Domain-based Frameworks and Embeddings for Dynamics over Networks

Bijaya Adhikari

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Application

B. Aditya Prakash (Chair)
Naren Ramakrishnan
Madhav Marathe
Chandan K. Reddy
Jimeng Sun

April 28, 2020
Blacksburg, Virginia

Keywords: data mining, networks, time-series, domain-based learning, graph summarization, network state inference, and data driven epidemiology

Domain-based Frameworks and Embeddings for Dynamics over Networks

Bijaya Adhikari

ABSTRACT (Academic)

Broadly this thesis looks into network and time-series mining problems pertaining to dynamics over networks in various domains. Which locations and staff should we monitor in order to detect *C. Difficile* outbreaks in hospitals? How do we predict the peak intensity of the influenza incidence in an interpretable fashion? How do we infer the states of all nodes in a critical infrastructure network where failures have occurred? Leveraging domain-based information should make it is possible to answer these questions. However, several new challenges arise, such as (a) *presence of more complex dynamics*. The dynamics over networks that we consider are complex. For example, *C. Difficile* spreads via both people-to-people and surface-to-people interactions and correlations between failures in critical infrastructures go beyond the network structure and depend on the geography as well. Traditional approaches either rely on models like Susceptible Infectious (SI) and Independent Cascade (IC) which are too restrictive because they focus only on single pathways or do not incorporate the model at all, resulting in sub-optimality. (b) *data sparsity*. Additionally, the data sparsity still persists in this space. Specifically, it is difficult to collect the exact state of each node in the network as it is high-dimensional and difficult to directly sample from. (c) *mismatch between data and process*. In many situations, the underlying dynamical process is unknown or depends on a mixture of several models. In such cases, there is a mismatch between the data collected and the model representing the dynamics. For example, the weighted influenza like illness (wILI) count released by the CDC, which is meant to represent the raw fraction of total population infected by influenza, actually depends on multiple factors like the number of health-care providers reporting the number and public tendency to seek medical advice. In such cases, methods which generalize well to unobserved (or unknown) models are required. Current approaches often fail in tackling these challenges as they either rely on restrictive models, require large volume of data, and/or work only for predefined models.

In this thesis, we propose to leverage domain-based *frameworks*, which include novel models and analysis techniques, and domain-based *low dimensional representation learning* to tackle the challenges mentioned above for networks and time-series mining tasks. By developing novel frameworks, we can capture the complex dynamics accurately and analyze them more efficiently. For example, to detect *C. Difficile* outbreaks in a hospital setting, we use a two-mode disease model to capture multiple pathways of outbreaks and discrete lattice-based optimization framework. Similarly, we propose an information theoretic framework which includes geographically correlated failures in critical infrastructure networks to infer the status of the network components. Moreover, as we use more realistic frameworks to accurately capture and analyze the mechanistic processes themselves, our approaches are effective even with sparse data. At the same time, learning low-dimensional domain-aware

embeddings capture domain specific properties (like incidence-based similarity between historical influenza seasons) more efficiently from sparse data, which is useful for subsequent tasks. Similarly, since the domain-aware embeddings capture the model information directly from the data without any modeling assumptions, they generalize better to new models.

Our domain-aware frameworks and embeddings enable many applications in critical domains. For example, our domain-aware frameworks for *C. Difficile* allows different monitoring rates for people and locations, thus detecting more than 95% of outbreaks. Similarly, our framework for product recommendation in e-commerce for queries with sparse engagement data resulted in a 34% improvement over the current Walmart.com search engine. Similarly, our novel framework leads to a near optimal algorithms, with additive approximation guarantee, for inferring network states given a partial observation of the failures in networks. Additionally, by exploiting domain-aware embeddings, we outperform non-trivial competitors by up to 40% for influenza forecasting. Similarly, domain-aware representations of subgraphs helped us outperform non-trivial baselines by up to 68% in the graph classification task. We believe our techniques will be useful for variety of other applications in many areas like social networks, urban computing, and so on.

Domain-based Frameworks and Embeddings for Dynamics over Networks

Bijaya Adhikari

ABSTRACT (General Audience)

Which locations and staff should we monitor to detect pathogen outbreaks in hospitals? How do we predict the peak intensity of the influenza incidence? How do we infer the failures in water distribution networks? These are some of the questions on dynamics over networks discussed in this thesis. Here, we leverage the domain knowledge to answer these questions. Specifically, we propose (a) novel optimization frameworks where we exploit domain knowledge for tractable formulations and near-optimal algorithms, and (b) low dimensional representation learning where we design novel neural architectures inspired by domain knowledge. Our frameworks capture the complex dynamics accurately and help analyze them more efficiently. At the same time, our low-dimensional embeddings capture domain specific properties more efficiently from sparse data, which is useful for subsequent tasks. Similarly, our domain-aware embeddings are inferred directly from the data without any modeling assumptions, hence they generalize better. The frameworks and embeddings we develop enable many applications in several domains. For example, our domain-aware framework for outbreak detection in hospitals has more than 95% accuracy. Similarly, our framework for product recommendation in e-commerce for queries with sparse data resulted in a 34% improvement over state-of-the-art e-commerce search engine. Additionally, our approach outperforms non-trivial competitors by up to 40% in influenza forecasting.

To my parents, Laxmi and Tanjan, and my lovely wife, Sonam.

Acknowledgments

First of all, I would like to thank my advisor B. Aditya Prakash for his support, guidance, and encouragement over the duration of my PhD. This thesis would not have been possible without him. I would also like to thank all the committee members: Naren Ramakrishnan and Chandan Reddy at Virginia Tech, Madhav Marathe at the University of Virginia, and Jimeng Sun at the University of Illinois Urbana-Champaign, for their invaluable suggestions to improve this thesis.

I was fortunate to work with an amazing group of collaborators: Naren Ramakrishnan, Anil Vullikanti, Bryan Lewis, Chris North, Srinivasan Venkatramanan, Yao Zhang, Sorour E. Amiri, Xinfeng Xu, Jose Jimenez, Aditya Bharadwaj, Karthik Subbian, Parikshit Sondhi, Mohit Sharma, Pavan Rangudu, Sathappan Muthiah, Mohhammad R. Islam, John Wenskovitch, Steve Han, Prathysuh Sambaturu, Alexander Rodriguez, Nikhil Rao and Liangyu Li. Thank you all. I would particularly like to thank Anil for his guidance over the years. Many chapters in this thesis could not have been completed without his thoughtful ideas and insights. I would also like to thank Naren for his mentorship in many of the deep learning works discussed in this thesis.

I would like to thank my friends, peers, and lab mates. Thank you for your feedback and help in improving my talks, papers, and slides. I am also grateful for two amazing internships at Walmartlabs and Amazon. I was fortunate to have Parikshit Sondhi as mentor at Walmartlabs and Liangyue li and Nikhil Rao and mentors and Karthik Subbian as a manager at Amazon.

Mom and Dad, thank you for your endless love and encouragement. I will always be indebted to the sacrifices you made for me. Finally, to my dear wife Sonam, thank you for unconditional support and limitless love.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Over the last few years, we have witnessed a tremendous increase in data collection. For example, consider the spread of influenza in the United States. The Centers for Disease Control and Prevention (CDC) employs an extensive surveillance network of healthcare providers throughout the country to collect the percentage of healthcare seekers who exhibit influenza like symptoms. CDC then processes the reported visitor percentages and release the weighted Influenza-like Illness (wILI) counts [41] every week at different geographical resolution including states level, HHS regional level, and the US national level. Let us now consider the spread of pathogens in a hospital. Due to availability of sensors such as Radio-frequency Identification Devices (RFIDs), mobility logs of who goes where in hospitals can be collected and recorded along with information on infections in people and contamination on surfaces [119]. These datasets help us get actionable insights to real world problems. Examples of such problems include how do we forecast the total number of infected individuals in an influenza epidemic? Which locations and health-care workers should we monitor to detect hospital acquired infections (HAIs) outbreaks? How do we determine the optimal time to distribute vaccines given that the outbreak has already started? These are very important questions (the current Covid-19 pandemic well highlights the need for answers).

Interestingly, the questions presented above can be abstracted as problems pertaining to dynamics over networks. Dynamics over networks are the processes which cause change in the states of the nodes in an underlying network [129, 165]. For example, pathogens like *C. Difficile* [168] spread over an underlying population contact network in hospitals changing states of individuals from healthy to infected. Many other important questions in public health (and other domains as we see later) can be expressed leveraging dynamics over networks.

Challenges: Several new challenges arise due to the increased complexity of incorporating the domain information and/or lack of data in the domain of concern. They are as follows:

a) *Presence of more complex dynamics.* The dynamics over networks we consider cannot

be readily captured by existing models. For example, *C. Difficile* spreads via multiple pathways of infections involving people-people as well as people-location interactions. Traditional approaches which often rely on restrictive models such as Independent Cascade (IC) or Susceptible Infectious (SI) result in sub-optimality as they focus only on single people-people infection pathways [142].

b) *Data sparsity.* Despite the increase in data collection in general, scarcity of data still persists due to observational constraints. For example, there are 2^n possible infection states (as each person can either be infected or healthy) of an underlying contact network in an epidemic outbreak. This data is high-dimensional and difficult to directly sample from.

c) *Mismatch between data and process.* The dynamical processes are not always easy to observe and this may lead to mismatch between the data collected and the dynamical model meant to represent the data generation process. For example, the weighted influenza like illness (wILI) count meant to represent the fraction of population infected by influenza actually depends on various factors like percentage of people seeking health-care and fraction of health-care providers in the surveillance network.

Dynamics over networks is not limited just to the public health and can be found in other domains as well. In fact, several important questions in various domains such as e-commerce, critical infrastructure and social media can also be expressed leveraging dynamics over networks. For example, we may ask how to infer the states of all the components of a power grid given a partial probes of the failed components? Here, the dynamics is the cascade of failures over the power-grid network. Similarly, problems on identifying users which facilitate flow of information in social media, determining most important queries in an e-commerce platform, and so on can be expressed using dynamics over networks. Indeed, the same challenges described above arise even in these settings.

Our Approach: In this thesis, we propose to overcome these challenges by leveraging domain knowledge. Specifically, we propose two-threads (a) *domain-aware optimization frameworks*, where we leverage domain knowledge (such as domain-specific dynamical models) to formulate tractable problems and propose near-optimal algorithms and (b) *domain-aware low dimensional representation learning*, where we design novel deep neural architectures to extract and exploit discriminating features from the data. Domain-aware frameworks enable better analysis of the actual underlying processes which generate the data. For example, to detect *C. Difficile* outbreaks in hospitals, we propose a discrete lattice-based optimization framework to complement a more accurate model, which captures multiple pathways of infections. Similarly, we incorporate geographically correlated failure model in an information theoretic optimization framework for failure inference in critical infrastructure networks. Additionally, as realistic frameworks allow one to analyze the mechanistic processes themselves at a finer resolution, it often leads to better generalization even with sparse data. On the other hand, learning domain-aware low-dimensional embeddings allows one to capture

domain specific properties more efficiently from sparse data, which is useful for downstream tasks. Similarly, since domain-aware embeddings capture information regarding the dynamical processes directly from the data, without any modeling assumptions, they generalize better to new or unknown processes.

Our Contributions and Impact: This thesis has made several contributions and has had major impacts. We proposed novel domain-aware frameworks and near-optimal algorithms as well as effective novel learning architectures which has enabled several new applications in public health, e-commerce, critical infrastructure, and social media. Our work on influenza forecasting has performed well in realtime challenges in past and is currently being leveraged for Covid-19 forecasting. Our work on outbreak monitoring in hospitals was featured in the PLOS complexity channel. Finally, several approaches presented here are at production in the industry.

Some Takeaways: The thesis is centered on leveraging domain knowledge for problems in dynamics over networks. A key lesson learnt in the early stages of building this thesis was that domain-specific characteristics are powerful and if incorporated properly could lead to useful tools. This can be done in multiple ways. Domain knowledge can be incorporated to formulate problems from the first principles and can be exploited for better algorithms. For example, formulation based on geographically correlated failure model specific to the critical infrastructure networks led to two chapters (Chapter 6 and 7) on missing failure inference in this thesis. Moreover, key domain-specific characteristics may inspire a better neural architecture. For example, the dynamic seasonal similarities between the historical influenza incidence curves motivated the architecture of `EpiDeep` in Chapter 10. Similarly, guidance from domain experts can be incorporated as constraints and loss function as demonstrated for epidemic forecasting in Chapter 11. These observations highlight a key point: domain knowledge, if incorporated correctly, leads to powerful approaches.

1.1 Thesis Statement and Structure

Our thesis statement is as follows:

> Incorporating domain-aware frameworks and low dimensional representations lead to more accurate algorithms and more effective learning architectures for dynamics-based networks and time-series mining tasks.

