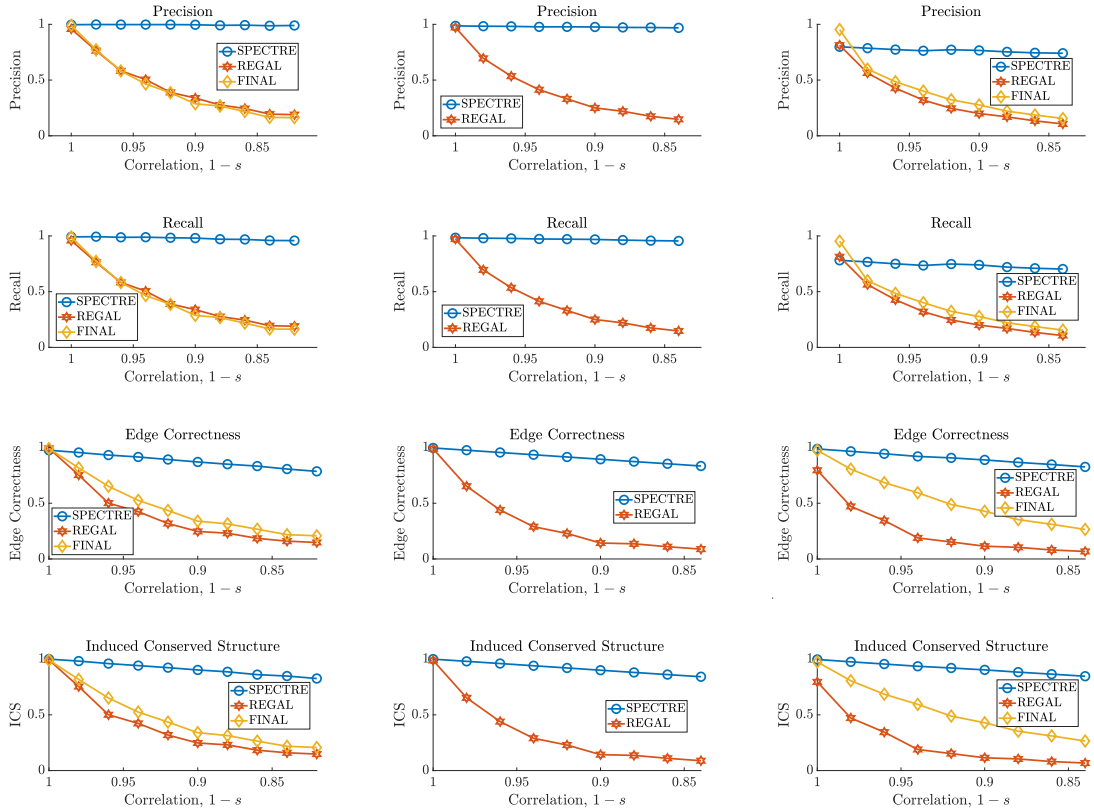
where $d_i(v)$ is the degree of node v in graph $\mathcal{G}_i$, and $i \leftrightarrow j$ means i corresponds to j in the ground-truth matching.

An issue with these metrics is that they require knowledge of the ground truth of the node correspondences. However, in most realistic scenarios, these correspondences are not available. Following the approach in [81], we measure the quality of our alignments using the *Edge Correctness* (EC) and *Induced Conserved Structure* (ICS) score. As described below, these two scores depend solely on topological information. In particular, Edge Correctness measures the fraction of matched edges from $\mathcal{E}_1$, denoted by $\mathcal{M}(\mathcal{E}_1)$, which are present in $\mathcal{E}_2$. In other words, EC measures the fraction of edges which are correctly matched by $\mathcal{M}$. However, EC does not penalize $\mathcal{M}$ for omitting edges in $\mathcal{E}_2$ which should be present. For this reason we also compute the ICS score, which measures the fraction of matched edges present in the subgraph of $\mathcal{G}_2$ induced by the nodes which are matched, i.e., those nodes in $\mathcal{M}(\mathcal{V}_1)$. The ICS score penalizes the matching both for omitting edges that are present in

$\mathcal{E}_1$ and those that are present in $\mathcal{E}_2$. Formally,

$$\mathrm{EC}(\mathcal{G}_1, \mathcal{G}_2, \mathcal{M}) = \frac{|\mathcal{M}(\mathcal{E}_1) \cap \mathcal{E}_2|}{|\mathcal{E}_1|},$$
$$\mathrm{ICS}(\mathcal{G}_1, \mathcal{G}_2, \mathcal{M}) = \frac{|\mathcal{M}(\mathcal{E}_1) \cap \mathcal{E}_2|}{|\{(i,j) \in \mathcal{E}_2 \ : \ i,j \in \mathcal{M}(\mathcal{V}_1)\}|}.$$

3.4.1. Correlated Networks



(a) *E. coli* PPI (b) GEMSEC FB-Artists[9] (c) arXiv Astrophysics

Figure 3.5: Performance of network alignment methods with varying edge correlation levels. `SPECTRE` (in blue) achieves consistently higher performance than its competitors, increasingly so as the correlation between the networks grows smaller.

In order to provide access to the ground-truth node correspondences, we run numerical experiments on correlated networks generated by randomly sampling the edges of a given arbitrary graph $\mathcal{G}$. In particular, we select each edge in $\mathcal{G}$ with a probability $1 - s$, inde-

[9]FINAL's MATLAB implementation ran out of memory when attempting to align the GEMSEC Facebook-Artists network, and was not able to produce a matching.

pendently of other samples. Performing this sampling twice and shuffling the labels of the resulting graphs we obtain two graphs, $\mathcal{G}_1$ and $\mathcal{G}_2$, to be aligned. Since the ground-truth information about node correspondences is available to us, we may measure the accuracy of the alignment `SPECTRE` produces. Moreover, by tuning the edge dropout probability s, we may test our algorithm on pairs of networks with different levels of correlation. Hence, we can measure the performance of `SPECTRE` as a function of the level of correlation of the networks to be aligned.

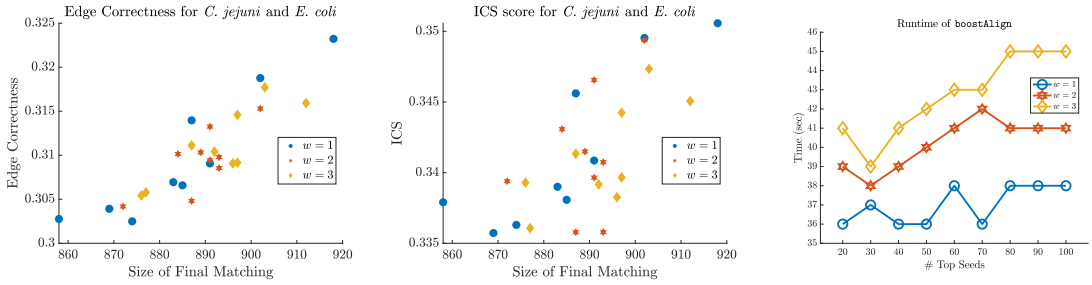
We compare our algorithm against 2 existing network alignment methods: REGAL [87] and FINAL [82]. REGAL constructs a node embedding using a low-rank representation and then matches greedily using a fast approximate algorithm. On the other hand, FINAL is an attributed fast alignment algorithm that extends IsoRank [91], which in the unattributed case utilizes a random-walk based fixed-point algorithm to create an embedding based off of an initial similarity matrix H. For REGAL we use the default parameters suggested by the authors, but for FINAL we adjust the prior-alignment matrix H to resemble our initialization strategy. We find that initializing H as a sparse matrix with a one for pairs of nodes that are within w positions in their respective centrality rankings considerably outperforms the authors' suggested *degree-similarity* initialization. The results for aligning two (generated) correlated graphs from three large-scale benchmark networks, at varying correlation levels, are presented in Figure 3.5. Notably, the PPI networks for *E. Coli* and *C. Jejuni* [101] have been used as a real application for testing alignment algorithms [73]–[75], [87]. GEMSEC Facebook-Artists [95] was used as a representative of a large real-world social network. Lastly, we tested on the benchmark network of collaborations in arXiv for Astrophysics [102].

3.4.2. Protein-Protein Interaction Networks

In the case of protein-protein interaction (PPI) networks, nodes correspond to proteins, and edges are placed between them if they participate in interactions together. We may not know the ground-truth matching, but we are looking for proteins that perform similar functions

across species. We will consider the PPI networks of the bacteria species *Campylobacter jejuni* (*C. jejuni*) and *Escherichia Coli* (*E. coli*) from the HitPredict Database [101], which have been used as a benchmark by other algorithms such as MI-GRAAL [80] and GHOST [81]. Using `SPECTRE`, we achieve an Edge Correctness of 32% and ICS score of 35%, which is a significant increase over both GHOST and MI-GRAAL.

Figure 3.6 summarizes the results of running `SPECTRE` on the PPI networks. In these experiments, we fix $f = 3/4$, $r = 4$, use eigenvector centrality to generate the initial seed estimate, and ran a sweep of our parameters with $k \in \{20, 30, \ldots, 100\}$ and $w \in \{1, 2, 3\}$. We permitted `SPECTRE` to run no more than five iterations of `SafeExpand` and `LooseExpand`. We observe a correlation between the size of the final matching and the quality of the alignment, both as measured by Edge Correctness and ICS score. The runtime of `SPECTRE` is lowest for $w = 1$ and, interestingly, the best alignment in terms of Edge Correctness and ICS score is for $w = 1$, with $k = 90$. These results illustrate that `SPECTRE` is robust even when aligning large-scale real-world networks without any prior information. Moreover, it outperforms most approaches found in the literature in terms of quality of the output matching, while being much more computationally scalable.



(a) Edge Correctness (b) ICS score (c) Runtime (sec)

Figure 3.6: Performance of `SPECTRE` on *C. jejuni* and *E. coli* PPI networks.

CHAPTER 4

Transferability of hyper-graph neural networks

M. Hayhoe, H. Riess, V. M. Preciado, and A. Ribeiro, "Stable and transferable hyper-graph neural networks," Submitted, available at arXiv:2211.06513*, 2022*

Graph Neural Networks (GNNs) are information processing and learning architectures for signals supported on graphs [26], [103]. They have been widely used in practice for problems ranging from text analysis [104] to recommendation [105] to control of multi-agent robotic systems [106], among many others [107], [108]. However, in many practical applications, relationships between entities are not inherently pairwise; examples include brain networks [109], biochemical reactions [110], social interactions [111], and more. These higher-order relationships are often modeled using every pairwise relationships between group members, but can be more faithfully represented via sets of entities where the cardinality of the set can be greater than two, and the interactions themselves can be nonlinear in nature. Hypergraphs (or higher-order graphs) and simplicial complices are tools for representing these higher-order relationship and have seen use in many applications [112]. Simplicial complices require the collection of node sets (called simplices) to be closed to taking subsets [113]; hence, they can be viewed as hypergraphs with added constraints that in turn provide added structure. While this additional simplicial constraint may be impractical, it endows rich topological structure known as *Hodge theory* (see, e.g., [112], [114]). The so-called *Hodge Laplacians* and their corresponding eigenvalue spectra can be understood as higher-order spectral analogues of the ordinary graph Laplacian, and have seen success for higher-order graph learning [115]–[118]. Building on these concepts, we introduce a hypergraph learning framework called Hyper-graph Expansion Neural Networks (HENNs) by combining graph representations of hypergraphs. While other approaches use graph representations for hypergraph signal processing [119]–[123], none combine all of the spectral interpretation and Hodge theory via simplicial complices, nonlinear interactions between nodes and hyperedges, and ability to process both node and hyperedge signals.

Informally, transferability is a type of generalization property for graph signal processing architectures which says that, given two graphs that describe the same or similar phenomenon, the graphs should process signals in a similar manner [23]. There have been three main approaches for codifying the similarity of graphs, i.e., that graphs model similar phenomena. The first compares the original graph to a mildly perturbed version [7], [9], [10], although the notions of perturbation and the measure of transferability differ. The second approach assumes a latent space model by which similar graphs are created. For example, the nodes of similar graphs may belong to the same latent measure space with signals sampled accordingly [23], or the graphs themselves may be random and drawn from the same distribution [24]. The third approach assumes similar graphs are obtained from a graphon, which can be understood as the continuous limit objects of sequences of graphs [25]. Transferability bounds using graphons have been explored in both the asymptotic sense [8] and the non-asymptotic sense [11], [26].

While these approaches describe transferability of GNNs across similar graphs, they may not be useful in practice. The core concept is that GNNs are spectral operators and, hence, should be transferable across graphs with similar spectra. This *spectral similarity* is achieved as a consequence of the assumptions under which the similar graphs are obtained, generated, or sampled. Unfortunately, given two real graphs of interest, it may not be possible to assert that they are small perturbations of one another, or are sampled from the same latent space or graphon. As such, it is of great practical interest to obtain transferability bounds across *arbitrary* graphs, without any assumptions on their origin. To this end, we provide the first GNN transferability bound directly between two arbitrary graphs of the same size, which can be computed before any training has occurred[10]. The key tool in our analysis is to explicitly measure the *spectral similarity* [27], [28] of the graphs for which the transferability bound is desired, via computing and comparing their spectra directly.

Using our approach, we provide what is to the best of our knowledge the first bound on

[10] As we will show, certain design choices for the GNN architecture will affect its transferability.

the transferability performance of neural networks that perform convolutions with signals supported on arbitrary hypergraphs. The key herein is to consider graph representations of hypergraphs (described in Section 4.1.3), which may have arbitrary structure; hence, previous results on GNN transferability which make assumptions on graph structure may not apply. In contrast, we may apply our results by simply measuring the spectral similarity of the graph expansions of the hypergraphs of interest, before any training has occurred. To apply our results when measuring spectral similarity may be difficult, we also explore transferability without direct computation of the eigenvalue spectrum in the context of small perturbations, random graphs, and graphs sampled from graphons to show that transferability bounds in each of these regimes may be recovered using our approach.

4.1. Preliminaries

4.1.1. Graph Neural Networks

A *graph* is a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ with a set of n nodes $\mathcal{V}$, m edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and a real edge weighting function $W : \mathcal{E} \to \mathbb{R}$. We assume throughout that $\mathcal{G}$ is undirected and connected. The graph $\mathcal{G}$ can be represented using a matrix $S \in \mathbb{R}^{n \times n}$ which respects its sparsity pattern, i.e., $[S]_{ij} = 0$ whenever $(i, j) \notin \mathcal{E}$, with popular examples including the adjacency matrix, graph and random walk Laplacians, and the normalized versions thereof. Since $\mathcal{G}$ is undirected with real edge weights the matrix representation S is symmetric and diagonalizable, with an orthonormal eigenvector basis $V = [\mathbf{v}_1, \ldots, \mathbf{v}_n]$ and eigenvalue matrix $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$, where the eigenvalues are real and ordered so that $\lambda_1 \leq \cdots \leq \lambda_n$. Node signals are data vectors $\mathbf{x} = [x_1, \ldots, x_n]^\intercal \in \mathbb{R}^n$ where x_i is associated to node i. We often assume the signal for an individual node is a scalar, but this can readily be generalized.

To process a signal $\mathbf{x}$ using the graph $\mathcal{G}$, we use the graph matrix S as a *graph shift operator* (GSO) via the linear map $\mathbf{y} = S^k \mathbf{x}$, where the k-fold application of S represent local exchanges of information from the signal $\mathbf{x}$ between a node and its neighbors which are at most a k-length path away in the graph $\mathcal{G}$ [26]. With this in mind, we define graph

filters as polynomials on the GSO below; the coefficients of this polynomial, called the *filter coefficients*, are the quantities that will be learned.

Definition 4.1.1 (Graph filter)**.** *A* graph convolutional filter $H(S)$ *with filter coefficients* $\{h_k\}_{k=0}^{\infty}$ *is defined as*

$$H(S) \coloneqq \sum_{k=0}^{\infty} h_k S^k. \tag{4.1}$$

Moreover, the graph frequency response *of the filter is*

$$h(\lambda) \coloneqq \sum_{k=0}^{\infty} h_k \lambda^k. \tag{4.2}$$

We frequently consider filters with an analytic frequency response, i.e., a finite number of coefficients, so that for some K, $h_k = 0 \; \forall k > K$. *Moreover, we assume* $|h(\lambda)| \leq 1$ *for all* $\lambda \in [\lambda_1(S), \lambda_n(S)]$ *so filters do not amplify signals, which is trivially satisfied via normalization with finite coefficients.*

To improve the representation power of graph filters, in practice they are stacked together with pointwise nonlinearities to create a *graph neural network* (GNN), defined below.

Definition 4.1.2 (Graph neural network)**.** *Graph neural networks are a cascade of* L *layers of graph filters, each followed by a pointwise nonlinearity. Let each layer have* f_l *graph signals (or* features*)* $\mathbf{x}_l^1, \ldots, \mathbf{x}_l^{f_l} \in \mathbb{R}^n$. *At layer* l, *we apply* $f_l f_{l-1}$ *graph filters of the form* $H_l^{ij}(S)$ *followed by a pointwise (or elementwise) nonlinear function* $\sigma : \mathbb{R} \to \mathbb{R}$ *to process the* f_{l-1} *input features into the* f_l *output features via*

$$\mathbf{x}_l^i = \sigma\left(\sum_{j=1}^{f_{l-1}} \sum_{k=0}^{\infty} h_{lk}^{ij} S^k \mathbf{x}_{l-1}^j\right), \quad i \in \{1, \ldots, f_l\}.$$

A graph neural network is the mapping $\Phi(\mathbf{x}_0; S, \Theta) = \mathbf{x}_L$, *where* Θ *represents the set of learned (or learnable) parameters. In this work we will consider normalized Lipschitz continuous nonlinearities so that* $|\sigma(x) - \sigma(y)| \leq |x - y|$, *e.g.,* ReLU, sigmoid, *and* tanh.

A property of graph filters which has been shown to be valuable for stability of GNNs [7], [26] is the so-called *integral Lipschitz* condition, defined below.

Definition 4.1.3 (Integral Lipschitz)**.** *A filter with frequency response h is* integral Lipschitz *on an interval $\mathcal{I}$ if there is some $C > 0$ such that, for all $\lambda_1, \lambda_2 \in \mathcal{I}$,*

$$|h(\lambda_1) - h(\lambda_2)| \leq C \frac{|\lambda_1 - \lambda_2|}{|\lambda_1 + \lambda_2|/2}, \tag{4.3}$$

which implies that the derivative of h satisfies $|\lambda h'(\lambda)| \leq C$. We omit the interval $\mathcal{I}$ when it is clear from context, e.g., for filters that will be applied to some GSOs $S_1, \ldots, S_m$ we have $\mathcal{I} = \cup_{i=1}^{m} [\lambda_1(S_i), \lambda_n(S_n)]$.

Integral Lipschitz graph filters can be arbitrarily discriminative for small eigenvalues, but must become effectively flat for larger eigenvalues. However, by applying nonlinearities to the output of these filters, the portion of the spectrum containing larger eigenvalues may be scattered to the lower portion. In other words, GNNs with integral Lipschitz filters can be both discriminative and stable [7]. Given some finite GSO S as well as a filter $H(S)$ with an analytic frequency response and support contained in $[\lambda_1(S), \lambda_n(S)]$, the integral Lipschitz condition is trivially satisfied with

$$C = \max \left\{ \left| \sum_{k=1}^{K} h_k k \lambda_1(S)^k \right|, \left| \sum_{k=1}^{K} h_k k \lambda_n(S)^k \right| \right\}. \tag{4.4}$$

Furthermore, we note that the value of the integral Lipschitz constant in a learning architecture may be affected by adding a penalty term to the loss function used for training.

4.1.2. Hypergraphs and simplicial complices

A *hypergraph* (or higher-order graph) is a generalization of a graph represented by the tuple $\mathcal{H} = (\mathcal{V}, \mathcal{E}, W)$ with a set of n nodes $\mathcal{V}$ and a collection of m *hyperedges* $\mathcal{E} \subseteq 2^{\mathcal{V}}$, where $2^{\mathcal{V}}$ is the powerset of $\mathcal{V}$, i.e., the collection of all subsets of $\mathcal{V}$. In contrast to a typical edge, hyperedges may join any number of nodes, i.e., elements of $\mathcal{E}$ are arbitrarily-sized subsets of nodes in $\mathcal{V}$, and not just pairs in $\mathcal{V} \times \mathcal{V}$. We ascribe some hyperedge weighting function

Symbol	Meaning
$A \in \mathbb{R}^{n \times n}$	Adjacency matrix
$B \in \mathbb{R}^{n \times m}$	Node-hyperedge incidence matrix
$D_v \in \mathbb{R}^{n \times n}$	Diagonal node degree matrix
$D_e \in \mathbb{R}^{m \times m}$	Diagonal hyperedge size matrix
$D_{ee} \in \mathbb{R}^{m \times m}$	Diagonal hyperedge intersection count matrix
$W \in \mathbb{R}^{m \times m}$	Diagonal hyperedge weight matrix

Table 4.1: Notation for a hypergraph with n nodes and m hyperedges

$W : \mathcal{E} \to \mathbb{R}$ and, with an abuse of notation, stack these hyperedge weights into a diagonal matrix W. If all hyperedges have cardinality two, we recover the ordinary definition of a graph.

Hypergraphs are commonly represented as matrices via the node-hyperedge *incidence matrix* $B \in \mathbb{R}^{n \times m}$, where

$$[B]_{ij} = \begin{cases} 1, & \text{node } i \text{ is in hyperedge } j, \\ 0, & \text{otherwise.} \end{cases} \tag{4.5}$$

The *energy* of a hypergraph node signal x on $\mathcal{H}$ is defined as

$$Q_{\mathcal{H}}(x) \coloneqq \sum_{e \in \mathcal{E}} \max_{i,j \in e} (x_i - x_j)^2. \tag{4.6}$$

The *hypergraph Laplacian* is then defined as the gradient of the energy functional, i.e.,

$$\mathcal{L}_{\mathcal{H}}(x) \coloneqq \frac{1}{2} \nabla Q_{\mathcal{H}}(x). \tag{4.7}$$

The hypergraph energy function is simply a generalization of the graph energy of a signal [28], which is defined for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_{\mathcal{G}}, W)$ as

$$Q_{\mathcal{G}}(x) = \sum_{(i,j) \in \mathcal{E}_{\mathcal{G}}} (x_i - x_j)^2 = x^\intercal B B^\intercal x = x^\intercal L_{\mathcal{G}} x, \tag{4.8}$$

where B is the node-edge incidence matrix, and $L_{\mathcal{G}}$ is the combinatorial graph Laplacian.

In contrast to many graph Laplacians, however, $\mathcal{L}_{\mathcal{H}}$ is a nonlinear operator. As such, its resultant diffusions are not easily-approximated with any linear graph Laplacian. Indeed, in contrast to a graph diffusion between nodes, such a hypergraph operator diffuses the signal x across both the nodes and hyperedges.

A *simplicial complex* is a hypergraph whose hyperedge set $\mathcal{E}$ is closed under taking subsets, i.e., any subset of nodes in a hyperedge is itself a hyperedge. By convention, we refer to hyperedges of cardinality $k+1$ as the order-k simplices, or simply as k-simplices; 0-simplices are the nodes, 1-simplices are the edges, and higher orders correspond to larger hyperedges. A *face* of a k-simplex $\mathcal{S}$ is a cardinality-k subset of $\mathcal{S}$, i.e., the subset obtained by omitting a single element of $\mathcal{S}$, and is itself a $(k-1)$-simplex. We call $\mathcal{S}$ a *coface* of such an order $k-1$ subset. If two k-simplices are both faces of the same $(k+1)$-simplex we call them *upper adjacent*, and if they share a common face we refer to them as *lower adjacent*. We now briefly define the higher-order incidence matrices B_k, which are generalizations of the classical node-to-edge incidence matrix B_1 and are related to the node-hyperedge incidence matrix as well; see [114] for full details. Given appropriate reference orientations on the simplices, these incidence matrices B_k can be understood as linear operators mapping from the space of signals on $(k-1)$-simplices to k-simplices. Signals on k-simplices, i.e., sequences of real numbers corresponding to k-simplices, are called *k-chains* and may be interpreted as flows on simplices (with a flow being negative if it opposes the reference orientation); for example, 0-chains are node signals, and 1-chains correspond to edge flows. We refer to the vector space of chains on k-simplices with real coefficients as $\mathcal{C}_k$, whose basis elements are all oriented simplices of order k. Then, the *boundary maps* $\delta_k : \mathcal{C}_k \to \mathcal{C}_{k-1}$ map such chains to the sum over the lower adjacent simplices, i.e., the boundary components. The higher-order incidence matrices B_k are the matrix representations of these boundary maps δ_k. Similarly, we may define the *co-boundary maps* $\delta_k^{\mathsf{T}} : \mathcal{C}_{k-1} \to \mathcal{C}_k$ as the operators mapping chains to the upper adjacent simplices, which are simply the adjoints of the boundary maps, and whose matrix representations are B_k^{T}. A property of boundary maps that will be relevant later is that their square is zero, i.e., $\delta_k \circ \delta_{k+1} = 0$.

Hodge Laplacians and the Hodge Decomposition

On its own, a simplicial complex may seem to be an artificial structure; few networks in practice are built on simplices. For example, in a coauthorship network, four authors collaborating on a paper does not mean each set of three have written a paper together. However, this added structure endows a rich theory known as *Hodge theory* [112], [114]. Using the higher-order incidence matrices defined above, we define a heirarchy of diffusion operators, denoted as the *combinatorial Hodge Laplacians*, by

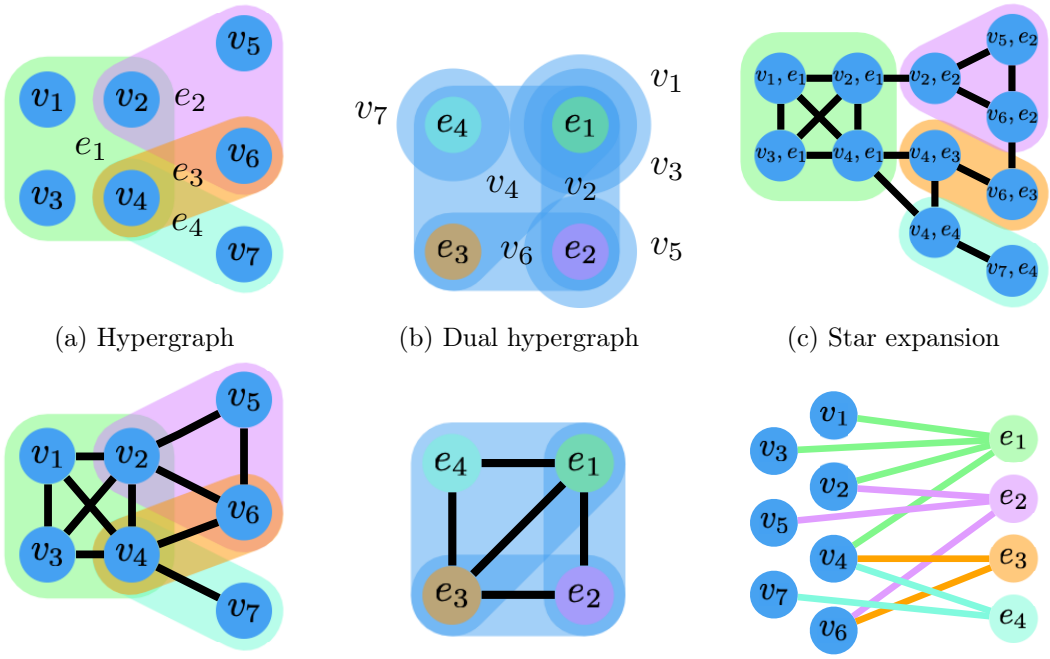
$$L_k \coloneqq B_k^\intercal B_k + B_{k+1} B_{k+1}^\intercal. \tag{4.9}$$

The first term above can be understood as *lower diffusion*, i.e., splitting the order-k signals across the lower adjacent order-$(k-1)$ faces and then summing these lower-order signals together into the upper adjacent order-k cofaces. Similarly, the second term can be understood as *upper diffusion*, by summing the order-k signals into the upper-adjacent cofaces and then splitting these into the lower-adjacent faces. As a special case, the standard combinatorial graph Laplacian is $L_0 = B_1 B_1^\intercal$ (since $B_0 = 0$), which performs only upper diffusion by summing node signals into the edges and then splitting back to the nodes.

Based on properties of the Hodge Laplacian L_k [124], we have that

$$\sigma(L_k) = \sigma(B_k^\intercal B_k) \oplus \sigma(B_{k+1} B_{k+1}^\intercal) \oplus \ker(L_k), \tag{4.10}$$

where by an abuse of notation we denote the whole eigenspace of a matrix using $\sigma(\cdot)$, and $\oplus$ denotes direct sum. This is known as the *Hodge decomposition*. Thus, we may decompose the eigenspaces (and hence eigenvectors and eigenvalues) into the components corresponding to the lower adjacent simplices, the upper adjacent simplices, and the harmonic components (elements of the kernel of L_k).



(a) Hypergraph (b) Dual hypergraph (c) Star expansion

(d) Clique expansion (e) Line graph (f) Bipartite expansion

Figure 4.1: Example hypergraph and several of its graph representations, none of which faithfully represent the original hypergraph on their own. To illustrate the connection to the original hypergraph (a) and dual hypergraph (b), edges are grouped and coloured in subfigures (c)-(f). The clique expansion in (d) cannot distinguish the real hyperedge e_2 from the clique $\{v_4, v_6, v_7\}$. The line graph in (e) cannot distinguish the dual hyperedge of v_4 from the clique $\{e_1, e_2, e_3\}$. The bipartite expansion in (f) has heterogeneous nodes and a different notion of signal processing. Finally, the star expansion in (c) adds many new nodes for each hyperedge intersection, which is impractical for dense hypergraphs and removes the interpretability of node convolutions.

4.1.3. Higher-order Graph Neural Networks

A common technique for processing hypergraph signals is to representations that are graphs [119]–[123]. While many graph representations are used in practice [112], the most common are the clique expansion, line graph, star expansion, and bipartite expansion (see Figure 4.1). The clique expansion is the graph generated by replacing each hyperedge by a clique, and the line graph is the clique expansion of the dual hypergraph (wherein the roles of nodes and hyperedges are reversed). The star expansion makes node-hyperedge pairs, and places edges between these pair-nodes if they shared a node or hyperedge in the hypergraph. The bipartite expansion partitions the nodes and hyperedges, and places edges between them

based on hyperedge inclusions in the hypergraph.

In principle, any graph representation(s) may be used in a hypergraph signal processing framework. However, in practice, the star expansion cannot represent large and/or dense hypergraphs due to its size. Moreover, the bipartite expansion loses the meaning of convolutions via applications of the graph shift operator, since the node set and, hence, the node signals are heterogeneous. In contrast, the clique expansion and line graph are homogeneous and of reasonable size; moreover, they are intimately related to the theory of simplicial complices. Indeed, since simplicial complices require closure under taking subsets, we can build a simplicial complex from a hypergraph by taking the original hyperedges and adding any missing subsets of these hyperedges as simplices. The clique expansion is then simply the 1-skeleton of this simplicial representation of the hypergraph, i.e., it includes only the 0-simplices (nodes) and 1-simplices (edges) and throws away all higher-order simplices. The line graph can also be seen as the 1-skeleton of the nerve complex constructed from the hypergraph [125], which is equivalent to the simplicial complex constructed from the dual hypergraph. Finally, we remark that the hypergraph Laplacian in (4.7) is a nonlinear operator that diffuses information across *both* the nodes and hyperedges and, hence, cannot be well-approximated via the linear graph Laplacian of one representation alone. For these reasons we introduce HENN below, which combines convolutions using the clique expansion and line graph.

Definition 4.1.4. *A Hyper-graph Expansion Neural Network (HENN) is a signal processing architecture of the form* $\Phi(\cdot; S_c, \Theta_c, S_l, \Theta_l)$*, which is composed of cascades of clique expansion layers*

$$X_v^{l+1} = \sigma\left(\sum_{k=0}^{K}\left(D_v^{-1/2}BWB^\intercal D_v^{-1/2}\right)^k X_v^l H_v^{k,l}\right), \tag{4.11}$$

and line graph layers

$$X_e^{l+1} = \sigma\left(\sum_{k=0}^{K}\left(D_{ee}^{-1/2}B^\intercal BWD_{ee}^{-1/2}\right)^k X_e^l H_e^{k,l}\right), \tag{4.12}$$

where D_{ee} is the hyperedge degree matrix from the line graph. We pool between these types of layers using max-pooling according to node-hyperedge membership, i.e., based on the entries of B, following the definition of the nonlinear hypergraph Laplacian in (4.7)*.*

There are several existing approaches for learning with hypergraphs via graph convolutions. We summarize their methodologies, benefits, and particular convolutions in our notation below as well as in Table 4.2.