More specifically, we leverage domain knowledge to formulate novel problems in a tractable space and propose near-optimal algorithms for dynamics-based network and time-series mining tasks. In contrast to prior work, we consider realistic frameworks which enable us to capture and analyze more complex dynamics in various domains such as public health, e-commerce, and critical infrastructures in an efficient manner. This in turn, leads to effective algorithms. Similarly, we also propose novel deep neural architectures to learn domain-aware low dimensional representations of networks and time-series. Unlike prior work, our

approaches are effective, interpretable, and generalizable.

Following the thesis statement, we organize the thesis into two parts: (1) Domain-aware optimization frameworks, and (2) Domain-aware representation learning. The outline is shown in Table 1.1.

1.1.1 Thesis Structure

In the first part of the thesis, Chapters 3 to 7, we discuss our work on optimization frameworks. We then present our work on representation learning in Chapters 8 to 11. The organization of the chapters with respect to the thesis is presented below.

Table 1.1: Structure of the thesis with references to the chapters.

	Domain-aware optimization frameworks	Domain-aware representation learning
Selection and Projection	• Near-Optimal Monitoring of HAI Outbreaks (Chapter 3) • Graph-based E-Commerce Query Relations Mining (Chapter 4) • Temporal Network Summarization (Chapter 5)	• Deep Learning for Task-based Network Summarization (Chapter 8) • Sub2Vec: Deep Representation Learning for Subgraph (Chapter 9)
Prediction and Inference	• Near-Optimal Network State Inference using Probes (Chapter 6) • Network State Inference using Connectivity Queries (Chapter 7)	• EpiDeep: Deep Learning for Influenza Forecasting (Chapter 10) • Incorporating Guidance for Deep Epidemic Forecasting (Chapter 11)

In each part, we categorize our problems into two buckets, namely selection and projection tasks, and prediction and inference tasks. In the first bucket, we tackle problems such as selecting people and locations and the rates to monitor them to detect HAI outbreaks, projecting large networks into smaller spaces such that the dynamics-based characteristics is preserved and learning meaningful feature representation of cascades. In the second bucket, we study the problems on prediction and inference such as inferring missing infections given partial observations and forecasting unobserved future events. Note that the first bucket involves optimization over observed data, whereas the second bucked involves problems asking to predict unobserved data instances.

We begin the first part of our thesis with a discussion on domain-aware frameworks for selection and projection tasks. The first problem we tackle is on *C. Difficile* outbreak detection in hospitals (Chapter 3). We propose a near optimal approach to select both people and locations as 'sensors' and corresponding rates to monitor each, while maximizing the detection probability and minimizing the detection time. Next we present the problem of e-commerce query relation mining (Chapter 4). Here we begin by constructing meaningful networks from the user interaction log and formalize the query relation mining tasks as node selection tasks. We then leverages special characteristics exhibited by the networks

to design effective and efficient approaches to solve them. In Chapter 5, we move on to a network projection problem, namely temporal network summarization. Here we develop a scalable approach for large scale temporal network summarization while preserving its dynamical characteristics. Then, we move onto the optimization-frameworks for prediction and inference tasks. Here, we first study the problem of inferring network state given a partial probe in Chapter 6. We present near-optimal algorithm with additive approximation guarantee which accurately infers the network states. We then extend this to the setting where the probes are based on the network connectivity as well as point failures in Chapter 7.

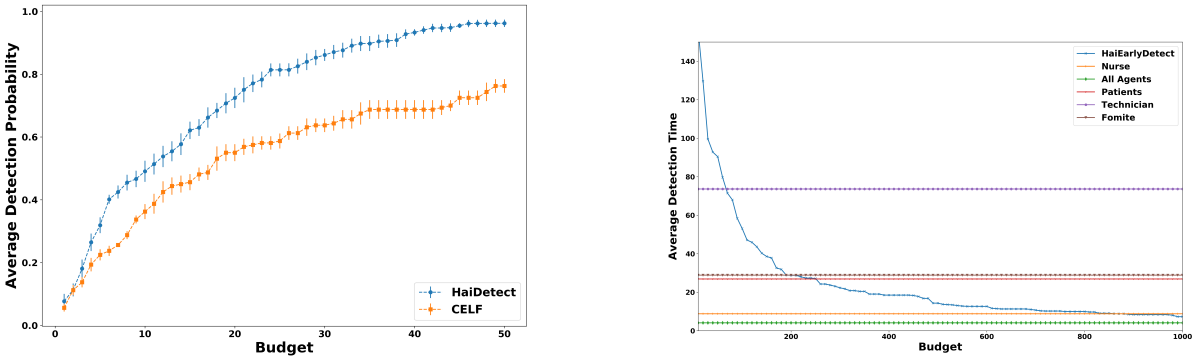
In the second part of the thesis, we present our work on domain-aware representation learning. Moving on from the model-based network projection, we study representation-based network projection problems including deep reinforcement learning for automatic generation task-based network summaries in Chapter 8 and learning low-dimensional representations of group of graphs/subgraphs into a continuous feature space in Chapter 9. We then move on to the prediction problems. Here we present deep learning approach for predicting influenza incidence and various influenza metrics defined by the Centers for Disease Control and Prevention (Chapter 10). In Chapter 11, we extend our influenza forecasting framework to incorporate guidance from epidemiological experts. Finally, we present future works and conclusions in chapter 12.

1.2 Summary of the Work

1.2.1 Part I: Domain-aware optimization frameworks

In this section, we discuss domain-aware frameworks for selection projection tasks as well as prediction and inference tasks. The problems we study include outbreak detection in hospitals, e-commerce query relation mining based on node selection tasks, and network state-inference.

Chapter 3 (Near-Optimal Monitoring of HAI Outbreaks) [4]. CDC reports that on any given day one in twenty-five hospital patients are infected by Hospital Acquired Infections (HAIs). Early detection of possible HAI outbreaks helps practitioners implement countermeasures before an infection spreads extensively. In this chapter, we develop an efficient data and model driven method to detect outbreaks with high accuracy. We leverage mechanistic modeling of *C. difficile* infection, a major HAI disease, to simulate its spread in a hospital wing, and design efficient near-optimal algorithms to select a sensor set to monitor using an optimization formulation. Our extensive experiments show that our senors detect up to 95% of 'future' *C. difficile* outbreaks. We design our method by incorporating specific hospital practices (like swabbing for infections) as well. As a result, our method outperforms state-of-the-art algorithms for finding sensor sets (See Figure 1.1).



(a) Detection likelihood (b) Detection time

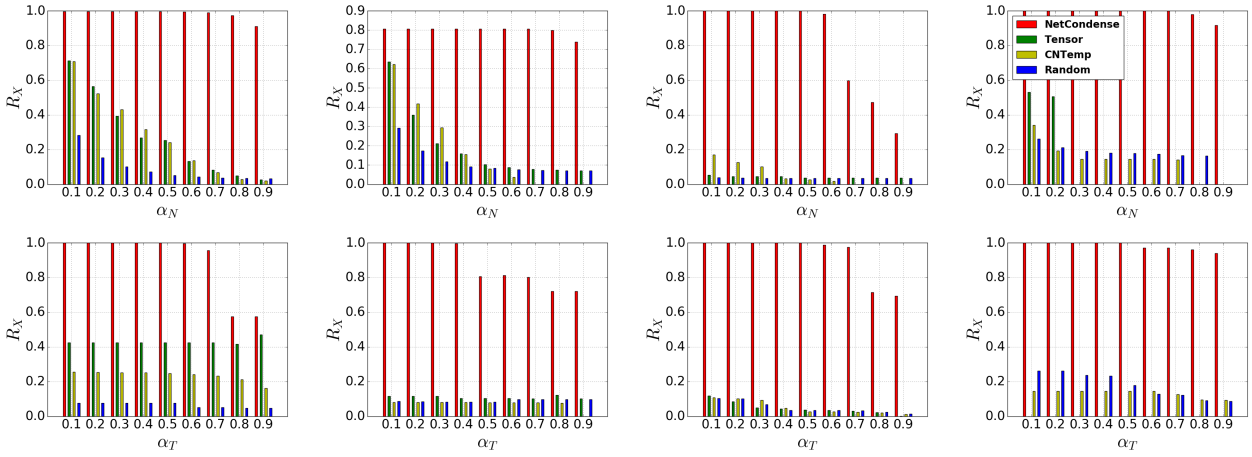
Figure 1.1: Our domain-aware framework detects upto 95% of *C. Difficile* outbreaks and reduces the detection time to a single day. Efficiency of our approaches in detecting unobserved HAI outbreaks in terms of (a) detection likelihood and (b) detection time.

Chapter 4 (Graph-based E-commerce Query Relations Mining) [7]. In the previous chapter, we presented optimization for selecting nodes and rates to monitor them for HAI outbreak detection. In this chapter, we focus on nodes selection tasks for query relations mining. Customer interactions begin with the submission of a query formulated based on an initial product intent, followed by a sequence of product engagement and query reformulation actions. The log of customer interaction contains rich data. In this work, we mine query-to-query and query-to-item relations by constructing Customer Interaction Networks (CINs) from Walmart.com's product search logs.

We construct various CINs each of which capture one aspect of the log at a time. We then study the properties of CINs. We observe that the properties exhibited by CINs make it possible to mine intent relationships between queries based purely on their structural information. We show how these relations can be exploited for a) clustering queries based on intents, b) significantly improve search quality for poorly performing queries, and c) identify the most influential (aka. 'critical') queries whose performance have the highest impact on performance of other queries. We solve these problems by designing node selection based techniques.

Chapter 5 (Temporal Network Summarization) [10, 9].

In this chapter, we focus on optimization for network projection. Specifically, we study the novel problem of obtaining a condensed representation of a temporal network which is diffusion-equivalent to the original one. First, we formulate a well-founded and general temporal-network condensation problem based on the so-called system-matrix of the network, which characterizes the diffusive property of the temporal network. We then propose

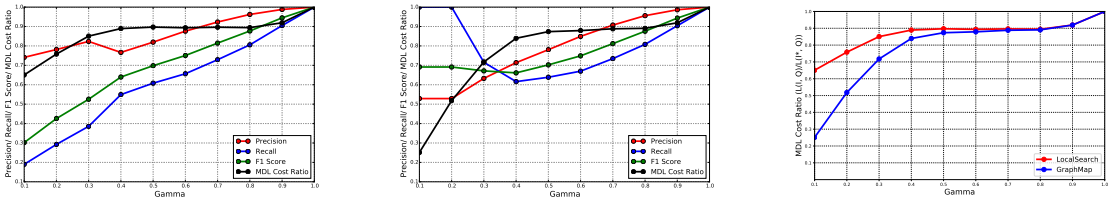


(a) `Arxiv` (b) `ProsperLoan` (c)`WikiTalk` (d) `DBLP`

Figure 1.2: Our framework allows large temporal networks to be reduced by up 60% and 70% in terms of number of nodes and time-stamps respectively, while preserving the dynamics-based characteristics. $R_{\mathbf{X}} = \lambda_{\mathbf{X}}^{\text{cond}}/\lambda_{\mathbf{X}}$ vs α_N (top row, $\alpha_T = 0.5$) and vs α_T (bottom row, $\alpha_N = 0.5$).

NetCondense, a scalable and effective algorithm which solves this problem using careful transformations in sub-quadratic running time, and linear space complexities. Our extensive experiments show that we can reduce the size of large real temporal networks (from multiple domains such as social, co-authorship and email) significantly without much loss of information as shown in Figure 1.2. We also show the wide-applicability of NetCondense by leveraging it for several tasks: for example, we use it to understand, explore and visualize the original datasets and to also speed-up algorithms for the influence-maximization and event detection problems on temporal networks.

Chapter 6 (Near-Optimal Network State Inference using Probes) [6].



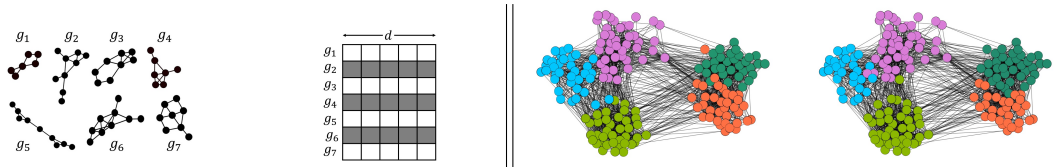
(a) (b) (c)

Figure 1.3: Our novel framework is highly effective in inferring network states. It infers failed nodes with F-1 score of 0.7 even when only 10% of the failures are observed. Performance of the (a) LocalSearch and (b) GraphMap on the JAM dataset with PlainCF probabilities. The MDL costs of the solutions is shown in (c).

In this chapter, we study the problem of network state inference. In many applications, such as the Internet and infrastructure networks, nodes fail or get congested dynamically. We study the problem of inferring the states of all the nodes, when only a sample of the failures is known and there exist correlations between node failures/congestion in networks. We formalize this as the GraphStateInf problem, using the Minimum Description Length (MDL) principle. We propose the GraphMap algorithm for minimizing the MDL cost, and show that it gives an additive approximation, relative to the optimal. We evaluate our methods on synthetic and real datasets, which includes one from WAZE which gives traffic incident reports for the city of Boston. We find that our method gives promising results in recovering the missing failures as shown in Figure 1.3.

Chapter 7 (Network State Inference using Connectivity Queries) [196]. In this chapter, we study the problem of inferring failed components in an infrastructure network given a sample of reachable nodes from a set of supply nodes. We formalize the problem using the MDL principle and propose two greedy heuristics to solve the problem. Our experiments show that our approach is indeed able to infer the failed components, especially the critical ones effecting the performance of the entire system.

1.2.2 Part II: Domain-aware representation learning



(a) Input subgraphs (b) Embeddings (c) Ground Truth (d) Our Approach

Figure 1.4: Embeddings learned by our approach are highly effective for downstream tasks like community detection. (a) and (b) An overview of `Sub2Vec`. Our input is a set of subgraphs $\mathcal{S}$. `Sub2Vec` learns $\mathcal{D}$ dimensional feature embedding of each subgraph. (c)-(d) Leveraging embeddings learned by `Sub2Vec` for community detection. (c) Communities in `School` network (different colors represent different communities). Communities discovered via `Sub2Vec` closely matches the ground truth.

Chapter 8 (Deep Learning for Task-based Network Summarization) [17]. Chapter 3 presented an optimization based approach for summarizing network with respect to the diffusive properties. However, preserving just the diffusive properties may not be sufficient for other tasks as different summary could be ideal for different tasks.

Hence, in this chapter we explore a promising alternative approach for automatically learning summaries for a given task instead. We propose NetGist, a framework which automatically learns *how* to generate a summary for a given task on a given network. In addition to gener-

ating the required summary, this also allows us to *reuse* the learned process on other similar networks. We formulate a novel task-based graph summarization problem and leverage reinforcement learning to design a flexible framework for our solution. Via extensive experiments, we show that NetGist robustly and effectively learns meaningful summaries, and helps solve challenging problems, and aids in complex task-based sense-making of networks.

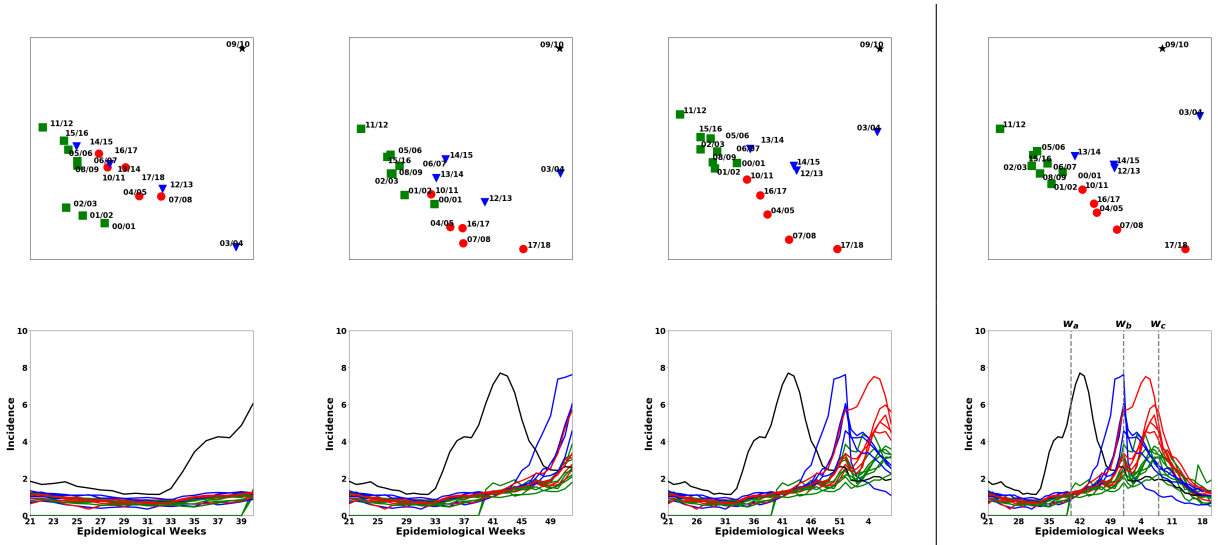
Chapter 9 (Sub2Vec: Deep Representation Learning for Subgraphs) [11, 12]. The previous two chapter explored projecting networks into a smaller networks. Here, in this chapter we study the problem of projecting a set of graphs/subgraphs into a continuous feature space.

Network embeddings study the problem of projecting network into continues feature space. They have become very popular in learning effective feature representations of networks. Motivated by the recent successes of embeddings in natural language processing, researchers have tried to find network embeddings in order to exploit machine learning algorithms for mining tasks like node classification and edge prediction. However, most of the work focuses on distributed representations of nodes that are inherently ill-suited to tasks such as community detection which are intuitively dependent on subgraphs. Consider the example in Figure 1.4. Our approach for community detection by embedding the ego-nets closely matches the ground truth.

Here, we formulate subgraph embedding problem based on two intuitive properties of subgraphs and propose `Sub2Vec`, an unsupervised algorithm to learn feature representations of arbitrary subgraphs. We also highlight the usability of `Sub2Vec` by leveraging it for network mining tasks, like community detection and graph classification. We show that `Sub2Vec` gets significant gains over state-of-the-art methods. In particular, `Sub2Vec` offers an approach to generate a richer vocabulary of meaningful features of subgraphs for representation and reasoning.

Chapter 10 (EpiDeep: Deep Learning for Influenza Forecasting) [8].

In the previous chapter, we focused on the learning based approach for prediction task for influenza incidence. Influenza leads to regular losses of lives annually and requires careful monitoring and control by health organizations. Annual influenza forecasts help policymakers implement effective countermeasures to control both seasonal and pandemic outbreaks. Existing forecasting techniques suffer from problems such as poor forecasting performance, lack of modelling flexibility, and/or lack of intepretability. We propose `EpiDeep`, a novel deep neural network approach for epidemic forecasting which tackles all of these issues by learning meaningful representations of incidence curves in a continuous feature space and accurately predicting future incidences, peak intensity, peak time, and onset of the upcoming season. Figure 1.5, shows the quality of the embeddings of historical data. We present extensive experiments on forecasting ILI (influenza-like illnesses) in the United States, leveraging multiple metrics to quantify success. Our results demonstrate that `EpiDeep` is successful at learning meaningful embeddings and, more importantly, that these embeddings evolve as the season progresses. Furthermore, our approach outperforms non-trivial baselines by up



(a) $w_a = 40$ (b) $w_b = 52$ (c) $w_c = 8$ (d) Whole Season

Figure 1.5: (d)-bottom row shows full season length influenza incidence curve for all historical seasons. Bottom row (a), (b), and (c) show snippets of the historical seasons till weeks w_a, w_b, and w_c respectively. The top row shows 2-d projection of learnt embeddings of corresponding snippets.

to 40%.

Chapter 11 (Incorporating Expert's Guidance on Epidemic Forecasting) [198]. Forecasting influenza like illnesses (ILI) has rapidly progressed in recent years from an art to a science with a plethora of data-driven methods. While these methods have achieved qualified success, their applicability is limited due to their inability to incorporate expert feedback and guidance systematically into the forecasting framework. In this chapter, we propose a new approach leveraging the Seldonian optimization framework from AI safety and demonstrate how it can be adapted to epidemic forecasting. We study two types of guidance: smoothness and regional consistency of errors, where we show that by its successful incorporation, we are able to not only bound the probability of undesirable behavior to happen, but also to reduce RMSE on test data by up to 17%.

1.3 Contributions and Impact

Works in this thesis have led to multiple publications in leading venues in data mining (KDD, ICDM, SDM, TKDE, WWW, and PAKDD) [6, 17, 110, 12, 205, 7, 9, 268, 10, 11, 8], public health (PLOS CompBio) [4], and industrial engineering [197]. Our specific contributions are as follows:

1. **Novel domain-aware frameworks:** We propose **novel frameworks** including both the**models** and **analysis techniques** for important problems in various domains. For example, to detect *C. Difficile* outbreaks, we leverage a two-mode model along with discrete-lattice based formulation. Similarly, to infer failures in critical infrastructure we leverage geographically correlated failure model along with information theoretic formulations.

2. **Near-optimal algorithms:** For various of the problems mentioned above, we propose **near-optimal algorithms**. For the network state inference problem, we propose an algorithm with **additive approximation guarantee**, where the additive term is in the logarithmic scale. Similarly, we propose an algorithm with (1-1/e) approximation guarantee for detecting outbreaks in hospitals and mining critical queries in E-commerce.

3. **Effective learning architectures:** We propose several **novel deep neural architectures**. Our learning approaches generate meaningful projections of graphs and time-series and are effective for downstream applications. For example, we augment deep architecture with clustering layers to learn similarity based embeddings of influenza seasons and to improve performance for forecasting.

4. **Enabling new applications:** We demonstrate that our approach has **numerous applications** in many domains such as critical infrastructure, epidemiology, viral marketing, e-commerce, and so on.

Impact: The impact of work in this thesis are as follows:

- The novel deep learning approach for influenza forecasting proposed in this thesis was the **best performing model** in the **CDC FluSight forecasting challenge in 2018-19 season** for HHS region 1.

- Our epidemic forecasting models are currently being used for **CDC COVID-ILI** forecasting projects.

- The near-optimal HAI outbreaks detection approached proposed in this thesis, which was developed with real epidemiologists, was selected by the editors to be **featured on the PLOS Complexity Channel**.

- Several approaches presented in this thesis are **at production in industry**. Our work on product recommendation for queries with no engagement data led to a 34% improvement over the current system at **Walmart** and our work on domain-aware embeddings for e-commerce fraud detection is currently being used at **Amazon**.

1.3.1 Publication List

The following are the author's published works which are discuss in this thesis.

1. Alexander Rodriguez, **Bijaya Adhikari**, Naren Ramakrishnan, and B. Aditya Prakash. "Incorporating Experts Guidance in Epidemic Forecasting". *In Submission to KDD 2020.* [PDF]

2. Alexander Rodriguez, **Bijaya Adhikari**, Andres D. Gonzalez, Charles Nicholson, Anil Vullikanti, and B. Aditya Prakash. "Mapping Network States using Connectivity Queries". *In Submission to CIKM 2020.* [PDF]

3. **Bijaya Adhikari**, Bryan Lewis, Anil Vullikanti, Jose Mauricio Jimenez, and B. Aditya Prakash. "Fast and Cheap Monitoring of Healthcare Acquired Infection Outbreaks". *PLoS Computational Biology 2019.* [PDF]

4. **Bijaya Adhikari**, Xinfeng Xu, Naren Ramakrishnan, and B. Aditya Prakash. "EpiDeep: Exploiting Embeddings for Epidemic Forecasting". *KDD 2019.* [PDF]

5. Sorour E. Amiri, **Bijaya Adhikari**, Aditya Bharadwaj, and B. Aditya Prakash. "NetGist: Learning to generate task-based network summaries". *ICDM 2018.* [PDF]

6. **Bijaya Adhikari**, Yao Zhang, Naren Ramakrishnan, and B. Aditya Prakash. "Sub2Vec: Feature Learning for Subgraphs". *PAKDD 2018.* [PDF]

7. **Bijaya Adhikari**, Parikshit Sondhi, Wenke Zhang, Mohit Sharma, and B. Aditya Prakash. "Mining E-Commerce Query Relations using Customer Interaction Networks". *WWW 2018.* [PDF]

8. **Bijaya Adhikari**, Pavan Rangudu, B. Aditya Prakash, and Anil Vullikanti. "Near-Optimal Mapping of Network States using Probes". *SDM 2018.* [PDF]

9. **Bijaya Adhikari**, Yao Zhang, Sorour E. Amiri, Aditya Bharadwaj, and B. Aditya Prakash. "Propagation based Temporal Network Summarization". *TKDE 2018.* [PDF]

10. **Bijaya Adhikari**, Yao Zhang, Naren Ramakrishnan, and B. Aditya Prakash. "Distributed Representations of Subgraphs". *ICDM Workshop DAMNET 2017.*[PDF]

11. Pavan Rangudu, **Bijaya Adhikari**, B. Aditya Prakash, and Anil Vullikanti. "Using Partial Probes to Infer Network States". *SIGKDD Workshop MLG 2017.* [PDF]

12. **Bijaya Adhikari**, Yao Zhang, Aditya Bharadwaj, and B. Aditya Prakash. "Condensing Temporal Networks using Propagation". *SDM 2017.* [PDF]

1.3.2 Excluded work

For a better coherence, a significant number of author's Ph.D. work have been excluded from this thesis. The excluded work includes the following

- **General graph mining:**Work on network summarization for visualization leveraging online learning [86, 87] and temporal vaccination [205].
- **Social Media:** Detecting communities of influential nodes and their influence sphere [268]. Predicting the who tweets next and when [110, 111].
- **E-commerce:** Work on e-commerce fraud detection [5].