- HGNN [119] performs convolutions using a hyperedge-normalized and weighted clique expansion of the hypergraph via the incidence matrix:

$$X^{l+1} = \sigma\left(D_v^{-1/2}BWD_e^{-1}B^\intercal D_v^{-1/2}X^l H\right). \tag{4.13}$$

- HGNN$^+$ [120] performs a convolution using the weighted and normalized incidence matrices, but the resulting convolution is interpreted as convolving the hyperedge signals via $WD_e^{-1}B^\intercal$ and then a node convolution via $D_v^{-1}B$:

$$X^{l+1} = \sigma\left(D_v^{-1}BWD_e^{-1}B^\intercal X^l H\right). \tag{4.14}$$

  This process may include multi-modal nodes and/or hyperedges, in which case the separately learned embeddings are joined together (stacked, pooled, etc.).

- HNHN [123] uses the incidence matrices as GSOs in a two-phase approach, similarly to HGNN$^+$ with an extra nonlinearity added between the hyperedge and node convolutions, which for node filters $H_v \in \mathbb{R}^{f\times k}$ and hyperedge filters $H_e \in \mathbb{R}^{k\times g}$ results in

the layers of the form

$$X_e = \sigma\left(D_e^{-1} B^\intercal D_v X^l H_v\right), \tag{4.15}$$

$$X^{l+1} = \sigma\left(D_v^{-1} B D_e X_e H_e\right). \tag{4.16}$$

Note the authors assume that $W = I$.

- HyperAtten [121] adapts HGNN by adding an attention module to adaptively learn the weights of the incidence matrix $\tilde{B}$, in cases where the hyperedges and nodes are from the same domain (e.g., hyperedges are formed from nearest neighbours of nodes).

- HyperGCN [122] builds a restricted clique expansion and performs convolutions. Instead of full cliques for each hyperedge, in each epoch, for each graph filtering layer and each hyperedge e it includes only the edge $(i, j) = \arg\max_{i,j \in e} \left\| x_i^l - x_j^l \right\|_2$ between the nodes with the largest pairwise signal difference, following (4.7). Edges between these two nodes and the other nodes in the hyperedge may also be included using much smaller weights. Note that the resulting GSO will depend on the node signals, of which there may be many in a dataset.

Architecture	GSOs	Spectral	HG Lapl.	SCs	Signals
HENN (ours)	$D_v^{-1/2} B W B^\intercal D_v^{-1/2}$ and $D_{ee}^{-1/2} B^\intercal B W D_{ee}^{-1/2}$	✔	✔	✔	✔
HGNN [119]	$D_v^{-1/2} B W D_e^{-1} B^\intercal D_v^{-1/2}$	✔		✔	
HGNN$^+$ [120]	$D_v^{-1} B W D_e^{-1} B^\intercal$	✔			✔
HyperAtten [121]	$D_v^{-1/2} \tilde{B} W D_e^{-1} \tilde{B}^\intercal D_v^{-1/2}$	✔			
HyperGCN [122]	$\tilde{D}_v^{-1/2} \tilde{B} W D_e^{-1} \tilde{B}^\intercal \tilde{D}_v^{-1/2}$		✔		
HNHN [123]	$D_e^{-1} B^\intercal D_v$ and $D_v^{-1} B D_e$	✔			✔

Table 4.2: Comparison of hypergraph signal processing approaches which use graph representations, including their connections with higher-order spectral theory, the nonlinear hypergraph Laplacian (4.7), simplicial complices (SCs), and whether they can process both node and hyperedge signals.

4.1.4. Spectral similarity

We quantify the similarity of connected, undirected graphs with the same number of nodes, including graph expansions of hypergraphs, by measuring the similarity of the spectra of their graph shift operators (GSOs), which we assume to be symmetric and positive semi-definite. We stress that a graph admits many different shift operators, such as the normalized graph Laplacian, and a hypergraph admits many graph representations, such as the clique expansion and line graph. While our notion of similarity will be dependent on which graph representations and/or GSOs we consider, this is an appropriate measure for similarity when signals are being processed by these particular GSOs.

Definition 4.1.5 (Spectral similarity [27])**.** *The symmetric and positive semi-definite matrices* $S \in \mathbb{R}^{n\times n}$ *and* $\tilde{S} \in \mathbb{R}^{n\times n}$ *are called* ϵ-spectrally similar *if* $(1-\epsilon)S \preceq \tilde{S} \preceq (1+\epsilon)S$*, i.e.,*

$$(1-\epsilon)x^\intercal Sx \leq x^\intercal \tilde{S}x \leq (1+\epsilon)x^\intercal Sx \quad \forall x \in \mathbb{R}^n, \tag{4.17}$$

which implies,

$$(1-\epsilon)\lambda_i(S) \leq \lambda_i(\tilde{S}) \leq (1+\epsilon)\lambda_i(S)\ \forall i \in \{1,\dots,n\}. \tag{4.18}$$

Spectral similarity can be seen as a notion of preservation of graph energy. Specifically, if the graph Laplacians L of $\mathcal{G}$ and $\tilde{L}$ of $\tilde{\mathcal{G}}$ are ϵ-spectrally similar then by (4.8) and (4.17),

$$(1-\epsilon)Q_{\mathcal{G}}(\mathbf{x}) \leq Q_{\tilde{\mathcal{G}}}(\mathbf{x}) \leq (1+\epsilon)Q_{\mathcal{G}}(\mathbf{x}) \qquad \forall \mathbf{x} \in \mathbb{R}^n. \tag{4.19}$$

Intuitively, this ϵ-spectral similarity preserves the energy of the signal $\mathbf{x}$ over $\tilde{\mathcal{G}}$ by deviating no more than $\epsilon\%$ from the value over $\mathcal{G}$, for *every* signal $\mathbf{x} \in \mathbb{R}^n$. Hence, in the context of higher-order graphs, an ϵ-spectrally similar hypergraph $\tilde{\mathcal{H}}$ of $\mathcal{H}$ should preserve the energy of *every* signal $\mathbf{x} \in \mathbb{R}^n$ over $\mathcal{H}$, using the notion of hypergraph energy in (4.6).

The notion of spectral similarity is less clear and intuitive for simplicial complices. Indeed, rather than a single GSO, the Hodge Laplacians are a heirarchy of matrices that have explicit interactions amongst one another, and the affect on the spectral similarity of these GSOs across simplicial complices is unclear. For this reason we will not explore spectral similarity of simplicial complices in this thesis, but for completeness we briefly discuss it below.

While some notions of energy on simplicial complices exist based on the simplices themselves (see e.g. [126]), we require a notion of energy in the sense of *node* signals. For example, considering only a single unweighted 2-simplex K, the definition of hypergraph energy would give us

$$\begin{aligned} Q_K(\mathbf{x}) &= \sum_{E \subseteq K} \max_{u,v \in E} (x_u - x_v)^2 \\ &= (x_1 - x_2)^2 + (x_1 - x_3)^2 + (x_2 - x_3)^2 + \max\{(x_1 - x_2)^2, (x_1 - x_3)^2, (x_2 - x_3)^2\}. \end{aligned}$$

In other words, we double-count the contribution of the pair of nodes with the largest difference, and then count the difference of all other pairs.

In this direction, Osting, Palande, and Wang [127] explore the generation of a simplicial complex at a specific order so that the spectra of the weighted up-Laplacians $L_i^{up} := W_i^{-1} B_{i+1} W_{i+1} B_{i+1}^\intercal$, i.e., the term in the Hodge Laplacian corresponding to upward diffusions, will be similar. In particular, given a simplicial complex K, the authors explore similarity at a dimension i (where $1 \leq i \leq \dim K$) by generating another complex $\tilde{K}$ such that the $(i-1)$ skeletons of K and $\tilde{K}$ are equal, all simplices of order greater than i are discarded (so $\dim \tilde{K} = i$), and

$$(1 - \epsilon) L_{i-1}^{up} \preceq \tilde{L}_{i-1}^{up} \preceq (1 + \epsilon) L_{i-1}^{up}, \tag{4.20}$$

where $\tilde{L}_{i-1}^{up}$ is the dimension-$(i-1)$ up-Laplacian of $\tilde{K}$. The authors in [127] show that (4.20) holds with probability at least $1/2$, which can readily be generalized to hold with

probability at least $1-\delta$ for some given $\delta > 0$. Unfortunately, investigating the spectrum of the down Laplacian $L_i^{down} := W_i^{-1} B_i^\intercal W_{i-1} B_i$ is much more difficult, since for the sampled simplex $\tilde{K}$ we have

$$\tilde{L}_i^{down} = (W_i S_i)^{-1} B_i^\intercal W_{i-1} B_i = S_i^{-1} L_i^{down}. \tag{4.21}$$

In particular, since some i-simplices will not be sampled, some entries of S_i^{-1} will not be defined. Even if we define S_i in a different way to account for this, computing $\mathbb{E}[S_i^{-1}]$ is non-trivial.

Computing spectral similarity

For some arbitrary graphs $\mathcal{G}$ and $\tilde{\mathcal{G}}$ both with n nodes and GSOs S and $\tilde{S}$, respectively, we may compute their coefficient of spectral similarity up to a precision κ in time polynomial in n and $\log(1/\kappa)$ [28]. For example, the following semi-definite program will yield the smallest ϵ that satisfies (4.17):

$$\begin{aligned} \underset{\epsilon}{\text{minimize}} \quad & \epsilon \\ \text{subject to} \quad & (1-\epsilon) S \preceq \tilde{S}, \\ & \tilde{S} \preceq (1+\epsilon) S. \end{aligned} \tag{4.22}$$

Typically the desired precision κ will depend on the smallest non-zero eigenvalues of S and $\tilde{S}$. Moreover, the complexity of this problem (i.e., the number of constraints) depends on the density of the graphs of interest. Note also that the multiplicity of the eigenvalue zero must be the same (possibly both zero) for S and $\tilde{S}$, which will be the case for most GSOs of connected graphs, such as the normalized Laplacian. This coefficient of spectral similarity, ϵ, will be the quantity by which we will provide bounds on the transferability of GNNs between two arbitrary graphs of the same size. This coefficient may *always* be measured between the GSOs of arbitrary graphs with the same number of nodes, but it may in general be quite large. However, for a GNN to be transferable we require only that the output is close for similar graphs. Indeed, if the output of a GNN was similar when considering graphs that

are not spectrally similar, the GNN would have poor ability to discriminate.

Finally, we mention that explicit computation of the coefficient of spectral similarity may be prohibitively expensive for very large graphs. Thankfully, if certain conditions on the graphs of interest are satisfied, we may still be able to bound and/or compute the spectral similarity. We explore the case of random graphs in Section 4.4.1, graphs sampled from graphons in Section 4.4.2, and small perturbations in Section 4.5.1, further showing the generality of our transferability bounds in the context of previous works.

4.2. Transferability via spectral similarity

To claim that an architecture is transferable, we need to show that using similar graphs to process signals produces similar results. In this paper, we will investigate transferability of graph filters, graph neural networks, and hypergraph neural networks that use graph representations. Since these tools are permutation equivariant [7], we need not be concerned with differences in node labelings between the graphs $\mathcal{G}$ and $\tilde{\mathcal{G}}$. To that end, given some signal processing architecture Φ along with GSOs S and $\tilde{S}$, we wish to examine the quantity

$$\left\|\Phi(\tilde{S}) - \Phi(S)\right\|_{\mathcal{P}} \coloneqq \min_{P\in\mathcal{P}} \max_{\mathbf{x}\in\mathbb{R}^n:\|\mathbf{x}\|_2=1} \left\|\Phi(\mathbf{x}; P^{\intercal}\tilde{S}P) - \Phi(\mathbf{x}; S)\right\|_2, \tag{4.23}$$

which is referred to as the *distance modulo permutation* [7]. Here the set of all permutation matrices is denoted $\mathcal{P} \coloneqq \{P \in \{0,1\}^{n\times n} : P\mathbf{1} = P^{\intercal}\mathbf{1} = \mathbf{1}\}$. Note that when Φ is linear, we may substitute the operator norm and ignore the unit-norm signal $\mathbf{x}$. If (4.23) is small, then the way Φ processes signals with S is similar to the way it processes signals using $\tilde{S}$ in a worst-case sense, regardless of any differences in the node labeling. In particular, if (4.23) is small whenever S and $\tilde{S}$ are ϵ-spectrally similar with small ϵ, then Φ has the transferability property. We formalize this intuition below.

Proposition 4.2.1. *For ϵ-spectrally similar symmetric GSOs S and $\tilde{S}$ and an integral Lipschitz filter with constant C, the operator difference modulo permutation between the filters*

$H(S)$ and $H(\tilde{S})$ satisfies

$$\left\| H(\tilde{S}) - H(S) \right\|_{\mathcal{P}} \leq C\epsilon + \mathcal{O}(\epsilon^2).$$

Moreover, if the filter applies only one shift operation with bias so $H(S) = h_0 I + h_1 S$, then $\left\| H(\tilde{S}) - H(S) \right\|_{\mathcal{P}} \leq C\epsilon$.

Proof. See Appendix A.2.1. □

As discussed in Section 4.1.1, GNNs are cascading layers of graph filters passed through pointwise nonlinearities. Thus we arrive at our main result, which is a bound on the transferability of graph neural networks based on spectral similarity.

Theorem 4.2.1. *Given ϵ-spectrally similar GSOs S and $\tilde{S}$ and a GNN $\Phi(\cdot; S, \Theta)$ with normalized Lipschitz nonlinearities and L layers with f features, each with filters that have unit operator norm and are C-integral Lipschitz, then*

$$\left\| \Phi(\cdot; S, \Theta) - \Phi(\cdot; \tilde{S}, \Theta) \right\|_{\mathcal{P}} \leq CLf^L\epsilon + \mathcal{O}(\epsilon^2).$$

Proof. See Appendix A.2.2. □

Theorem 4.2.1 is not surprising; informally, since GNNs are spectral operators, our result says that their actions on any signal $x \in \mathbb{R}^n$ for graphs with similar spectra will be similar. Indeed, with this result in hand, the task of GNN transferability is reduced to measuring the difference of graph spectra. While this bound depends on the design choices of the architecture (such as the number of features and layers), it is independent of the learned parameters of the GNN, Θ, excepting possibly the filters' integral Lipschitz constant C. Indeed, if the filters satisfy $|h(\lambda)| \leq 1$ and the integral Lipshitz constant is constrained[11],

[11]The former may be achieved via normalization during training, and the latter by adding a penalty term to the training loss, such as a log-barrier function on the integral Lipschitz constant, which may be computed via (4.4).

we may obtain this bound on the transferability error between *any arbitrary graphs* $\mathcal{G}$ and $\tilde{\mathcal{G}}$ before any training has taken place. In practice, the coefficient of spectral similarity can then be computed via, for example, the SDP (4.22). For large graphs this may be computationally expensive; hence, in Sections 4.4 and 4.5 we provide bounds for spectral similarity in many regimes of practical interest. Thus, transferability is entirely characterized by parameters of the architecture (the integral Lipschitz constant, number of features, and number of layers), and the spectral similarity between the graphs of interest. Naturally, worse spectral similarity results in looser transferability bounds, as does a larger integral Lipschitz constant C and more features f or convolutional layers L. However, larger C, f, and L suggest enhanced discriminability, since the GNN can produce sharper filters as C is larger, and more of those filters can be composed as f and L grow. Together, these insights pose a tradeoff between transferability and discriminability; if a GNN is indifferent to large differences in graph spectra, it cannot also treat very similar graphs differently.

We remark that in practice the filters may not be normalized, the integral Lipschitz constants may differ, and each layer may have a different number of features. We explicitly compute our transferability bound in this context in Appendix A.2.2.

4.3. Transferability of Hyper-Graph Neural Networks

In this section we will provide what is, to the best of our knowledge, the first bound on the transferability performance of neural networks that perform convolutions with signals supported on arbitrary hypergraphs. The key herein is to consider graph expansions of hypergraphs (described in Section 4.1.3), which may have arbitrary structure; hence, previous results on GNN transferability which make assumptions on graph structure may not apply. In contrast, we may use our results by simply measuring the spectral similarity of the graph expansions of the hypergraphs of interest.

Consider a hypergraph $\mathcal{H}$ and another (similar) hypergraph $\tilde{\mathcal{H}}$. We may consider $\tilde{\mathcal{H}}$ as a perturbation of $\mathcal{H}$, analogously to Section 4.5.1, or it may simply be a related hypergraph that models similar phenomena. Furthermore, consider any hypergraph signal processing

framework $\Phi(\cdot; \{S_i, \Theta_i\}_{i=1}^r)$ comprised of graph filtering layers for r graph representations of the original hypergraph. The particular GSOs of the graph representations that are used by many such hypergraph learning frameworks are listed in Table 4.2; note that most consider only one GSO. By computing the spectral similarities of these graph representations of $\mathcal{H}$ and $\tilde{\mathcal{H}}$, the result below allows us to understand how similar the output of the hypergraph learning framework $\Phi(\cdot; \{S_i, \Theta_i\}_{i=1}^r)$ will be when applied to the GSOs $\tilde{S}_r, \ldots, \tilde{S}_r$.

Theorem 4.3.1. *Consider two hypergraphs $\mathcal{H}$ and $\tilde{\mathcal{H}}$ and r graph representations with GSOs $\{S_i\}_{i=1}^r$ and $\{\tilde{S}_i\}_{i=1}^r$, respectively, such that S_i and $\tilde{S}_i$ are ϵ_i-spectrally similar. If the hypergraph learning framework $\Phi(\cdot; \{S_i, \Theta_i\}_{i=1}^r)$ has a normalizing pooling function between graph representations with normalized Lipschitz nonlinearities, with L_i layers having f_i features for graph representation i, each with filters that have unit operator norm and are C-integral Lipschitz, then*

$$\left\| \Phi(\cdot; \{S_i, \Theta_i\}_{i=1}^r) - \Phi(\cdot; \{\tilde{S}_i, \Theta_i\}_{i=1}^r) \right\| \leq \sum_{i=1}^{r} C L_i \epsilon_i \prod_{j=1}^{r} f_j^{L_j} + \mathcal{O}(\epsilon_1^2 + \cdots + \epsilon_r^2).$$

Proof. Similar to Theorem 4.2.1 with different GSOs across layers; see Appendix A.2.2. □

Since we make no assumptions on the structure of the hypergraphs and, hence, their graph representations beyond connectedness, Theorem 4.3.1 provides a transferability bound for *any* hypergraph learning framework which uses graph representations, including all those in Table 4.2. This result also lends intuition to the formation of HENN. From the connection of the clique expansion and line graph with simplicial complices, we know these graph representations are associated with the higher-order spectral theory of Hodge Laplacians [112], [114]. Together with the results of Section 4.5, this suggests that HENN will be stable to structural perturbations in the hypergraph, which is not a statement that can be asserted for the other hypergraph learning methods described earlier. Hence, for two related hypergraphs, HENN will be *both* stable and transferable if the GSOs of their clique expansions and line graphs are spectrally similar.

4.4. Transferability between random graphs

In this section we show that two graphs of the same size drawn from an appropriate random graph distribution will be spectrally similar with a coefficient ϵ that shrinks as n grows. In other words, random graphs become *more* spectrally similar as they grow larger. By combining this notion with Theorem 4.2.1, we obtain results in agreement with works that investigate the transferability of GNNs applied to large random graphs [24], [128] and convergent graph sequences [8], [11].

For practicality and to build intuition throughout this section we will assume that all graph shift operators are normalized Laplacians. As mentioned in earlier chapters, normalized Laplacians are used as graph matrices in many applications, from multi-agent robotic systems to social networks. However, we stress that this is *not* a requirement of our approach, and any GSO that satisfies the conditions we provide can be used in practice. We begin with a general result which provides the conditions under which a family of random graphs will produce GSOs that grow more spectrally similar as they grow larger, with high probability.

Proposition 4.4.1. *Let $\{\mathcal{G}_n\}_{n=1}^{\infty}$ and $\{\tilde{\mathcal{G}}_n\}_{n=1}^{\infty}$ be two independent sequences of graphs drawn from the same family of random graph distributions so that $\mathcal{G}_n$, $\tilde{\mathcal{G}}_n \sim P_{\mathcal{G}}(n)$ for each n, with graph shift operators S_n and $\tilde{S}_n$, respectively, having eigenvalues $\{\lambda_i\}_{i=1}^{n}$ and $\{\tilde{\lambda}_i\}_{i=1}^{n}$. Assume the following:*

*(*A1*) Multiplicity of zero: the zero eigenvalue has almost surely constant multiplicity independent of n (potentially zero);*

*(*A2*) Bounded spectral gap: there exists $c > 0$ independent of n such that $|\lambda_i| \geq c$ almost surely for all non-zero eigenvalues;*

*(*A3*) Concentration: given $\epsilon_c > 0$ and $\delta_c > 0$ there exist some values γ_i, $i \in \{1 \dots, n\}$, such*

that for large enough n,

$$P\left(|\lambda_i - \gamma_i| < \epsilon_c,\ \forall i \in \{1, \dots, n\}\right) > 1 - \delta_c. \tag{4.24}$$

Then for any $\epsilon > 0$ and $\delta > 0$, there exists $N = N(\epsilon, \delta)$ such that for any $n \geq N$,

$$P\Big((1-\epsilon)\lambda_i < \tilde{\lambda}_i < (1+\epsilon)\lambda_i,\ \forall i \in \{1, \dots, n\}\Big) > 1-\delta. \tag{4.25}$$

In other words, for appropriate graph shift operators of large random graphs whose eigenvalues concentrate, the coefficient of spectral similarity converges in probability to 0 as $n \to \infty$.

Proof. See Appendix A.2.6. □

This proposition applies to a large class of random graphs and corresponding shift operators. Assumption (*A1*) is necessary for spectral similarity to hold but is trivially satisfied by the normalized Laplacian of a connected graph. As we will show in Section 4.4.1, (*A2*) and (*A3*) are satisfied by the normalized Laplacian of many families of random graphs including the Erdös-Rényi and Chung-Lu models, as well as random power-law graphs with given expected degree sequences and large enough minimum expected degree [129]. Furthermore, we will show in Section 4.4.2 that graphs sampled from the same well-structured graphon will satisfy all of these conditions and, hence, become arbitrarily spectrally similar as their size grows larger. The implications of these results in terms of transferability of GNNs for random graph families is summarized below.

Theorem 4.4.1. *Let $\{\mathcal{G}_n\}_{n=1}^{\infty}$ and $\{\tilde{\mathcal{G}}_n\}_{n=1}^{\infty}$ be two independent sequences of random graphs with graph shift operators $\{S_n\}_{n=1}^{\infty}$ and $\{\tilde{S}_n\}_{n=1}^{\infty}$, respectively, such that $\mathcal{G}_n, \tilde{\mathcal{G}}_n \sim P_{\mathcal{G}}(n)$, and let $P_{\mathcal{G}}(n)$ satisfy (*A1*)-(*A3*) almost surely for all n. Then, for a sequence of GNNs $\{\Phi(\cdot; S_n, \Theta)\}_{n=1}^{\infty}$ trained on the GSOs S_n with normalized Lipschitz nonlinearities, each with the same number of layers and features, as well as integral Lipschitz filters that have unit*

operator norm,

$$\left\| \Phi(\cdot; S_n, \Theta) - \Phi(\cdot; \tilde{S}_n, \Theta) \right\|_{\mathcal{P}} \to 0,$$

in probability as $n \to \infty$.

Proof. Direct result of Theorem 4.2.1 and Proposition 4.4.1. □

The statement above has far-reaching implications. For example, GNNs which use the normalized Laplacian as a GSO will exhibit *better* transferability properties as the size of the original training graph grows, with high probability. This agrees with previous results on GNN transferability in the regime of large random graphs [24], [128], as well as the regime of convergent random graph sequences [8]. In particular, these results suggests that for a family of large enough random graphs, when using the normalized Laplacian as a GSO, it suffices to train on one graph realization to achieve comparable performance across all realizations of the same size. However, Theorem 4.4.1 also presents an issue of discriminability; with high probability, the action of a GNN trained on a large random graph will look very similar to the action of that same GNN applied to a different random graph of the same family. While this may appear to be an artifact of the choice to consider normalized GSOs, we remark that these are used almost exclusively in practice for large graphs. Indeed, without normalization the graph frequency response of a filter can become unbounded as n grows.

4.4.1. Transferability of large random graphs

To make the results in Proposition 4.4.1 and Theorem 4.4.1 concrete, in this section we will explore specific families of random graphs that satisfy (*A1*)-(*A3*). To this end, and in order to motivate the results pertaining to concentration of eigenvalues, we will introduce distributions on the spectra of random graphs.

Definition 4.4.1 (Empirical spectral distribution)**.** *The* empirical spectral distribution *(ESD) of a real symmetric matrix* X*,* μ_X*, is the atomic distribution that assigns equal*

mass to each of the eigenvalues of X, i.e.,

$$\mu_X \coloneqq \frac{1}{n}\sum_{i=1}^{n}\delta_{\lambda_i(X)}. \tag{4.26}$$

Since we wish to study the spectra of random graphs, in our setting the ESD will be a *random measure*, i.e., a random variable in the space of probability measures on $\mathbb{R}$ (see [93] for more details). We will also study a common limiting distribution of the ESD of random matrices, called Wigner's semicircle law.

Definition 4.4.2 (Wigner's semicircle law)**.** Wigner's semicircle law μ *is the distribution with density defined on* $[-1,1]$ *as*

$$\mu(x) \coloneqq \frac{2}{\pi}\sqrt{1-x^2}, \tag{4.27}$$

and zero otherwise.

Wigner's semicircle law is to random matrices what the central limit theorem is to random variables. A result of note, called Wigner's eigenvalue rigidity theorem [130, Theorem 2.2], states that the eigenvalues of a matrix whose ESD converges to the semicricle law concentrate around specific distinct values, i.e., $(A3)$ is satisfied. The exact characterization of the general class of matrices whose empirical spectral distributions converge to Wigner's semicircle law is beyond the scope of this paper (see [130] for a full treatment). However, many graphs of interest have corresponding shift operators that satisfy these conditions, including the Erdös-Rényi and Chung-Lu models, as well as random power law graphs with large enough minimum degree. Indeed, in [129, Theorem 6] it was shown that the eigenvalues of the normalized Laplacian of any random graph with an appropriate given expected degree distribution converge to Wigner's semicircle law. Moreover, [129, Theorem 5] provides bounds which may be used to show $(A2)$ is satisfied when the minimum degree is large enough. Furthermore, in [131] it was shown that inhomogeneous Erdös-Rényi random graphs (where the connection probabilities may all be different) admit limiting distributions

that are related to Wigner's semicircle law, although they may be difficult to compute in general. The implications of these results in the context of transferability of GNNs are summarized below.

Corollary 4.4.1. *Let $\{\mathcal{G}_n\}_{n=1}^{\infty}$ and $\{\tilde{\mathcal{G}}_n\}_{n=1}^{\infty}$ be two independent sequences of almost surely connected random graphs drawn from the same distribution with given expected degrees, with normalized Laplacians $\{S_n\}_{n=1}^{\infty}$ and $\{\tilde{S}_n\}_{n=1}^{\infty}$, respectively. If the minimum and average degrees are large enough, then for a sequence of GNNs $\{\Phi(\cdot; S_n, \Theta)\}_{n=1}^{\infty}$ following the conditions of Theorem 4.4.1,*

$$\left\| \Phi(\cdot; S_n, \Theta) - \Phi(\cdot; \tilde{S}_n, \Theta) \right\|_{\mathcal{P}} \to 0,$$

in probability as $n \to \infty$.

Proof. Result of [129, Theorems 5 and 6], [130, Theorem 2.2] and Theorem 4.4.1. □

It may be possible to achieve almost sure convergence above, depending on the family of random graphs under consideration [93], [130]. While this result is asymptotic in nature, it is possible to obtain convergence rates which may be informative for finite n. Concretely, in the context of Proposition 4.4.1, the eigenvalue rigidity result in [130, Theorem 2.2] suggests that for some fixed n, $\epsilon \approx O(n^{-1})$ and $\delta \approx O(n^{-c})$ for some small c. In other words, GNNs trained on GSOs of size n satisfying the conditions of Corollary 4.4.1 will have transferability bounds of the order $1/n$ with high probability.

We remark that these results also apply to the transferability of large random hypergraphs. Consider a random hypergraph with n nodes and m hyperedges where the expected degrees of the hyperedges, i.e., the number of other hyperedges that share at least one node, are given by some fixed $w_1, \ldots, w_m$ and the expected cardinality of the hyperedges are $k_1, \ldots, k_m$. In this case both the clique expansion and line graph will be random with given expected degrees and, hence, we may apply Corollary 4.4.1 to assert with high probability that the

HENN trained using one such random hypergraph will be transferable to others of the same size with a bound of the order $1/n$.

4.4.2. Graphon Transferability

A *graphon* is a symmetric, measurable function (sometimes called a *kernel*) $W : [0,1]^2 \to [0,1]$ which can be understood as the limit of a sequence of dense undirected graphs where the node index is a continuous set [25]. While many authors explore transferability for sequences of graphs that converge to the same graphon [8], [11], we will focus on the particular related case of random graphs sampled from graphons. This allows us to make use of existing results which explore the spectrum of the normalized Laplacian of such sampled graphs [132]; however, it may be possible to obtain more general results to measure the spectral similarity of sequences of GSOs obtained from graphs converging to the same graphon.

We assume that the graphon is bounded away from zero, so that $\inf_{x,y\in[0,1]} W(x,y) > 0$. In order to generate a random graph with n nodes via the graphon W, following [132] we sample some points $x_1, \ldots, x_n \sim \text{Uniform}([0,1])$ and create the edge (i,j) with a probability $W(x_{(i)}, x_{(j)})$, where $x_{(i)}$ denotes the i-th order statistic of the sampled points. We say the resulting graph $\mathcal{G}_n$ is sampled from the graphon W. There is an explicit connection to the results of Section 4.4, as it has been shown that graphs sampled from graphons in this manner have fixed expected degree distributions [132]. Indeed, [132, Lemma 3] shows that the eigenvalues of these sampled graphs concentrate, and [132, Proposition 3] bounds the spectral gap. Moreover, the graphon being bounded away from zero ensures the sampled graphs will be almost surely connected if they are large enough. Hence, all conditions of Proposition 4.4.1 are satisfied. With this in mind, we present our result on the transferability of sequences of random graphs sampled from the same graphon.

Corollary 4.4.2. *Let $\{\mathcal{G}_n\}_{n=1}^{\infty}$ and $\{\tilde{\mathcal{G}}_n\}_{n=1}^{\infty}$ be two independent sequences of almost surely connected graphs sampled from the same graphon W, which is bounded away from zero, with normalized Laplacians $\{S_n\}_{n=1}^{\infty}$ and $\{\tilde{S}_n\}_{n=1}^{\infty}$, respectively. Then, for a sequence of GNNs*

$\{\Phi(\cdot; S_n, \Theta)\}_{n=1}^{\infty}$ *following the conditions of Theorem 4.4.1,*

$$\left\| \Phi(\cdot; S_n, \Theta) - \Phi(\cdot; \tilde{S}_n, \Theta) \right\|_{\mathcal{P}} \to 0,$$

in probability as $n \to \infty$.

Proof. Direct result of [132, Lemma 3 and Proposition 3] and Theorem 4.4.1. □

4.5. Stability of Graph Neural Networks via spectral similarity

Stability is a generalization property of graph learning architectures that is closely related to transferability. In particular, an architecture with the stability property is one for which small changes in the underlying topology or structure of the graph have a small effect on the way signals are processed. Hence, stability can be understood as a special case of transferability, where one graph is a perturbed version of another graph. However, stability is no less important, as it characterizes the robustness of a graph learning architecture to small changes in the underlying graph, which could arise for example via measurement noise. In the following sections, we will explore stability of graph neural networks by computing the spectral similarity of a graph and its perturbed version. By applying the results from Section 4.2 together with the similarity bounds we produce herein, we will show that graph neural networks can be stable to perturbations of the graph structure. In contrast to prior results [7], the results herein along with Section 4.4.1 show that stability does not decay as the size of the graph grows.

4.5.1. Spectral similarity of perturbed matrices

To investigate stability, we are interested in small changes to some graph $\mathcal{G}$ that results in a perturbed version which we call $\tilde{\mathcal{G}}$. In particular, since we are processing graph signals, we will focus on the graph shift operator S of the original graph $\mathcal{G}$ and the GSO $\tilde{S}$ of the perturbed graph $\tilde{\mathcal{G}}$. We will explore two types of perturbations, both relative and additive, in order to model how a graph may be changed slightly.

Fundamentally, graph neural networks are spectral operators; the graph frequency response of a filter (4.2) makes this relationship explicit. Well-known results such as Weyl's inequality or the Weilandt-Hoffman inequalities [93] show that that the spectrum of a real symmetric matrix is stable to small perturbations. Thus, it stands to reason that if small perturbations of a graph lead to small perturbations in the eigenvalues, then these small perturbations should not change the graph frequency response of a filter very much. To this end, we will explore the spectral similarity of a graph and its perturbed version. In doing so, we can bound the change in the eigenvalues of a graph after a perturbation has been applied.