1.4 Thesis Outline

The outline of the thesis is as follows. We first present survey of related work in Chapter 2. Then we discuss our work on domain-aware optimization frameworks in the first part of the thesis (Chapters 3 to 7). Then in the second part of the thesis (Chapters 8-12) we present our work on domain-aware representation learning. Finally, we conclude and present future works in Chapter 13.

Chapter 2

Survey

In this chapter, we present a survey of related work in general graph mining, dynamics over networks, as well as work in specific domains.

Early work like [50, 88] were the first to study the macroscopic structure of large scale networks. Similar studies characterizing real network from several domains such as biology [114, 16], online collaborations [264], e-commerce marketplace [216], and Wikipedia pages [274] have followed. Several important mining techniques developed for large networks include, community detection, which is one of the best studied problem in a network setting [100, 171], including for bipartite networks [32] and heterogeneous networks [145]. Other important tasks include discovering influential nodes in the networks [46, 129] and measuring node centrality [180, 45]. Dynamic graphs, where the network evolves over time, have gained a lot of interest recently (see [14] for a survey). Many graph mining tasks on static graphs have been introduced to dynamic graphs, including community detection [233] and link prediction [207].

Dynamics over networks has been widely studied, including in epidemiology [21, 107], information diffusion [129], cyber-security [130] and product marketing [199]. Based on the models, several papers studied propagation related applications, such as propagation of memes [141], tweets [62], and computer viruses [242]. Significant work has been done on determining the epidemic threshold i.e. the conditions under which a virus causes an epidemic [94, 183, 184]. Examples of propagation-based optimization problems are influence maximization [129, 96, 15], and immunization [270]. Remotely related work deals with weak and strong ties over time for diffusion [127].

2.1 Data-Driven Epidemiology

Interventions: There also has been extensive work on developing strategies for interventions. Early work in this space include [159, 24]. Shim proposed age-structured dynamical model of influenza to determine the best intervention policy [219]. Tong et al. presented spectral approach for selecting best nodes to remove from a graph (vaccination) [239] and the best edges (quarantining) to remove [238] to control epidemic outbreaks. Shah et al. proposed approximation algorithm to reduce spectral radius of large graphs [203]. Zhang et al. studied the vaccination problem under several scenarios such as after the onset of an outbreak [270], with uncertain data [271], for groups [269], and with auxiliary medical data [272].

Surveillance: Surveillance of an epidemic outbreak is an important problem. Several practice-based approaches have been proposed for hospital acquired infections [150, 149, 71]. Christakis et al. proposed monitoring central nodes in social media for early outbreak [66]. Data driven approaches for surveillance often pose the problem as 'sensor' selection in large networks [208, 142].

Source and Missing Infection Inference:

A topic closely related to this thesis is the inference of the source of an infection and other missing infections, in the case of epidemic spread on networks. Epidemics are modeled as stochastic processes, e.g., SI/SIR, in which the infection spreads from an infected node to its susceptible neighbors. Usually, only partial information about the infections is known, and some of the problems that have been studied include identifying the source of an infection and finding other missing nodes [185, 202, 225, 211, 210, 255]. The Minimum Description Length (MDL) principle [104, 195] has been successfully used in [185, 225] for these problems, whereas [211] develop an MLE method.

2.2 Network State Inference

Related works include network and state inference in communication networks. The area of network tomography involves inferring link states, such as delays and failures, in the Internet and other communication networks; see, e.g., [121, 254, 173, 20, 85]. Xia et al. [254] assume link delays are exponentially distributed. Ni et al. [173] study different kinds of probing models, including multicast probes which can give estimates on a tree, and develop methods for inferring the topology in dynamic networks. There has also been work on designing probes to infer part of the network structure, as in [20].

A different class of failure models motivated by settings such as disaster events, e.g., [38, 13, 204] has been extensively studied. These studies assume an initial failure, and subsequent failures whose probability is correlated with the source. For instance, in [13], the probability

$p(j|i)$ that node j fails, given that i is the source is a function of the distance from i to j, with the probabilities decaying with the distance.

2.3 Network Summarization

Network summarization seeks to find a compact representation of a large graph by leveraging global and local graph properties like local neighborhood structure [167], node/edge attributes [265], action logs [187], eigenvalue of the adjacency matrix [186], and key subgraphs. It is also related to graph sparsification algorithms [155]. The goal is to either reduce storage and manipulation costs, or simplify structure. Summarizing temporal networks has not seen much work, except recent papers based on bits-storage-compression [144], or extracting a list of recurrent sub-structures over time [212]. Unlike these, we are the first to focus on hierarchical condensation: using *structural* merges, giving a *smaller propagation-equivalent* temporal network.

2.4 Graph Representation Learning

The network embedding problem, which seeks to generate low dimensional feature representation of nodes, has been well studied. Early work includes Laplacian Eigenmap [36], IsoMap [234], locally linear embedding [201], and spectral techniques [25, 250]. However, these methods are slow and do not scale to large networks. Recently, several deep learning based network embeddings algorithms were proposed. DeepWalk [182] and Node2Vec [103] extend skip-Gram model [161] to networks and learn feature representation based on contexts generated by random walks. SDNE [248] and LINE [231] learn feature representation of nodes while preserving first and second order proximity. Other works include embedding Signed networks [65], using structural identity [193] and so on.

The most similar network embedding literature includes [194, 258, 166]. Risen and Bunke propose to learn vector representations of graphs based on edit distance to a set of pre-defined prototype graphs. Yanardag et. al. and Narayanan et al. learn vector representation of the subgraphs using the Word2Vec by generating "corpus" of subgraphs where each subgraph is treated as a word.

Another closely related filed to this thesis is the deep graph networks. Kipf et al. proposed a semi-supervised approach for node classification [132]. Several extensions have also been proposed including attentions on graph convolution [244], tree-structured sequential models [229], and adaptive receptive fields [147].

2.5 Forecasting

Epidemic Forecasting: Epidmic forecasting models can be broadly categorized into statistical [236, 67] and modelling based approaches [214, 266]. Additionally, orthogonal to this thesis, there has also been much interest in leveraging signals from external data sources such as search engine [99, 261], social media [62, 139], environmental and weather reports [213, 230], and a combination of heterogeneous data [59]. Deep learning for flu forecasting has barely been explored except for [245] which basically uses a simple LSTM with geographical and climate constraints and [246] which uses LSTM to predict influenza activities specifically in the military population by incorporating twitter data. LSTM does not perform well as it requires a large amount of data. Another related filed is cascade prediction. Here the goal is to forecast whether the cascade will go viral in the future [113, 74, 224] or predict the size of cascade [241].

Time Series Analysis: Time-series prediction is a well-studied area with several methods from different perspectives including auto-regression, kalman-filters and groups/panels [48, 206, 115]. Recently recurrent neural architectures [108, 70] have also become popular. However these prediction methods are ill suited for flu forecasting as they are too specialized or usually not flexible enough to capture the seasonal inconsistency in wILI activity [177]. Temporal point processes [77] while having been applied to modeling information diffusion [83, 232] in a wide range of domains (e.g. finance [26], sociology [72], recommendation [109]), model only the temporal dynamics of the diffusion networks unlike ours. Attention mechanisms are one of the most recent and exciting advances in deep learning but have been primarily utilized for NLP tasks (e.g. [29, 105, 260]) or computer vision (e.g. [135, 79, 257]) unlike our application to cascade prediction.

2.6 Graph Analytics for E-commerce

Beeferman and Berger introduced click graphs [35], which are bipartite graphs with directed edges between queries and clicked urls, for clustering similar queries and urls. Several subsequent works utilized click-graphs for other applications [93, 92], including a prominent work by Craswell and Szummer [73] who used them for query-suggestion, document-search, relevance feedback and url-annotation. Baeza-Yates also proposed various variants of query relation graphs in [27] including cover graphs, where two queries have an edge if and only if they share an item.

In other related works, Boldi et al. studied query-flow graphs [43], and used them for related query suggestion [43, 44]. Song et al. proposed the term graph and leverage it for the same task [221]. Many techniques have been employed in mining relations between queries. Query clustering has been exploited to group similar queries [35, 28, 251]. Some other works are based on association rules [90] and modeling user [273].

Part I

Domain-aware optimization frameworks

Chapter 3

Near-Optimal Monitoring of HAI Outbreaks

Since the time of Hippocrates, the "father of western medicine", a central tenet of medical care has been to "do no harm." Unfortunately, the scourge of hospital acquired infections (HAI) challenges the medical system to honor this tenet. When patients are hospitalized they are seeking care and healing, however, they are simultaneously being exposed to risky infections from others in the hospital, and in their weakened state are much more susceptible to these infections than they would be normally. Acquiring these infections increases the chances of either dying or becoming even sicker, which also lengthens the time the patient needs to stay in the hospital (increasing costs). These infections can range from pneumonias and gastro-intestinal infections *Clostridium difficile* to surgical site infections and catheter associated infections, which puts nearly any patient in the hospital at risk. Antibiotic treatments intended to aid in recovery from one infection, may open the door for increased risk of infection from another.

Hospital acquired infections are a significant problem in the United States and around the world. Some estimates put the annual cost between 28 and 45 billion US dollars per year in the US [223]. More importantly, they inflict a significant burden on human health. A recent study estimated more than 2.5M new cases per year in Europe alone, inflicting a loss of just over 500 disability-adjusted life years (DALYS) per 100,000 population [55]. Given their burden and cost, their prevention is a high priority for infection control specialists. A simple approach to monitor HAI outbreaks would be to test every patients and staff in the hospital and swab every possible location for HAI infection. However, such a naive process is too expensive to implement. A better strategy is required to efficiently monitor HAI outbreaks.

A recent review article [138] included 29 hospital outbreak detection algorithms described in the literature. They found these fall into five main categories: simple thresholds, statistical process control, scan statistics, traditional statistical models, and data mining methods. Comparing the performance of these methods is challenging given the myriad diseases, defini-

tions of outbreaks, study environments, and ultimately the purpose of the studies themselves. However, the authors identify that few of these studies were able to leverage important covariates in their detection algorithms. For example, including the culture site or antibiotic resistance was shown to boost detectability. Past simulation based approaches [143] tackle optimal surveillance system design, by choosing clinics as sensors, to increase sensitivity and time to detection for outbreaks in a population. In contrast, our approach selects most vulnerable people and locations to infections as sensors to detect outbreaks in a hospital setting. Different kinds of mechanistic models have also been used for studying HAI spread [150, 149, 71, 133]. Most of these are differential equation based models. We refer to [243] for a review of mechanistic models of HAI transmission.

On a broader level, sensor selection problem for propagation (of contents, disease, rumors and so on) over networks has gained much attention in the data mining community. Traditional sensor selection approaches [142, 66] typically select a set of nodes which require constant monitoring. Instead, in this paper, we select sensor set as well as the rate to monitor each sensor. Hence, our approach is novel from the data mining perspective as well. Recently Shao et al. [215] proposed selecting a set of users on social media to detect outbreaks in the general population. Similarly, Reis et al. [191] proposed an epidemiological network modeling approach for respiratory and gastrointestinal disease outbreaks. Other closely related data mining problems include selecting nodes for inhibiting epidemic outbreaks (vaccination) [270, 49, 269] and inferring missing infections in an epidemic outbreak [202].

We employ a simulation and data optimization based approach to design our algorithm and to provide robust bounds on its performance. Additionally, our simulation model is richly detailed in terms of the class of individuals and locations where sampling can occur. None of the prior works explicitly model the 'two-mode' nature of HAIs and fail in separating the location contamination and infections in people. We formalize the sensor set problem as an optimization problem over the space of rate vectors, which represent the rates at which to monitor each location and person. We consider two objectives, namely the probability of detection and the detection time, and show that these satisfy a mathematical property called submodularity, which enable efficient algorithms. In addition, we leverage data generated from carefully a calibrated simulation using real data collected from a local hospital. Our extensive experiments show that our approach outperforms a state-of-the-art general sensor detection algorithm. We also show that our approach acheives the minimum outbreak detection time compared to other alternatives. To the best of our knowledge, we are the first to provide a principled data driven optimization based approach for HAI outbreak detection.

3.1 Materials and methods

Data

As previously mentioned, we propose a data-driven approach in selecting the sensors. There are multiple challenges in obtaining actual HAI spread data such as high cost, data sparsity, and the ability to safeguard patient personal information. For this reason, we rely on simulated HAI contagion data. We use a highly-detailed agent based simulation that employs a mobility log obtained from local hospitals [120, 119] to produce realistic contagion data. The data generation process we leverage is explained in [120] with greater detail. Fig 3.1 shows a visualization of simulated HAI spread. In our simulation, people (human agents) move across various locations (static agents) as defined by the mobility log and spread HAI in stochastic manner.

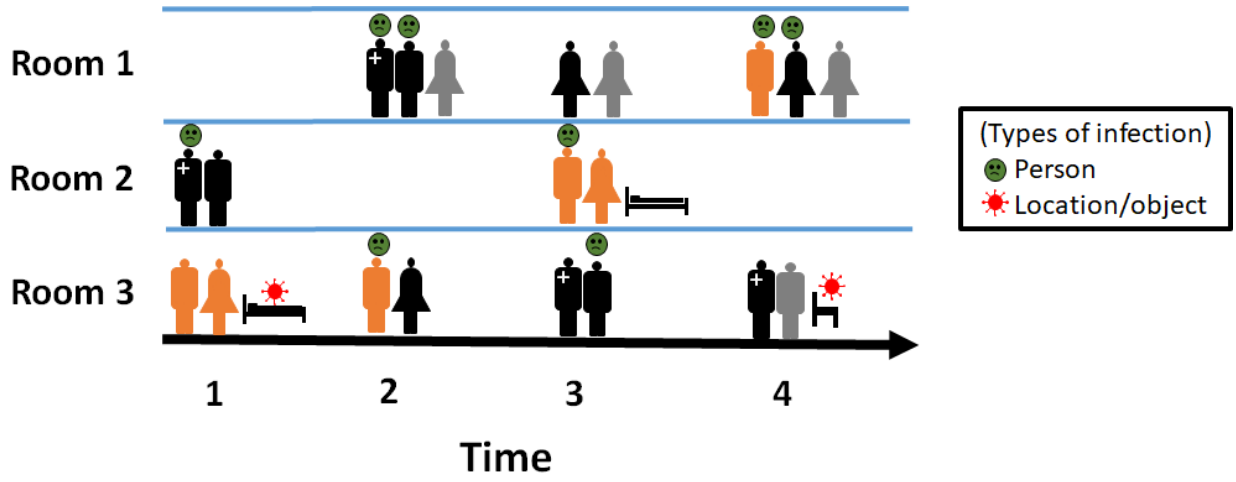


Figure 3.1: Visualization of a possible HAI spread. As human agents move through various locations, they infect other agents and contaminate the locations.

The simulation was developed by following three steps: design of an *in-silico* or computer-based population and its activities, conceptualization of a disease model for a pathogen of interest, and the employment of a highly-detailed simulation. The following sections describe the data creation process in more detail.

In-silico Population and Mobility Log

The first step in developing an HAI contagion model is the creation of a complete hospital population based on realistic hospital parameters. This will ensure high resolution and fidelity for the simulation. The hospital population was initially developed by capturing actual

patient schedule data from regional hospitals in Southwest Virginia. The de-identified data contains information regarding patient location, disposition, and activities within a hospital. In addition to the patient schedule, the population incorporates the activities of health care workers such as nurses, doctors, therapists, and environmental services personnel among others. Their activities were compiled into the population based on direct observation of each type of health care worker. One additional level of realism was added to the population by including detailed information of locations, such as patient rooms and static objects such as furniture. We also use the term *fomite* to mean a location. The locations/fomites were modeled as static agents.

The final mobility log of the hospital population consists of a detailed database that specifies the type of agent, its location, specific agent, and the duration of the activity. Formally, the mobility logs are represented as a bipartite temporal network $G(P, L, E, T)$, with partition P representing population of human agents, partition L representing locations, E representing who-visits-what-location relationship and T representing time/duration of the visit. The mobility log includes 72,146 unique locations with 96,281 unique human agents, and their interaction for a total duration of 200 days.

Disease Model

The next step in generating a realistic simulation is to model the disease accurately. The disease model we use is a probabilistic finite-state machine (FSM). Once the simulation is initiated, each agent will move through the different disease/health states described in the disease model beginning with the uninfected state. As the simulation progresses, agents are either colonized or not with the *C. difficile* bacterium. It is important to note that the transition probabilities for each disease/health state utilize current infection and recovery rates to capture the actual behavior of the pathogen in the hospital setting. Figure 3.2 shows the disease model for humans.

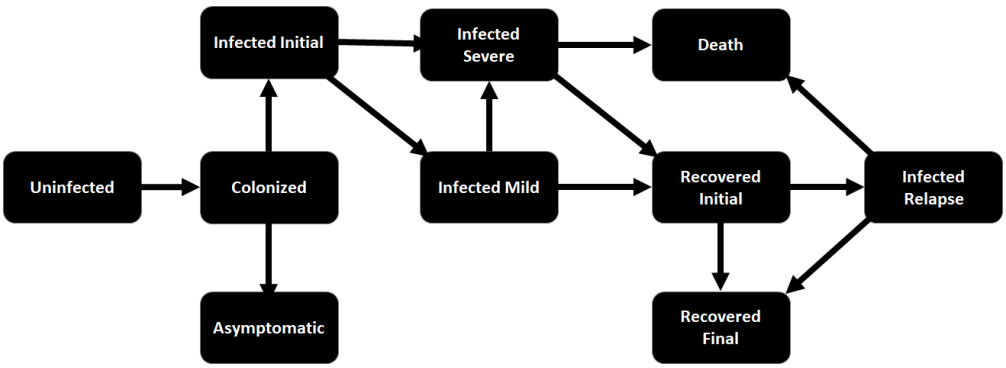


Figure 3.2: Human Infection Model for *C. Difficile*

Since the infection in people and the contamination of fomites are inherently different, we

use two separate FSMs to model the spread of the pathogen throughout the hospital. The fomite FSM includes states capturing low, mid, and high level of contamination, which can trigger infection on human agents in the simulation. Figure 3.3, shows the disease model for fomites in different hospital locations. These carefully designed FSMs with meticulously calibrated transition probabilities along with the agent mobility logs constitute the input for the simulation software to generate the HAI contagion.

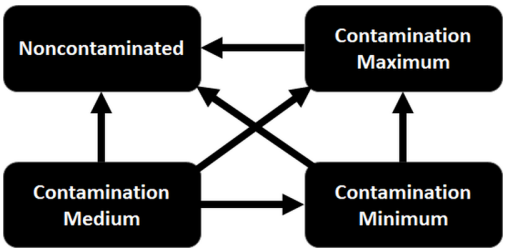


Figure 3.3: Fomite Contamination Model for *C. Difficile*

Simulation

The final step in the simulation process is to use a powerful simulation software capable of processing a large amount of agents through the disease models. This study utilizes EpiSimdemics [33], a simulation software developed by the Network Dynamics and Simulation Science Laboratory (NDSSL). Episimdemics is an agent based simulation, in which every person is represented as an agent. Each agent performs activities as described by a master schedule. The schedule determines the location of each agents at a given time. If two or more agents are in the same location, then they can interact with each other. For example, an infected agent can infect a susceptible agent if they are in the same location based on the transition probabilities of the disease/health state. The ability of EpiSimdemics to simulate large populations (millions of agents) has been largely documented [33].

It is important to note that patient and healthcare workers mobility data leveraged here was obtained directly from the hospitals. Hence the actions performed by human agents in the simulation are heavily detailed and have high resolution. Moreover, to capture the correct dynamics of *C. difficile* outbreak, community acquired and hospital acquired cases were collected to calibrate the simulation parameters. We ensured that the simulation output of hospital acquired cases closely matched the real value, given the community acquired cases. The details of simulation calibration is presented in [120]. Once the model is correctly calibrated, simulation can be leveraged for multiple initial conditions.

Simulation Outputs

We ran EpiSimdemics multiple times with various initial conditions. Each simulation instance produced a cascade (infection dendogram) consisting of information on identity of newly infected agents, time of infection, the source of infection, and the activity the agent was performing when infected. In the following, we use the terms *simulation instances* and *cascades* interchangeably. Each simulation also produces information regarding the infection status of each agent on each day for all 200 days. Formally, we have a set $\mathcal{I}$ of individual simulation instances i of HAI spread. Each simulation i produces a cascade (dendogram) of HAI spread starting from a particular initial stage.

Figure 3.4 shows the distribution of infections over different categories of agents. The infections are dominated by nurses, physicians, fomites, and patients. Patients have the highest infection rate as they are the largest group by population. Nurses and physicians have high infection levels as well, given that they are more mobile and hence are more exposed to HAIs. They also have higher interactions with other health care workers. The box plot in Figure 3.5 shows that roughly 50% of the simulations infect between 19 and 25 agents.

In summary, our HAI contagion simulation data follows meticulous stages of development, modeling and calibration. The simulation results produce intuitive and realistic infection patterns.

Sensor Set and Rate Selection

Recall that our goal is to select a set of agents as sensors, and *the rate at which each such sensor should be monitored*, such that future HAI outbreaks are detected with high probability, and as early as possible. However, these have to be selected within given resource constraints. We start with a formalization of these problems. Finding a minimum cost sensor set is a challenging optimization problem, and we present efficient algorithms by using the notion of submodularity.

We first define some notation. Let bold letters represent vectors. Let P and L denote the sets of human agents and locations respectively; let $n = |P \cup L|$—this will be the total number of agents in our simulations. Let B denote the budget on a number of samples that is permitted (weighted by cost of agents), i.e, it is the sum of expected number of swabs to detect whether a location is contaminated or a human is infected. As mentioned earlier, the mobility logs are represented as a bipartite temporal network $G(P, L, E, T)$, with two partitions P and L representing agents, E representing who-visits-what-location relationship and T representing time/duration of the visit. We consider each agent to be a *node* in the temporal network. Hence we use the terms *node* and *agent* interchangeably. Now, let $\mathbf{c} \in R^n$, be the vector of costs, i.e, $\mathbf{c}[v]$ is the cost of node v. Let $\mathbf{r} \in R^n$ be the vector of monitoring rates, where $\mathbf{r}[v]$ denotes that the probability that node v is monitored (e.g., swabbed) each day. Finally, let T_{max} denote the maximum time in each simulation instance.

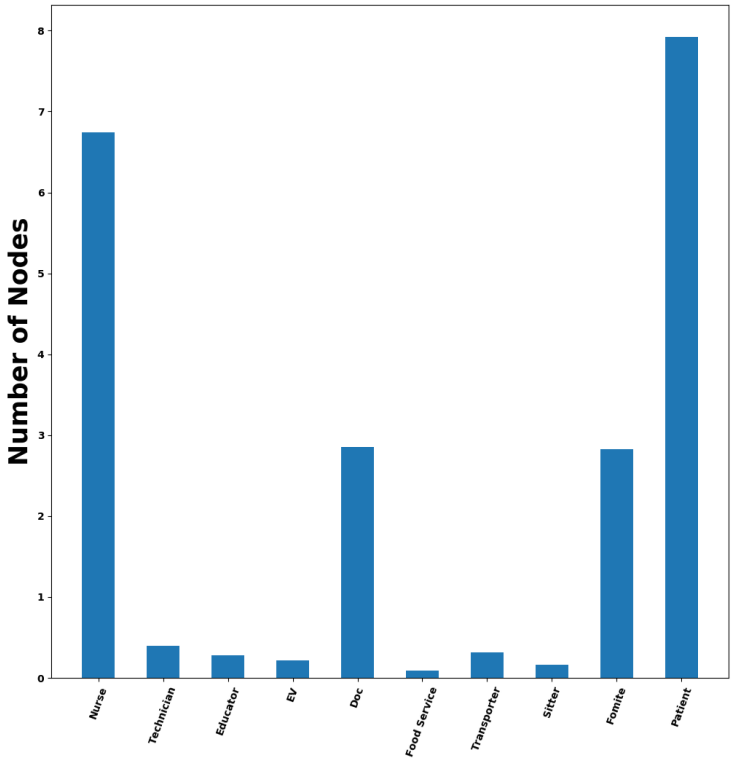


Figure 3.4: Infection distribution for various agents. High susceptibility of health-care workers can be attributed to their high mobility. Patients have the highest number of infections as they are the largest population group.