We will study two types of perturbations herein, called *additive* and *relative* perturbations. The first and simplest type involves adding a small perturbation to the GSO S of the original graph $\mathcal{G}$, so that the GSO $\tilde{S}$ of the perturbed graph $\tilde{\mathcal{G}}$ becomes

$$\tilde{S} = S + E. \tag{4.28}$$

The perturbation is small in the sense of the operator norm, so that $\|E\|_{op} \leq \delta$. Such perturbations can be understood as adding or removing edge weight regardless of the original magnitude of the edges, and could be as extreme as adding or removing edges entirely. The other type of perturbation instead affects the edge weights in a manner that is relative to their magnitude, so that

$$\tilde{S} = S + \frac{1}{2}(SE + ES), \tag{4.29}$$

again with $\|E\|_{op} \leq \delta$ for some small $\delta > 0$. In both cases we assume that the perturbed GSO $\tilde{S}$ will be positive semi-definite. We explore spectral similarity between the original GSOs and their perturbed versions in the following results, beginning with relative perturbations.

Proposition 4.5.1. *Given a symmetric, positive semi-definite GSO S and its symmetric, positive semi-definite relatively perturbed version $\tilde{S} = S + \frac{1}{2}(SE + ES)$, where E is diagonalizable with $\|E\|_{op} \leq \delta$, the matrices are δ-spectrally similar.*

Proof. See Appendix A.2.3. □

Consider a relative perturbation via dilation where $E = \delta I$ and so $\|E\| = \delta$, and $\tilde{S} = (1+\delta)S$. We thus observe that δ-similarity is tight, i.e., for a *general* E with $\|E\| \leq \delta$, we cannot hope for a better bound on the similarity coefficient. Since our results require spectral similarity to hold regardless of the structure of the perturbation E, the bound in Proposition 4.5.1 is thus tight. However, for some arbitrary relative perturbation, it may be the case that a spectral similarity coefficient $\epsilon < \delta$ suffices. Thus, this result and those that follow should be viewed as tight *worst-case* bounds on the coefficient of spectral similarity for general relative perturbations. Next, let us explore additive perturbations.

Proposition 4.5.2. *Given a symmetric, positive semi-definite GSO S and its symmetric, positive semi-definite additively perturbed version $\tilde{S} = S + E$, where $\|E\|_{op} \leq \delta$, if ker$(E) \subseteq$ ker(S) then the matrices are $(\delta/\bar{\lambda}(S))$-spectrally similar, where $\bar{\lambda}(S)$ is the smallest non-zero eigenvalue of S.*

Proof. See Appendix A.2.4. □

The condition $\ker(E) \subseteq \ker(S)$ mirrors *(A1)* from Proposition 4.4.1. If S is the normalized Laplacian, this condition enforces that the perturbation not cause the graph to become disconnected, which is reasonable in practice. Indeed, from a graph signal processing perspective, disconnected components have no effect on each other. Further, similarly to *(A2)*, if S is the normalized Laplacian then Cheeger's inequality states $h_{\mathcal{G}}^2/2 \leq \bar{\lambda}(S) \leq 2h_{\mathcal{G}}$, where $h_{\mathcal{G}}$ is the Cheeger constant (or conductance) of the graph $\mathcal{G}$ [2]. Hence, our result in Proposition 4.5.2 tells us that graphs with higher Cheeger constant, i.e., better-connected graphs, will be more similar to their additively perturbed versions. In other words, graphs which are well-connected will result in trained GNNs that exhibit better stability properties than more sparsely connected graphs.

In practice, perturbations to graphs can be modeled as both relative and additive perturba-

tions. Thus, below we consider spectral similarity when both perturbations are present.

Proposition 4.5.3. *Given a symmetric, positive semi-definite GSO S, an additive perturbation matrix D such that $\|D\|_{op} \leq \delta_A$ and $ker(D) \subseteq ker(S)$, a diagonalizable relative perturbation matrix E such that $\|E\|_{op} \leq \delta_R$, and the symmetric, positive semi-definite perturbed GSO $\tilde{S} = S + \frac{1}{2}(SE + ES) + D$, the matrices are $(\delta_R + \delta_A/\bar{\lambda}(S))$-spectrally similar.*

Proof. See Appendix A.2.5. □

While our bounds on spectral similarity are tight for relative perturbations, they are maximal in the sense that we require them to hold for all eigenvalues, i.e., they are equivalent to $(1-\epsilon)\lambda_i(S) \leq \lambda_i(\tilde{S}) \leq (1+\epsilon)\lambda_i(S)$ for all $i \in \{1, \ldots, n\}$. However, perturbations in practice will not affect the eigenvalues in such a uniform manner, and thus the following stability bounds may not be tight for arbitrary perturbations. Moreover, for additive perturbations, in Proposition 4.5.2 the quantity $1/\bar{\lambda}(S)$ arises due to the necessity of satisfying spectral similarity for all signals $\mathbf{x} \in \mathbb{R}^n$, including the eigenvector $\bar{\mathbf{v}}$ associated with the smallest non-zero eigenvalue of S. If the signals to be used in practice are not well-aligned with $\bar{\mathbf{v}}$, we may be able to tighten this bound.

4.5.2. Stability via spectral similarity

To achieve our main result on stability of graph neural networks, we combine the above results on perturbations with the transferability bounds from Section 4.2.

Theorem 4.5.1. *Consider a symmetric, positive semi-definite GSO S and its perturbed version $\tilde{S} = S + \frac{1}{2}(SE + ES) + D$, where D is an additive perturbation matrix such that $\|D\|_{op} \leq \delta_A$ and $ker(D) \subseteq ker(S)$, and E is a diagonalizable relative perturbation matrix such that $\|E\|_{op} \leq \delta_R$. The GNN $\Phi(\cdot; S, \Theta)$ with normalized Lipschitz nonlinearities and L layers with f features, each with filters that have unit operator norm and are C-integral*

Lipschitz, satisfies

$$\left\|\Phi(\cdot; S, \Theta) - \Phi(\cdot; \tilde{S}, \Theta)\right\| \leq CLf^L(\delta_R + \delta_A/\bar{\lambda}(S)) + \mathcal{O}((\delta_R + \delta_A)^2)$$

Proof. Follows from Theorem 4.2.1 and Proposition 4.5.3. □

In contrast to previous results on the stability of graph neural networks [7], this bound has no dependence on the size of the graph n. This can be explained by the effect of the eigenvector differences which appear in the result in [7]. Via spectral similarity, we instead upper-bound the difference between the matrices (in the sense of the positive semi-definite cone), i.e., we consider $(1+\epsilon)S$ instead of $\tilde{S}$. In this way we can bound the transferability error by considering the eigenvalues, which are known to be stable to perturbations, without considering any changes whatsoever to the eigenvectors, which are known to be unstable [93]. Indeed, the results of Section 4.4 suggest that stability in fact *improves* as the number of nodes n grows larger. Thus, stability is entirely characterized by parameters of the architecture (the integral Lipschitz constant and number of layers), the structure of the original graph (via the smallest nonzero eigenvalue of S), and the magnitude of the perturbations that occur. Naturally, larger magnitude perturbations result in looser stability bounds, as does a larger integral Lipschitz constant C and more layers L or features f. However, larger C, L, and f suggest enhanced discriminability, since the GNN can produce sharper filters as C is larger, and more of those filters can be composed as L and f grow. A larger value of $\bar{\lambda}(S)$ when S is the normalized Laplacian suggests better connectedness and, hence, more local smoothing, which explains how it tightens the stability bound. Together, these insights pose a tradeoff between stability and discriminability; if a GNN is indifferent to large perturbations in a graph's structure, it cannot also tell very similar graphs apart.

4.6. Simulations

To evaluate the performance of HENN, we train several graph and hypergraph neural networks to solve a hyperedge source localization problem, wherein the goal is to determine the

Model	Validation accuracy	Test accuracy
Clique Expansion	0.631 ± 0.027	0.552 ± 0.016
Line Graph	0.625 ± 0.057	0.588 ± 0.016
HGNN [119]	0.638 ± 0.024	0.580 ± 0.012
HENN (ours)	$\mathbf{0.889 \pm 0.155}$	$\mathbf{0.852 \pm 0.123}$

Table 4.3: Cross-validated comparison & ablation test.

source hyperedge of a diffusion process that has been occurring over the nodes of a hypergraph for an unknown period of time. Hence, the input is a node signal, but the desired output is a hyperedge signal. This nonlinear diffusion of both node and edge signals occurs via the hypergraph Laplacian (4.7). To make the problem more difficult, we add both input and measurement noise to the signals. We perform an ablation test for HENN by comparing with GNNs that use only the clique expansion or line graph of the original hypergraph, as well as a comparison with HGNN [119], showing the improvement in performance of HENN relative to approaches which diffuse the signal across *only* the nodes or hyperedges. Since none of these other approaches can natively handle both node and hyperedge signals, where appropriate we pool all signals according to hyperedge inclusions, in the same manner as HENN. The full experimental setup is detailed in Appendix B.2. In particular, we note that all methods (including HGNN) were constrained to have normalized and integral Lipschitz filter weights to match the setting of Theorem 4.2.1.

The results of this study are presented in Table 4.3. Performance metrics were obtained after randomly shuffling the data five times, and represent the mean and standard deviation of the model with hyperparameters exhibiting the highest upper-confidence bound cross-validation score (mean plus standard deviation). HENN shows an improvement in test accuracy of $45 - 54\%$ over the GNNs which use only the clique expansion or line graph, illustrating the value of combining these representations for problems that involve a nonlinear diffusion process across both the nodes and hyperedges of a hypergraph. Moreover, it shows a 47% improvement over HGNN when both methods have normalized and integral Lipschitz filter weights, suggesting superior performance when transferability is desired.

CHAPTER 5

Data-driven modeling and control of epidemics

M. Hayhoe, F. Barreras, and V. M. Preciado, "Multitask learning and nonlinear optimal control of the covid-19 outbreak: A geometric programming approach," Annual Reviews in Control*, 2021.* DOI*: https://doi.org/10.1016/j.arcontrol.2021.04.014*

Ever since the first COVID-19 case was reported on December 31st 2019 [133], the SARS-CoV-2 pandemic has spread world-wide in an unprecedented manner for our modern age [134]. The response to the first wave of COVID-19 by governments was the implementation of large scale non-pharmaceutical interventions (NPIs) ranging from contact tracing, quarantines and mask usage, to more aggressive measures like city wide shelter-in-place orders, air-travel restrictions and closures of non-essential businesses [135]. In the absence of pharmaceutical treatment, prevention, or herd immunity, NPIs remain the only tool to curb the spread of an epidemic in its earlier stages. In the case of COVID-19, governments across the world have implemented strategies to relax mobility restriction measures and reactivate the economy [136] while, at the same time, preventing the collapse of healthcare systems. However, relaxing mobility restrictions too fast or carelessly can result in resounding waves, as was repeatedly observed for COVID-19. In fact, as long as enough people in the population are susceptible, the danger of recurrent waves is not only real, but probable. In this situation, it is of utmost societal importance to develop reopening strategies in a principled manner utilizing the wealth of data readily available.

Several epidemic models have been proposed in the recent literature to simulate the effects of social distancing on the evolution of the pandemic; see, e.g., [137]–[139]. Although the majority of epidemic models in recent years are variations of the seminal mathematical models on theoretical epidemiology (see [29] and references therein), the availability of rich datasets describing human mobility and behavior is rapidly changing the field of mathematical modelling of epidemics. Companies like Google, Foursquare, Safegraph, Baidu, and

others have provided public access to massive datasets describing human mobility, enabling the development of data-driven epidemic models capturing the effects of mobility restrictions. Indeed, the choices faced by decision makers regarding disease management involve the use of multiple control actuations such as vaccination, quarantine, treatment or, as is the case for COVID-19, non-pharmaceutical interventions such as social distancing. These decisions must face the trade-off of minimizing the impact of the disease and the economic cost associated with the implementation of non-pharmaceutical interventions.

In order to increase predictive power and utility for policy decision-makers, epidemic models have gradually increased their complexity to account for a multitude of features of real epidemics such as disease-specific compartmental models [140], resurgence [141], multi-scale effects [142], seasonality [143], differential risk structure in the population [144], healthcare system capacity [145], and uncertainty in testing data [146], among others. This increased sophistication in the modeling often comes at the cost of mathematical intractability, a limitation often circumvented by formulating policies based on heuristics and/or simulations [137], [139], [143], [144], [147], [148]. Although informative for certain scenarios, these proposed interventions are not the result of rigorously formulated optimal control problems and, thus, lack the guarantees and flexibility of mathematical optimization frameworks.

Conversely, the control of epidemics does not usually admit straightforward solutions from optimal control theory due to the presence of nonlinearities and/or the lack of convexity [29]. There are important theoretical results in optimal control of epidemics which achieve mathematical tractability, for example, optimal resource allocation aiming to asymptotically drive the epidemic to extinction [5], [149]–[154]. Other theoretical results are concerned with applications of Pontryagin's maximum principle (PMP), and find exact solutions to resource allocation problems under some variations of the SIS and SEIR, for example in [155]–[157]. In contrast, data-driven control frameworks in real epidemics have found limited applicability due to the difficulty in incorporating real data and the challenges in solving non-convex programs exactly. Recent work [149] has proposed a solution to the problem of optimal social

distancing with economic constraints using a static convex program which seeks to ensure the decrease of the epidemic at all times; a condition that is too stringent and fails to consider more efficient dynamic strategies. Piguillem and Shi [145] derive a quarantine and testing policy in an optimal control framework, but the control actions and objective functions are restrictive and chosen ad-hoc for tractability. Other recent applications have tackled the non-convex optimal control of social distancing policies using model predictive control (MPC), an approach which has been applied to a plethora of non-linear control problems in industry [158]. For example, Köhler, Schwenkel, Koch, Berberich, Pauli, and Allgöwer [159] propose an optimal predictive control problem where a control input representing social distancing affects the infectivity rates directly and the number of fatalities is approximately minimized using a nonlinear program solver. Morato, Bastos, Cajueiro, and Normey-Rico [146] present a related application of MPC for optimal social distancing policies in which, instead of controlling the infectivity rates directly, the control input represents a binary lockdown policy enacted by the government which has a delayed effect in the population's level of isolation, in turn affecting the infectivity rate.

A practical concern is whether it may be possible to design optimal control strategies based on mobility restrictions that are fully data-driven, in the sense that human mobility is measured and explicitly incorporated in the epidemic model. In this chapter, we propose a model of the spread of COVID-19 and a data-driven optimal control problem that directly minimizes the number of predicted cumulative deaths by implementing mobility restrictions in the population. We use real mobility data from Google [160] to learn a nonlinear mapping representing the impact of human mobility on the parameters of a dynamical epidemic model and propose a novel nonlinear optimal control problem that can be solved efficiently using tools from geometric programming [30].

Our model consists of an extension of the classic SEIR model, augmented with compartments for asymptomatic and hospitalized agents. We assume that the rate at which agents become infected in any given day is a function of the mobility trends in the population for that

same day, reflecting the fact that an increase in mobility leads to more infections. We rely on data regarding case counts and deaths from The New York Times [161] and from Google's COVID-19 Community Mobility Reports [160] to capture the changes in visitation patterns to different Places of Interest (POIs). The mobility data consists of several time series measuring visits to various categories of places such as Retail & Recreation, Grocery & Pharmacy, and Workplaces, although in general we may produce our own categorizations from data (see Appendix B.3.4). The dataset is organized into separate time series for all counties in the United States, and measures visits to multiple categories of places against a benchmark established in January and February of 2020. The key observation is that a decision maker can enforce restrictions on visits to each of these categories to reduce the spread of the epidemic while incurring a cost to the economy. Hence, our objective is to design optimal strategies for mobility restrictions to contain the spread of COVID-19 while taking in to account the associated economic cost.

Our approach is similar to that in [146], [159] in that we solve the problem of optimal social distancing policies to curb an epidemic under state constraints, which can be implemented with a receding horizon. However, we stress two important differences: first, we explicitly model and learn the impact of human mobility on the evolution of the disease spread using real mobility data and can formulate granular continuous mobility restriction policies that vary across economic sectors; second, unlike [146], [159] our optimization problem is reduced to a convex program which can be solved with great efficiency, has global optimality guarantees, and can accommodate a large number of variables and longer optimization horizons.

The structure of the chapter is as follows. In Section 5.1 we introduce the notation used as well as some necessary background in geometric programming. In Section 5.2 we discuss the specifics of our data-driven model, consisting of a mobility layer and an epidemic layer. In Section 5.3 we discuss the details of our learning strategy to identify the parameters of the model. In Section 5.4 we present our optimal control framework and present simulations

showing the effectiveness of our method. Appendix A.3 contains proofs of the results herein, and Appendix B.3 includes additional results of practical interest.

5.1. Background and Notation

Throughout this chapter bold characters are used to denote vectors and upper-case characters denote either matrices or compartments of the epidemic model. For the following definitions let $x_1, \ldots, x_n \geq 0$ denote n non-negative variables, and let $\mathbf{x} = (x_1, \ldots, x_n)$. When considering our epidemic model we use tildes for the states of the true (nonlinear) model, e.g. $\tilde{S}(t)$, and omit the tildes for the linearized model, e.g., $S(t)$.

Definition 5.1.1 (Monomial)**.** *A function $f(\mathbf{x})$ is called a monomial if it has the form*

$$f(\mathbf{x}) = c x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n},$$

for $a_1, \ldots, a_n \in \mathbb{R}$ and $c > 0$.

Definition 5.1.2 (Posynomial)**.** *A sum of one or more monomials is called a posynomial, that is, a function of the form*

$$f(\mathbf{x}) = \sum_{i=1}^{k} c_i x_1^{a_{i,1}} x_2^{a_{i,2}} \ldots x_n^{a_{i,n}}.$$

Since posynomials admit negative exponents but do not admit negative coefficients they are not necessarily polynomials, and vice versa. We remark that posynomials are closed under addition, multiplication, and positive scalar multiplication. This implies that if the entries of two matrices $A \in \mathbb{R}^{m \times k}$ and $B \in \mathbb{R}^{k \times n}$ are posynomials of the same variables, then so are the entries of their product AB, since $[AB]_{i,j} = \sum_{t=1}^{k} A_{i,t} B_{t,j}$, which is a sum of products of posynomials. This result extends trivially to the product of an arbitrary number of matrices with posynomial entries.

Definition 5.1.3 (Convex in log-scale)**.** *A function $f(\mathbf{x})$ is convex in log-scale if the function $F(y) := \log f(\exp(\mathbf{y}))$ is convex (where the exponentiation is component wise).*

A careful application of Hölder's inequality shows that posynomials are convex in log-scale.

We solve the epidemic control problems presented herein using a quasi-convex optimization framework called *geometric programming* [30], [162], which has found wide applicability in fields such as communication systems [163], epidemiology [153], and control [164], among others. A *geometric program* (GP) is a mathematical optimization program of the form

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & q_i(\mathbf{x}) \leq 1 \qquad \text{for } i \in 1, \dots m \\ & h_i(\mathbf{x}) = 1 \qquad \text{for } i \in 1, \dots p, \end{aligned}$$

where $f(\mathbf{x}), q_1(\mathbf{x}), \dots, q_m(\mathbf{x})$ are posynomials and $h_1(\mathbf{x}), \dots, h_p(\mathbf{x})$ are monomials. Due to the convexity in log-scale, one can exactly transform[12] a GP to a convex program by means of the logarithmic change of variables $y_i = \log(x_i)$ and transforming the objective and constraints with the logarithmic transformations $F(\mathbf{y}) = \log f(\exp(\mathbf{y}))$, $Q_i(\mathbf{y}) = \log q_i(\exp(\mathbf{y}))$ and $H(\mathbf{y}) = \log h(\exp(\mathbf{y}))$ to obtain

$$\begin{aligned} \underset{\mathbf{y}}{\text{minimize}} \quad & F(\mathbf{y}) \\ \text{subject to} \quad & Q_i(\mathbf{y}) \leq 0 \qquad \text{for } i \in 1, \dots m \\ & H_i(\mathbf{y}) = 0 \qquad \text{for } i \in 1, \dots p, \end{aligned}$$

which is a convex program that can be efficiently solved using, for example, primal-dual interior-point methods; see [165] for more details. In practice, geometric programs with tens of thousands of decision variables and constraints can be solved to find the global optimum in a matter of seconds on a standard laptop computer.

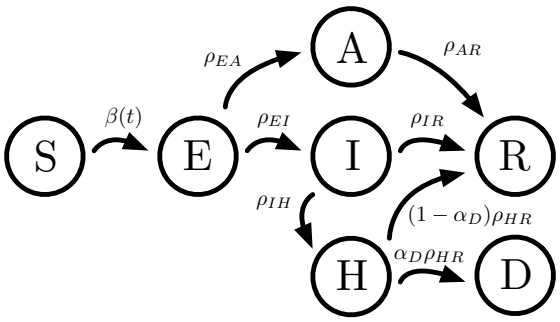


Figure 5.1: Illustration of the epidemic model under consideration.

5.2. Model

We now describe the epidemiological model under analysis. We consider a region with N individuals and propose a population model with *homogeneous mixing*, i.e., every pair of individuals come in contact with a probability that depends on aggregated mobility variables. Population models are commonly used in the absence of granular data on the network of social contacts in a region [29]. We assume that each individual can visit points of interest (POIs) belonging to different categories $1, 2, \ldots, K$; let $\mathbf{m}(t) = (m_1(t), m_2(t), \ldots, m_K(t))$ denote a vector of human mobility variables capturing the percentage change in volume of visits to each of those categories at a particular discrete time t (e.g., days) relative to a pre-established baseline; hence, $m_k(t) \in [0, \infty)$. In particular, we use publicly available mobility data from Google's COVID-19 Community Mobility Reports [160] which capture daily changes in visitation patterns to public places, for example, Retail & recreation, Grocery & pharmacy, and Parks, as well as time spent at work. For each category, Google reports the relative change in visits compared to a baseline of mobility measured before the lockdown measures took place. This baseline corresponds to the median daily visits to each category over the period comprising January 3 through February 6, 2020.

Our model consists of two layers: a *mobility layer* and an *epidemic layer*. The mobility layer captures the effect of human mobility on the spread of the disease and influences the

[12]In particular, we remark that the globally optimal solution to this convex program corresponds exactly to the globally optimal solution of the original GP, i.e., it is not a relaxation.

dynamics taking place on the epidemic layer. In the epidemic layer, we consider an extension of the classic SEIR epidemic model [166], [167] which explicitly accounts for asymptomatic hosts and hospitalizations, as in [168]. Each of the individuals in a region belongs to one of seven possible compartments described below. In this model, $\tilde{S}(t)$ represents the number of healthy individuals susceptible to becoming infected at a discrete time t. In our optimal control problems, we will consider a finite horizon T_c over which we can assume an almost constant number of susceptible individuals; hence, in these problems we assume $\tilde{S}(t) \approx S_0$ for all $0 \leq t \leq T_c$, which linearizes the model. The variable $\tilde{E}(t)$ represents the number of individuals who have contracted the virus (exposed) but are in an incubation period at time t. After the incubation period, agents can move to one of the infectious compartments; $\tilde{I}(t)$ represents the number of symptomatic individuals and $\tilde{A}(t)$ represents the number of asymptomatic individuals. The asymptomatic compartment is included since asymptomatic individuals play a crucial role in the spread of COVID-19, with transmission rates that are different from symptomatic individuals [169], [170]. Asymptomatic individuals eventually recover on their own and move on to the recovered compartment, represented by the variable $R(t)$. Symptomatic individuals can recover on their own, or their symptoms can worsen and they subsequently require hospitalization, in which case they are moved into a hospitalized compartment, represented by the variable $\tilde{H}(t)$. Since hospital capacity is a principal concern with the treatment of COVID-19, we include this compartment to constrain the control problems described in Section 5.4. In particular, our mobility-based control input will be constrained to prevent a hospital capacity overflow. Finally, individuals that are hospitalized may either recover and transition to the compartment represented by $\tilde{R}(t)$ or may die, and subsequently transition to the compartment represented by $\tilde{D}(t)$. With explicit data on the number of deaths in every region, we may train our model to predict the population of this compartment. We make the simplifying assumption that only individuals with severe symptoms are at a risk of dying and, hence, all of them have been previously hospitalized.

All parameters related to the dynamics of this model are summarized in Table 5.1. Using these parameters, the discrete-time evolution of the number of individuals in each compart-

Parameter	Meaning
$\beta(t)$	Rate at which susceptible individuals become infected due to contacts with infectious individuals at time t; $\beta(t)$ is a function of the mobility variables $\mathbf{m}(t)$ at time t
γ_A	Weight representing lower risk of infection when infectious individuals are asymptomatic
ρ_{EI}	Rate at which exposed individuals become symptomatic
ρ_{EA}	Rate at which exposed individuals become asymptomatic
ρ_{IR}	Rate at which symptomatic individuals recover on their own
ρ_{IH}	Rate at which symptomatic individuals develop severe symptoms and become hospitalized
ρ_{AR}	Rate at which asymptomatic individuals recover on their own
ρ_{HR}	Rate at which hospitalized individuals recover
α_D	Proportion of hospitalized individuals that die, relative to those that recover

Table 5.1: Summary of parameters in epidemic model. We assume $\rho_{IR} + \rho_{IH} < 1$, $\rho_{EI} < 1$, $\rho_{AR} < 1$, and $\rho_{HR} < 1$, since these parameters capture the fraction of individuals in each compartment that transition to other compartments, which must be less than one for the model to be meaningful. We also assume $\gamma_A \in [0, 1]$, reflecting the fact that asymptomatic individuals are less likely to spread the disease.

ment, illustrated in Figure 5.1, is given by:

$$\tilde{S}(t+1) = \tilde{S}(t) - \tilde{S}(t)\beta\left(t\right)\left(\gamma_A \tilde{A}\left(t\right) + \tilde{I}\left(t\right)\right), \tag{5.1}$$

$$\tilde{E}(t+1) = \left(1 - \rho_{EI} - \rho_{EA}\right)\tilde{E}\left(t\right) + \tilde{S}(t)\beta\left(t\right)\left(\gamma_A \tilde{A}\left(t\right) + \tilde{I}\left(t\right)\right), \tag{5.2}$$

$$\tilde{I}(t+1) = \left(1 - \rho_{IR} - \rho_{IH}\right)\tilde{I}\left(t\right) + \rho_{EI}\tilde{E}\left(t\right), \tag{5.3}$$

$$\tilde{A}(t+1) = \left(1 - \rho_{AR}\right)\tilde{A}\left(t\right) + \rho_{EA}\tilde{E}\left(t\right), \tag{5.4}$$

$$\tilde{H}(t+1) = \left(1 - \rho_{HR}\right)\tilde{H}\left(t\right) + \rho_{IH}\tilde{I}\left(t\right), \tag{5.5}$$

$$\tilde{R}(t+1) = \tilde{R}(t) + \rho_{IR}\tilde{I}(t) + \rho_{AR}\tilde{A}(t) + (1 - \alpha_D)\rho_{HR}\tilde{H}(t), \tag{5.6}$$

$$\tilde{D}(t+1) = \tilde{D}(t) + \alpha_D \rho_{HR}\tilde{H}(t). \tag{5.7}$$

In our model, we assume that susceptible individuals can transition into the exposed compartment when in contact with either Infected (symptomatic) or Asymptomatic individuals. We assume that the rate at which asymptomatic individuals infect others is weighted by a constant (unknown) parameter γ_A. We also assume that the portion of hospitalized in-

dividuals who die, relative to those that recover, is equal to an unknown constant α_D. In Section 5.3, we will introduce a methodology to learn these (and other) unknown parameters in our model. As mentioned above, this model is intended to solve optimal control problems over a finite time horizon T_c over which the number of new infected individuals is small compared to the entire population, so we can assume that the number of susceptible individuals at any time $0 \leq t \leq T_c$, $S(t)$, is well approximated with a constant; hence, in our control problems we set $\tilde{S}(t) = S_0$. Moreover, this assumption linearizes[13] the dynamics of the states; for notational clarity, we will omit the tildes when considering the states of the linear model. As we will see in Section 5.4, this linearization ensures that the entries of the state vector at any given time are posynomials on the parameter $\beta(t)$. However, we incorporate a non-linear dependency of the parameter $\beta(t)$ on the mobility restriction variables, which we will use as our external control variable, rendering the resulting model non-linear and multiplicative in the control input. Fortunately, the states of this linearized model upper-bound the states of the true model, as shown in the lemma below; Appendix B.3.2 contains simulation results to examine the tightness of these bounds in practice. Hence, by reducing the number of deaths in the linearized model, $D(t)$, we are guaranteed to reduce the number of deaths in the true nonlinear model, $\tilde{D}(t)$.

Lemma 5.2.1. *The states of the linearized model (wherein $S(t) = S_0$) upper-bound the states of the model in* (5.1)-(5.7)*. Thus, for all $t > 0$,*

$$\tilde{S}(t) \leq S(t) = S_0, \quad \tilde{E}(t) \leq E(t), \quad \tilde{I}(t) \leq I(t),$$
$$\tilde{A}(t) \leq A(t), \quad \tilde{H}(t) \leq H(t), \quad \tilde{R}(t) \leq R(t), \quad \tilde{D}(t) \leq D(t).$$

Proof. See Appendix A.3.1. □

The mobility layer of our model incorporates the effects of non-pharmaceutical interventions,

[13] In general the resulting dynamics are bilinear due to the terms $\beta(t)\tilde{A}(t)$ and $\beta(t)\tilde{I}(t)$, but in the original SEIR model $\beta(t) = \beta \ \forall t$, rendering these dynamics linear in the states. For this reason we refer to these dynamics as linear throughout this chapter.

such as social distancing and other forms of mobility restrictions, which a decision maker may employ to curb an epidemic. By reducing human mobility, a decision maker induces fewer contacts between susceptible and infected individuals and, thus, reduces the risk of infection. In particular, we relate the infection rate $\beta(t)$ to a time series $\mathbf{m}(t)$ of human mobility variables by means of a learnable function $f(\cdot)$. We choose f to be a parametric function whose parameters will be learned from data (described in detail in Section 5.3.1).

In order to employ non-pharmaceutical interventions, a decision maker designs a mobility control strategy to set the human mobility variables $\mathbf{m}(t)$ for some finite horizon $t \in \{0, \dots, T\}$. In mathematical terms, the decision maker designs an input $\{\mathbf{u}(t)\}_{t=0}^{T_c}$ which affects future values of the mobility variables. For simplicity, we assume an identity mapping between mobility and the input, so that $\mathbf{m}(t) = \mathbf{u}(t)$, and $\mathbf{u}(t)$ is in some set of admissible actions $\mathcal{U}$. In other words, we assume that mobility patterns may be directly actuated to a desired level by a decision maker, subject to feasibility constraints.

Intuitively, lower values of $\mathbf{u}(t)$ correspond to more restrictions on human mobility. The decision maker may have fine-grained control over their control strategy, for example by closing individual establishments, imposing occupancy limits, or restricting hours of operation, and as such we treat the individual components of the control action $u_k(t)$ as continuous variables. Moreover, some categories may have different admissible actions, e.g., it may not be possible to close down all pharmacies but closing all gyms is reasonable. In mathematical terms, we will consider a set of allowable control actions $\mathcal{U}$ that is described using posynomial inequalities and monomial equalities.

Furthermore, implementing mobility restrictions in this manner cannot be done without incurring a financial loss. Closure of businesses causes economic losses, which need to be taken into consideration when selecting an appropriate control strategy. In particular, applying the temporal control strategy $\mathbf{u}(t)$ of mobility restrictions incurs a cost $C_t(\mathbf{u}(t))$ which we assume to be monotonically decreasing with $\mathbf{u}$ and convex in log-scale, reflecting that the costs on society of restricting mobility are marginally increasing.

5.3. Parameter estimation via multitask learning

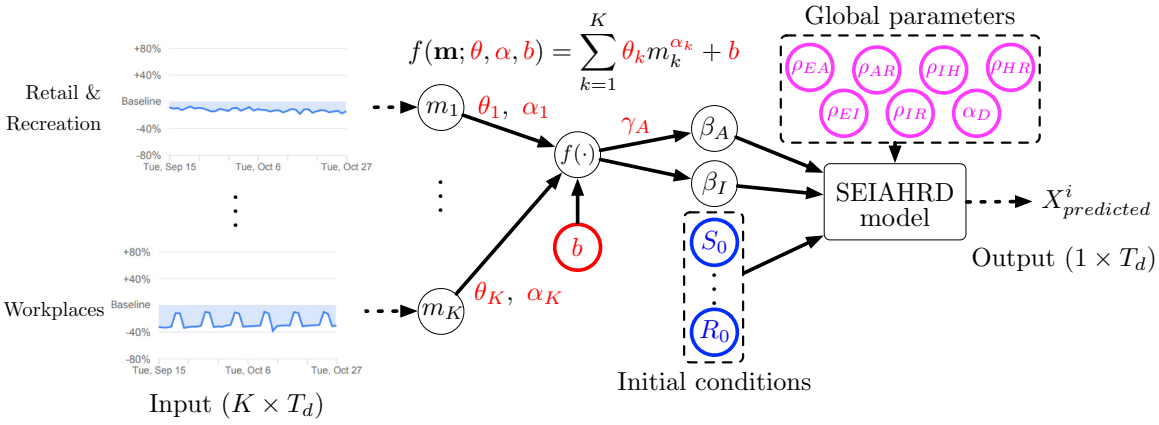


Figure 5.2: The learning pipeline for a given region. Global parameters, learned using all available data, are shown in magenta; initial conditions, learned for each region, are shown in blue; mobility mapping parameters, again different across regions, are shown in red.