Optimizing Probability of Detection

Consider an arbitrary simulation instance i and an arbitrary sensor set. Agent v is monitored at the rate of $\mathbf{r}[v]$ in a rate vector $\mathbf{r}$. Let the number of days in which node v is in infected state in simulation instance i be $\tau(v, i)$. Then, the probability that node v is detected in simulation i is

$$P(v|i, \mathbf{r}) = 1 - (1 - \mathbf{r}[v])^{\tau(v,i)} \tag{3.1}$$

Next, the probability of at least one node being detected in simulation i, given the rate vector $\mathbf{r}$, is

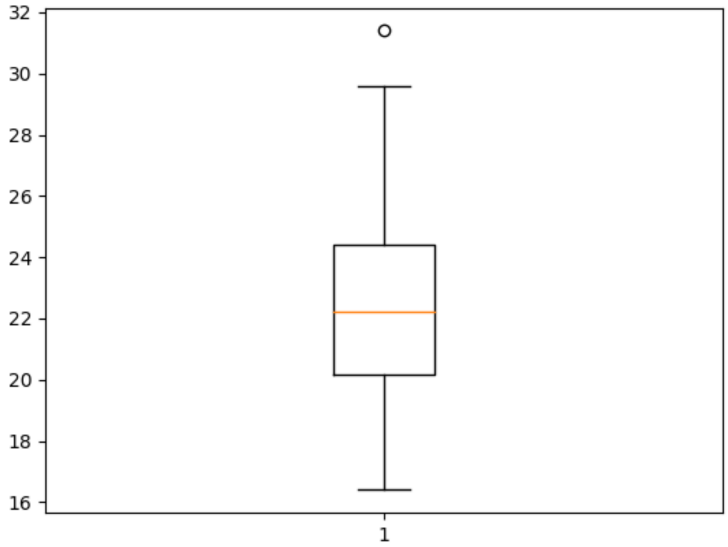


Figure 3.5: Box plot for variation in number of infection per simulation. Consistent with prior works, our simulation results in a few infections over 200 days of simulation.

$$P(i|\mathbf{r}) = 1 - \prod_{v \in P \cup L} (1 - P_d(v|i, \mathbf{r})) \tag{3.2}$$

$$P(i|\mathbf{r}) = 1 - \prod_{v \in P \cup L} \left[1 - \left(1 - (1 - \mathbf{r}[v])^{\tau(v,i)}\right)\right] \tag{3.3}$$

$$P(i|\mathbf{r}) = 1 - \prod_{v \in P \cup L} (1 - \mathbf{r}[v])^{\tau(v,i)} \tag{3.4}$$

Recall that $\mathcal{I}$ denotes the set of simulations. It follows that the expected number of cascades, where at least one node is detected, given the rate vector $\mathbf{r}$, is $\sum_{i \in \mathcal{I}} P(i|\mathbf{r})$. The probability that a sensor node is detected equals the fraction of cascades in which at least one sensor node is infected. This motivates the first problem we study.

Problem 1 *Find the vector* $\mathbf{r}^*$*, such that*

$$\mathbf{r}^* = \arg\max_{\mathbf{r}} \sum_{i \in \mathcal{I}} P(i|\mathbf{r}) \tag{3.5}$$

and $\sum_{i=0} r(v) \cdot c(c) \leq B$.

Optimizing Expected Time of Detection.

As discussed above, the node v is sampled/swabbed for infection each day with rate $\mathbf{r}[v]$. Let the ordered list of days in which node v is infected in simulation instance i be $\gamma(v,i) = \{t_1^v, t_2^v, \ldots t_n^v\}$, where t_i^v represents the i^{th} day in which node v is infected.

Here, we are interested in the first day until node v is detected to be infected. In the days where node v is not infected, sampling node v does not result in outbreak detection. The detection time is the first time at which v is infected, and is sampled. Since node v is sampled each day with probability $r[v]$, the first time in $\gamma(v,i)$ in which it gets detected (restricted to the times in which v is infected) is a geometric process. Therefore, in expectation, node v is detected to be infected on the $1/\mathbf{r}[v]^{\text{th}}$ day in $\gamma(v,i)$. Let that be denoted as $Det(v|i,\mathbf{r})$. For example, let us consider a case when a node gets infected in day 5 of simulation and remains infected till day 15. During this time, if we sample this node with a rate of 0.2, the node is sampled and detected to be infected on the 5^{th} day of infection which is the day 9 of the simulation in expectation. If $1/\mathbf{r}[v]$ is greater than the length of $\gamma(v,i)$, we consider that the infection to be undetected and set $Det(v|i,\mathbf{r})$ as T_{max}, the last time-stamp in any of the simulation instances.

Formally, the minimum detection day for simulation instance i is,

$$D(i|\mathbf{r}) = \min_v Det(v|i,\mathbf{r}) \tag{3.6}$$

Our goal is to minimize $D(i|\mathbf{r})$ over all $i, \mathbf{r}$. This turns out to be a challenging computational problem, and instead we consider its converse, which turns out to have useful properties. Now, the problem to optimize for expected time of detection can be posed as following:

Problem 2 *Find the vector* $\mathbf{r}^*$*, such that*

$$\mathbf{r}^* = \arg\max_{\mathbf{r}} \sum_{i \in \mathcal{I}} [T_{max} - D(i|\mathbf{r})] \tag{3.7}$$

and $\sum_{i=0} r(v) \cdot c(c) \leq B$.

Our Methods

Problems 1 and 2 are both computationally very challenging, specifically they are in the computational class NP-hard even for simplistic instances [220], and cannot be solved optimally in polynomial time, unless P = NP. Since the instances we consider are pretty large, naive exhaustive search for the optimal solution is not feasible. Therefore, we focus on near-optimal approximate solutions. We show that the objective function in Problem 1 is a *submodular*

lattice function. Informally, this means that the objective value has a property of diminishing returns, for a small increase in the rate in any dimension (this is defined formally in the Supplementary Information section)—this property implies that a natural greedy algorithm which maximizes the objective marginally at each step gurantees a $(1 - 1/e)$-approximation to the optimal solution. Without such a submodularity property, solving Problem 1, even for a small budget B would be challenging.

Our HaiDetect algorithm for Problem 1 selects the sensor set and rates such that nodes which tend to get infected across multiple simulation instances have higher infection rates. It consists of the following steps.

1. For each feasible initial vector $\mathbf{r}_0$
   (a) Intiliaze the rate vector $\mathbf{r} = 0$
   (b) While summation of rates is less than the budget:
      i. Find a node v and rate r maximizing average marginal gain among all possible values
      ii. Let $\mathbf{r}[v] = r$
      iii. Remove all candidate pairs of nodes and rates which are not feasible
2. Return the best rate vector $\mathbf{r}$

The most expensive computational step is the estimation of the node v and rate r that gives the maximum average marginal gain (Step (i) of 1(b)). This can be expedited using lazy evaluations and memoization. Hence, the algorithm is quite fast in practice. Moreover, it also embarrassingly parallelizable. The steps (a) and (b) for each initial vector can be performed in parallel.

We also propose a similar algorithm HaiEarlyDetect for Problem 2. The main idea here is that we assign higher rates to nodes which tend to get infected earlier in many simulation instances. HaiEarlyDetect optimizes the marginal gain in the objective in Problem 2 in each iteration. It turns out that the objective in Problem 2 is not submodular. However, as shown by our empirical results, the greedy approach we propose works well in practice and outperforms the baselines.

Baseline Method

A natural heuristic to monitor HAI outbreaks would be to sample each agent every day. However, such a method is too expensive to implement is practice. Other natural baselines include, monitoring all patients, swabbing all locations for fomites, monitoring all nurses, and so on. These methods too get expensive as the number of personnel and locations increase.

An interesting baseline would be methods for general sensor set selection problems. Here we compare our approach against CELF [142], a state-of-the-art method for general sensor set selection problem. We run CELF in the same set simulation as our methods. CELF is also a greedy algorithm designed for a submodular set function(as opposed to a lattice function in our case). CELF tries to add node v to the sensor set with rate $\mathbf{r}[v] = 1$, such that the number of newly detectable simulation instances is maximized. CELF has been previously used for selecting sensors in water distribution network and in other network settings.

3.2 Results

We ran HAIDETECT and HAIEARLYDETECT and compared them with CELF in various settings for both qualitative and quantitative studies. We ask questions like: "How does the performance of the methods change with different budget constraints and with more data?", "Are the sensors selected by our methods qualitatively the same?" , "Is there an advantage in using HAIDETECT or HAIEARLYDETECT over CELF?", and so on.

Sensor Quality with Budget

First we compare the performance of HAIDETECT and CELF with respect to the budget. For this experiment, we performed a 5-fold cross validation on 200 simulations. Specifically, we divided the simulations into 5 groups, and at each turn we selected the sensors in the first four groups and computed the sum of outbreak detection probability as shown in Equation 1 in the fifth group (the test set). Then we normalize the resulting sum of outbreak detection probability by total number of simulation instances in the the same group. The normalized value can be intuitively described as the average probability of detecting a future outbreak. We repeat this process five times ensuring each group is used for success evaluation. We then compute the overall average and its standard error. We repeat the entire process for budget size from 1 to 50.

The result of our experiment is show in Figure 3.6. The first observation is that HAIDETECT consistently outperforms CELF for all values of the budget. The disparity between the methods is more apparent for larger values of budget. The difference in quality of the sensors can be explained by the fact that CELF only assigns rate of 0 or 1. However, HAIDETECT can strategically assign non-integer rates so as to maximize the likelihood of detection.

We can also observe that the standard error for the HAIDETECT decreases and is negligible for larger budgets. However, it is not the case for CELF. This shows that not only the quality of sensors detected by HAIDETECT is better, but it is more stable as well. Finally, we see that probability of an outbreak being detected by sensors selected by HAIDETECT is 0.96 when budget is equal to 50, whereas it is only around 0.75 for CELF. Similarly, a budget of only 25 is required to detect an outbreak with probability of 0.8 for HAIDETECT. For the

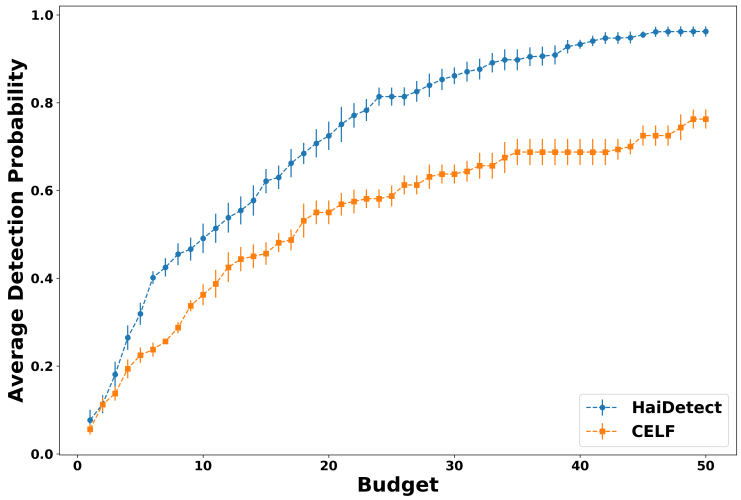


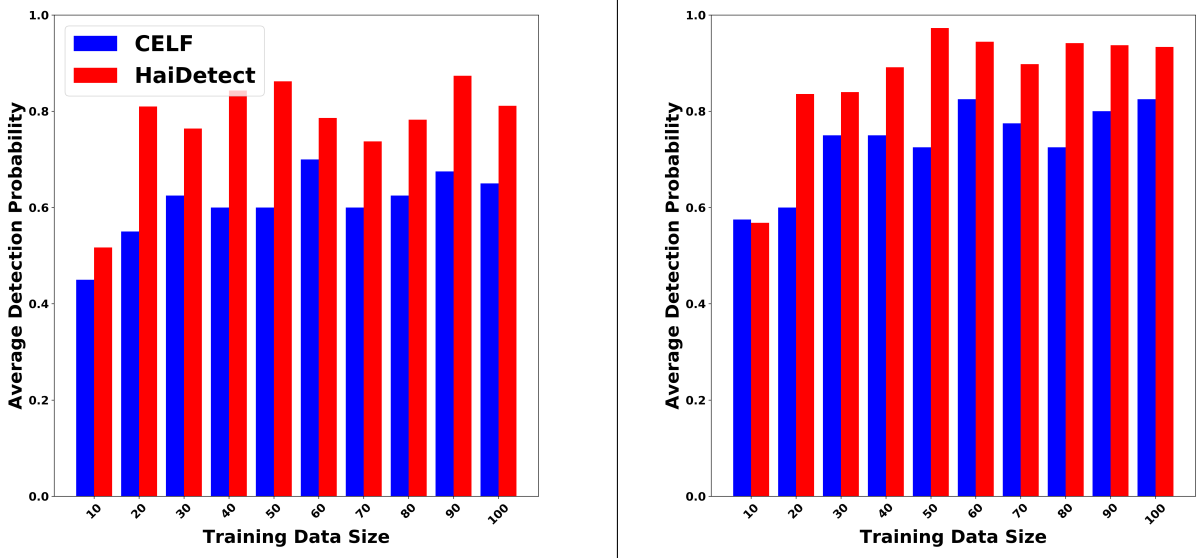
Figure 3.6: Probability of detecting future outbreaks (normalized) for different budgets. HaiDetect significantly outperforms Celf implying that monitoring sensors selected by HaiDetect have higher chance of detecting an outbreak.

same budget, sensors selected by Celf detect cascades with probability of 0.55. The result highlights that HaiDetect produces more reliable monitoring strategy for HAI outbreak detection.