Several recent epidemic prediction methods opt to set some (or all) of the parameters in their models to estimations from the medical and virology literature (e.g. [137]–[139], [147], [149]). However, these parameters often have wide confidence intervals and are commonly inferred from statistical models that do not take into account the effects of social distancing and hospital capacity [171]. Other recent works learn these parameters from data [138]–[140], [150], but do not explicitly model the infectivity of the epidemic using mobility patterns, making the design of related control strategies difficult. In contrast to these approaches, our model is entirely data-driven in that all parameters used (including initial conditions) are learned directly from data, and we learn an explicit mapping between mobility patterns and the spread of the epidemic. In particular, we employ a multitask learning approach [172] by leveraging both data on the number of deaths across a set of regions, and mobility data describing how often individuals in a region visit different points of interest. The key idea is to consider each region of interest as a separate prediction subtask, while many parameters of the models describing the evolution of the epidemic are shared across regions. Indeed, certain parameters of the epidemic model are intrinsic to the disease; hence, they do not depend on the geographical location from which data is collected. As such, when calibrating this model to a given region, we can benefit from the data from other regions by employing a

multitask learning framework, providing better parameter estimates and avoiding overfitting. Towards this goal, we pool data from multiple regions (e.g., U.S. counties) and minimize a global cost function in which the global parameters are shared across regions but the mobility parameters and initial conditions are specific to each region. This provides *statistical data amplification*, as the prediction subtask for each county is given additional data with which to learn the global epidemic parameters. Another relevant facet of multitask learning is *representation bias*; the learning procedure will avoid local minima intrinsic to only a few (or just one) of the counties. The learning pipeline from data to predictions for a single region is shown in Figure 5.2. We fit our models using publicly available mobility data from Google's COVID-19 Community Mobility Reports [160] which captures daily changes in visitation patterns to public places, for example, Retail & Recreation, Grocery & Pharmacy, and Parks, as well as time spent at Workplaces. This dataset is organized as different time series for six different categories. For each category, Google reports the relative change in visits compared to a baseline of mobility measured before the lockdown measures took place. This baseline corresponds to the median daily visits to each category over the period comprising January 3 through February 6, 2020. Furthermore, we use public data from The New York Times, based on reports from state and local health agencies [161], consisting of daily and cumulative caseloads and deaths attributed to COVID-19 in the United States. To account for inconsistencies and lags in reporting, we compute a seven-day rolling average on the original time series for the calibration of our model.

As mentioned above, there are two layers to our model, namely the *mobility layer*, which is a mapping from mobility data to the infection rate $\beta(t)$ (described in Section 5.3.1) and the *epidemic layer*, which describes the dynamics of the disease itself (discussed in Section 5.3.2). Since parameters such as the latency period, ratio of infected individuals who develop symptoms, and case fatality ratio are intrinsic to the disease and should not vary greatly based on the geographical area being studied, we group these together across regions as global parameters and learn them jointly with all the available data. However, the mapping from mobility data to the infection rate and initial conditions of the regional

epidemic are dependent on the locality, and thus they are learned using only the data from their region.

5.3.1. The mobility layer

To learn the function $f : \mathbf{m} \mapsto \beta$ from mobility data to the infection rate, we must first select an appropriate class of functions for such a mapping. Although we could use any parametric family of functions, such as neural networks, to estimate f, not all choices are tractable. In particular, neural networks may provide great prediction performance but would render an intractable control problem. In order to obtain a tractable control problem, we choose to model the function f using a parametric posynomial function [30]. As we will show in Section 5.4, this choice allows us to use geometric programming to efficiently solve several nonlinear optimal control problems of interest. Thus, we model f in a parametric way as

$$f(\mathbf{m}; \boldsymbol{\theta}, \boldsymbol{\alpha}, b) = \sum_{k=1}^{K} \theta_k m_k^{\alpha_k} + b, \tag{5.8}$$

where we recall that $m_k \in [0, \infty)$. From a practical standpoint, this posynomial approximation is justified because β can be viewed as the product of the contact rate (the expected number of contacts an individual has with others) and the transmission risk, which is constant over time. Moreover, the number of contacts within a category should exponentially increase with the number of visits to points of interest in that category. Since the mobility data is stratified across K different categories, we allow the parameters to be different across the categories. Thus, in the parametric function $f(\mathbf{m}; \boldsymbol{\theta}, \boldsymbol{\alpha}, b)$ the probability of transmission is captured by $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_K) \in \mathbb{R}^K_{\geq 0}$, the exponential growth of infectivity is captured by $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}^K$, and the bias term b accounts for infections that are unrelated to human mobility.

Since susceptible individuals may become infected by either symptomatic or asymptomatic individuals, the mobility mapping f is incorporated into the epidemic layer in two terms, as seen in (5.2). Firstly, it is used to model new infections from symptomatic individuals via the term $\beta(\mathbf{m}(t))S_0 I(t)$; secondly, we include a weighting term γ_A in the term $\gamma_A \beta(\mathbf{m}(t))S_0 A(t)$

to model the rate of new infections from asymptomatic infectious individuals. Thus, altogether we denote the set of parameters corresponding to the mobility mapping function for region i as $\Psi^i_{mobility} \coloneqq \{\boldsymbol{\theta}, \boldsymbol{\alpha}, b, \gamma_A\}$.

5.3.2. The epidemic layer

Our model is a latent-space model; hence, the states are not fully observable. Following common practice with these models, we treat the unobserved initial conditions as unknown parameters to be identified from the data. Since the dynamical trajectories of the epidemic are different across regions (e.g., U.S. counties), for each region i we learn a set of initial conditions $\Psi^i_0 \coloneqq \{S^i_0, \ldots, R^i_0\}$. As mentioned previously, we follow a multitask learning approach and, thus, the remaining global parameters, which we assume to be intrinsic to the disease, are shared across all regions and learned collectively using data available from all regions. The set of global parameters is denoted by $\Psi_{global} \coloneqq \{\rho_{EA}, \rho_{EI}, \rho_{AR}, \rho_{IH}, \rho_{IR}, \rho_{HR}, \alpha_D\}$, which includes all clinical parameters that depend on the nature of the virus alone (i.e., they are not influenced by geography or human mobility).

5.3.3. Simulated predictions for Philadelphia and surrounding counties

In order to validate the predictive accuracy of our data-driven model, we conducted a case study using counties from the Delaware Valley metropolitan statistical area (commonly known as the *Philadelphia metropolitan area*), which includes Philadelphia County and nine surrounding regions, including counties in Maryland, New Jersey, and Delaware. Using county-level data, we learned both the local mobility mapping functions and initial conditions, as well as the global clinical parameters of the epidemic. Due to the known inconsistencies and lags in reporting of cases and deaths, we use the rolling seven-day average of cumulative deaths for both training and prediction. Formally, if we have training data for M counties over T_d days, the training loss for the set of parameters $\Psi \coloneqq \Psi_{global} \cup_{i=1}^{M} \Psi^i_{mobility} \cup_{i=1}^{M} \Psi^i_0$ is the following mean-squared error loss function:

$$\ell_{train}(\Psi) = \frac{1}{MT_d} \sum_{i=1}^{M} \sum_{t=1}^{T_d} \frac{1}{N_i} \left\| X^i(t) - \hat{X}^i(t; \Psi) \right\|_2^2, \tag{5.9}$$

(a) Philadelphia County, PA. (b) Montgomery County, PA. (c) Bucks County, PA.

(d) Chester County, NJ. (e) Camden County, NJ. (f) Gloucester County, NJ.

Figure 5.3: Example of predicted cumulative deaths for several US counties, trained on 90 days of mobility and death count data, and tested on 30 days of the same. Predictions are shown in blue, and the rolling seven-day averages of real deaths data are shown as dashed lines. The area in white denotes training data, and the area in red denotes test data.

where N_i is the population of region i, $X^i(t)$ is the measured rolling seven-day average of cumulative deaths on day t for region i, and $\hat{X}^i(t;\Psi)$ is the predicted value for the same region for a set of parameters Ψ. We normalize the number of deaths by the population in each county to avoid biasing our predictions towards counties with a larger population. Note that given data on the populations of additional compartments, such as the number of hospitalizations, the loss in (5.9) generalizes readily. Moreover, any differentiable loss function may be substituted in this approach; mean-squared error was chosen for simplicity.

Our model was trained by computing gradients of the loss function in (5.9) with respect to all parameters in Ψ via the automatic differentiation package `autograd` [173] and running stochastic gradient descent using `adam` [174] over several independent trials using different initial guesses to account for the non-convexity of the training problem. In particular, `autograd` performs reverse-mode differentiation (i.e., backpropagation) to compute numerical gradients; hence, differentiability of the loss function ℓ_{train} and the dynamical equations

governing our system help to ensure the existence of such gradients. Global clinical parameters were initialized by using plausible values from the medical literature [138], [175]–[181], while local parameters are initialized randomly in each trial. Different counties are split across random batches, allowing the global parameters (trained using all data in our multitask approach) to converge more quickly, which in turn allows the local parameters to converge to values which fit more closely with the global parameters. From these trials, we select the set of parameters with lowest testing error, and present some examples of predictions using these chosen parameters in Figure 5.3.

5.4. Optimal control using geometric programming

Traditional optimal control techniques are not directly applicable to compartmental epidemic models for a number of reasons. First, epidemic models are typically nonlinear and the effect of non-pharmaceutical interventions is not additive, but multiplicative. Therefore, standard techniques such as LQR cannot be readily employed. Due to the modeling choices proposed in this chapter, we obtain a mobility-driven epidemic dynamics amenable to certain optimal control problems which may be solved using tools from geometric programming [30]. In particular, we can solve the data-driven optimal control problems aiming to minimize the final number of deaths while respecting budget constraints on the economic costs associated with implementing mobility restrictions, as well as avoiding hospital overflows, with guarantees of global optimality. In practice, all people may not exactly follow the desired mobility restrictions; thus, such a strategy may be implemented in receding horizon. For example, if a decision maker sets mobility restrictions on a weekly basis, they may plan ahead for several months, implement a week's restrictions, then re-plan the following week and implement the resulting strategy, and repeat.

In order to employ geometric programming in this setting, it is necessary that the states can be expressed as posynomial functions of the control input. The negative term in (5.1) bars us from directly using our model; however, the horizon $0 \leq t \leq T_c$ over which our control problem takes place is small enough that the number of new infected individuals

is small relative to the total population, and so we may assume a nearly constant number of susceptible individuals exist, i.e., $\tilde{S}(t) \approx S_0$. This assumption linearizes our model, and allows us to express the states as posynomial functions of the control input; hence, we will consider this linearized model for the remainder of this section. Fortunately, this linearization has a modest effect on our predictions; indeed, this linearized model upper-bounds the states of the true model, as illustrated in Lemma 5.2.1 and Appendix B.3.2.

We start our analysis by considering the minimization of the final number of deaths. Since we will minimize deaths in the linear model, $D(t)$, we remark by Lemma 5.2.1 that we induce conservatism in our control framework by minimizing an upper bound on the number of deaths in the nonlinear model, $\tilde{D}(t)$. As the eventual control strategies will be implemented in a receding horizon, we allow the inclusion of a daily discount factor γ_D and a terminal cost γ_∞ on the number of deaths at the end of the horizon T_c. Hence, the decision maker aims to minimize the following objective function:

$$J \coloneqq \sum_{t=1}^{T_c-1} \gamma_D^t \alpha_D \rho_{HR} H(t) + \gamma_\infty \alpha_D \rho_{HR} H(T_c). \tag{5.10}$$

We remark from (5.7) that the incident deaths on day t, i.e., the number of new deaths $D(t) - D(t-1)$, is exactly $\alpha_D \rho_{HR} H(t)$. The discount factor $\gamma_D \in (0, 1]$ is motivated by the uncertainty of deaths in the future which might be prevented by interventions not available in the present day. For example, the probability that a vaccine is widely available in the future increases as time passes. In particular, $1 - \gamma_D$ can be considered the probability of a vaccine being widely available on each day in the future, so the probability of no vaccine being widely available by day t is γ_D^t and, hence, deaths predicted at day t are only accounted for if there is not a vaccine that could prevent them. The terminal cost γ_∞ illustrates the desire to keep the number of deaths low beyond the time horizon in consideration. For example, let us assume that beyond the time horizon T_c the epidemic has been curbed and the number of new daily deaths falls below $\alpha_D \rho_{HR} H(T_c)$. In the worst case scenario we

would have $H(t) = H(T_c)$ for $t \geq T_c$, hence,

$$\sum_{t=T_c}^{\infty} \gamma_D^t H(t) = H(T_c) \frac{\gamma_D^{T_c}}{1-\gamma_D}.$$

Defining $\gamma_\infty := \gamma_D^{T_c}/(1-\gamma_D)$, and $\gamma_\infty = 0$ if $\gamma_D = 1$, the discounted number of deaths beyond T_c is given by the terminal cost $\gamma_\infty D(T_c)$.

Given that the infection rate $\beta(t)$ depends on mobility, we assume that a decision maker can restrict mobility dynamically to curb the number of deaths by designing a mobility control strategy $\mathbf{u}(t)$ so that $\beta(t) = f(\mathbf{u}(t))$. Furthermore, we assume that $\mathbf{u}(t)$ is constrained to be within a set $\mathcal{U}$ reflecting that essential businesses cannot be severely restricted and that some mobility restrictions are only partially effective.

These mobility restrictions incur a cost which could be measured in terms of a pecuniary cost to the economy, absolute number of visits lost by businesses, or impact on the utility of citizens. In our framework, we quantify the economic cost of imposing a mobility control strategy $\mathbf{u}(t)$ using a cost function $C_t(\mathbf{u}(t))$ which, in general, can be time-varying; for example, we can use different costs for mobility restrictions on workdays and weekends. Moreover, such a cost function may incorporate terminal costs to account for economic losses beyond the time horizon T_c. We choose to model $C_t(\mathbf{u}(t))$ as a posynomial on $\mathbf{u}(t)$, since this is amenable to a geometric programming approach. We thus investigate the problem of choosing an optimal mobility control strategy $\mathbf{u}^\star(t)$ that minimizes the number of cumulative deaths while keeping the total cost of the intervention, given by $\sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t))$, below a pre-specified budget $\mathcal{B}$, while obeying a limit on hospitalizations τ_H.

As we will show, this problem can be expressed as a geometric program; this stems from the fact that the states $H(t)$ and $D(t)$ can be expressed as posynomial functions of the mobility control variables $\mathbf{u}(t)$, as shown in the following lemma.

Lemma 5.4.1. *The functions $H(t)$ and $D(t)$, representing the number of hospitalized individuals and deaths at time t, are posynomials on the entries of $\mathbf{u}(t)$ for $t = 0, 1, \ldots, T_c$.*

Proof. See Appendix A.3.2. □

Since a positively weighted sum of posynomials is also a posynomial, we obtain our main result below.

Theorem 5.4.1. *If $C_t(\mathbf{u}(t))$ is a posynomial cost function for all t, and the set of admissible control actions $\mathcal{U}$ is described by posynomial inequalities and monomial equalities, then the following is a geometric program:*

$$\begin{aligned} \underset{\mathbf{u}(0),\mathbf{u}(1),\ldots,\mathbf{u}(T_c-1)}{\text{minimize}} \quad & \sum_{t=1}^{T_c-1} \gamma_D^t D(t) + \gamma_\infty D(T_c) \\ \text{subject to} \quad & \sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t)) \leq \mathcal{B} \\ & H(t) \leq \tau_H, && t = 1, \ldots, T_c, \\ & \mathbf{u}(t) \in \mathcal{U}, && t = 0, \ldots, T_c-1, \end{aligned} \tag{5.11}$$

where $\mathbf{u}(t)$ are the NPI control actions, and $H(t)$ and $D(t)$ are the number of individuals who are hospitalized and who die at time t, respectively.

Proof. See Appendix A.3.3. □

The choice of the cost function and the budget $\mathcal{B}$ may be difficult to discern. It may be unclear how reductions of mobility in a given category impact the economy, and the choice of a budget for those mobility restrictions engenders an implicit trade-off between minimizing deaths and economic costs. The choice of these cost functions is up to the decision maker and, in practice, could leverage available economic data about employment, revenue losses and social-wellness. To choose such a budget $\mathcal{B}$ once a cost function has been defined, we propose a principled approach to obtain the minimal budget under which hospitals remain below capacity, preventing the steep increase in deaths due to an overwhelmed healthcare system. This is achieved by solving an auxiliary optimal control problem that aims to find the minimum cost required to keep the number of hospitalized individuals $H(t)$ below a

threshold τ_H at all times. This auxiliary optimal control problem is also a geometric program as long as the cost functions $C_t(\mathbf{u}(t))$ are posynomials (for example, found via posynomial fitting [30] on economic data); this is stated in Theorem 5.4.2 and the proof follows from Lemma 5.4.1.

Theorem 5.4.2. *If $C_t(\mathbf{u}(t))$ is a posynomial cost function for all t, and the set of admissible control actions $\mathcal{U}$ is described by posynomial inequalities and monomial equalities, then the minimal budget required to keep hospitalizations below a given threshold τ_H is given by*

$$\mathcal{B}^\star = \sum_{t=0}^{T_c-1} C_t(\mathbf{u}^\star(t)), \tag{5.12}$$

where $\mathbf{u}^\star$ is the solution to the geometric program

$$\begin{aligned} \underset{\mathbf{u}(0),\ldots,\mathbf{u}(T_c-1)}{\text{minimize}} \quad & \sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t)) \\ \text{subject to} \quad & H(t) \leq \tau_H, \quad t = 1,\ldots,T_c, \\ & \mathbf{u}(t) \in \mathcal{U}, \quad t = 0,\ldots,T_c-1. \end{aligned} \tag{5.13}$$

where $\mathbf{u}(t)$ are the NPI control actions, and $H(t)$ is the number of individuals who are hospitalized at time t.

Proof. See Appendix A.3.4. □

The budget $\mathcal{B}^\star$ obtained from Theorem 5.4.2 can be seen as a conservative cost which only guarantees that hospital operations remain within capacity, avoiding overflow. In particular, this conservative budget leaves little room to intervene and fully curb the spread of an epidemic in its earlier stages. The decision maker should thus use some budget $\mathcal{B} \geq \mathcal{B}^\star$ in implementing the results from Theorem 5.4.1 to obtain a less conservative control input $\mathbf{u}(t)$.

5.4.1. Practical considerations

As mentioned previously, in practice some individuals may not follow guidance regarding mobility restrictions; thus, our control strategy may be implemented in receding horizon. Since a decision maker may wish to set mobility restrictions on a rolling basis (e.g., weekly), they may plan ahead for several months, implement restrictions within the time frame of interest, then re-plan at the end of the time period, and repeat. As the decision maker may tune the different categories of mobility variables independently, we allow each of the components $k = 1, \ldots, K$ of the mobility control action $\mathbf{u}(t)$ to vary between a lower bound $\underline{u}_k > 0$ (representing full lockdown) and an upper bound $\overline{u}_k$ (representing no restrictions at all). Moreover, to reflect the fact that decision makers want to avoid dramatic changes to restrictions on a consistent basis, we enforce that the policy must remain fixed for periods of one week (seven days), and that each component of mobility may change no more than $\Delta\%$, i.e., $(1-\Delta)u_k(t-1) \leq u_k(t) \leq (1+\Delta)u_k(t-1)$, $\forall k, t \in \{1, \ldots, T\}$. However, it may be the case that categories of mobility are dependent, and cannot be tuned arbitrarily; we explore this idea in detail in Appendix B.3.1. These conditions lead to the set of admissible control actions

$$\begin{aligned} \mathcal{U} = \{u_1(0), \ldots, u_K(0), \ldots, u_1(T_c), \ldots u_k(T_c) \in \mathbb{R}^{K \times T_c} \ : \\ u_k(t) \in [\underline{u}_k, \overline{u}_k] \cap [(1-\Delta)u_k(t-1), (1+\Delta)u_k(t-1)], \\ u(s) = u(s+1) = \cdots = u(s+6), s \in \{0, 7, \ldots\}\}. \end{aligned} \tag{5.14}$$

In practice the cost function $C_t(\mathbf{u}(t))$ may be supplied by a decision maker, or may be found via posynomial fitting [30] using economic data from a region. In the absence of such data, we choose a time-invariant cost function $C(\mathbf{u}(t))$ that satisfies the requirements of being convex in log-scale and decreasing, given by

$$C(\mathbf{u}(t)) = \sum_{k=1}^{K} c_k \frac{u_k(t)^{-1} - \overline{u}_k^{-1}}{\underline{u}_k^{-1} - \overline{u}_k^{-1}}, \tag{5.15}$$

where $\mathbf{c} = (c_1, \ldots, c_K)$ is a relative cost weighting of the mobility categories. For example, this relative cost could reflect that visits lost to healthcare facilities are more costly than visits lost to retail venues. As shown in the lemma below, this selection of cost function and set of allowable control actions renders (5.11) and (5.13) geometric programs.

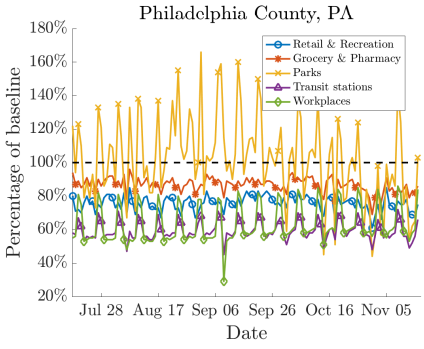
Lemma 5.4.2. *The set of admissible control actions in* (5.14)*, where* $0 < \underline{u}_k < \overline{u}_k$ *for all* $k \in \{1, \ldots, K\}$*,* $\Delta > 0$*, and the cost function in* (5.15)*, where* $c_k \geq 0$ *for all* $k \in \{1, \ldots, K\}$*, are sufficient for* (5.11) *and* (5.13) *to be geometric programs.*

Proof. See Appendix A.3.5. □

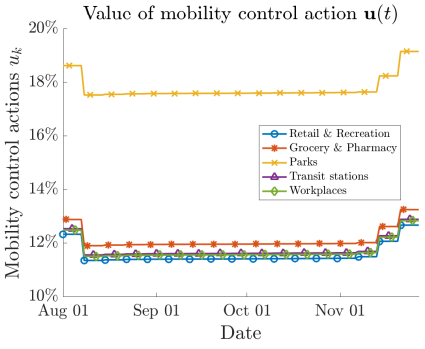
5.4.2. Control simulations

We demonstrate the effectiveness of our control approach with a case study for the counties in the greater Philadelphia area, illustrated in Figure 5.4 and Figure 5.5. All simulations were performed on a laptop computer with 16GB of RAM and a 2.2GHz Intel Core i7 CPU. The parameters of our compartmental model as well as the mobility mapping $\beta(\mathbf{u}(t))$ are learned from data as described in Section 5.3.3 using the proposed multitask learning framework. Our models are trained using recent data before the widespread usage of vaccines, from August 1st to October 31st, 2020, and tested from November 1st-30th, 2020. In particular, in Figure 5.5(c) we illustrate the number of cumulative deaths predicted using the optimal control strategy $\mathbf{u}^\star(t)$ as compared to the number of cumulative deaths predicted based on the true mobility data $\mathbf{m}(t)$ from Philadelphia. We invoke Lemma 5.4.2 to select the cost function $C_t(\mathbf{u}(t))$ and set of allowable control actions $\mathcal{U}$. To elucidate the set of allowable control actions $\mathcal{U}$, we select the values $\underline{u}_k$ and $\overline{u}_k$ independently for each category based on the mobility data used in our multitask learning framework. We set the limit on the relative change in mobility, Δ, to 10%. We further re-scale the mobility data to be in $[0, 1]^K$, which has been shown to speed up convergence of learning methods [182]. For simplicity, we assign equal costs to each category of mobility, so that $c_k = 1$ for all $k \in \{1, \ldots, K\}$.

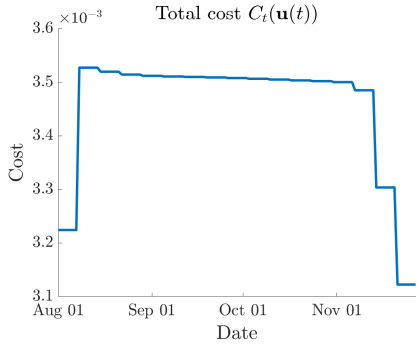
We show the minimal-cost control action $\mathbf{u}^\star(t)$ computed by solving (5.13) in Figure 5.4.



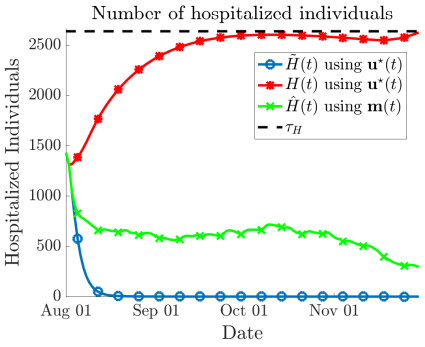
(a) Measured mobility data $\mathbf{m}(t)$ for Philadelphia, PA.



(b) Value of minimal-cost control action $\mathbf{u}^\star(t)$ (lower means less mobility allowed).



(c) Actuation cost $C_t(\mathbf{u}^\star(t))$ for control action $\mathbf{u}^\star(t)$.



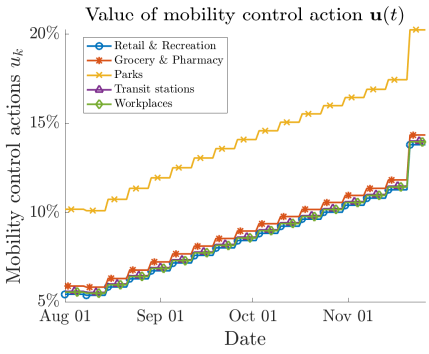
(d) Hospitalized individuals, with hospital bed threshold τ_H shown as the dashed line.

Figure 5.4: Minimal-cost control strategy $\mathbf{u}^\star(t)$ for Philadelphia County, PA, obtained by solving the geometric program (5.13). For comparison, we plot the true mobility data $\mathbf{m}(t)$ over the same time period. As shown in Lemma 5.2.1, number of hospitalized individuals in the linear model, $H(t)$, upper-bounds the number of hospitalized individuals in the nonlinear model, $\tilde{H}(t)$. We also include the hospitalizations predicted based on the true mobility data $\mathbf{m}(t)$ as a baseline. Parameters of the models used herein were learned as described in Section 5.3.3. We obtain the budget $\mathcal{B}^\star$ to use in geometric program (5.11) by taking the total cost of the minimal-cost control strategy, i.e., $\mathcal{B}^\star = \sum_{t=0}^{T_c-1} C_t(\mathbf{u}^\star(t))$. Control actions in Figure (a) for categories except transit stations are similar, and are overlaid.

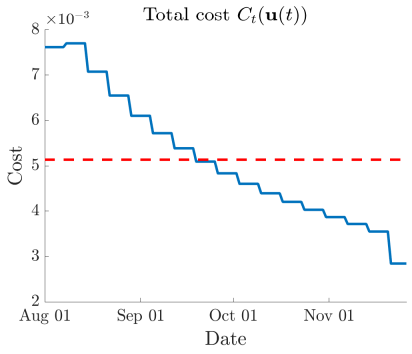
As mentioned previously, interior point methods solve these problems very efficiently; full discussions on the computational complexity of our approach are in Appendix B.3.3. Figure 5.4(a) illustrates the measured mobility data $\mathbf{m}(t)$ in Philadelphia County in order to compare with the designed control strategies. In Figure 5.4(b) we show the individual mobility control strategies for each category; recall that a lower value of $u_k^\star(t)$ corresponds to a lower value of mobility allowed for category k, which incurs a higher cost. As an example

of the efficacy of the learned mapping between mobility categories and infectivity, mobility in parks is allowed to be larger than all other categories since it is much easier to socially distance outdoors. Mobility is kept consistently low across the time horizon to keep the number of hospitalized individuals under control, but allowed to increase later on, reducing the overall cost as an economic trade-off. In Figure 5.4(d) we see the result of Lemma 5.2.1, namely that the number of hospitalizations in the linear model $H(t)$, which is used in the GP (5.13), upper-bounds the hospitalizations in the nonlinear model $\tilde{H}(t)$. Indeed, we see that our GP is providing a conservative estimate for the control action $\mathbf{u}^\star(t)$: even though $H(t)$ is near the hospitalization threshold τ_H, the predicted hospitalizations under the nonlinear model $\tilde{H}(t)$ are approaching zero at the end of the time horizon; we explore this phenomenon further in Appendix B.3.2.

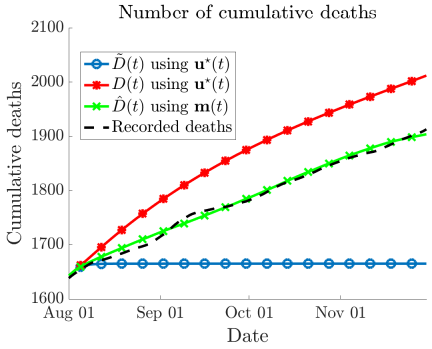
In Figure 5.5 we show the results of Theorem 5.4.1 by solving the GP (5.11). While this problem is related to the GP (5.13), the objective in this case is to minimize deaths subject to the budget $\mathcal{B}^\star$ computed by solving (5.13). Hence, while the mobility control strategy $\mathbf{u}^\star(t)$ and corresponding cost are similar to what is shown in Figure 5.4, there is a sharper initial restriction to mobility which is lifted in a more gradual manner. Figure 5.5(c) shows the number of cumulative deaths using the mobility control action $\mathbf{u}^\star(t)$ in blue, as compared to the deaths incurred without any control, i.e., with the measured mobility pattern data $\mathbf{m}(t)$. To confirm the accuracy of these predictions, we also show the number of deaths recorded in the same time horizon as the dashed black line. Indeed, by solving the GP (5.11) and implementing the associated control strategy, our model predicts over 200 lives could have been saved between August 1st and November 30th. As before we show the predictions using the linear model, which upper-bound the predictions in the nonlinear model; this further illustrates the conservatism of our control approach.



(a) Value of optimal minimal-death control action $\mathbf{u}^\star(t)$ (lower means less mobility allowed).



(b) Actuation cost $C(\mathbf{u}^\star(t))$ with control action $\mathbf{u}^\star(t)$; average daily budget $\mathcal{B}^\star/T_c$ shown dashed in red.



(c) Number of cumulative deaths, with true recorded deaths shown as the dashed line.

Figure 5.5: Optimal minimal-death control strategy $\mathbf{u}^\star(t)$ for Philadelphia County, PA, obtained by solving the minimal-death GP (5.11), from August 1st to November 30th, 2020. In Figure (c) we compare the cumulative deaths predicted using the optimal control strategy $\mathbf{u}^\star(t)$ in the nonlinear model, $\tilde{D}(t)$ (shown in blue), cumulative deaths predicted in the linear model, $D(t)$ (shown in red), and the cumulative deaths predicted based on the true mobility data $\mathbf{m}(t)$ as a prediction baseline (shown in green). Parameters of the models used herein were learned as described in Section 5.3.3, with the budget $\mathcal{B}^\star$ taken from the solution to the minimal-cost GP (5.13). Control actions in Figure (a) for categories except transit stations are similar, and are overlaid.

CHAPTER 6

Conclusion and Future Work

This thesis studied the learning and control of phenomena on networks. First we presented SPECTRE, a robust algorithm for aligning networks which outperforms state-of-the-art competitors in terms of the quality of produced matchings on moderately correlated networks. Indeed, SPECTRE is the first scalable algorithm to exhibit such high precision without using prior information. Unlike most algorithms, SPECTRE uses no prior or side information. Instead, a noisy seed set estimate is generated using spectral centrality measures. SPECTRE then uses bootstrap percolation techniques along with a backtracking strategy that allows it to iteratively improve the quality of the alignment. We compared its performance with other algorithms found in the literature on a number of benchmark real-world networks from different sources. As the correlation between the networks to be aligned decreases, SPECTRE remains able to obtain high-quality alignments.

The robustness of SPECTRE's performance is due to the bootstrap percolation framework. Future lines of research can extend this framework to the attributed network alignment problem and to the construction of network embeddings. Furthermore, these results motivate several theoretical questions regarding the ranking induced by spectral centrality measures and its apparent robustness to edge deletions, which we explored in the context of graph neural networks.