Sensor Quality with Increasing Simulations

Here, we investigate the change in performance of HaiDetect and Celf as the number of simulations used to detect the sensor increases. For this experiment, we used 150 distinct simulations. We divided the simulations into two categories, 'training' and 'testing' sets. We used the cascades in the training set to select the sensors and used the ones in the testing set to measure quality. First we decided on a budget of 30 and training size of 10 cascades. We ran both HaiDetect and Celf for this setting and measured the quality using the cascades in the testing set. We then increased the training size by 10 till we reached the size of 100. We repeated the same procedure for budget of 50. We compute the average probability of detection in the same manner as described above.

Figure 3.7 summarizes the result. We can observe that HaiDetect outperforms Celf consistently. It reinforces the previous observation that HaiDetect selects good sensors for the HAI outbreak detection. An interesting observation is that the performance tails off after training size of 20. Which implies that not many cascades have to be observed before we can select a good quality sensor. This is an encouraging finding as gathering large number



(a) Budget = 30.

(b) Budget = 50.

Figure 3.7: Average probability of detecting future outbreaks for sensors computed using $\mathbf{T}$ simulations (training) for different values of $\mathbf{T}$, and tested on the remaining simulations for a budget of (a) 30 and (b) 50. Note that HaiDetect required only 20 simulation instances to detect future outbreak with probability of 0.8

of real cascades of HAI spread is not feasible.

Next we study the change in performance of HaiDetect with the training size for various budget sizes. Here we tracked the performance of HaiDetect for budgets of 10, 30, and 50 for training sets of various size. The result is summarized in Figure 3.8.

As shown in the figure, the difference between performance of HaiDetect for budgets 30 and 10 is much larger than that for budgets 50 and 30. The normalized objective, or the probability of detection, is close to 1 at budget 50, indicating that monitoring sensors at rates assigned by HaiDetect detects almost all the HAI outbreaks. Hence, in expectation, roughly 50 swabs a day is enough to monitor an outbreak in a hospital wing. Again, we observe that performance of HaiDetect tails off after the training size of 20. It provides extra validation for the observation that a limited number of observed cascades are enough to select high quality sensors.

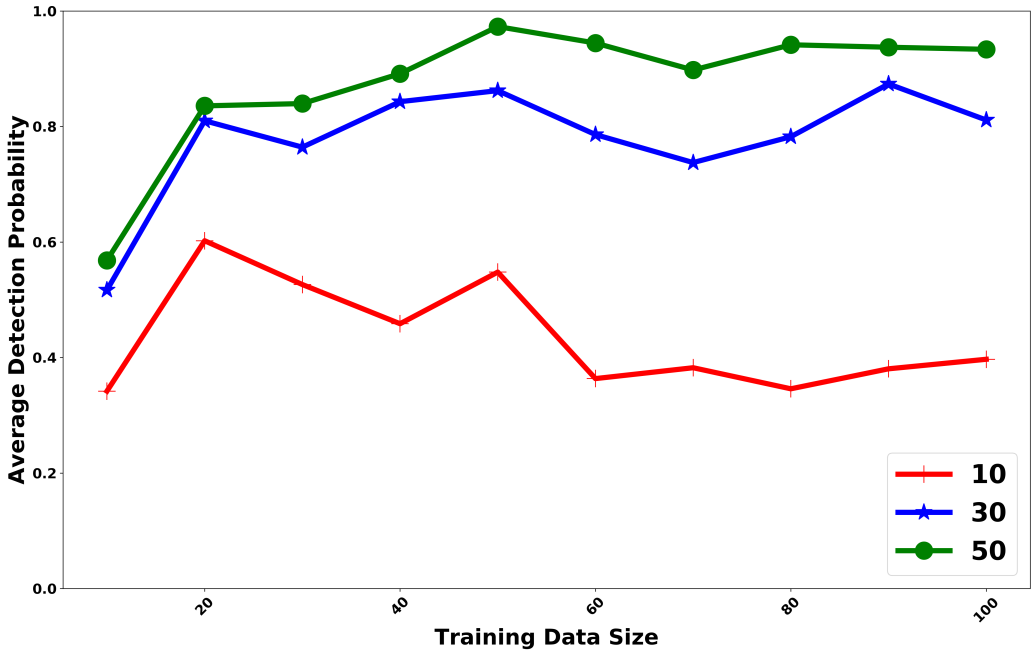


Figure 3.8: Average probability of detecting future outbreaks for sensor sets computed using **T** simulations (training), and evaluated using the remaining simulations for different budget values.

Time of Detection

A desirable property of sensors is that they aid in early detection of outbreaks. Here we study the average time of outbreak detection time of future outbreaks using the sensors and rates selected by HaiEarlyDetect. In this experiment, we first divided our simulations into equally sized training and testing sets, each having 100 simulations. We ran HaiEarlyDetect on the training set to detect sensors and rates at which to monitor them. Then, we monitored the selected sensors at the inferred rates and measured the detection time for each simulation in the testing set. We repeated the entire process for various budgets. The detection time averaged over 100 simulated outbreaks in the testing set is summarized in Figure 3.9 and the variance in the detection time is shown in Figure 3.10.

As shown in the figure, as the budget increases the average detection time decreases. According to our results, the average time to detect an outbreak in the testing set while monitoring sensors selected for budget of 1000 is roughly six days. This is impressive considering the fact that monitoring all agents results in detection time of 4 days monitoring all of more than 1200 nurses results in detection time of 8 days.

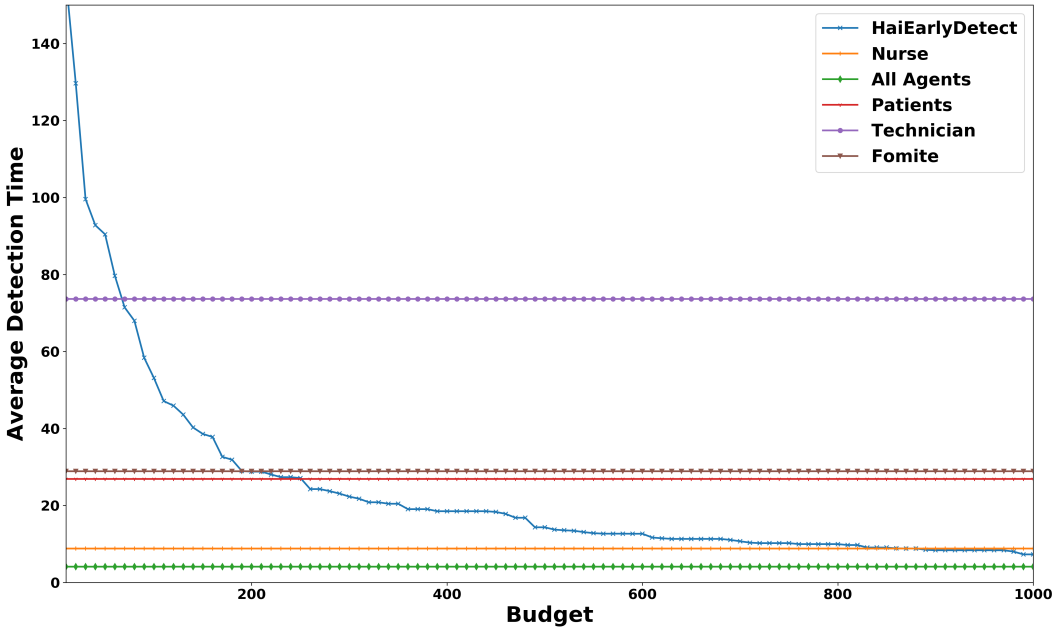


Figure 3.9: Average dectection time for various budgets. The flat lines are the detection time for monitoring all members of different categories of agents. Note that for a budget of 1000, monitoring sensors selected by HaiDetect detect future outbreaks earlier than monitoring all nurses.

Hence, our sensors detect the outbreak earlier with fewer sensors. Another advantage of our sensors is that they are diverse. Significant proportion of the selected sensors include patients and fomites, which are easier to monitor than the nurses. Hence, monitoring our sensors also has an economic advantage.

An interesting observation seen in Figure 3.10 is that the variability in average detection time decreases with the increase in budget. Hence, we expect the performance of our sensors to be fairly consistent in detecting future outbreaks for larger budgets. Moreover, the median time to detect an outbreak (as shown by the box plots) is always less than the average. Hence, we expect that performance of HaiEarlyDetect to be generally better than that suggested by the average detection time. For budget of 1000, the median detection time is just 5 days. Note that monitoring all agents results in detection time of 4 days. This implies that in practice our approach requires only 1000 swabs per day to detect an outbreak within a single day of the first infection.

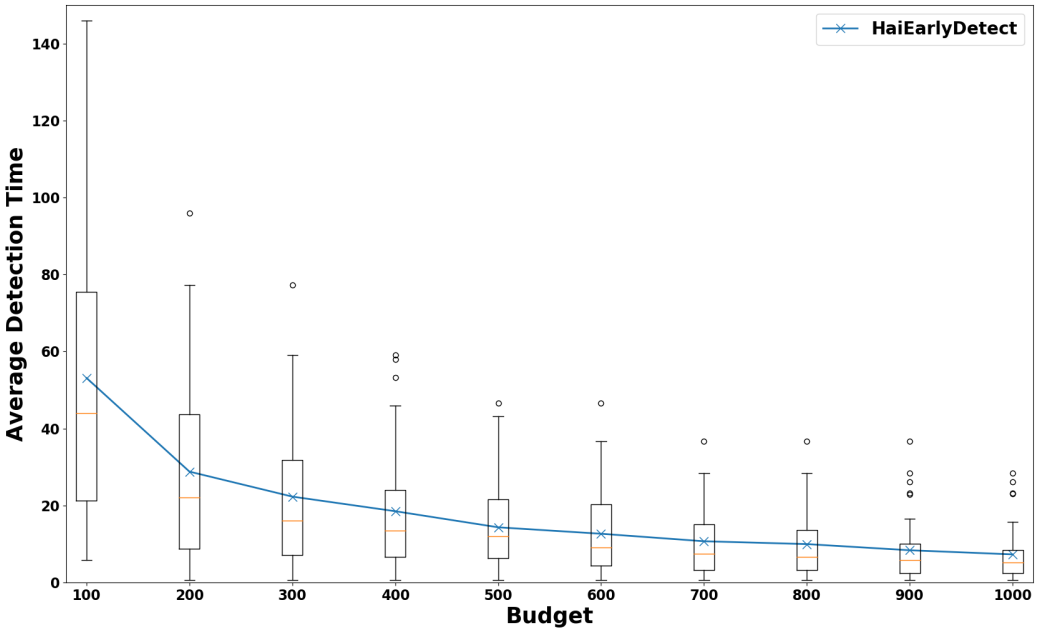


Figure 3.10: Variation in detection time for various budgets. As budget increases, the variation decreases.

Number of Cases Prevented

An interesting question is how many potential cases can be prevented by monitoring the sensors selected by HaiEarlyDetect. Here we study how many nodes get infected before an outbreak is detected and how many potential infection can be prevented by monitoring our sensors for various budgets. As in the previous experiment, for a given budget, we leverage 100 simulations to select sensors and their monitoring rates. Once the sensors are selected, we count the number of infection that occur in a test simulation before a sensor is infected and how many further infection occur following the infection of sensors. We then average these numbers over 100 test simulations. The results are summarized in Table 3.1.

As shown in Table 3.1, for the budget of 10 samples/swabs, 4.31 potential future infections could prevented. Note that there are only 22 infections on average per simulation. For the budget of only 200, 15.02 infections could be prevented, which is about 69% of potential infections. The number goes up to 17, or 77% for the budget of 1000. The result shows that even for a low budget (less than 200 swabs per day), our approach could help prevent a significant number of future infections.

Table 3.1: Average number of infections before an outbreak is detected by monitoring sensors selected by HaiEarlyDetect and potential number of infections prevented by detection the outbreak. For a budget of 1000 roughly 77% of potential cases are prevented.