Next, we proposed an efficient methodology for estimating the eigenvalues of any arbitrary graph matrix of a network of interacting dynamical agents using a set of dynamical measurements in the presence of periodic control inputs. The graph matrix may be directed, have edge weights, incorporate self-loops, and render the system unstable. Unlike other methods, we require only a single finite sequence of discrete, temporal measurements from the multiagent network of length, at most, $2n + s$ (with n agents and input period s). Moreover, we need no prior knowledge of the network topology, initial condition, or which agents are

contributing to the measurements. For any arbitrary random initial condition, our approach recovers all eigenvalues corresponding to observable eigenmodes of the pair $(S, \mathbf{c}^\intercal)$, almost surely. We developed our technique for systems in both discrete- and continuous-time. Our simulation results showed that we can recover the observable spectrum of the graph matrix in all cases with high accuracy.

We then introduced an architecture for processing signals supported on hypergraphs via graph neural networks (GNNs), which we call Hyper-graph Expansion Neural Networks (HENNs), and provided the first bounds on the stability and transferability error of a hypergraph signal processing model. To do so, we provided a framework for bounding the stability and transferability error of GNNs across *arbitrary* graphs via spectral similarity. By bounding the difference between two graph matrices in the positive semi-definite sense via their eigenvalue spectrum, we showed that this error depends only on the properties of the GNN and the magnitude of spectral similarity of the graph matrices. Moreover, we showed that existing transferability results that assume the graphs are small perturbations of one another, or that the graphs are random and drawn from the same distribution or sampled from the same graphon can be recovered using our approach. Thus, both GNNs and our HENNs (trained using normalized Laplacians as graph shift operators) are increasingly stable and transferable as the graphs become larger. Experimental results illustrated the importance of considering multiple graph representations in HENN, and showed its superior performance when transferability is desired.

Future work may explicitly prove that the hypergraph energies of two similar hypergraphs will be within a factor of ϵ when the energy of their clique expansions and line graphs are also within ϵ. In this manner, we can explicitly establish a measure for spectral similarity of hypergraphs and relate it to our Hyper-graph Expansion Neural Networks. Furthermore, we may explore definitions for spectral similarity that apply to simplicial complices. In particular, we may investigate the conditions under which the Hodge Laplacians of two simplicial complices will be spectrally similar across the same order. More generally, we

may study convolutions over simplicial complices, as Hodge theory for simplicial complices is a special case of the more general Hodge-de Rham theory of manifolds. Indeed, due to this theory simplicial complices can be understood as discretizations of manifolds in the sense of the diffusion operators defined on them. This viewpoint may explain in a formal sense that convolutions on simplicial complices are the appropriate way to discretize convolutions over manifolds.

Finally, we presented a multitask learning and nonlinear optimal control framework that aims to bridge the gap between optimal control theory of epidemic models and applicable data-driven models for analyzing the spread of COVID-19 and other future epidemics. To estimate the parameters of our model, we proposed a multitask learning approach that leverages mobility and epidemic data from multiple regions to capture how daily changes in mobility patterns affect the spread of the disease, and to accurately predict the resulting daily and cumulative deaths. Using this data-driven model we presented a nonlinear optimal control framework using geometric programming to efficiently design non-pharmaceutical interventions which limit the spread of the epidemic while obeying a budget constraint on the economic loss incurred. Furthermore, we presented a principled method for determining such a budget based on eliminating excess deaths due to over-utilization of hospital resources. We validated both our model and our control framework in a case study on the greater Philadelphia area.

In the future, this work could be extended to accommodate for robustness considerations as well as stochastic transitions in the epidemic layer, which introduce the additional challenge of expressing chance constraints as posynomial functions. The success of geometric programming in our work comes from expressing states of the system as posynomials on the mobility variables, allowing for the potential extension to models with more sophisticated mappings from human mobility to epidemic dynamics. For example, generalized geometric programming admits functions that are max-monomials or posynomials with fractional exponents, which opens the door to modeling epidemic dynamics and human-mobility using

ReLU neural networks or posynomial approximations to arbitrary functions. Furthermore, since geometric programs can be solved efficiently our approach could be applied to models with higher complexity, such as networked meta-population models which can make use of more granular datasets.

APPENDIX A

Proofs

In this appendix we collect all proofs from the results in Chapters 2, 4 and 5.

A.1. Proofs for Chapter 2

A.1.1. Proof of Lemma 2.2.1

Lemma 2.2.1. *For periodic control inputs* $\mathbf{u}[k]$ *with period* r*, the differences of output measurements satisfy*

$$\Delta y[r+k] \coloneqq y[r+k] - y[k] = \mathbf{c}^\intercal S^k (\mathbf{x}[r] - \mathbf{x}_0). \tag{2.6}$$

Proof. By definition of $y[k]$ in (2.5), we have

$$\begin{aligned}
&\Delta y[r+k] = y[r+k] - y[k] \\
&= \mathbf{c}^\intercal \left[S^k (S^r - I)\, \mathbf{x}_0 + \sum_{s=0}^{r+k-1} S^{r+k-1-s} \mathbf{u}[s] - \sum_{s=0}^{k-1} S^{k-1-s} \mathbf{u}[s] \right] \\
&= \mathbf{c}^\intercal \left[S^k \left(S^r - I\right) \mathbf{x}_0 + \sum_{s=0}^{r-1} S^{r+k-1-s} \mathbf{u}[s] + \sum_{s=r}^{r+k-1} S^{r+k-1-s} \mathbf{u}[s] - \sum_{s=0}^{k-1} S^{k-1-s} \mathbf{u}[s] \right] \\
&= \mathbf{c}^\intercal \left[S^k \left(S^r - I\right) \mathbf{x}_0 + \sum_{s=0}^{r-1} S^{r+k-1-s} \mathbf{u}[s] + \sum_{\tilde{s}=0}^{k-1} S^{k-1-\tilde{s}} \mathbf{u}[\tilde{s}] - \sum_{s=0}^{k-1} S^{k-1-s} \mathbf{u}[s] \right] \\
&= \mathbf{c}^\intercal S^k \left[S^r \mathbf{x}_0 + \sum_{s=0}^{r-1} S^{r-1-s} \mathbf{u}[s] - \mathbf{x}_0 \right] \\
&= \mathbf{c}^\intercal S^k (\mathbf{x}[r] - \mathbf{x}_0),
\end{aligned}$$

where we perform the substitution $\tilde{s} = s - r$, noting that since the control input is periodic with a period r, $\mathbf{u}[s] = \mathbf{u}[\tilde{s} + r] = \mathbf{u}[\tilde{s}]$. □

A.1.2. Proof of Lemma 2.2.2

Lemma 2.2.2. *The rank of H in* (2.13) *satisfies*

$$rk(H) = \sum_{i \in \mathcal{I}} \tilde{m}_i,$$

where $\tilde{m}_i$ is defined in (2.11).

Proof. Recall that the set of indices corresponding to observable eigenvalues is defined as $\mathcal{I} = \{i \in \{1, \dots, n\} : \lambda_i \in \mathcal{S}_S\}$, and the total weights corresponding to each unique eigenvalue $\bar{\omega}_i^{(s)} = \sum_{j:\lambda_j = \lambda_i} \omega_j^{(s)}$ from (2.9). Now let $\mathbf{v}_i \coloneqq \left[1, \lambda_i, \lambda_i^2 \dots, \lambda_i^{n-1}\right]$ and $b_k^n \coloneqq \binom{n}{k}$. Considering the (j,k)-th entry of the matrix $\mathbf{v}_i \mathbf{v}_i^\intercal$, notice that

$$\begin{aligned}\frac{d^s}{d\lambda_i^s}[\mathbf{v}_i \mathbf{v}_i^\intercal]_{jk} &= \frac{d^s}{d\lambda_i^s} \lambda_i^{j+k-2} = \frac{(j+k-2)!}{(j+k-2-s)!} \lambda_i^{j+k-2-s} \\ &= s! b_s^{j+k-2} \lambda_i^{j+k-2-s}.\end{aligned}$$

Combining (2.7), (2.12), and (2.13) we thus obtain

$$\begin{aligned}H &= \sum_{i \in \mathcal{I}} \sum_{s=0}^{\hat{m}_i - 1} \bar{\omega}_i^{(s)} \begin{bmatrix} b_s^0 & \cdots & b_s^{n-1} \lambda_i^{n-1-s} \\ \vdots & \ddots & \vdots \\ b_s^{n-1} \lambda_i^{n-1-s} & \cdots & b_s^{2n-2} \lambda_i^{2n-2-s} \end{bmatrix} \\ &= \sum_{i \in \mathcal{I}} \sum_{s=0}^{\hat{m}_i - 1} \frac{\bar{\omega}_i^{(s)}}{s!} \frac{d^s}{d\lambda_i^s} \left(\mathbf{v}_i \mathbf{v}_i^\intercal\right) =: \sum_{i \in \mathcal{I}} H_i.\end{aligned}$$

Notice that for all s and any given $i, j \in \mathcal{I}$ the Hankel matrices $\frac{d^s}{d\lambda_i^s}\left(\mathbf{v}_i \mathbf{v}_i^\intercal\right)$ and $\frac{d^s}{d\lambda_j^s}\left(\mathbf{v}_j \mathbf{v}_j^\intercal\right)$ have orthogonal ranges since the λ_i for $i \in \mathcal{I}$ are distinct, and so $\mathbf{v}_i$ and $\mathbf{v}_j$ are linearly independent.

Let us now examine the ranks of the matrices $D_i^{(s)} \coloneqq \frac{d^s}{d\lambda_i^s}\left(\mathbf{v}_i \mathbf{v}_i^\intercal\right)$ for a particular $i \in \mathcal{I}$. We will proceed via induction on s to show that $\mathrm{rk}(D_i^{(s)}) = s + 1$. For the base case we have $D_i^{(0)} = \mathbf{v}_i \mathbf{v}_i^\intercal$, which clearly has rank 1. Now assume $\mathrm{rk}(D_i^{(s-1)}) = s$. The j-th column of

$D_i^{(s)}$ is of the form

$$\begin{aligned} d_{i,j}^{(s)} &:= s! \left[b_s^{j-1} \lambda_i^{j-1-s}, b_s^j \lambda_i^{j-s}, \ldots, b_s^{j+n-2} \lambda_i^{j+n-2-s} \right]^\intercal \\ &= s! \lambda_i^{j-1-s} \left[b_s^{j-1}, b_s^j \lambda_i, \ldots, b_s^{j+n-2} \lambda_i^{n-1} \right]^\intercal. \end{aligned}$$

Recall that $b_s^k = \binom{k}{s} = 0$ for $k < s$. By the leading-zero structure of $D_i^{(s)}$, wherein the first column has s leading zeros followed by a nonzero value, the second has $s-1$ leading zeros followed by a nonzero value, all the way to the s-th column having a nonzero value in the first component, we can see that $\mathrm{rk}(D_i^{(s)}) \geq s+1$. Now take any collection of $s+2$ columns of $D_i^{(s)}$, and we will show they must be linearly dependent. Via the identity $\binom{j}{s} - \binom{j-k}{s} = \sum_{l=1}^{k} \binom{j-l}{s-1}$, we may write

$$\frac{d_{i,j}^{(s)}}{\lambda_i^{j-1-s}} - \frac{d_{i,j-k}^{(s)}}{\lambda_i^{j-k-1-s}} = \frac{s!}{(s+1)!} \sum_{l=1}^{k} \frac{d_{i,j-l}^{(s-1)}}{\lambda_i^{j-l-1-s}}.$$

In other words, we may express the j-th and $(j-k)$-th columns of $D_i^{(s)}$ as a linear combination of exactly k columns from $D_i^{(s-1)}$. Since we have a collection of $s+2$ columns of $D_i^{(s)}$, we will need at least $s+1$ unique columns of $D_i^{(s-1)}$ to express linear combinations of our entire collection (in the case where the columns are sequential), but may need more. However, the rank of $D_i^{(s-1)}$ is s, so any collection of at least $s+1$ unique columns of $D_i^{(s-1)}$ must be linearly dependent; hence, our collection of $s+2$ columns of $D_i^{(s)}$ must be linearly dependent. Thus, $\mathrm{rk}\left(D_i^{(s)}\right) = s+1$.

We will now examine the ranges of the matrices $D_i^{(s)}$ for a particular $i \in \mathcal{I}$. For $1 \leq s < j-k$ and $1 \leq k < j$ we have the identity $\binom{j}{s} - \binom{j-k}{s} = \binom{j-k}{s-k}$. Thus, $d_{i,j-k}^{(s-k)} = \frac{(s-k)!}{s!} \left[d_{i,j}^{(s)} - \lambda_i^k d_{i,j-k}^{(s)} \right]$. In other words, we may write the $(j-k)$-th column of $D_i^{(s-k)}$ as a linear combination of the j-th and $(j-k)$-th columns of $D_i^{(s)}$ for $1 \leq k \leq s$ and $k < j \leq n$. Recall that $\mathrm{rk}(D_i^{(s-k)}) = s-k+1$. Since we may write the first $s-k+1$ columns of $D_i^{(s-k)}$ as linear combinations of the columns of $D_i^{(s)}$, the same is true for all columns of $D_i^{(s-k)}$.

Thus $\text{range}(D_i^{(s-k)}) \subseteq \text{range}(D_i^{(s)})$ for $1 \leq k \leq s \leq \tilde{m}_i - 1$. Hence,

$$\text{range}\,(H_i) = \text{range}\left(\sum_{s=0}^{\tilde{m}_i-1} \frac{\bar{\omega}_i^{(s)}}{s!} D_i^{(s)}\right) = \text{range}(D_i^{(\tilde{m}_i-1)}) \quad \Rightarrow \quad \text{rk}(H_i) = \text{rk}(D_i^{(\tilde{m}_i-1)}),$$

where $\tilde{m}_i$, as defined in (2.11), is the largest index with a nonzero total weight. Thus, the rank of H_i is simply the largest s for which $\bar{\omega}_i^{(s-1)} \neq 0$, i.e., $\text{rk}(H_i) = \tilde{m}_i$. Then, since for all s and any $i \neq j \in \mathcal{I}$ the matrices $D_i^{(s)}$ and $D_j^{(s)}$ have orthogonal ranges, we have that $\text{rg}(H) = \text{rg}(\sum_{i\in\mathcal{I}} H_i) = \oplus_{i\in\mathcal{I}}\text{rg}(H_i)$, and hence $\text{rk}(H) = \sum_{i\in\mathcal{I}} \text{rk}(H_i)$. Therefore, the rank of H is equal to the sum of the sizes of the largest observable Jordan blocks for each unique eigenvalue, which is $\sum_{i\in\mathcal{I}} \tilde{m}_i$. □

A.1.3. Proof of Theorem 2.2.1

Theorem 2.2.1. *Given the sequence of observations $(y\,[k])_{k=0}^{3n-1}$ from the discrete-time system in (2.4), consider the matrix H from (2.13) and denote its rank by r. For periodic control inputs $\mathbf{u}[k]$ with period r and the differences of measurements $\Delta y[r+k] = y[r+k] - y[k]$, the observable eigenvalues are roots of the polynomial*

$$p_S\,(x) = x^r + \alpha_{r-1}x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

where the coefficients $\alpha_0, \ldots, \alpha_{r-1}$ are given by

$$\begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{r-1} \end{bmatrix} = -\begin{bmatrix} \Delta y[r] & \cdots & \Delta y[2r-1] \\ \vdots & \ddots & \vdots \\ \Delta y[2r-1] & \cdots & \Delta y[3r-2]] \end{bmatrix}^{-1} \begin{bmatrix} \Delta y[2r] \\ \vdots \\ \Delta y[3r-1] \end{bmatrix}.$$

Moreover, $\lambda_i \in \mathcal{S}_S$ is a root of $p_S(x)$ with multiplicity $\tilde{m}_i$.

Proof. By definition, we know that at most we may recover all eigenvalues corresponding to observable eigenmodes, i.e., $\lambda_i \in \mathcal{S}_S$. As before, let $\mathcal{I} = \{i \in \{1, \ldots, n\} : \lambda_i \in \mathcal{S}_S\}$. By Lemma 2.2.2, we know that $\text{rk}\,(H) = \sum_{i\in\mathcal{I}} \tilde{m}_i$, which we denote by r. Define the following

polynomial:

$$p_S(x) \coloneqq \prod_{i \in \mathcal{I}} (x - \lambda_i)^{\tilde{m}_i} = x^r + \alpha_{r-1} x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

where $\tilde{m}_i$ is defined in (2.11). Notice that, since the eigenvalues are unknown, the coefficients of the polynomial are also unknown.

Let us calculate $p_S(J_i)$ for each $i \in \mathcal{I}$. Recall that there may be multiple Jordan blocks associated with a single eigenvalue, and that the Jordan block J_l is of size $m_l \times m_l$. First consider the case that there exists some Jordan block i such that $m_i = \tilde{m}_i$. By Cayley-Hamilton theorem we know $(J_i - \lambda_i I_{m_i})^{\tilde{m}_i} = \mathbf{0}_{m_i \times m_i}$, and so

$$p_S(J_i) = J_i^r + \alpha_{r-1} J_i^{r-1} + \cdots + \alpha_1 J_1 + \alpha_0 = \mathbf{0}_{m_i \times m_i}.$$

Note from (2.7) that each upper diagonal of J_i^r contains the same values. For ease of exposition, define $b_k^n = \binom{n}{k}$. Hence, since J_i^r is of size $m_i \times m_i$, we in fact have m_i separate equations (one per upper diagonal) of the form

$$b_s^r \lambda_i^{r-s} + \alpha_{r-1} b_s^{r-1} \lambda_i^{(r-s)-1} + \cdots + \alpha_{s+1} b_s^{s+1} \lambda_i + \alpha_s = 0,$$

for $s \in \{0, \ldots, m_i - 1\}$. If there is no Jordan block i such that $m_i = \tilde{m}_i$, then pick one such that $m_i > \tilde{m}_i$, and consider the first $\tilde{m}_i$ upper diagonals of $(J_i - \lambda_i I_{m_i})^{\tilde{m}_i}$, which will be zero. Multiplying the equations above by the corresponding total weights $\bar{\omega}_i^{(s)}$, some of which may be zero, we obtain for $s \in \{0, \ldots, \tilde{m}_i - 1\}$

$$\bar{\omega}_i^{(s)} \left(b_s^r \lambda_i^{r-s} + \alpha_{r-1} b_s^{r-1} \lambda_i^{(r-s)-1} + \cdots + \alpha_{s+1} b_s^{s+1} \lambda_i + \alpha_s \right) = 0.$$

Summing these equations, noting $b_s^r = \binom{r}{s} = 0$ for $r < s$,

$$\sum_{s=0}^{\tilde{m}_i - 1} \bar{\omega}_i^{(s)} \left(b_s^r \lambda_i^{r-s} + \alpha_{r-1} b_s^{r-1} \lambda_i^{(r-s)-1} + \cdots + \alpha_{s+1} b_s^{s+1} \lambda_i + \alpha_s \right)$$

$$= \sum_{s=0}^{\tilde{m}_i-1} \bar{\omega}_i^{(s)} \sum_{l=0}^{r} \alpha_l \binom{l}{s} \lambda_i^{l-s} = 0.$$

Now, let us sum over all eigenvalues $\lambda_i \in \mathcal{S}_S$:

$$\begin{aligned}
&\sum_{i\in\mathcal{I}} \sum_{s=0}^{\tilde{m}_i-1} \bar{\omega}_i^{(s)} \sum_{l=0}^{r} \alpha_l \binom{l}{s} \lambda_i^{l-s} \\
&= \sum_{l=0}^{r} \alpha_l \sum_{i\in\mathcal{I}} \sum_{s=0}^{\tilde{m}_i-1} \bar{\omega}_i^{(s)} \binom{l}{s} \lambda_i^{l-s} = \sum_{l=0}^{r} \alpha_l \Delta y[r+l] = 0,
\end{aligned}$$

by definition of the differences of observations $\Delta y[r+l]$ from (2.12) with control inputs with period r. Now, for $k \in \{1, \ldots, r-1\}$, let us examine the equations

$$J^k p_{\mathcal{G}}(J) = J^{r+k} + \alpha_{r-1} J^{r+k-1} + \cdots + \alpha_1 J^{k+1} + \alpha_0 J^k.$$

Repeating the same process from above, we obtain for $k \in \{1, \ldots, r-1\}$

$$\begin{aligned}
&\sum_{i\in\mathcal{I}} \sum_{s=0}^{\tilde{m}_i-1} \bar{\omega}_i^{(s)} \sum_{l=0}^{r} \alpha_l \binom{l+k}{s} \lambda_i^{l+k-s} \\
&= \sum_{l=0}^{r} \alpha_l \sum_{i\in\mathcal{I}} \sum_{s=0}^{\tilde{m}_i-1} \bar{\omega}_i^{(s)} \binom{l+k}{s} \lambda_i^{l+k-s} \\
&= \sum_{l=0}^{r} \alpha_l \Delta y[r+l+k] = 0.
\end{aligned}$$

In summary, we have r equations of the form

$$\Delta y[2r+k] + \alpha_{r-1} \Delta y[2r-1+k] + \cdots + \alpha_0 \Delta y[r+k] = 0,$$

where $k \in \{0, \ldots, r-1\}$. In matrix form,

$$\begin{bmatrix} \Delta y[r] & \cdots & \Delta y[2r-1] \\ \vdots & \ddots & \vdots \\ \Delta y[2r-1] & \cdots & \Delta y[3r-2] \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{r-1} \end{bmatrix} = - \begin{bmatrix} \Delta y[2r] \\ \vdots \\ \Delta y[3r] \end{bmatrix}.$$

By Lemma 2.2.2 we know $\mathrm{rk}(H) = r$ and hence we may find the values of the coefficients $\alpha_0, \ldots, \alpha_{r-1}$ by a simple matrix inversion. Using these coefficients we can compute the roots of p_S to recover the eigenvalues of S that are in the set $\mathcal{S}_S$, i.e., those eigenvalues λ_i corresponding to the observable eigenmodes of the dynamics. Moreover, the multiplicity of the root λ_i will be $\tilde{m}_i$; hence, we recover λ_i with multiplicity of exactly $\tilde{m}_i$. □

A.1.4. Proof of Theorem 2.2.2

Theorem 2.2.2. *Given the sequence of observations $(y[k])_{k=0}^{2n-1}$ from the system in (2.14), consider the Hankel matrix H defined in (2.13) with $\Delta y[k] = y[k]$ and denote its rank by r. The weighted sums of eigenvalues $\sigma_k \coloneqq \sum_{i=1}^{d} \sum_{s=0}^{m_i-1} \omega_i^{(s)} \binom{k}{s} \lambda_i^{k-s}$ satisfy the following equality:*

$$\begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_{2r-1} \end{bmatrix} = \begin{bmatrix} b_0^0 \nu_0 & 0 & \cdots & 0 \\ b_0^1 \nu_1 & b_1^1 \nu_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_0^{2r-1} \nu_{2r-1} & b_1^{2r-1} \nu_{2r-2} & \cdots & b_{2r-1}^{2r-1} \nu_0 \end{bmatrix}^{-1} \begin{bmatrix} y[0] \\ y[1] \\ \vdots \\ y[2r-1] \end{bmatrix}$$

where $\nu_{k-s} \coloneqq \boldsymbol{\gamma}^\intercal A^{k-s} \boldsymbol{\beta}$, $b_s^k \coloneqq \binom{k}{s}$, and the matrix is invertible when $\boldsymbol{\gamma}^\intercal \boldsymbol{\beta} \neq 0$. Then, the observable eigenvalues of S are roots of the polynomial

$$p_S(x) = x^r + \alpha_{r-1} x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

where the coefficients $\alpha_0, \ldots, \alpha_{r-1}$ satisfy

$$\begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_{r-1} \end{bmatrix} = - \begin{bmatrix} \sigma_0 & \cdots & \sigma_{r-1} \\ \vdots & \ddots & \vdots \\ \sigma_{r-1} & \cdots & \sigma_{2r-2} \end{bmatrix}^{-1} \begin{bmatrix} \sigma_r \\ \vdots \\ \sigma_{2r-1} \end{bmatrix}.$$

Proof. Considering the Jordan decomposition $S = VJV^{-1}$,

$$\begin{aligned}(I_n \otimes A + S \otimes I_d)^k &= \left[(V \otimes I_d)\,(I_n \otimes A + J \otimes I_d)\,\left(V^{-1} \otimes I_d\right)\right]^k \\ &= (V \otimes I_d)\,(I_n \otimes A + J \otimes I_d)^k\,\left(V^{-1} \otimes I_d\right) \\ &= (V \otimes I_d)\left[\sum_{s=0}^{k} \binom{k}{s} \left(I_n \otimes A^{k-s}\right)(J^s \otimes I_d)\right]\left(V^{-1} \otimes I_d\right).\end{aligned}$$

Thus,

$$\begin{aligned}y\,[k] &= (\mathbf{c} \otimes \boldsymbol{\gamma})^\intercal\,(I_n \otimes A + S \otimes I_d)^k\,(\mathbf{x}_0 \otimes \boldsymbol{\beta}) \\ &= \sum_{s=0}^{k} \binom{k}{s} (\mathbf{c}^\intercal V \otimes \boldsymbol{\gamma}^\intercal)\Big(I_n \otimes A^{k-s}\Big)(J^s \otimes I_d)\big(V^{-1}\mathbf{x}_0 \otimes \boldsymbol{\beta}\big) \\ &= \sum_{s=0}^{k} \binom{k}{s} \left(\mathbf{c}^\intercal V J^s V^{-1}\mathbf{x}_0\right)\left(\boldsymbol{\gamma}^\intercal A^{k-s}\boldsymbol{\beta}\right) \\ &= \sum_{s=0}^{k} \binom{k}{s} \nu_{k-s} \sum_{i=1}^{d} \sum_{s=0}^{m_i-1} \omega_i^{(s)} \binom{k}{s} \lambda_i^{k-s} \\ &= \sum_{s=0}^{k} \binom{k}{s} \nu_{k-s}\sigma_s, \end{aligned} \tag{A.1}$$

where $\sigma_s \coloneqq \sum_{i=1}^{d} \sum_{s=0}^{m_i-1} \omega_i^{(s)} \binom{k}{s} \lambda_i^{k-s}$ and $\nu_{k-s} \coloneqq \boldsymbol{\gamma}^\intercal A^{k-s}\boldsymbol{\beta}$. From the sequence $(y\,[k])_{k=0}^{2n-1}$, we obtain a lower triangular system of linear equations that can be solved to find the sequence $(\sigma_k)_{k=0}^{2n-1}$. Specifically, if we collect $2r$ observations, with $b_s^k = \binom{k}{s}$, we have that (A.1) for $k = 0, \ldots, 2r-1$ results in

$$\begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_{2r-1} \end{bmatrix} = \begin{bmatrix} b_0^0\nu_0 & 0 & \cdots & 0 \\ b_0^1\nu_1 & b_1^1\nu_0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ b_0^{2r-1}\nu_{2r-1} & b_1^{2r-1}\nu_{2r-2} & \cdots & b_{2r-1}^{2r-1}\nu_0 \end{bmatrix} \begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \vdots \\ \sigma_{2r-1} \end{bmatrix}$$

As long as $\nu_0 = \boldsymbol{\gamma}^\intercal\boldsymbol{\beta} \neq 0$, the above matrix is full-rank. We may then recover the values σ_s by a simple inversion, and apply Theorem 2.2.1 to find the eigenvalues of S. □

A.1.5. Proof of Corollary 2.3.1

Corollary 2.3.1. *Given the sequence of observations* $(y_k)_{k=0}^{3n-1}$ *from the continuous-time system in* (2.15) *with fixed sampling rate* $\tau > 0$ *and control inputs with period* $r\tau$*, the observable eigenvalues of the graph matrix* S *may be obtained via*

$$\lambda_i = \log(\eta_i)/\tau,$$

where η_i *are the roots of the polynomial*

$$p_S(x) = x^r + \alpha_{r-1}x^{r-1} + \cdots + \alpha_1 x + \alpha_0,$$

whose coefficients $\alpha_0, \ldots, \alpha_{r-1}$ *are obtained from the observations* $(y_k)_{k=0}^{3n-1}$ *as in Theorem 2.2.1.*

Proof. With sampling period $\tau > 0$, recall from (2.17) that $\Delta y_{r+k} = y((r+k)\tau) - y(k\tau)$. Then, similarly to Lemma 2.2.1, we may rewrite Δy_{r+k} as

$$\begin{aligned}
&\mathbf{c}^\intercal e^{Sk\tau}(e^{Sr\tau}\mathbf{x}_0 - \mathbf{x}_0) + \int_0^{r\tau} \mathbf{c}^\intercal e^{S((r+k)\tau - s)}\mathbf{u}(s)ds \\
&+ \int_{r\tau}^{(r+k)\tau} \mathbf{c}^\intercal e^{S((r+k)\tau - s)}\mathbf{u}(s)ds - \int_0^{k\tau} \mathbf{c}^\intercal e^{S(k\tau - s)}\mathbf{u}(s)ds.
\end{aligned}$$

Recall that, with period $r\tau$, $u(t + r\tau) = u(t)$. Via the substitution $\tilde{s} = s - r\tau$ we have that $u(s) = u(\tilde{s} + r\tau) = u(\tilde{s})$, thus

$$\begin{aligned}
\int_{r\tau}^{(r+k)\tau} \mathbf{c}^\intercal e^{S((r+k)\tau - s)}\mathbf{u}(s)ds &= \int_0^{k\tau} \mathbf{c}^\intercal e^{S(k\tau - \tilde{s})}\mathbf{u}(\tilde{s})d\tilde{s} \\
\Rightarrow \Delta y_{r+k} &= \mathbf{c}^\intercal e^{Sk\tau}\left(e^{Sr\tau}\mathbf{x}_0 + \int_0^{r\tau} e^{S(r\tau - s)}\mathbf{u}(s)ds - \mathbf{x}_0\right) \qquad \text{(A.2)} \\
&= \mathbf{c}^\intercal e^{Sk\tau}\left(\mathbf{x}(r\tau) - \mathbf{x}_0\right).
\end{aligned}$$

Analogously to Theorem 2.2.1 we define the polynomial

$$p_S(x) \coloneqq \prod_{i \in \mathcal{I}} \left(x - e^{\lambda_i \tau} \right)^{\tilde{m}_i} = x^r + \cdots + \alpha_1 x + \alpha_0.$$

Examining (2.16) suggests the substitution $b_k^n = \frac{n^k}{k!}$ in the proof of Theorem 2.2.1, whose application yields the values $\eta_i \coloneqq e^{\lambda_i \tau}$. Then, the eigenvalues corresponding to observable eigenmodes may be obtained via $\lambda_i = \log(\eta_i)/\tau$. □

A.2. Proofs of Chapter 4

A.2.1. Transferability bounds via spectral similarity

Proposition 4.2.1. *For ϵ-spectrally similar symmetric GSOs S and $\tilde{S}$ and an integral Lipschitz filter with constant C, the operator difference modulo permutation between the filters $H(S)$ and $H(\tilde{S})$ satisfies*

$$\left\|H(\tilde{S}) - H(S)\right\|_{\mathcal{P}} \leq C\epsilon + \mathcal{O}(\epsilon^2).$$

Moreover, if the filter applies only one shift operation with bias so $H(S) = h_0 I + h_1 S$, then $\left\|H(\tilde{S}) - H(S)\right\|_{\mathcal{P}} \leq C\epsilon$.

Proof. By spectral similarity, we have that

$$(1-\epsilon)S \preceq \tilde{S} \preceq (1+\epsilon)S \tag{A.3}$$

$$\Rightarrow \ (1-\epsilon)\mathbf{x}^{\intercal}S\mathbf{x} \leq \mathbf{x}^{\intercal}\tilde{S}\mathbf{x} \leq (1+\epsilon)\mathbf{x}^{\intercal}S\mathbf{x} \quad \forall \mathbf{x} \in \mathbb{R}^n. \tag{A.4}$$

Hence,

$$\begin{aligned} H(\tilde{S}) - H(S) &= \sum_{k=0}^{\infty} h_k(\tilde{S}^k - S^k) = \sum_{k=1}^{\infty} h_k(\tilde{S}^k - S^k) \\ &\preceq \sum_{k=1}^{\infty} h_k(((1+\epsilon)S)^k - S^k). \end{aligned}$$

For symmetric matrices $\|A\|_{op} \leq \|B\|_{op}$ if $A \preceq B$. Thus, using the first-order expansion $((1+\epsilon)S)^k = (1+k\epsilon)S^k + O(\epsilon^2)$, since graph filters are permutation equivariant we have

$$\begin{aligned} \left\|H(\tilde{S}) - H(S)\right\|_{\mathcal{P}} &= \left\|H(\tilde{S}) - H(S)\right\|_{op} \\ &\leq \left\|\sum_{k=1}^{\infty} h_k(((1+\epsilon)S)^k - S^k)\right\|_{op} \end{aligned}$$

$$= \left\| \sum_{k=1}^{\infty} h_k((1+k\epsilon)S^k - S^k) + O(\epsilon^2) \right\|_{op}$$
$$\leq \left\| \sum_{k=1}^{\infty} h_k k\epsilon S^k \right\|_{op} + \mathcal{O}(\epsilon^2).$$

Next, notice that the derivative of the frequency response is $\tilde{h}'(\lambda) = \sum_{k=1}^{\infty} h_k k\lambda^{k-1}$. Thus, using the graph Fourier representation of $\mathbf{x}$,

$$\begin{aligned}
\sum_{k=1}^{\infty} h_k k\epsilon S^k \mathbf{x} &= \epsilon \sum_{k=1}^{\infty} h_k k S^k \left(\sum_{i=1}^{n} \tilde{x}_i \mathbf{v}_i \right) \\
&= \epsilon \sum_{i=1}^{n} \tilde{x}_i \sum_{k=1}^{\infty} h_k k S^k \mathbf{v}_i \\
&= \epsilon \sum_{i=1}^{n} \tilde{x}_i \sum_{k=1}^{\infty} h_k k \lambda_i^k \mathbf{v}_i \\
&= \epsilon \sum_{i=1}^{n} \tilde{x}_i \lambda_i \tilde{h}'(\lambda_i) \mathbf{v}_i.
\end{aligned}$$

By the integral Lipschitz assumption of the filter, $\lambda\tilde{h}'(\lambda) \leq C$. Moreover, since the signal is assumed to have a unit norm, $\|\mathbf{x}\|_2 = \|\tilde{\mathbf{x}}\|_2 = 1$. Thus, by orthonormality of the $\mathbf{v}_i$,

$$\begin{aligned}
\left\| \sum_{k=1}^{\infty} h_k k\epsilon S^k \mathbf{x} \right\|_2^2 &= \left\| \epsilon \sum_{i=1}^{n} \tilde{x}_i \lambda_i \tilde{h}'(\lambda_i) \mathbf{v}_i \right\|_2^2 \\
&= \epsilon^2 \sum_{i=1}^{n} (\tilde{x}_i \lambda_i \tilde{h}'(\lambda_i))^2 \\
&\leq (C\epsilon)^2 \sum_{i=1}^{n} \tilde{x}_i^2 = (C\epsilon)^2.
\end{aligned}$$

Finally, we thus have

$$\left\| H(\tilde{S}) - H(S) \right\|_{\mathcal{P}} \leq \left\| \sum_{k=1}^{\infty} h_k k\epsilon S^k \right\|_{op} + \mathcal{O}(\epsilon^2) \leq C\epsilon + \mathcal{O}(\epsilon^2),$$

as desired. Moreover, note that if we apply only one shift operation with a bias term, i.e., $H(S) = h_0 I + h_1 S$, then the first-order expansion performed earlier is exact and

$$\left\|H(\tilde{S}) - H(S)\right\|_{\mathcal{P}} \leq C\epsilon. \qquad \square$$

A.2.2. Transferability of GNNs

Theorem 4.2.1. *Given ϵ-spectrally similar GSOs S and $\tilde{S}$ and a GNN $\Phi(\cdot; S, \Theta)$ with normalized Lipschitz nonlinearities and L layers with f features, each with filters that have unit operator norm and are C-integral Lipschitz, then*

$$\left\|\Phi(\cdot; S, \Theta) - \Phi(\cdot; \tilde{S}, \Theta)\right\|_{\mathcal{P}} \leq CLf^{L}\epsilon + \mathcal{O}(\epsilon^2).$$

Proof. We wish to bound the quantity $||\Phi(\cdot; S, \Theta) - \Phi(\cdot; \tilde{S}, \Theta)||_{\mathcal{P}}$, and proceed similarly to [7, Theorem 4]. At an arbitrary layer $l \in \{1, \ldots, L\}$ of $\Phi(\cdot; S, \Theta)$ with f_l features (where $l = L$ is the output layer), the (possibly hidden) node signals are some $\{\mathbf{x}_l^f\}_{f=1}^{f_l}$, where $\mathbf{x}_l^f \in \mathbb{R}^n$. Similarly, the node signals in the GNN $\Phi(\cdot; \tilde{S}, \Theta)$ at layer l are denoted by $\{\tilde{\mathbf{x}}_l^f\}_{f=1}^{f_l}$. We thus wish to bound quantities of the form $||\mathbf{x}_l^f - \tilde{\mathbf{x}}_l^f||_2$, i.e.,

$$\left\|\sigma\left(\sum_{g=1}^{f_{l-1}} H_l^{fg}(S)\mathbf{x}_{l-1}^g\right) - \sigma\left(\sum_{g=1}^{f_{l-1}} H_l^{fg}(\tilde{S})\tilde{\mathbf{x}}_{l-1}^g\right)\right\|_2. \tag{A.5}$$

Since the nonlinearities are assumed to be normalized Lipschitz, $|\sigma(x) - \sigma(y)| \leq |x - y|$. So, by the triangle inequality,

$$\begin{aligned}
\left\|\mathbf{x}_l^f - \tilde{\mathbf{x}}_l^f\right\|_2 &\leq \sum_{g=1}^{f_{l-1}} \left\|H_l^{fg}(S)\mathbf{x}_{l-1}^g - H_l^{fg}(\tilde{S})\tilde{\mathbf{x}}_{l-1}^g\right\|_2 \\
&= \sum_{g=1}^{f_{l-1}} \left\|H_l^{fg}(S)\big(\mathbf{x}_{l-1}^g - \tilde{\mathbf{x}}_{l-1}^g\big) + \Big(H_l^{fg}(S) - H_l^{fg}(\tilde{S})\Big)\tilde{\mathbf{x}}_{l-1}^g\right\|_2 \\
&\leq \sum_{g=1}^{f_{l-1}} \left\|H_l^{fg}(S)\right\|_{op} \left\|\mathbf{x}_{l-1}^g - \tilde{\mathbf{x}}_{l-1}^g\right\|_2 + \left\|H_l^{fg}(S) - H_l^{fg}(\tilde{S})\right\|_{op} \left\|\tilde{\mathbf{x}}_{l-1}^g\right\|_2.
\end{aligned}$$

Notice similarly that

$$\left\|\mathbf{x}_l^{g_l}\right\|_2 = \left\|\sigma\left(\sum_{g_{l-1}=1}^{f_{l-1}} H_l^{g_l g_{l-1}}(S)\mathbf{x}_{l-1}^{g_{l-1}}\right)\right\|_2$$

$$
\begin{aligned}
&\leq \sum_{g_{l-1}=1}^{f_{l-1}} \left\| H_l^{g_l g_{l-1}}(S) \mathbf{x}_{l-1}^{g_{l-1}} \right\|_2 \\
&\leq \sum_{g_{l-1}=1}^{f_{l-1}} \left\| H_l^{g_l g_{l-1}}(S) \right\|_{op} \left\| \mathbf{x}_{l-1}^{g_{l-1}} \right\|_2 \\
&\leq \sum_{g_{l-1}=1}^{f_{l-1}} \cdots \sum_{g_0=1}^{f_0} \left\| \mathbf{x}_0^{g_0} \right\|_2 \prod_{s=1}^{l} \left\| H_s^{g_s g_{s-1}}(S) \right\|_{op}.
\end{aligned}
$$

Since the GSOs S and $\tilde{S}$ are ϵ-spectrally similar, by Proposition 4.2.1 we have $||H_1^{g_l g_{l-1}}(S) - H_1^{g_l g_{l-1}}(\tilde{S})||_{op} \leq C_l^{g_l g_{l-1}} \epsilon + \mathcal{O}(\epsilon^2)$ for all g_l, g_{l-1}, where $C_l^{g_l g_{l-1}}$ is the integral Lipschitz constant of the filter in the l-th layer for the g_{l-1}-th input feature and g_l-th output feature. Combining these results,

$$
\begin{aligned}
\left\| \mathbf{x}_l^{g_l} - \tilde{\mathbf{x}}_l^{g_l} \right\|_2 &\leq \sum_{g_{l-1}=1}^{f_{l-1}} \left\| H_l^{g_l g_{l-1}}(S) \right\|_{op} \left\| \mathbf{x}_{l-1}^{g_{l-1}} - \tilde{\mathbf{x}}_{l-1}^{g_{l-1}} \right\|_2 + (C_l^{g_l g_{l-1}} \epsilon + \mathcal{O}(\epsilon^2)) \left\| \tilde{\mathbf{x}}_{l-1}^{g} \right\|_2 \\
&\leq \sum_{g_{l-1}=1}^{f_{l-1}} \left\| H_l^{g_l g_{l-1}}(S) \right\|_{op} \left\| \mathbf{x}_{l-1}^{g_{l-1}} - \tilde{\mathbf{x}}_{l-1}^{g_{l-1}} \right\|_2 \\
&\quad + \sum_{g_{l-2}=1}^{f_{l-2}} \cdots \sum_{g_0=1}^{f_0} \left\| \tilde{\mathbf{x}}_0^{g_0} \right\|_2 C_l^{g_l g_{l-1}} \epsilon \prod_{s=1}^{l-1} \left\| H_s^{g_s g_{s-1}}(S) \right\|_{op} + \mathcal{O}(\epsilon^2).
\end{aligned}
$$

This establishes a recurrence relation for $\left\| \mathbf{x}_l^{g_l} - \tilde{\mathbf{x}}_l^{g_l} \right\|_2$. Notice that for $l = 1$ and any $g_1 \in \{1, \ldots, f_1\}$,

$$
\left\| \mathbf{x}_1^{g_1} - \tilde{\mathbf{x}}_1^{g_1} \right\|_2 \leq \sum_{g_0=1}^{f_0} C_1^{g_1 g_0} \epsilon \left\| \mathbf{x}_0^{g_0} \right\|_2 + \mathcal{O}(\epsilon^2),
$$

since the input signals to both GNNs are the same and, hence, $\left\| \mathbf{x}_0^{g_0} \right\|_2 = \left\| \tilde{\mathbf{x}}_0^{g_0} \right\|_2$ and $\left\| \mathbf{x}_0^{g_0} - \tilde{\mathbf{x}}_0^{g_0} \right\|_2 = 0$ for all $g_0 \in \{1, \ldots, f_0\}$. Thus we may solve the recurrence relation above to obtain

$$
\left\| \mathbf{x}_l^{g_l} - \tilde{\mathbf{x}}_l^{g_l} \right\|_2 \leq \sum_{g_{l-1}=1}^{f_{l-1}} \cdots \sum_{g_0=1}^{f_0} \left\| \mathbf{x}_0^{g_0} \right\|_2 \sum_{s=1}^{l} C_s^{g_s g_{s-1}} \epsilon \prod_{\substack{t=1 \\ t \neq s}}^{l} \left\| H_t^{g_t g_{t-1}}(S) \right\|_{op} + \mathcal{O}(\epsilon^2).
$$

This inequality makes explicit the effects of the integral Lipschitz constants $C_l^{g_l g_{l-1}}$ as well as the operator norms $\left\|H_t^{g_t g_{t-1}}(S)\right\|_{op}$ of each filter, which are functions of both the (learned) filter coefficients and the spectrum of the GSO S used for training. Both quantities control the amplification of the input signals; if they are smaller, the magnitude of the difference in the signals at layer l is smaller. The output signals of the GNNs are the quantities $\{\mathbf{x}_L^f\}_{f=1}^{f_L}$ and $\{\tilde{\mathbf{x}}_L^f\}_{f=1}^{f_L}$. Hence, since we wish to bound the difference in the output of the GNNs, for an arbitrary input signal $\mathbf{x}_0 = \{\mathbf{x}_0^f\}_{f=1}^{f_0}$, we have

$$\begin{aligned}\left\|\Phi(\mathbf{x}_0; S, \Theta) - \Phi(\mathbf{x}_0; \tilde{S}, \Theta)\right\|_2 &= \sum_{g_L=0}^{f_L} \left\|\mathbf{x}_L^{g_L} - \tilde{\mathbf{x}}_L^{g_L}\right\|_2 \\ &\leq \sum_{g_L=0}^{f_L} \cdots \sum_{g_0=1}^{f_0} \|\mathbf{x}_0^{g_0}\|_2 \sum_{s=1}^{L} C_s^{g_s g_{s-1}} \epsilon \prod_{\substack{t=1 \\ t\neq s}}^{L} \left\|H_t^{g_t g_{t-1}}(S)\right\|_{op} + \mathcal{O}(\epsilon^2).\end{aligned}$$

If the filter responses are bounded, i.e., $\left\|H_t^{g_t g_{t-1}}(S)\right\|_{op} \leq c$ (which can be achieved via normalization during training), all filters share an integral Lipschitz constant C (which is possible by adding a penalty to the training loss), and the input signals have unit norm (via normalization before training), then

$$\left\|\Phi(\mathbf{x}_0; S, \Theta) - \Phi(\mathbf{x}_0; \tilde{S}, \Theta)\right\|_2 \leq CLc^{L-1} \prod_{s=0}^{L} f_s \epsilon + \mathcal{O}(\epsilon^2).$$

The constant c can be understood as controlling the amplification or contraction of the input and hidden signals, while the number of features f_s and layers L determine how many filters are stacked together, each obeying the bound from Proposition 4.2.1, which depends on the integral Lipschitz constant C and the spectral similarity coefficient ϵ. The desired result is achieved by setting $c = 1$ and $f_s = f$ for all $s \in \{0, \ldots, L\}$. □

A.2.3. Spectral similarity under relative perturbations

Proposition 4.5.1. *Given a symmetric, positive semi-definite GSO S and its symmetric, positive semi-definite relatively perturbed version $\tilde{S} = S + \frac{1}{2}(SE + ES)$, where E is diagonalizable with $\|E\|_{op} \leq \delta$, the matrices are δ-spectrally similar.*

Proof. Recall by definition of ϵ-spectral similarity (4.17) that

$$(1-\epsilon)S \preceq \tilde{S} \preceq (1+\epsilon)S \Leftrightarrow (1-\epsilon)\mathbf{x}^\intercal S\mathbf{x} \leq \mathbf{x}^\intercal \tilde{S}\mathbf{x} \leq (1+\epsilon)\mathbf{x}^\intercal S\mathbf{x} \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

Given some perturbation matrix E, we wish to determine the coefficient of spectral similarity, ϵ. Let us begin with the latter inequality,

$$\begin{aligned}
\tilde{S} \preceq (1+\epsilon)S &\Leftrightarrow S + \frac{1}{2}(SE+ES) \preceq (1+\epsilon)S \\
&\Leftrightarrow \epsilon S - \frac{1}{2}(SE+ES) \succeq 0 \\
&\Leftrightarrow \mathbf{x}^\intercal \left(\epsilon S - \frac{1}{2}(SE+ES)\right)\mathbf{x} \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n.
\end{aligned}$$

Note that, since S is PSD, we may always choose ϵ to be large enough that $\epsilon S - \frac{1}{2}(SE+ES)$ is PSD. Let us diagonalize $E = UMU^\intercal$, and recall that $\|E\|_{op} = |\lambda_{max}(E)| \leq \delta$. Hence,

$$\begin{aligned}
\mathbf{x}^\intercal \left(\epsilon S - \frac{1}{2}(SE+ES)\right)\mathbf{x} &= \mathbf{x}^\intercal S((\epsilon/2)I - E/2)\mathbf{x} + \mathbf{x}^\intercal((\epsilon/2)I - E/2)S\mathbf{x} \\
&= \mathbf{x}^\intercal SU((\epsilon/2)I - M/2)U^\intercal\mathbf{x} + \mathbf{x}^\intercal U((\epsilon/2)I - M/2)U^\intercal S\mathbf{x} \\
&\geq \frac{1}{2}\mathbf{x}^\intercal SU(\epsilon I - \delta I)U^\intercal\mathbf{x} + \frac{1}{2}\mathbf{x}^\intercal U(\epsilon I - \delta I)U^\intercal S\mathbf{x} \\
&= \frac{1}{2}(\epsilon-\delta)\mathbf{x}^\intercal SUU^\intercal\mathbf{x} + \frac{1}{2}(\epsilon-\delta)\mathbf{x}^\intercal UU^\intercal S\mathbf{x} \\
&= (\epsilon-\delta)\mathbf{x}^\intercal S\mathbf{x},
\end{aligned}$$

with equality above when $E = \delta I$. Since S is PSD, $\mathbf{x}^\intercal S\mathbf{x} \geq 0 \; \forall \mathbf{x} \in \mathbb{R}^n$. Thus, if $\epsilon \geq \delta$, then $\tilde{S} \preceq (1+\epsilon)S$. Moreover, when $E = \delta I$ then $\tilde{S} = (1+\delta)S = (1+\epsilon)S$, and the δ-similarity is tight.

Next, notice similarly that

$$\begin{aligned}
(1-\epsilon)S \preceq \tilde{S} &\Leftrightarrow \epsilon S + \frac{1}{2}(SE+ES) \succeq 0 \\
&\Leftrightarrow \mathbf{x}^\intercal \left(\epsilon S + \frac{1}{2}(SE+ES)\right)\mathbf{x} \geq 0 \; \forall \mathbf{x} \in \mathbb{R}^n.
\end{aligned}$$

Then, as above,

$$\begin{aligned}\mathbf{x}^\intercal\left(\epsilon S+\frac{1}{2}(SE+ES)\right)\mathbf{x} &= \mathbf{x}^\intercal SU((\epsilon/2)I+M/2)U^\intercal\mathbf{x}+\mathbf{x}^\intercal U((\epsilon/2)I+M/2)U^\intercal S\mathbf{x}\\ &\geq \frac{1}{2}\mathbf{x}^\intercal SU(\epsilon I+\lambda_{min}(E)I)U^\intercal\mathbf{x}\\ &\quad+\frac{1}{2}\mathbf{x}^\intercal U(\epsilon I+\lambda_{min}(E)I)U^\intercal S\mathbf{x}\\ &= (\epsilon+\lambda_{min}(E))\mathbf{x}^\intercal S\mathbf{x}.\end{aligned}$$

Since $|\lambda_{min}(E)|\leq|\lambda_{max}(E)|=\delta$, $\epsilon\geq\delta$ implies $(1-\epsilon)S\preceq\tilde{S}$. Hence if $\epsilon\geq\delta$, we have that $(1-\epsilon)S\preceq S+SE+ES\preceq(1+\epsilon)S$. □

A.2.4. Spectral similarity under additive perturbations

Proposition 4.5.2. *Given a symmetric, positive semi-definite GSO S and its symmetric, positive semi-definite additively perturbed version $\tilde{S}=S+E$, where $\|E\|_{op}\leq\delta$, if ker$(E)\subseteq$ ker(S) then the matrices are $(\delta/\bar{\lambda}(S))$-spectrally similar, where $\bar{\lambda}(S)$ is the smallest non-zero eigenvalue of S.*

Proof. As in Proposition 4.5.1, let us find the value of ϵ which satisfies the inequality

$$\begin{aligned}\tilde{S}\preceq(1+\epsilon)S &\Leftrightarrow S+E\preceq(1+\epsilon)S\\ &\Leftrightarrow \epsilon S-E\succeq 0\\ &\Leftrightarrow \mathbf{x}^\intercal(\epsilon S-E)\mathbf{x}\geq 0\quad\forall\mathbf{x}\in\mathbb{R}^n.\end{aligned}$$

Now, for any $\mathbf{x}\in\ker(S)$, $\mathbf{x}^\intercal(\epsilon S-E)\mathbf{x}=-\mathbf{x}^\intercal E\mathbf{x}$, but recall that by assumption $\ker(E)\subseteq\ker(S)$. Hence, regardless of ϵ, $\mathbf{x}^\intercal(\epsilon S-E)\mathbf{x}=0$ for $\mathbf{x}\in\ker(S)$. Now, for $\mathbf{x}\notin\ker(S)$, since $\|E\|_{op}\leq\delta$,

$$\begin{aligned}\mathbf{x}^\intercal(\epsilon S-E)\mathbf{x} &= \epsilon\mathbf{x}^\intercal S\mathbf{x}-\mathbf{x}^\intercal E\mathbf{x}\\ &\geq \epsilon\mathbf{x}^\intercal S\mathbf{x}-\delta\|\mathbf{x}\|_2\end{aligned}$$

$$\geq \epsilon\bar{\lambda}(S)\left\|\mathbf{x}\right\| - \delta\left\|\mathbf{x}\right\|_2$$
$$= (\epsilon\bar{\lambda}(S) - \delta)\left\|\mathbf{x}\right\|_2,$$

where we have used the fact that $\min\{\mathbf{x}^\intercal S\mathbf{x} \mid \mathbf{x} \notin \ker(S)\} = \bar{\lambda}(S)\left\|\mathbf{x}\right\|_2$. Thus, we must have that $\epsilon \geq \delta/\bar{\lambda}(S)$. Now, for the second inequality,

$$(1-\epsilon)S \preceq \tilde{S} \Leftrightarrow \mathbf{x}^\intercal(\epsilon S + E)\mathbf{x} \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n.$$

Clearly for $\mathbf{x} \in \ker(S)$, $\mathbf{x}^\intercal(\epsilon S + E)\mathbf{x} = 0$. Then, for $\mathbf{x} \notin \ker(S)$,

$$\mathbf{x}^\intercal(\epsilon S + E)\mathbf{x} = \epsilon\mathbf{x}^\intercal S\mathbf{x} + \mathbf{x}^\intercal E\mathbf{x} \geq (\epsilon\bar{\lambda}(S) + \lambda_{min}(E))\left\|\mathbf{x}\right\|_2.$$

Thus, if $\epsilon \geq \max\{-\lambda_{min}(E)/\bar{\lambda}(S), 0\}$ then $(1-\epsilon)S \preceq \tilde{S}$. However, $\delta = |\lambda_{max}(E)| \geq |\lambda_{min}(E)|$. Hence, it is sufficient for $\epsilon \geq \delta/\bar{\lambda}(S)$ to ensure $(1-\epsilon)S \preceq S + E \preceq (1+\epsilon)S$. □

A.2.5. Spectral similarity under relative and additive perturbations

Proposition 4.5.3. *Given a symmetric, positive semi-definite GSO S, an additive perturbation matrix D such that $\left\|D\right\|_{op} \leq \delta_A$ and ker$(D) \subseteq$ ker(S), a diagonalizable relative perturbation matrix E such that $\left\|E\right\|_{op} \leq \delta_R$, and the symmetric, positive semi-definite perturbed GSO $\tilde{S} = S + \frac{1}{2}(SE + ES) + D$, the matrices are $(\delta_R + \delta_A/\bar{\lambda}(S))$-spectrally similar.*

Proof. As before, we wish to find a value of ϵ such that,

$$(1-\epsilon)S \preceq S + \frac{1}{2}(SE + ES) + D \preceq (1+\epsilon)S. \tag{A.6}$$

Starting with the latter inequality, i.e.,

$$(1+\epsilon)\mathbf{x}^\intercal S\mathbf{x} - \mathbf{x}^\intercal\left(S + \frac{1}{2}(SE + ES) + D\right)\mathbf{x} \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n,$$

we thus have

$$\begin{aligned}\mathbf{x}^\intercal\left(\epsilon S - \frac{1}{2}(SE+ES)\right)\mathbf{x} - \mathbf{x}^\intercal D\mathbf{x} &= \frac{1}{2}\mathbf{x}^\intercal SU(\epsilon I - M)U^\intercal\mathbf{x} + \frac{1}{2}\mathbf{x}^\intercal U(\epsilon I - M)U^\intercal S\mathbf{x} - \mathbf{x}^\intercal D\mathbf{x} \\ &\geq \frac{1}{2}\mathbf{x}^\intercal SU(\epsilon - \delta_R)U^\intercal\mathbf{x} + \frac{1}{2}\mathbf{x}^\intercal U(\epsilon - \delta_R)U^\intercal S\mathbf{x} - \mathbf{x}^\intercal D\mathbf{x} \\ &= (\epsilon - \delta_R)\mathbf{x}^\intercal S\mathbf{x} - \mathbf{x}^\intercal D\mathbf{x}.\end{aligned}$$

For $\mathbf{x} \in \ker(S)$, the above is zero since $\ker(D) \subseteq \ker(S)$ by assumption. Then, for $\mathbf{x} \notin \ker(S)$,

$$\begin{aligned}(\epsilon - \delta_R)\mathbf{x}^\intercal S\mathbf{x} - \mathbf{x}^\intercal D\mathbf{x} &\geq (\epsilon - \delta_R)\mathbf{x}^\intercal S\mathbf{x} - \delta_A \left\|\mathbf{x}\right\|_2 \\ &\geq (\epsilon - \delta_R)\bar{\lambda}(S)\left\|\mathbf{x}\right\|_2 - \delta_A \left\|\mathbf{x}\right\|_2.\end{aligned}$$

Hence, we have the condition $\epsilon \geq \delta_R + \delta_A/\bar{\lambda}(S)$. Then, for the former inequality, we have

$$\begin{aligned}&(1-\epsilon)S \preceq S + \frac{1}{2}(SE+ES) + D \\ &\Leftrightarrow \mathbf{x}^\intercal\left(\epsilon S + \frac{1}{2}(SE+ES)\right)\mathbf{x} + \mathbf{x}^\intercal D\mathbf{x} \geq 0 \quad \forall \mathbf{x} \in \mathbb{R}^n.\end{aligned}$$

Thus, following the proofs of Proposition 4.5.1 and Proposition 4.5.2,

$$\begin{aligned}\mathbf{x}^\intercal\left(\epsilon S + \frac{1}{2}(SE+ES)\right)\mathbf{x} + \mathbf{x}^\intercal D\mathbf{x} &\geq (\epsilon + \lambda_{min}(E))\mathbf{x}^\intercal S\mathbf{x} + \mathbf{x}^\intercal D\mathbf{x} \\ &\geq (\epsilon + \lambda_{min}(E))\bar{\lambda}(S)\left\|\mathbf{x}\right\|_2 + \lambda_{min}(D)\left\|\mathbf{x}\right\|_2.\end{aligned}$$

Since $|\lambda_{min}(E)| \leq |\lambda_{max}(E)| \leq \delta_R$ and $|\lambda_{min}(D)| \leq |\lambda_{max}(D)| \leq \delta_A$, S and $\tilde{S}$ are ϵ-spectrally similar whenever $\epsilon \geq \delta_R + \delta_A/\bar{\lambda}(S)$. □

A.2.6. Spectral similarity of random graphs

Proposition 4.4.1. *Let $\{\mathcal{G}_n\}_{n=1}^\infty$ and $\{\tilde{\mathcal{G}}_n\}_{n=1}^\infty$ be two independent sequences of graphs drawn from the same family of random graph distributions so that $\mathcal{G}_n$, $\tilde{\mathcal{G}}_n \sim P_{\mathcal{G}}(n)$ for each n, with graph shift operators S_n and $\tilde{S}_n$, respectively, having eigenvalues $\{\lambda_i\}_{i=1}^n$ and $\{\tilde{\lambda}_i\}_{i=1}^n$. Assume the following:*

(A1) Multiplicity of zero: the zero eigenvalue has almost surely constant multiplicity independent of n (potentially zero);

(A2) Bounded spectral gap: there exists $c > 0$ independent of n such that $|\lambda_i| \geq c$ almost surely for all non-zero eigenvalues;

(A3) Concentration: given $\epsilon_c > 0$ and $\delta_c > 0$ there exist some values γ_i, $i \in \{1 \dots, n\}$, such that for large enough n,

$$P\left(|\lambda_i - \gamma_i| < \epsilon_c,\ \forall i \in \{1, \dots, n\}\right) > 1 - \delta_c. \tag{4.24}$$

Then for any $\epsilon > 0$ and $\delta > 0$, there exists $N = N(\epsilon, \delta)$ such that for any $n \geq N$,

$$P\Big((1-\epsilon)\lambda_i < \tilde{\lambda}_i < (1+\epsilon)\lambda_i,\ \forall i \in \{1, \dots, n\}\Big) > 1-\delta. \tag{4.25}$$

In other words, for appropriate graph shift operators of large random graphs whose eigenvalues concentrate, the coefficient of spectral similarity converges in probability to 0 as $n \to \infty$.

Proof. For ease of notation, fix some n and set $\lambda_i \coloneqq \lambda_i(S_n)$ and $\tilde{\lambda}_i \coloneqq \lambda_i(\tilde{S}_n)$. Hence, denoting $\mathcal{I} \coloneqq \{i \in \{1, \dots, n\} \ : \ \lambda_i(S_n) \neq 0\}$, note that *(A1)* implies almost surely that we can order the eigenvalues of $\tilde{S}_n$ so that $\mathcal{I} = \left\{i \in \{1, \dots, n\} \ : \ \lambda_i(\tilde{S}_n) \neq 0\right\}$ and, thus, $|\lambda_i - \tilde{\lambda}_i| = 0$ for all $i \notin \mathcal{I}$. Thus, it suffices to show that $\forall \epsilon > 0, \delta > 0$ there exists some N such that for any $n \geq N$,

$$P\left((1-\epsilon)\lambda_i \leq \tilde{\lambda}_i \leq (1+\epsilon)\lambda_i\ \forall i \in \mathcal{I}\right) > 1 - \delta \Leftrightarrow P\left(|\lambda_i - \tilde{\lambda}_i| \leq \epsilon|\lambda_i|\ \forall i \in \mathcal{I}\right) > 1 - \delta. \tag{A.7}$$

By *(A2)*, the non-trivial eigenvalues of S_n and $\tilde{S}_n$ are bounded away from zero by some

$c > 0$ almost surely. Thus, since $P(A) > P(B)$ if $A \supseteq B$,

$$\begin{aligned} P\left(|\lambda_i - \tilde{\lambda}_i| \leq \epsilon|\lambda_i| \ \forall i \in \mathcal{I}\right) &\geq P\left(|\lambda_i - \tilde{\lambda}_i| \leq c\epsilon \ \forall i \in \mathcal{I}\right) \\ &= P\left(|\lambda_i - \gamma_i - (\tilde{\lambda}_i - \gamma_i)| \leq c\epsilon \ \forall i \in \mathcal{I}\right) \\ &\geq P\left(|\lambda_i - \gamma_i| + |\tilde{\lambda}_i - \gamma_i| \leq c\epsilon \ \forall i \in \mathcal{I}\right) \\ &\geq P\Big(|\lambda_i - \gamma_i| \leq c\epsilon/2 \ \forall i \in \mathcal{I}\Big) P\Big(|\tilde{\lambda}_i - \gamma_i| \leq c\epsilon/2 \ \forall i \in \mathcal{I}\Big), \end{aligned}$$

where the last line follows by independence. Then by $(A3)$, for any $\epsilon > 0$, $\delta > 0$ we can choose N_1 large enough such that $P(|\lambda_i - \gamma_i| \leq c\epsilon/2 \ \forall i \in \mathcal{I}) > \sqrt{1-\delta}$ (and similarly choose N_2 for all $\tilde{\lambda}_i$), and the result follows by picking $N = \max\{N_1, N_2\}$. □

A.3. Proofs of Chapter 5

A.3.1. Proof of Lemma 5.2.1

Lemma 5.2.1. *The states of the linearized model (wherein $S(t) = S_0$) upper-bound the states of the model in* (5.1)-(5.7)*. Thus, for all $t > 0$,*

$$\tilde{S}(t) \leq S(t) = S_0, \quad \tilde{E}(t) \leq E(t), \quad \tilde{I}(t) \leq I(t),$$
$$\tilde{A}(t) \leq A(t), \quad \tilde{H}(t) \leq H(t), \quad \tilde{R}(t) \leq R(t), \quad \tilde{D}(t) \leq D(t).$$

Proof. Clearly, since in our model there is no possibility of reinfection, $\tilde{S}(t) \leq S_0 = S(t)$. We now proceed by induction on t. For the base case $t = 1$, recall the initial conditions are identical for both the linearized and true model. Thus, from (5.2)-(5.7), the compartments have identical values for $t = 1$. Now, assuming the bounds hold for $t = k - 1$, we have

$$\begin{aligned}
\tilde{E}(k) &= (1 - \rho_{EI} - \rho_{EA})\tilde{E}(k-1) + \tilde{S}(k-1)\beta(t)(\gamma_A \tilde{A}(k-1) + \tilde{I}(k-1)) \\
&\leq ((1 - \rho_{EI} - \rho_{EA})E(k-1) + S_0\beta(t)(\gamma_A A(k-1) + I(k-1)) \\
&= E(k), \\
\tilde{I}(k) &= (1 - \rho_{IH} - \rho_{IR})\tilde{I}(k-1) + \rho_{EI}\tilde{E}(k-1) \\
&\leq (1 - \rho_{IH} - \rho_{IR})I(k-1) + \rho_{EI}E(k-1) \\
&= I(k), \\
\tilde{A}(k) &= (1 - \rho_{AR})\tilde{A}(k-1) + \rho_{EA}\tilde{E}(k-1) \\
&\leq (1 - \rho_{AR})A(k-1) + \rho_{EA}E(k-1) \\
&= A(k), \\
\tilde{H}(k) &= (1 - \rho_{HR})\tilde{H}(k-1) + \rho_{IH}\tilde{I}(k-1) \\
&\leq (1 - \rho_{HR})H(k-1) + \rho_{IH}I(k-1) \\
&= H(k-1), \\
\tilde{R}(k) &= \tilde{R}(k-1) + \rho_{IR}\tilde{I}(k-1) + \rho_{AR}\tilde{A}(k-1) + (1 - \alpha_D)\rho_{HR}\tilde{H}(t)
\end{aligned}$$

$$
\begin{aligned}
&\leq R(k-1) + \rho_{IR} I(k-1) + \rho_{AR} A(k-1) + (1-\alpha_D)\rho_{HR} H(t) \\
&= R(k), \\
\tilde{D}(k) &= \tilde{D}(k-1) + \alpha_D \rho_{HR} \tilde{H}(k-1) \\
&\leq D(k-1) + \alpha_D \rho_{HR} H(k-1) \\
&= D(k),
\end{aligned}
$$

and the result follows. □

A.3.2. Proof of Lemma 5.4.1

Lemma 5.4.1. *The functions $H(t)$ and $D(t)$, representing the number of hospitalized individuals and deaths at time t, are posynomials on the entries of $\mathbf{u}(t)$ for $t = 0, 1, \ldots, T_c$.*

Proof. We can rewrite equations (5.2)-(5.5) in matrix form by defining a state vector $\mathbf{x}(t) = [E(t), I(t), A(t), H(t)]^\intercal$ to obtain the dynamics

$$
\begin{aligned}
\mathbf{x}(t+1) &= \begin{bmatrix} 1-\rho_{EI}-\rho_{EA} & S_0\beta(\mathbf{u}(t)) & \gamma_A S_0 \beta(\mathbf{u}(t)) & 0 \\ \rho_{EI} & 1-\rho_{IR}-\rho_{IH} & 0 & 0 \\ \rho_{EA} & 0 & 1-\rho_{AR} & 0 \\ 0 & \rho_{IH} & 0 & 1-\rho_{HR} \end{bmatrix} \mathbf{x}(t) \\
&=: M_t \mathbf{x}(t). \qquad \text{(A.8)}
\end{aligned}
$$

It follows that

$$
H(t) = [0,0,0,1]\, M_{t-1} \cdots M_1 M_0 \mathbf{x}(0) =: f_H^t \left(\{\mathbf{u}(s)\}_{s=0}^{t-3} \right). \qquad \text{(A.9)}
$$

Recalling that the mobility mapping $\beta(\mathbf{u}(s))$ is a posynomial, each of the matrices M_s have posynomial entries on $\beta(\mathbf{u}(s))$, and thus, on $\mathbf{u}(s)$; thus, it follows that the function $f_H^t \left(\{\mathbf{u}(s)\}_{s=0}^{t-3} \right)$ in (A.9) is a posynomial on $\{\mathbf{u}(s)\}_{s=0}^{t-3}$ as it is the product of matrices with entries that are posynomials on $\mathbf{u}(s)$ for $s \in \{0, 1, \ldots, t-3\}$. Moreover, from (5.7) we can

see that $D(t)$ is simply a sum of positive constants and positive scalar multiples of $H(t)$ and is also a posynomial. □

A.3.3. Proof of Theorem 5.4.1

Theorem 5.4.1. *If $C_t(\mathbf{u}(t))$ is a posynomial cost function for all t, and the set of admissible control actions $\mathcal{U}$ is described by posynomial inequalities and monomial equalities, then the following is a geometric program:*

$$\begin{aligned} \underset{\mathbf{u}(0),\mathbf{u}(1),\ldots,\mathbf{u}(T_c-1)}{\text{minimize}} \quad & \sum_{t=1}^{T_c-1} \gamma_D^t D(t) + \gamma_\infty D(T_c) \\ \text{subject to} \quad & \sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t)) \leq \mathcal{B} \\ & H(t) \leq \tau_H, \qquad\qquad t = 1, \ldots, T_c, \\ & \mathbf{u}(t) \in \mathcal{U}, \qquad\qquad t = 0, \ldots, T_c - 1, \end{aligned} \tag{5.11}$$

where $\mathbf{u}(t)$ are the NPI control actions, and $H(t)$ and $D(t)$ are the number of individuals who are hospitalized and who die at time t, respectively.

Proof. By Lemma 5.4.1, the objective function J in (5.10) is posynomial on $\{\mathbf{u}(t)\}_{t=0}^{T_c-4}$ as it is the sum of posynomials, since $\gamma_D \in (0, 1]$ and thus $\gamma_\infty \in [0, \infty)$ (defining $\gamma_\infty = 0$ when $\gamma_D = 1$). Moreover, the constraint $H(t) \leq \tau_H$ is clearly posynomial on $\{\mathbf{u}(s)\}_{s=0}^{t-3}$ for each $1 \leq t \leq T_c$ by Lemma 5.4.1. By assumption the cost function $C_t(\mathbf{u}(t))$ is posynomial on the decision variables $\mathbf{u}(t)$ for all $0 \leq t \leq T_c - 1$, and thus the constraint $\sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t)) \leq \mathcal{B}$ is posynomial on $\{\mathbf{u}(t)\}_{t=0}^{T_c-1}$. Finally, by assumption the set $\mathcal{U}$ is described via posynomial inequalities and monomial equalities on $\mathbf{u}(t)$ for every $0 \leq t \leq T_c - 1$. Thus, (5.11) is a geometric program, and the globally optimal solution $\mathbf{u}^\star$ may be found efficiently. □

A.3.4. Proof of Theorem 5.4.2

Theorem 5.4.2. *If $C_t(\mathbf{u}(t))$ is a posynomial cost function for all t, and the set of admissible control actions $\mathcal{U}$ is described by posynomial inequalities and monomial equalities, then the*

minimal budget required to keep hospitalizations below a given threshold τ_H is given by

$$\mathcal{B}^\star = \sum_{t=0}^{T_c-1} C_t(\mathbf{u}^\star(t)), \tag{5.12}$$

where $\mathbf{u}^\star$ is the solution to the geometric program

$$\begin{aligned} \underset{\mathbf{u}(0),\ldots,\mathbf{u}(T_c-1)}{\text{minimize}} \quad & \sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t)) \\ \text{subject to} \quad & H(t) \leq \tau_H, \quad t = 1,\ldots,T_c, \\ & \mathbf{u}(t) \in \mathcal{U}, \quad t = 0,\ldots,T_c-1. \end{aligned} \tag{5.13}$$

where $\mathbf{u}(t)$ are the NPI control actions, and $H(t)$ is the number of individuals who are hospitalized at time t.

Proof. Similarly to the proof of Theorem 5.4.1, we may invoke Lemma 5.4.1 to show that the constraint $H(t) \leq \tau_H$ is posynomial on $\{\mathbf{u}(s)\}_{s=0}^{t-3}$ for each $1 \leq t \leq T_c$. Again similarly to the proof of Theorem 5.4.1, by assumption on $C_t(\mathbf{u}(t))$ the sum $\sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t))]$ and, hence, the objective function, is posynomial on $\{\mathbf{u}(t)\}_{t=0}^{T_c-1}$, and the set $\mathcal{U}$ is described via posynomial inequalities and monomial equalities on $\mathbf{u}(t)$ for every $0 \leq t \leq T_c - 1$. Hence, (5.13) is a geometric program, whose globally optimal solution $\mathbf{u}^\star$ may be found efficiently. Using this solution, we may compute the minimal budget to prevent hospital overflow as $\mathcal{B}^\star = \sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t))$. □

A.3.5. Proof of Lemma 5.4.2

Lemma 5.4.2. *The set of admissible control actions in* (5.14)*, where $0 < \underline{u}_k < \overline{u}_k$ for all $k \in \{1,\ldots,K\}$, $\Delta > 0$, and the cost function in* (5.15)*, where $c_k \geq 0$ for all $k \in \{1,\ldots,K\}$, are sufficient for* (5.11) *and* (5.13) *to be geometric programs.*

Proof. By definition in (5.14), for all $\{(u_1(t),\ldots,u_K(t))\}_{t=0}^{T_c} \in \mathcal{U}$ we have $u_k(t)^{-1} \leq \underline{u}_k^{-1}$, $u_k(t) \leq \overline{u}_k$, and we may express the other inequalities as $u_k(t)^{-1}u_k(t-1) \leq (1-\Delta)^{-1}$, and

$u_k(t)u_k(t-1)^{-1} \leq 1+\Delta$ for each $k \in \{1,\ldots,K\}$. Moreover, the equality constraints may be satisfied by considering only one decision variable for each seven day period. Hence, $\mathcal{U}$ may be described by posynomial inequalities. Next, notice that

$$\begin{aligned} C_t(\mathbf{u}(t)) &= \sum_{k=1}^{K} c_k \frac{u_k(t)^{-1} - \overline{u}_k^{-1}}{\underline{u}_k^{-1} - \overline{u}_k^{-1}} \\ &= \sum_{k=1}^{K} \frac{c_k}{\underline{u}_k^{-1} - \overline{u}_k^{-1}} u_k(t)^{-1} - \sum_{k=1}^{K} \frac{\overline{u}_k^{-1}}{\underline{u}_k^{-1} - \overline{u}_k^{-1}}. \end{aligned}$$

Clearly, since the values $c_k, \underline{u}_k$, and $\overline{u}_k$ are constant, and $\underline{u}_k < \overline{u}_k$ implies $\underline{u}_k^{-1} - \overline{u}_k^{-1} > 0$, the left-hand term above is a posynomial in the entries of $\mathbf{u}(t)$, and the right-hand term is constant. Hence, the function $C_t(\mathbf{u}(t))$ is a posynomial in $\mathbf{u}(t) = (u_1(t),\ldots,u_K(t))$ shifted by a constant (which does not affect the optimal values of $\mathbf{u}(t)$ in (5.13)). Considering the budget-constraint inequality in (5.11), from above we have

$$\sum_{t=0}^{T_c-1} C_t(\mathbf{u}(t)) \leq \mathcal{B} \quad \Leftrightarrow \quad \sum_{t=0}^{T_c-1} \sum_{k=1}^{K} \frac{c_k}{\underline{u}_k^{-1} - \overline{u}_k^{-1}} u_k(t)^{-1} \leq \mathcal{B} + \sum_{t=0}^{T_c-1} \sum_{k=1}^{K} \frac{\overline{u}_k^{-1}}{\underline{u}_k^{-1} - \overline{u}_k^{-1}},$$

which is clearly a posynomial inequality amenable to geometric programming. □

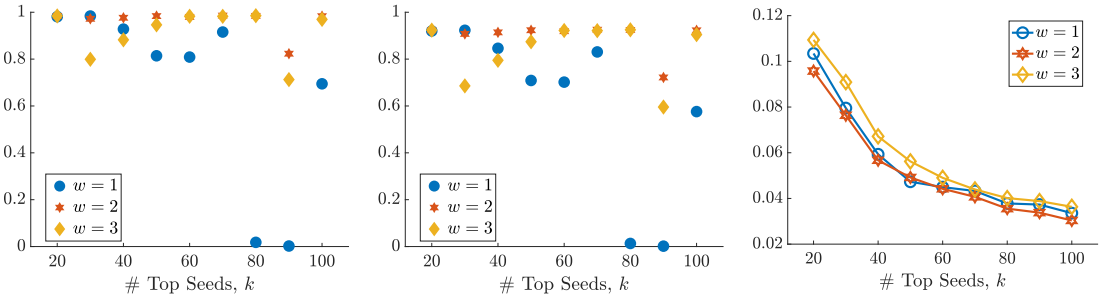
APPENDIX B

Additional Details and Experimental Results

This appendix contains experimental details and supplementary materials for all chapters.

B.1. Additional experimental results for Chapter 3

B.1.1. Parameter Selection



(a) Precision (b) Recall (c) Percentage of correct pairs in $\mathcal{S}$

Figure B.1: Effect of changing k and w for GEMSEC-Artists network with 60% correlation.

As discussed in Section 3.3.1, the process of seed selection requires the parameters k and w to be chosen. In Figure B.1 we do a sweep of the parameters with $k \in \{20, 30, \ldots, 100\}$ and $w \in \{1, 2, 3\}$ on the GEMSEC Facebook-Artists network [95] to study their influence on the performance of `SPECTRE`. As suggested in Section 3.3, we fix $f = 3/4$, $r = 4$, and use eigenvector centrality to build the initial noisy seed estimate. In all cases, we limit the number of iterations of `SafeExpand` and `LooseExpand` to be no more than five. Intuitively, increasing k should allow for the possibility of more correct pairs to be included, and a larger w increases the probability of placing correct pairs in the initial seed estimate. This intuition is supported by our empirical results; however, increasing these parameters is not free since a larger proportion of incorrect (or noisy) pairs are included in the seed set, as shown in Figure B.1(c). In a situation with no prior or side information about the network, our empirical results suggest that $k = O(\log n)$, where $n = \min\{|\mathcal{V}_1|, |\mathcal{V}_2|\}$, and $w = 2$ are good choices. Even when the networks to be aligned exhibit a moderate correlation of

60%, as shown in Figure B.1(a), choosing $k = 20$ and $w = 2$ yields near-perfect precision and similarly high recall, suggesting `SPECTRE` was able to correctly percolate throughout the networks.

We can see the percolation behavior in Figures B.1(a) and B.1(b); specifically, the efficacy of repeatedly re-running `SafeExpand` and `LooseExpand` to boost the alignment performance. In these figures, we observe thresholding behavior; in particular, the matching will either succeed in percolating throughout the networks, achieving high precision and recall, or it will fail and perform poorly. This empirical result is similar to the percolation phenomenon in Erdös-Rényi graphs [79], as well as empirical studies on real-world networks [183]. The main reason for performing the boosting step in `SPECTRE` is to encourage the success of this percolation. By seeding `SafeExpand` with the previous matching that did not spread well, we may enable the percolation to succeed after further iterations have taken place. In practice this effect induces a much larger and more accurate network alignment. As Figure B.1(a) shows, we may achieve high-quality alignments even in networks which are not very correlated. Another practical consideration is that larger values of k and w tend to induce higher runtimes, as larger noisy seed estimates takes longer to spread in the `SafeExpand` subroutine. This effect can clearly be seen in Figure 3.6(c); notice that the behavior is not exactly monotonic due to the randomness in the way `SafeExpand` breaks ties, which in some cases may result in poor seed estimates. However, the largest impact on running time is from the boosting rounds, as each additional iteration of `SafeExpand` and `LooseExpand` takes several minutes on large networks.

B.1.2. Time Complexity

To observe how the runtime of `SPECTRE` is effected by the number of edges in the networks to align, we performed several experiments using publicly available datasets of varying size. In each case, we generated two 90% correlated networks using the process described earlier. A plot of the runtime of `SPECTRE` on each of these networks is given in Figure B.2. The networks include: a word-noun adjacency graph (adjnoun) [184]; connections between US

airports (USair97) [185]; a yeast PPI network (yeastl) [185]; a graph of hyperlinks between political blogs (polblogs) [186]; a network of athletes' pages from Facebook (Athletes) [95]; and the relationships of Hungarian users of the music streaming service Deezer (HR) [95]. The properties of all networks used for numerical experiments are shown in Table B.1. All experiments were performed on a quad-core Intel Core i7 at 2.2GHz with 16GB of RAM.

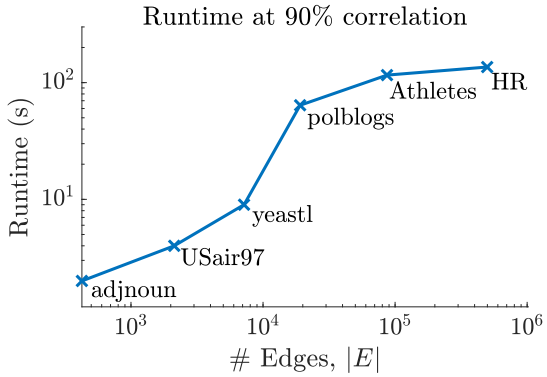


Figure B.2: Runtime of `SPECTRE` on multiple networks with varying numbers of edges.

Data set	$\lvert\mathcal{V}\rvert$	$\lvert\mathcal{E}\rvert$	Avg.Deg.	Max.Deg.
C. jejuni	3,294	19,643	11.93	699
E. coli	1,290	11,100	17.21	154
Facebook-Artists	50,515	819,306	32.44	1,292
arXiv Astrophysics	18,772	198,110	21.11	354
Adjective-Noun	112	425	7.58	49
US Air	332	2,126	12.80	139
Yeast PPI	2,284	6,646	5.81	64
Political Blogs	1,224	19,087	31.18	468
Facebook-Athletes	13,866	86,858	12.53	468
Deezer-HR	54,573	498,202	18.26	420

Table B.1: Properties of all networks used in numerical experiments.

B.2. Experimental setup for Chapter 4

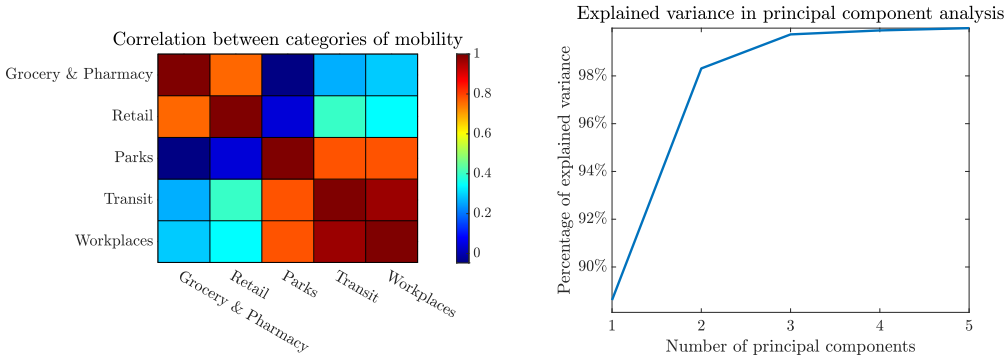
The hypergraph is created by randomly sampling 500 points on a 3-dimensional torus with inner radius 1 and outer radius 2, and keeping maximal hyperedges of a Vietoris-Rips complex [187] with radius 0.4, i.e., constructing simplices between any points jointly within a 2-norm distance 0.4. 10 hyperedges are selected at random to be the sources, making the source localization problem a 10-class classification problem. To generate K data samples $\{(\mathbf{x}_k, y_k)\}_{k=1}^K$, we pick 10 hyperedges at random to be the sources. For each source hyperedge i, we construct a node source signal for node j:

$$x_{0,j}^i = \begin{cases} 1+z, & \text{node } i \text{ is in the source hyperedge,} \\ z, & \text{otherwise.} \end{cases}$$

where $z \sim \text{Normal}(0, 0.01)$ is independent Gaussian white noise. This noisy source signal is diffused for $t_{max} = 30$ time steps via the nonlinear hypergraph Laplacian (4.7) to generate the sequence $\{\mathbf{x}_t^i\}_{t=0}^{t_{max}}$. Samples are then of the form $(\mathbf{x}_t^i + \mathbf{z}, y_i)$, where we pick $\mathbf{x}_t^i$ by randomly sampling a source hyperedge i and time t and add measurement noise $\mathbf{z} \sim \text{Normal}(\mathbf{0}, 0.01 I_n)$, and the hyperedge label is $y_i = i$. We use 500 of these signals for training (including cross-validation), and 300 of these signals for testing. Note that while we may sample a signal from the same source at the same time more than once, they will not be identical due to the added independent measurement noise.

All GNNs have two graph filtering layers with one input feature per node and a fixed, non-learnable readout layer, which selects the signals from the 10 candidate source hyperedges. The filters are normalized during training to ensure $|h(\lambda)| \leq 1$ and the integral Lipschitz constant is constrained to be less than 10 via a loss penalty term. Pooling from node to hyperedge signals is done based on node inclusion in the hyperedges. We train using `adam` with weights 0.9 and 0.999 with a learning rate of 0.0005, decay rate of 0.99 and decay period of 20. These hyperparameters, as well as others such as the number of filters and number of filter taps were chosen via 5-fold cross-validation.

B.3. Additional practical results for Chapter 5



(a) Pairwise correlation coefficients between different categories of mobility data.

(b) Percentage of explained variance in principal component analysis.

Figure B.3: Exploration of mobility pattern data in Philadelphia county. Analysis of pairwise correlation suggests two correlated groupings, which is supported by two principal components explaining over 98% of observed variance in the data.

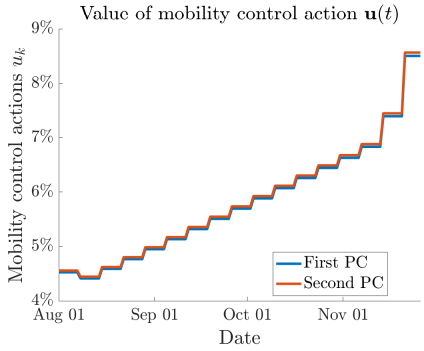
B.3.1. Dimensionality reduction

While mobility data may be expressed in terms of several different categories, in practice these categories are highly correlated; the correlation between mobility categories in Philadelphia is shown in Figure B.3(a). As such, it may not be reasonable to assume that a decision maker may tune these categories independently and arbitrarily. However, it may also be difficult to elucidate the explicit relationships present from data alone. As a compromise, we may perform dimensionality reduction on the measured mobility pattern data via principal component analysis (PCA). In particular, since our data is in the form of time series, we take the first differences of the mobility data $\mathbf{m}_d(t) \coloneqq \mathbf{m}(t) - \mathbf{m}(t-1)$ to ensure stationarity, find the $K \times K$ covariance matrix C of the K-dimensional observations $\mathbf{m}_d(1), \ldots, \mathbf{m}_d(T_c)$, and compute the principal components $\mathbf{w}_k$ of C [182]. We then project into the space of the mobility pattern data as $\tilde{\mathbf{m}}(t) \coloneqq [\mathbf{w}_1, \ldots, \mathbf{w}_l]^\intercal \mathbf{m}(t)$ for some number of principal components l. As shown in Figure B.3(b), the first two principal components are enough to describe over 98% of the variance in mobility data in the greater Philadelphia region over the time period of interest. For this reason, we present additional results in which this two-dimensional principal component projection, $\tilde{\mathbf{m}}(t)$, is used in place of the full mobility data from the Google Mobility Report, $\mathbf{m}(t)$. In particular, our multitask learning

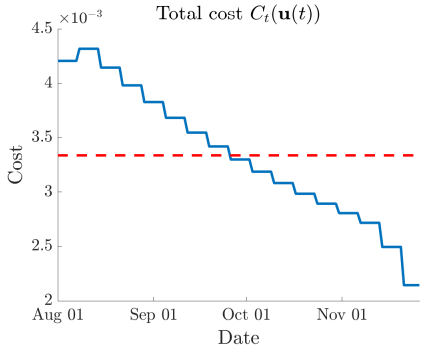
approach uses $\tilde{\mathbf{m}}(t)$ to learn the mobility mapping $\beta(\cdot)$, as well as all other parameters in our model. We then use the mapping and parameters to design control strategies via the GPs (5.11) and (5.13). A reasonable budget $\mathcal{B}^\star$ is deduced by solving (5.13) using this two-dimensional mobility data, and then a control strategy to minimize deaths $\mathbf{u}^\star(t)$ is designed by solving (5.11). These results are illustrated in Figure B.4. Compared to using the full-dimensional mobility data, this approach does not control the pandemic as quickly and, hence, saves fewer lives. These results illustrate that, should it be possible, it is advantageous to independently control different categories of mobility. However, even if it is not possible to execute fine-grained control across many categories of mobility, this multitask learning-driven nonlinear optimal control framework is still able to reduce the number of deaths while respecting limits on the economic costs incurred.

B.3.2. Relationship between the linear and nonlinear model

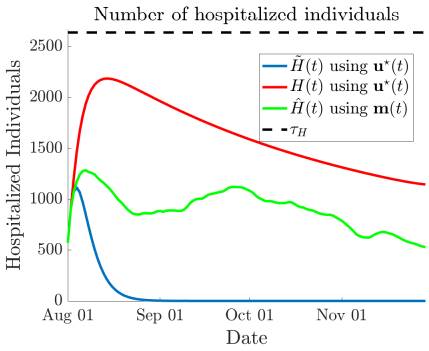
As shown in Lemma 5.2.1, the linear epidemic model is always an upper bound to the nonlinear model. However, it is of practical importance to investigate how tight this bound may be. As such, we present results on the gap between the final states in the linear and nonlinear model in Figure B.5. For each value of the hospitalization threshold τ_H, we solve the minimal cost GP (5.13) to find the minimal budget $\mathcal{B}(\tau_H)$. As mentioned in Section 5.4, this budget is minimal in the sense that it is the lowest cost incurred while respecting the threshold on the number of hospitalizations, τ_H, but in practice a larger budget is desirable. As such, we vary the budget for the minimal death GP (5.11) between $\mathcal{B}(\tau_H)$ and $10\mathcal{B}(\tau_H)$, and record the final number of hospitalizations in the linear model, $H(T_c)$, and the nonlinear model, $\tilde{H}(T_c)$, for $T = 61$ days. We also record the cumulative number of deaths in the linear and nonlinear models ($D(T_c)$ and $\tilde{D}(T_c)$, respectively). We then plot the difference between the quantities for the linear and nonlinear models as functions of the hospitalization threshold τ_H and the respective budgets $\mathcal{B}(\tau_H)$. Lower values of this difference correspond to a smaller gap between the linear and nonlinear models. As we can see in Figure B.5, budgets closer to $\mathcal{B}(\tau_H)$, i.e., the minimal feasible budget to respect τ_H, lead to a much larger gap between the models; this trend is consistent across values of τ_H. We observe threshold



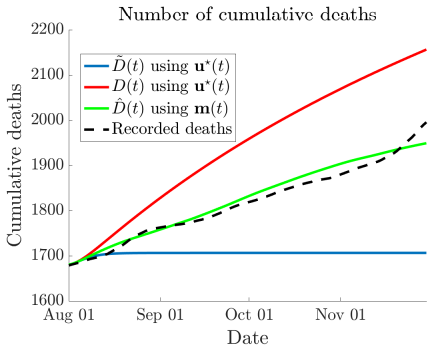
(a) Value of optimal minimal-death control action $\mathbf{u}^\star(t)$ (lower means less mobility allowed).



(b) Actuation cost $C_t(\mathbf{u}^\star(t))$ with control action $\mathbf{u}^\star(t)$; average daily budget $\mathcal{B}^\star/T_c$ shown dashed in red.



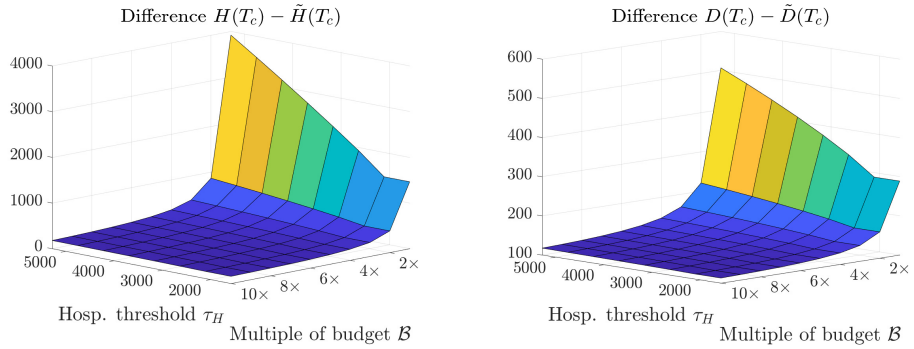
(c) Hospitalized individuals, with hospital bed threshold τ_H shown as the dashed line.



(d) Number of cumulative deaths, with actual recorded deaths shown as the dashed line.

Figure B.4: Example optimal minimal-death control strategy $\mathbf{u}^\star(t)$ with lower-dimensional mobility data for Philadelphia County, PA, obtained by solving the minimal-death GP (5.11), from September 1st to November 30th, 2020. In (c) and (d), we plot the prediction using the linear model with the strategy $\mathbf{u}^\star(t)$, $H(t)$ and $D(t)$, predictions using the nonlinear model with the strategy $\mathbf{u}^\star(t)$, $\tilde{H}(t)$ and $\tilde{D}(t)$, and the predictions using the true mobility data $\mathbf{m}(t)$ as a baseline for no intervention, $\hat{H}(t)$ and $\hat{D}(t)$. Parameters of the models used herein were learned as described in Section 5.3.3 using the principal component projections of mobility data $\tilde{\mathbf{m}}(t)$, with the budget $\mathcal{B}^\star$ taken from the solution to the minimal-cost GP (5.13).

behavior, whereby a moderate increase in the budget leads to a dramatic decrease in the gap between the models. This threshold behavior explains the difference between the linear and nonlinear models in the simulations of Section 5.4 and B.3.1, as we employ a budget close to the minimal cost obtained from solving the GP (5.13). Hence, there exists a moderate trade-off between the prescribed budget and the tightness of the upper bounds given by the states of the linear model. Fortunately, these results illustrate that implementing the



(a) Difference between hospitalizations. (b) Difference between cum. deaths.

Figure B.5: Difference between final hospitalizations (resp. cumulative deaths) in the linear model, $H(T_c)$ (resp. $D(T_c)$), and in the nonlinear model, $\tilde{H}(T_c)$ (resp. $\tilde{D}(T_c)$) as a function of the hospitalization threshold τ_H and the budget $\mathcal{B}(\tau_H)$. For each value of τ_H, we solve the minimal-cost GP (5.13) to find the budget $\mathcal{B}(\tau_H)$, then take multiples of this budget from $1\times$ to $10\times$. Trends appear consistent across hospitalization thresholds τ_H, but the difference between the models quickly shrinks as the budget $\mathcal{B}(\tau_H)$ grows.

designed strategies on the true nonlinear system can lead to better outcomes than expected using any budget.

B.3.3. Time complexity

Instance	T_c	Posynomial variables	Posynomial constants	Convex variables	Convex constants	Time (s)
GP (5.11)	30	16,365	7,291	52,445	29,841	18.2
GP (5.13)	30	17,287	7,693	55,287	31,443	14.7
GP (5.11)	60	69,220	31,506	219,660	125,531	104.8
GP (5.13)	60	71,112	32,338	225,472	128,813	86.4
GP (5.11)	90	158,570	72,686	501,490	287,011	379.1
GP (5.13)	90	161,402	73,938	510,162	291,913	380.8
GP (5.11)	120	294,055	135,296	928,247	531,666	2204.0
GP (5.13)	120	297,960	137,017	940,208	538,422	737.2

Table B.2: Space and time complexity of minimal-cost GP (5.13) and minimal-deaths GP (5.11) instances solved herein, over multiple time horizons T_c. Space complexity is measured in terms of the number of decision variables and constraints of the GP in posynomial form and in the equivalent convex form, the latter of which the solver uses to find the optimal solution. Time complexity measures the running time for the solver to instantiate and solve the program.

In order to validate the efficiency and scalability of our approach, we present data on the size and time complexity of the GPs constructed herein. These results are not exhaustive,

but meant to illustrate the practical applicability of our approach. For example, computational complexity may be reduced by dimensionality reduction techniques as in B.3.1. All experiments were performed on a laptop with an Intel Core i7 processor running at 2.2GHz and 16GB of RAM. These results illustrate the tractability of using geometric programming to solve problems with a large number of decision variables and contraints.

B.3.4. Clustering points of interest

If data is available on a more granular level, specifically for individual points of interest (POIs), it may be of use to consider how to cluster these POIs together in order to build categories of mobility directly from data. In particular, we wish to cluster places of interest based on both their geographical location as well as their patterns of visitations. For this reason, we may employ an approach that couples the clustering of time-series and networks, called the CCTN algorithm [188]. The underlying idea is to construct an embedding of the POIs, i.e., a low-dimensional representation in which "similar" POIs are close to one another, and then to find clusters in the latent embedding space. Since the geographical interpretation of clusters is important in our context, we may perform a further clustering on any large groups of POIs to split them according to distance from one another.

In this setting we have N POIs, each with a (commonly sparse) time series of daily visits per square foot over T days, which we stack into the rows of the matrix $X \in \mathbb{R}^{N\times T}$. We use visits per square foot since we wish to cluster POIs with similar *patterns* of visitations, not just numbers of visits. The main idea of the embedding is to reconstruct the matrix X as $\tilde{X} = CW$, where $W \in \mathbb{R}^{d\times T}$ is the basis matrix, representing d time series patterns learned from data, and $C \in \mathbb{R}^{N\times d}$ is the embedding matrix which reconstructs the original time series of visits for each POI using weighted combinations of the d basis patterns in W. To incorporate the geographical locations of the POIs, we may build a network wherein nodes represent POIs and edge weights are inversely proportional to geographical distance (dropping to zero, i.e., no edge, after a fixed distance). Then, the embedding will be regularized for smoothness across the network: we ensure nodes' embeddings are not dramatically different than their

network neighbors' embeddings. Hence, the embedding matrix C and basis patterns W may be found by solving the problem

$$\arg\min_{C,W} \left\| X - \tilde{X} \right\|_F^2 + \gamma \cdot \mathrm{tr}(C^\intercal L C), \tag{B.1}$$

where L denotes the graph Laplacian matrix, and γ is the parameter which controls the strength of the network regularization. In order to solve this problem, the CCTN algorithm performs an alternating process, fixing C and solving a quadratic problem for W, then fixing W and solving a mixed-integer program to find C. For full implementation details, see [188]. We may then perform k-means clustering in the d-dimensional embedding space described by C in order to cluster the POIs. This primary clustering focuses on grouping POIs with similar visitation patterns and regularizes for local smoothness in the network, but it does not necessarily disincentivize POIs which are geographically distant from being clustered together. Thus, in any clusters with many POIs, we may perform a further k-means clustering using the geographical location of the POIs to split these larger clusters based solely on distance. The final result of this approach will be clusters of POIs that are geographically close, and exhibit similar visitation patterns (in terms of daily visits per square foot) to those within their group.

Bibliography

[1] D. Easley, J. Kleinberg, *et al.*, "Networks, crowds, and markets: Reasoning about a highly connected world," *Significance*, vol. 9, pp. 43–44, 2012.

[2] F. Chung, *Spectral graph theory*. American Mathematical Soc., 1997.

[3] D. A. Spielman, "Spectral graph theory and its applications," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, IEEE, 2007, pp. 29–38.

[4] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," Stanford InfoLab, Tech. Rep., 1999.

[5] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, "Epidemic spreading in real networks: An eigenvalue viewpoint," in *22nd International Symposium on Reliable Distributed Systems, 2003. Proceedings.*, IEEE, 2003, pp. 25–34.

[6] D. Acemoglu and A. Ozdaglar, "Opinion dynamics and learning in social networks," *Dynamic Games and Applications*, vol. 1, no. 1, pp. 3–49, 2011.

[7] F. Gama, J. Bruna, and A. Ribeiro, "Stability properties of graph neural networks," *IEEE Transactions on Signal Processing*, vol. 68, pp. 5680–5695, 2020.

[8] L. Ruiz, L. Chamon, and A. Ribeiro, "Graphon neural networks and the transferability of graph neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1702–1712, 2020.

[9] R. Levie, E. Isufi, and G. Kutyniok, "On the transferability of spectral graph filters," in *2019 13th International conference on Sampling Theory and Applications (SampTA)*, IEEE, 2019, pp. 1–5.

[10] H. Kenlay, D. Thanou, and X. Dong, "Interpretable stability bounds for spectral graph filters," in *International conference on machine learning*, PMLR, 2021.

[11] S. Maskey, R. Levie, and G. Kutyniok, "Transferability of graph neural networks: An extended graphon approach," *arXiv preprint arXiv:2109.10096*, 2021.

[12] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on automatic control*, vol. 48, no. 6, pp. 988–1001, 2003.

[13] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[14] L. M. Pecora and T. L. Carroll, "Master stability functions for synchronized coupled systems," *Physical review letters*, vol. 80, no. 10, p. 2109, 1998.

[15] F. Dörfler, M. Chertkov, and F. Bullo, "Synchronization in complex oscillator networks and smart grids," *Proceedings of the National Academy of Sciences*, vol. 110, no. 6, pp. 2005–2010, 2013.

[16] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[17] M. Hayhoe, F. Barreras, H. Hassani, and V. M. Preciado, "SPECTRE: Seedless network alignment via spectral centralities," *arXiv preprint arXiv:1811.01056*, 2018.

[18] M. Hayhoe, F. Barreras, and V. M. Preciado, "Sparse estimation of Laplacian eigenvalues in multiagent networks," in *IFAC 2020 World Congress*, IFAC, 2020, pp. 1043–1048. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1288.

[19] M. Hayhoe, F. Barreras, and V. M. Preciado, "Multitask learning and nonlinear optimal control of the covid-19 outbreak: A geometric programming approach," *Annual Reviews in Control*, 2021. DOI: https://doi.org/10.1016/j.arcontrol.2021.04.014.

[20] M. Hayhoe, H. Riess, V. M. Preciado, and A. Ribeiro, "Stable and transferable hypergraph neural networks," *Submitted, available at arXiv:2211.06513*, 2022.

[21] M. Hayhoe, F. Barreras, and V. M. Preciado, "A dynamical approach to efficient eigenvalue estimation in general multiagent networks," *Automatica*, vol. 140, 2022. DOI: https://doi.org/10.1016/j.automatica.2022.110234.

[22] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub) graph isomorphism algorithm for matching large graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 10, pp. 1367–1372, 2004.

[23] R. Levie, W. Huang, L. Bucci, M. M. Bronstein, and G. Kutyniok, "Transferability of spectral graph convolutional neural networks," *J. Mach. Learn. Res.*, vol. 22, 2021.

[24] N. Keriven, A. Bietti, and S. Vaiter, "Convergence and stability of graph convolutional networks on large random graphs," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 512–21 523, 2020.

[25] L. Lovász, *Large networks and graph limits*. American Mathematical Soc., 2012, vol. 60.

[26] L. Ruiz, F. Gama, and A. Ribeiro, "Graph neural networks: Architectures, stability, and transferability," *Proceedings of the IEEE*, vol. 109, no. 5, pp. 660–682, 2021.

[27] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[28] J. Batson, D. A. Spielman, N. Srivastava, and S.-H. Teng, "Spectral sparsification of graphs: Theory and algorithms," *Communications of the ACM*, vol. 56, no. 8, pp. 87–94, 2013.

[29] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Analysis and control of epidemics: A survey of spreading processes on complex networks," *IEEE Control Systems Magazine*, vol. 36, no. 1, pp. 26–46, 2016.

[30] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and engineering*, vol. 8, no. 1, p. 67, 2007.

[31] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Autograd: Effortless gradients in numpy," in *ICML 2015 AutoML workshop*, vol. 238, 2015, p. 5.

[32] M. Fiedler, "Algebraic connectivity of graphs," *Czechoslovak Mathematical Journal*, vol. 23, no. 2, pp. 298–305, 1973.

[33] R. Merris, "Laplacian matrices of graphs: A survey," *Linear algebra and its applications*, vol. 197, pp. 143–176, 1994.

[34] V. M. Preciado, "Spectral analysis for stochastic models of large-scale complex dynamical networks," Ph.D. dissertation, Massachusetts Institute of Technology, 2008.

[35] M. Mesbahi and M. Egerstedt, *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010, vol. 33.

[36] F. Bullo, *Lectures on Network Systems*, 1.3. Kindle Direct Publishing, 2019, Contributions by J. Cortes, F. Dorfler, and S. Martinez, ISBN: 978-1986425643.

[37] C. O. Becker, S. Pequito, G. J. Pappas, *et al.*, "Spectral mapping of brain functional connectivity from diffusion imaging," *Scientific reports*, vol. 8, no. 1, p. 1411, 2018.

[38] B. Mohar, "Some applications of Laplace eigenvalues of graphs," in *Graph symmetry*, Springer, 1997, pp. 225–275.

[39] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, 8 2000.

[40] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Analysis and control of epidemics: A survey of spreading processes on complex networks," *IEEE Control Systems Magazine*, vol. 36, no. 1, pp. 26–46, 2016.

[41] D. Kempe and F. McSherry, "A decentralized algorithm for spectral analysis," *Journal of Computer and System Sciences*, vol. 74, no. 1, pp. 70–83, 2008.

[42] G. H. Golub and C. Van Loan, *Matrix computations*, 4th ed. Johns Hopkins University Press, 2013.

[43] M. Franceschelli, A. Gasparri, A. Giua, and C. Seatzu, "Decentralized estimation of Laplacian eigenvalues in multi-agent systems," *Automatica*, vol. 49, no. 4, pp. 1031–1036, 2013.

[44] C. Li and Z. Qu, "Distributed estimation of algebraic connectivity of directed networks," *Systems & Control Letters*, vol. 62, no. 6, pp. 517–524, 2013.

[45] R. Aragues, G. Shi, D. V. Dimarogonas, C. Sagüés, K. H. Johansson, and Y. Mezouar, "Distributed algebraic connectivity estimation for undirected graphs with upper and lower bounds," *Automatica*, vol. 50, no. 12, pp. 3253–3259, 2014.

[46] H. A. Poonawala and M. W. Spong, "Decentralized estimation of the algebraic connectivity for strongly connected networks," in *2015 American Control Conference*, IEEE, 2015, pp. 4068–4073.

[47] C. Li, Z. Qu, D. Qi, and F. Wang, "Distributed finite-time estimation of the bounds on algebraic connectivity for directed graphs," *Automatica*, vol. 107, pp. 289–295, 2019.

[48] S. Leonardos, V. M. Preciado, and K. Daniilidis, "Distributed spectral computations: Theory and applications," *under review*, 2019.

[49] T.-M.-D. Tran and A. Y. Kibangou, "Distributed estimation of Laplacian eigenvalues via constrained consensus optimization problems," *Systems & Control Letters*, vol. 80, pp. 56–62, 2015.

[50] A. Gusrialdi and Z. Qu, "Distributed estimation of all the eigenvalues and eigenvectors of matrices associated with strongly connected digraphs," *IEEE control systems letters*, vol. 1, no. 2, pp. 328–333, 2017.

[51] A. Mauroy and J. Hendrickx, "Spectral identification of networks using sparse measurements," *SIAM Journal on Applied Dynamical Systems*, vol. 16, no. 1, pp. 479–513, 2017.

[52] A. Mesbahi and M. Mesbahi, "Identification of the Laplacian spectrum from sparse local measurements," in *2019 American Control Conference*, IEEE, 2019, pp. 3388–3393.

[53] Y. Yuan, G.-B. Stan, L. Shi, M. Barahona, and J. Goncalves, "Decentralised -time consensus," *Automatica*, vol. 49, no. 5, pp. 1227–1235, 2013.

[54] T. Charalambous, M. G. Rabbat, M. Johansson, and C. N. Hadjicostis, "Distributed finite-time computation of digraph parameters: Left-eigenvector, out-degree and spectrum," *IEEE Transactions on Control of Network Systems*, vol. 3, no. 2, pp. 137–148, 2015.

[55] A. Gusrialdi and Z. Qu, "Data-driven distributed algorithms for estimating eigenvalues and eigenvectors of interconnected dynamical systems," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 52–57, 2020.

[56] D. Potts and M. Tasche, "Parameter estimation for exponential sums by approximate Prony method," *Signal Processing*, vol. 90, no. 5, pp. 1631–1642, 2010.

[57] S. Kunis, T. Peter, T. Römer, and U. von der Ohe, "A multivariate generalization of Prony's method," *Linear Algebra and its Applications*, vol. 490, 2016.

[58] I. N. Herstein, *Topics in algebra*. John Wiley & Sons, 2006.

[59] V. M. Preciado and A. Jadbabaie, "Moment-based spectral analysis of large-scale networks using local structural information," *IEEE/ACM Transactions on Networking*, vol. 21, no. 2, pp. 373–382, 2013.

[60] V. M. Preciado, A. Jadbabaie, and G. C. Verghese, "Structural analysis of Laplacian spectral properties of large-scale networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 9, pp. 2338–2343, 2013.

[61] X. Chen, M. Ogura, and V. M. Preciado, "Bounds on the spectral radius of digraphs from subgraph counts," *SIAM Journal on Matrix Analysis and Applications*, vol. 41, no. 2, pp. 525–553, 2020.

[62] F. Barreras, M. Hayhoe, H. Hassani, and V. M. Preciado, "Measure-theoretic bounds on the spectral radius of graphs from walks," *Linear Algebra and its Applications*, vol. 625, pp. 126–145, 2021.

[63] T. J. Aprille and T. N. Trick, "Steady-state analysis of nonlinear circuits with periodic inputs," *Proceedings of the IEEE*, vol. 60, no. 1, pp. 108–114, 1972.

[64] F. Colonius, *Optimal periodic control*. Springer, 2006, vol. 1313.

[65] J. P. Hespanha, *Linear systems theory*. Princeton University Press, 2018.

[66] K. B. Petersen, M. S. Pedersen, *et al.*, *The Matrix Cookbook*. Technical University of Denmark, 2008, vol. 7, p. 510.

[67] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[68] A. Narayanan and V. Shmatikov, "De-anonymizing social networks," in *30th IEEE Symp. on Sec. and Priv.*, 2009, pp. 173–187.

[69] Y. Zhang, "Browser-oriented universal cross-site recommendation and explanation based on user browsing logs," in *Proc. ACM Conf. on Recommender Sys.*, ACM, 2014, pp. 433–436.

[70] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *IEEE Conf. on Comp. Vision and Patt. Recognition*, IEEE, vol. 1, 2005, pp. 26–33.

[71] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *IEEE Conf. on Computer Vision*, IEEE, vol. 2, 2005, pp. 1482–1489.

[72] J. H. Hays, M. Leordeanu, A. A. Efros, and Y. Liu, "Discovering texture regularity via higher-order matching," in *9th European conference on computer vision*, 2006, pp. 522–535.

[73] V. Saraph and T. Milenković, "MAGNA: Maximizing accuracy in global network alignment," *Bioinformatics*, vol. 30, pp. 2931–2940, 2014.

[74] E. Kazemi, H. Hassani, M. Grossglauser, and H. P. Modarres, "PROPER: Global protein interaction network alignment through percolation matching," *BMC bioinformatics*, vol. 17, no. 1, p. 527, 2016.

[75] N. Malod-Dognin, K. Ban, and N. Pržulj, "Unified alignment of protein-protein interaction networks," *Scientific Reports*, 2017.

[76] R. Sharan, S. Suthram, R. M. Kelley, *et al.*, "Conserved patterns of protein interaction in multiple species," *Proc. Nat. Acad. Sciences*, vol. 102, no. 6, pp. 1974–1979, 2005.

[77] J. Ni, H. Tong, W. Fan, and X. Zhang, "Inside the atoms: Ranking on a network of networks," in *Proc. ACM SIGKDD Int. Conf. Know. Disc. and Data Mining*, ACM, 2014, pp. 1356–1365.

[78] K. Zetter, "Arvind Narayanan isn't anonymous, and neither are you," *Wired*, Jun. 18, 2012. [Online]. Available: https://www.wired.com/2012/06/wmw-arvind-narayanan.

[79] L. Yartseva and M. Grossglauser, "On the performance of percolation graph matching," in *Proc. 1st ACM Conf. Online Soc. Netw.*, ACM, 2013, pp. 119–130.

[80] O. Kuchaiev and N. Pržulj, "Integrative network alignment reveals large regions of global network similarity in yeast and human," *Bioinformatics*, vol. 27, no. 10, pp. 1390–1396, 2011.

[81] R. Patro and C. Kingsford, "Global network alignment using multiscale spectral signatures," *Bioinformatics*, 2012.

[82] S. Zhang and H. Tong, "FINAL: Fast attributed network alignment," in *Proc. ACM SIGKDD Int. Conf. Know. Disc. and Data Mining*, 2016, pp. 1345–1354.

[83] O. Kuchaiev, T. Milenković, V. Memišević, W. Hayes, and N. Pržulj, "Topological network alignment uncovers biological function and phylogeny," *Journ. Royal Society Interface*, 2010.

[84] E. Mossel and J. Xu, "Seeded graph matching via large neighborhood statistics," *arXiv preprint arXiv:1807.10262*, 2018.

[85] S. Feizi, G. Quon, M. R. Mendoza, M. Médard, M. Kellis, and A. Jadbabaie, "Spectral alignment of networks," *arXiv preprint arXiv:1602.04181*, 2016.

[86] A. Yaşar and Ü. V. Çatalyürek, "An iterative global structure-assisted labeled network aligner," *Proc. ACM SIGKDD Int. Conf. Know. Disc. and Data Mining*, 2018.

[87] M. Heimann, H. Shen, and D. Koutra, "REGAL: Representation learning-based graph alignment," *arXiv preprint arXiv:1802.06257*, 2018.

[88] S. Janson, T. Łuczak, T. Turova, T. Vallier, *et al.*, "Bootstrap percolation on the random graph $G_{n,p}$," *Annals of Applied Prob.*, vol. 22, no. 5, pp. 1989–2047, 2012.

[89] E. Kazemi, S. H. Hassani, and M. Grossglauser, "Growing a graph matching from a handful of seeds," *Proceedings of the VLDB Endowment*, 2015, ISSN: 21508097.

[90] P. Pedarsani and M. Grossglauser, "On the privacy of anonymized networks," in *Proc. ACM SIGKDD Int. Conf. Know. Disc. and Data Mining*, ACM, 2011, pp. 1235–1243.

[91] R. Singh, J. Xu, and B. Berger, "Pairwise global alignment of protein interaction networks by matching neighborhood topology," in *Int. Conf. on Research in Computational Molecular Biology*, Springer, 2007, pp. 16–31.

[92] M. Aizenman and J. L. Lebowitz, "Metastability effects in bootstrap percolation," *Journal of Physics A: Mathematical and General*, vol. 21, no. 19, p. 3801, 1988.

[93] T. Tao, *Topics in random matrix theory*. American Mathematical Soc., 2012, vol. 132.

[94] E. Kazemi and M. Grossglauser, "Mpgm: Scalable and accurate multiple network alignment," *arXiv preprint arXiv:1804.10029*, 2018.

[95] B. Rozemberczki, R. Davies, R. Sarkar, and C. Sutton, "GEMSEC: Graph embedding with self clustering," *arXiv preprint arXiv:1802.03997*, 2018.

[96] L. A. Zager and G. C. Verghese, "Graph similarity scoring and matching," *Appl. Math. Letters*, vol. 21, 2008.

[97] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambridge University Press, 1994.

[98] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comp. Networks and ISDN Sys.*, vol. 30, no. 1-7, pp. 107–117, 1998.

[99] I. Poulakakis, G. F. Young, L. Scardovi, and N. E. Leonard, "Information centrality and ordering of nodes for accuracy in noisy decision-making networks," *IEEE Trans. Auto. Cont.*, vol. 61, no. 4, pp. 1040–1045, 2016.

[100] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997.

[101] Y. López, K. Nakai, and A. Patil, "Hitpredict version 4: Comprehensive reliability scoring of physical protein–protein interactions from more than 100 species," *Database*, 2015.

[102] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 1, p. 2, 2007.

[103] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[104] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," *Advances in neural information processing systems*, vol. 29, 2016.

[105] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[106] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," in *Conference on robot learning*, PMLR, 2020, pp. 671–682.

[107] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[108] J. Zhou, G. Cui, S. Hu, *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[109] C. Giusti, R. Ghrist, and D. S. Bassett, "Two's company, three (or more) is a simplex," *Journal of computational neuroscience*, vol. 41, no. 1, pp. 1–14, 2016.

[110] S. Klamt, U.-U. Haus, and F. Theis, "Hypergraphs and cellular networks," *PLoS computational biology*, vol. 5, no. 5, e1000385, 2009.

[111] K. F. Kee, L. Sparks, D. C. Struppa, and M. Mannucci, "Social groups, social media, and higher dimensional social structures: A simplicial model of social aggregation for computational communication research," *Communication Quarterly*, vol. 61, no. 1, pp. 35–58, 2013.

[112] M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra, "Signal processing on higher-order networks: Livin'on the edge... and beyond," *Signal Processing*, vol. 187, p. 108 149, 2021.

[113] V. Salnikov, D. Cassese, and R. Lambiotte, "Simplicial complexes and complex systems," *European Journal of Physics*, vol. 40, no. 1, p. 014 001, 2018.

[114] M. T. Schaub, A. R. Benson, P. Horn, G. Lippner, and A. Jadbabaie, "Random walks on simplicial complexes and the normalized Hodge 1-Laplacian," *SIAM Review*, vol. 62, no. 2, pp. 353–391, 2020.

[115] T. M. Roddenberry, N. Glaze, and S. Segarra, "Principled simplicial neural networks for trajectory prediction," in *International Conference on Machine Learning*, PMLR, 2021, pp. 9020–9029.

[116] M. Hajij, G. Zamzmi, and X. Cai, "Simplicial complex representation learning," *arXiv preprint arXiv:2103.04046*, 2021.

[117] C. Bodnar, F. Frasca, Y. Wang, *et al.*, "Weisfeiler and Lehman go topological: Message passing simplicial networks," in *International Conference on Machine Learning*, PMLR, 2021, pp. 1026–1037.

[118] M. Yang, E. Isufi, and G. Leus, "Simplicial convolutional neural networks," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2022, pp. 8847–8851.

[119] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3558–3565.

[120] Y. Gao, Y. Feng, S. Ji, and R. Ji, "HGNN$^+$: General hypergraph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[121] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognition*, vol. 110, p. 107 637, 2021.

[122] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "HyperGCN: A new method for training graph convolutional networks on hypergraphs," *Advances in neural information processing systems*, vol. 32, 2019.

[123] Y. Dong, W. Sawin, and Y. Bengio, "HNHN: Hypergraph networks with hyperedge neurons," *arXiv preprint arXiv:2006.12278*, 2020.

[124] S. Barbarossa and S. Sardellitti, "Topological signal processing over simplicial complexes," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2992–3007, 2020.

[125] B. Grünbaum, "Nerves of simplicial complexes," *Aequationes Mathematicae*, vol. 4, no. 1, pp. 63–73, 1970.

[126] O. Knill, "The energy of a simplicial complex," *Linear Algebra and its Applications*, vol. 600, pp. 96–129, 2020.

[127] B. Osting, S. Palande, and B. Wang, "Spectral sparsification of simplicial complexes for clustering and label propagation," *arXiv preprint arXiv:1708.08436*, 2017.

[128] N. Keriven, A. Bietti, and S. Vaiter, "On the universality of graph neural networks on large random graphs," *Advances in Neural Information Processing Systems*, vol. 34, pp. 6960–6971, 2021.

[129] F. Chung, L. Lu, and V. Vu, "The spectra of random graphs with given expected degrees," *Internet Mathematics*, vol. 1, no. 3, pp. 257–275, 2004.

[130] L. Erdős, H.-T. Yau, and J. Yin, "Rigidity of eigenvalues of generalized Wigner matrices," *Advances in Mathematics*, vol. 229, no. 3, pp. 1435–1515, 2012.

[131] A. Chakrabarty, R. S. Hazra, F. den Hollander, and M. Sfragara, "Spectra of adjacency and Laplacian matrices of inhomogeneous Erdős–Rényi random graphs," *Random matrices: Theory and applications*, vol. 10, no. 01, p. 2 150 009, 2021.

[132] R. Vizuete, F. Garin, and P. Frasca, "The Laplacian spectrum of large graphs sampled from graphons," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1711–1721, 2021.

[133] W. H. Organization, "Novel coronavirus (2019-nCoV): Situation report 1," World Health Organization, Technical documents, 2020-01-20.

[134] W. H. Organization, "Novel coronavirus (2019-nCoV): Weekly epidemiological update, 21 September 2020," World Health Organization, Technical documents, 2020-09-20.

[135] S. at-home orders across the country., *COVID-19 community mobility reports*, Available at https://www.nbcnews.com/health/health-news/here-are-stay-home-orders-across-country-n1168736, 2020.

[136] J. Kaplan and L. Frias, "Our ongoing list of how countries are reopening, and which ones remain under lockdown," *BUSINESS INSIDER*, 2020. [Online]. Available: \url{https://www.businessinsider.com/countries-on-lockdown-coronavirus-italy-2020-3}.

[137] M. A. Achterberg, B. Prasse, L. Ma, S. Trajanovski, M. Kitsak, and P. Van Mieghem, "Comparing the accuracy of several network-based COVID-19 prediction algorithms," *International Journal of Forecasting*, 2020.

[138] M. A. Bhouri, F. S. Costabal, H. Wang, *et al.*, "COVID-19 dynamics across the US: A deep learning study of human mobility and social behavior," *medRxiv*, 2020.

[139] S. Y. Chang, E. Pierson, P. W. Koh, *et al.*, "Mobility network modeling explains higher SARS-CoV-2 infection rates among disadvantaged groups and informs reopening strategies," *medRxiv*, 2020.

[140] W. Van den Broeck, C. Gioannini, B. Gonçalves, M. Quaggiotto, V. Colizza, and A. Vespignani, "The GLEaMviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale," *BMC infectious diseases*, vol. 11, no. 1, p. 37, 2011.

[141] D. J. Watts, R. Muhamad, D. C. Medina, and P. S. Dodds, "Multiscale, resurgent epidemics in a hierarchical metapopulation model," *Proceedings of the National Academy of Sciences*, vol. 102, no. 32, pp. 11 157–11 162, 2005.

[142] M. Hayhoe, F. Alajaji, and B. Gharesifard, "A Polya contagion model for networks," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1998–2010, 2018.

[143] D. Balcan, B. Gonçalves, H. Hu, J. J. Ramasco, V. Colizza, and A. Vespignani, "Modeling the spatial spread of infectious diseases: The global epidemic and mobility computational model," *Journal of computational science*, vol. 1, no. 3, pp. 132–145, 2010.

[144] N. Ferguson, D. Laydon, G. Nedjati Gilani, *et al.*, *Report 9: Impact of non-pharmaceutical interventions (NPIs) to reduce COVID-19 mortality and healthcare demand*, 2020.

[145] F. Piguillem and L. Shi, "Optimal covid-19 quarantine and testing policies," *The Economic Journal*, vol. 132, no. 647, pp. 2534–2562, 2022.

[146] M. M. Morato, S. B. Bastos, D. O. Cajueiro, and J. E. Normey-Rico, "An optimal predictive control strategy for COVID-19 (SARS-CoV-2) social distancing policies in Brazil," *Annual Reviews in Control*, vol. 50, pp. 417–431, 2020.

[147] A. Aleta, D. Martín-Corral, A. P. y Piontti, *et al.*, "Modelling the impact of testing, contact tracing and household quarantine on second waves of COVID-19," *Nature Human Behaviour*, vol. 4, no. 9, pp. 964–971, 2020.

[148] L. Lorch, W. Trouleau, S. Tsirtsis, A. Szanto, B. Schölkopf, and M. Gomez-Rodriguez, "A spatiotemporal epidemic model to quantify the effects of contact tracing, testing, and containment," *arXiv preprint arXiv:2004.07641*, 2020.

[149] J. R. Birge, O. Candogan, and Y. Feng, "Controlling epidemic spread: Reducing economic losses with targeted closures," *University of Chicago, Becker Friedman Institute for Economics Working Paper No. 2020-57.*, 2020.

[150] A. R. Hota, J. Godbole, P. Bhariya, and P. E. Paré, "A closed-loop framework for inference, prediction and control of SIR epidemics on networks," *arXiv preprint arXiv:2006.16185*, 2020.

[151] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Optimal resource allocation for control of networked epidemic models," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 159–169, 2015.

[152] V. M. Preciado, M. Zargham, C. Enyioha, A. Jadbabaie, and G. J. Pappas, "Optimal vaccine allocation to control epidemic outbreaks in arbitrary networks," in *52nd IEEE conference on decision and control*, IEEE, 2013, pp. 7486–7491.

[153] V. M. Preciado, M. Zargham, C. Enyioha, A. Jadbabaie, and G. J. Pappas, "Optimal resource allocation for network protection against spreading processes," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 1, pp. 99–108, 2014.

[154] P. Van Mieghem, J. Omic, and R. Kooij, "Virus spread in networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 17, no. 1, pp. 1–14, 2009.

[155] S. Eshghi, M. H. R. Khouzani, S. Sarkar, and S. S. Venkatesh, "Optimal patching in clustered epidemics of malware," *IEEE Trans. Network*, vol. 24, no. 1, pp. 283–298, 2015.

[156] M. H. R. Khouzani, S. S. Venkatesh, and S. Sarkar, "Market-based control of epidemics," in *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, 2011, pp. 314–320.

[157] X. Yan and Y. Zou, "Optimal and sub-optimal quarantine and isolation control in sars epidemics," *Mathematical and Computer Modelling*, vol. 47, no. 1-2, pp. 235–245, 2008.

[158] C. E. Garcia, D. M. Prett, and M. Morari, "Model predictive control: Theory and practice—a survey," *Automatica*, vol. 25, no. 3, pp. 335–348, 1989.

[159] J. Köhler, L. Schwenkel, A. Koch, J. Berberich, P. Pauli, and F. Allgöwer, "Robust and optimal predictive control of the covid-19 outbreak," *Annual Reviews in Control*, 2020.

[160] Google, *Covid-19 community mobility reports*, Available at https://google.com/covid19/mobility/, Accessed: 10-31-2020, 2020.

[161] N. Y. Times, *Coronavirus (COVID-19) data in the United States*, Available at https://www.nytimes.com/interactive/2020/us/coronavirus-us-cases.html, Accessed: 10-31-2020, 2020.

[162] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[163] M. Chiang, *Geometric programming for communication systems*. Now Publishers Inc, 2005.

[164] M. Ogura, M. Kishida, and J. Lam, "Geometric programming for optimal positive linear systems," *IEEE Transactions on Automatic Control*, 2019.

[165] J. Dahl and E. D. Andersen, "A primal-dual interior-point algorithm for nonsymmetric exponential-cone optimization," *Optimization Online*, 2019.

[166] F. Brauer, C. Castillo-Chavez, and Z. Feng, *Mathematical models in epidemiology*. Springer, 2019.

[167] M. Martcheva, *An introduction to mathematical epidemiology*. Springer, 2015, vol. 61.

[168] G. Giordano, F. Blanchini, R. Bruno, *et al.*, "Modelling the covid-19 epidemic and implementation of population-wide interventions in italy," *Nature medicine*, vol. 26, no. 6, pp. 855–860, 2020.

[169] Y. Bai, L. Yao, T. Wei, *et al.*, "Presumed asymptomatic carrier transmission of covid-19," *Jama*, vol. 323, no. 14, pp. 1406–1407, 2020.

[170] M. Gandhi, D. S. Yokoe, and D. V. Havlir, *Asymptomatic transmission, the Achilles' heel of current strategies to control covid-19*, 2020.

[171] S. T. Ali, L. Wang, E. H. Y. Lau, *et al.*, "Serial interval of SARS-CoV-2 was shortened over time by nonpharmaceutical interventions," *Science*, vol. 369, no. 6507, pp. 1106–1109, 2020.

[172] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[173] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Autograd: Effortless gradients in NumPy," in *ICML 2015 AutoML Workshop*, vol. 238, 2015, p. 5.

[174] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[175] M. Day, *COVID-19: Four fifths of cases are asymptomatic, China figures indicate*, 2020.

[176] H. Nishiura, T. Kobayashi, T. Miyama, *et al.*, "Estimation of the asymptomatic ratio of novel coronavirus infections (COVID-19)," *International journal of infectious diseases*, vol. 94, p. 154, 2020.

[177] S. A. Lauer, K. H. Grantz, Q. Bi, *et al.*, "The incubation period of coronavirus disease 2019 (COVID-19) from publicly reported confirmed cases: Estimation and application," *Annals of internal medicine*, vol. 172, no. 9, pp. 577–582, 2020.

[178] X. He, E. H. Y. Lau, P. Wu, *et al.*, "Temporal dynamics in viral shedding and transmissibility of COVID-19," *Nature medicine*, vol. 26, no. 5, pp. 672–675, 2020.

[179] H. Rahmandad, T. Y. Lim, and J. Sterman, "Estimating COVID-19 under-reporting across 86 nations: Implications for projections and control," *medRxiv preprint*, 2020.

[180] S. Pei, S. Kandula, and J. Shaman, "Differential effects of intervention timing on COVID-19 spread in the united states," *medRxiv*, 2020.

[181] R. Woelfel, V. M. Corman, W. Guggemos, *et al.*, "Clinical presentation and virological assessment of hospitalized cases of coronavirus disease 2019 in a travel-associated transmission cluster," *MedRxiv*, 2020.

[182] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[183] T. A. Schieber, L. Carpi, A. Díaz-Guilera, P. M. Pardalos, C. Masoller, and M. G. Ravetti, "Quantification of network structural dissimilarities," *Nature Communications*, vol. 8, 2017.

[184] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Physical Review E*, vol. 74, no. 3, p. 036 104, 2006.

[185] V. Batagelj and A. Mrvar, *Pajek datasets*, 2006. [Online]. Available: http://vlado.fmf.uni-lj.si/pub/networks/data/.

[186] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 us election: Divided they blog," in *Proceedings of the 3rd international workshop on Link discovery*, ACM, 2005, pp. 36–43.

[187] A. Zomorodian, "Fast construction of the Vietoris-Rips complex," *Computers & Graphics*, vol. 34, no. 3, pp. 263–271, 2010.

[188] Y. Liu, L. Zhu, P. Szekely, A. Galstyan, and D. Koutra, "Coupled clustering of time-series and networks," in *Proceedings of the 2019 SIAM International Conference on Data Mining*, SIAM, 2019.