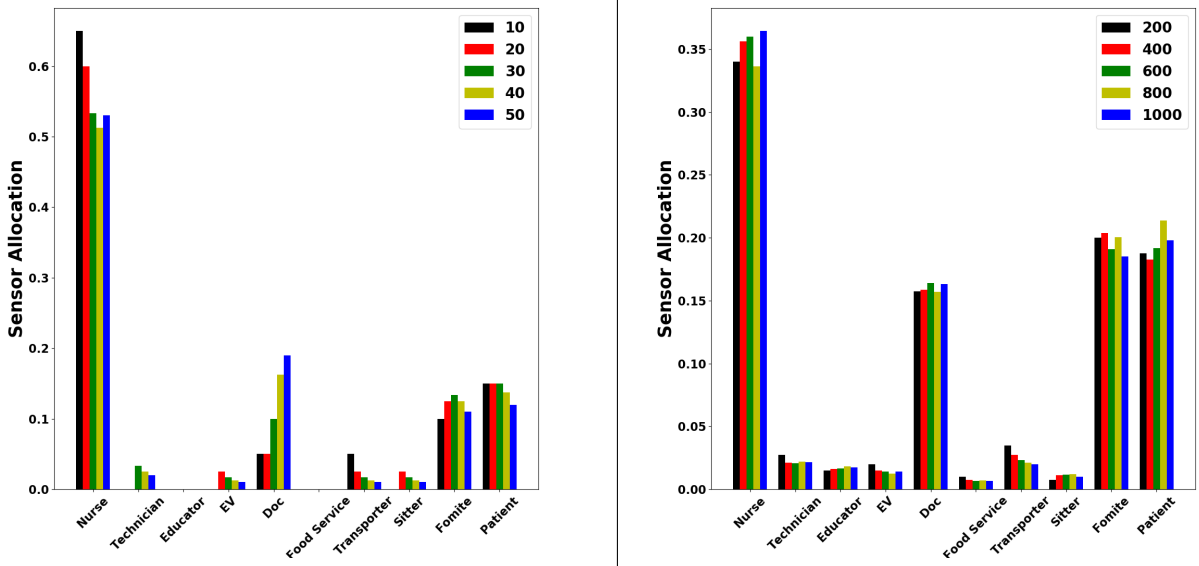
Budget	# infections when detected	# potential infections prevented
10	17.1	4.3
50	13.9	6.5
100	11.3	8.2
200	6.5	15.0
500	6.0	16.2
1000	5.4	17.0

Qualitative Distribution of Sensors

Next we study the types of agents that are selected by HaiDetect as sensors. For this experiment, we use 100 randomly selected simulations to detect sensors for wide range of budgets. After the sensors are selected, we sum up the rates of each category of agents like nurses, doctors, patients, and so on.

Figure 3.11 (a) shows the distribution of sensor allocation for each category of agents at low budgets. We observe that for a budget of 10, nearly 60% of the total budget is spent on selecting nurses. Since, nurses are the most mobile agents, the result highlights the fact that HaiDetect selects the most important agents as sensors early on. Similarly, Figure 3.11 (b) shows the distribution of sensors for higher budgets. Here we observe that nearly 35% of the budget is allocated for nurses. Fomites and patients have roughly equal allocations of about 20%. 17% of the budget is allocated to doctors. The rest of the categories have minimal allocation. The distribution shows that HaiDetect selects heterogeneous sensors including both people and objects/locations as intended.

Finally, we are also interested on the scheduling implications of the sensors selected by HaiDetect. To this end, we measure the aggregated proportion of budget assigned to each rate for the sensors we select. The results are summarized in Figure 3.12. As shown in 3.12 (a), most of the sensors have rate of 0.1. Very few sensors have rate from 0.2 to 0.5. Finally, there is a sudden spike at rate = 1.0. When we look at rate distribution for each category separately, interestingly we observe that only nurses have rates of 1.0. This implies that certain nurses have to be monitored each day to detect HAI outbreak. The reason behind this unexpected behaviour can be attributed to the fact that the hospital from where the mobility log was collected, required all the nurses to attend a daily meeting. Hence, all the nurses were in contact with each other every day and it is likely that nurses infect each other in case of an outbreak. Hence, there is an advantage in monitoring some of the nurses everyday to quickly detect HAI outbreak.



(a) Low Budgets (b) High Budgets

Figure 3.11: Variation of sensor set distribution for HAIDETECT with budget. HAIDETECT selects intuitively meaningful sensors even for lower budgets.

3.3 Discussion

Our results show that our sensor sets have good performance with respect to both the detection probability and detection time objectives, compared to the CELF baseline, as well as other natural heuristics motivated by standard practices in a hospital. A relatively low budget ensures a high detection probability, with low variance. Further, we find that training with a small set of simulation cascades (e.g., 20) is able to ensure reasonably high detection probability (about 0.8). This suggests the strategy is quite practical, since, in general, detailed knowledge of how HAIs cascade through a hospital system are difficult to obtain, thus motivating this agent-based modeling approach.

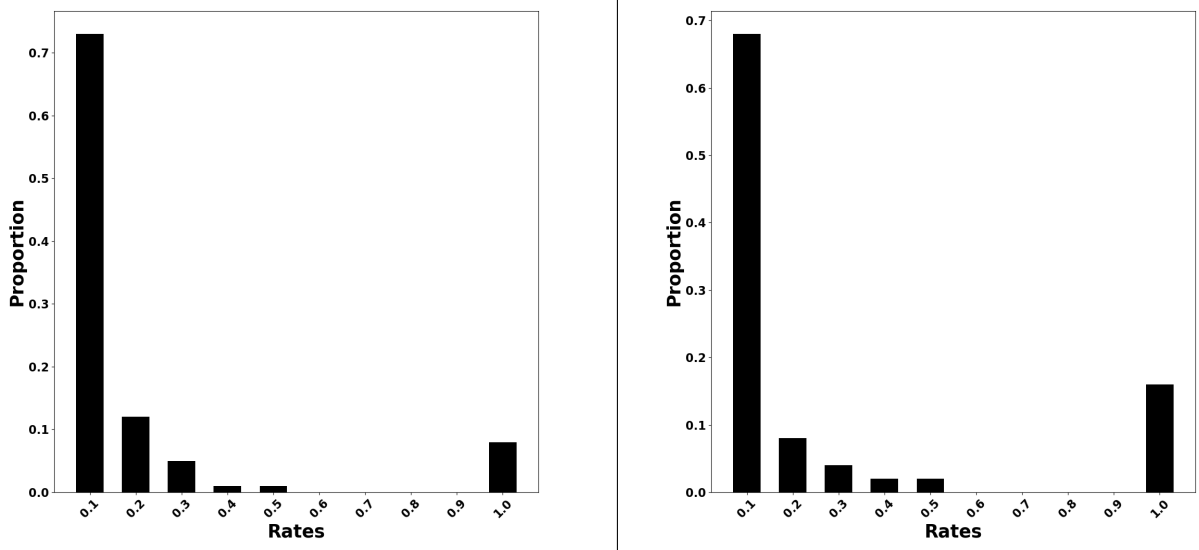
It is important to note that for simplicity of interpretation, we assume perfect detection at each sensor, thus these results to provide an estimate of best case performance in the real world. The sensitivity and specificity of detecting C. difficile on different fomites varies widely and the stochastic nature of this process would require extensive sensitivity analyses, which is beyond the scope of this particular study. Additionally, we did not simulate the course of care for patients in this simulation, which is also highly stochastic, thus we did not include detection outside of the designed sensor scheme. This assumption removes clinically appropriate detections, which would further improve the probability of detections reduce the time to detection.

An area for improvement of the algorithm is the lengthy time of detection anticipated for limited budgets under 100 sensors. A much larger budget is needed for quick detection times, which is a harder problem for myriad reasons. In particular, a budget of about 1000 is needed to get the detection time under a week. In contrast, monitoring strategies which monitor all agents of a specific type (e.g., all patients, all technicians, all nurses) lead to much lower detection time bounds. However, the number of nurses is almost 20% larger than this budget; further, monitoring all nurses is not a very practical strategy. Our sensor sets are much more practical in the sense that they include different types of agents (including fomites). Only when all nurses are monitored, is the detection time is close to a week. It is important to note that this time to detection is being measured from the very beginning of infection from the first case in the cascade. Many HAI infection cascades can remain undetected for many generations of transmission, which is impossible to measure in the real-world, and indeed may persist in a hospital for years. Thus a monitoring system that increases the probability of eventually detecting a cascade and provides expected detection times in the order of several months can represent a significant improvement.

Our approach requires mobility and activity data in a hospital, along with data on disease incidence. This type of data are difficult to obtain even well resourced field trials, especially in the quantity needed for the robust sensitivity analyses presented here. While these limitations were the primary motivation for the use of agent-based simulations, this simulated data remains a major limitation. While a field trial would provide more convincing evidence of the actual real-world performance of HaiDetect, the purpose here was to evaluate its performance and motivate its potential use in a resource intensive field trial. We also note that no prior work on HAI modeling and control studies consider problems at the level of detail we consider here.

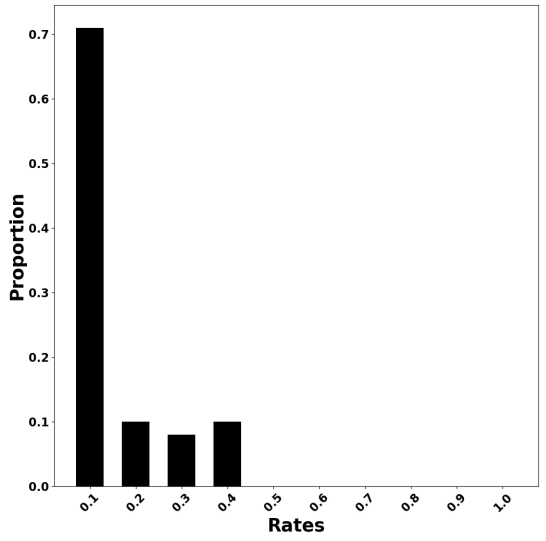
3.4 Conclusion

Effective and early detection of HAI outbreaks are important problems in hospital infection control, and have not been studied systematically so far. While these are challenging problems, understanding their structure can help in designing effective algorithms and optimizing resources. Current practices in hospitals are fairly simple, and do not attempt to optimize resources. Our algorithms perform better than many natural heuristics, and our results show that a combination of data and model driven approach is effective in detecting HAIs. Since there is limited data on disease incidence, good models and simulations play an important role in designing algorithms and evaluating them.

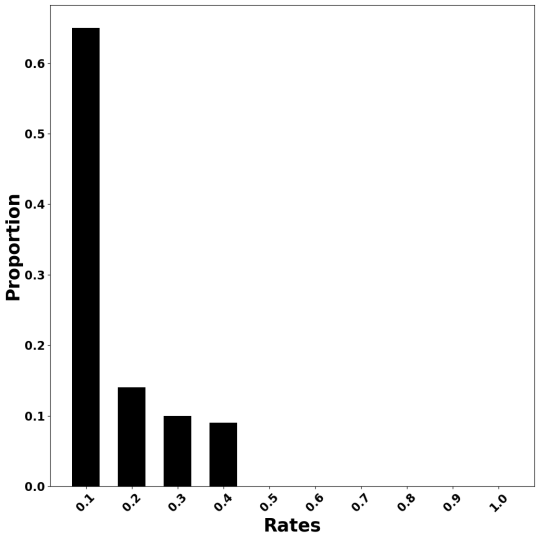


(a) Overall rate Distribution

(b) Rate Distribution for Nurses



(c) Rate Distribution for Fomite



(d)Rate Distribution for Doctors

Figure 3.12: Distribution of allocation for different rates. Since most of the sensors selected by HaiDetect have low rates they have to be monitored only sporadically.

Chapter 4

Graph-based E-Commerce Query Relations Mining

Search engine logs serve as an invaluable resource of customer interactions with a search engine. Each search session in the log, begins with the submission of a query formulated based on an initial intent, followed by a sequence of result engagement and query reformulation actions. Engagement with a result (e.g. clicks), signals its relevance to the customer's intent. Reformulation to a new query indicates either dissatisfaction with current results, or evolution of intent. Considerable attention is therefore paid toward mining meaningful information from search logs, and using it to improve various aspects of the system. In web-search domain, this is reflected in several prior works [27, 35, 117, 81], which propose novel search log representations, formulate various log mining tasks, and evaluate the utility of mined information in delivering measurable system improvements. Graph based representations such as click-graphs [118], cover graphs [27], query flow graphs [43], term graphs [221] etc. are frequently used. Popular mining tasks include identification of relationships (e.g. synonymy, generalization, specialization etc.) between query-query pairs, and relevance

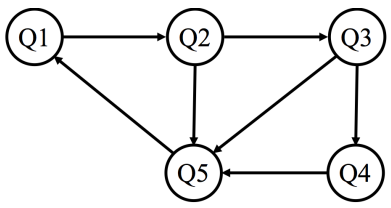
Query	Item	Time
Q1	I1	07 :00
Q2	I2	07:15
Q3	I3	07:20

Query	Item	Time
Q2	I3	09 :00
Q3	I4	09:15
Q4	I3	09:30

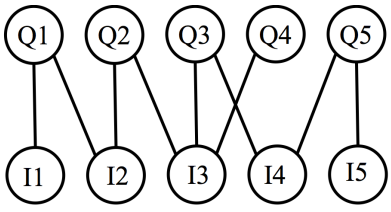
Query	Item	Time
Q2	I3	17 :00
Q3	I4	17:15
Q5	I5	17:30

Query	Item	Time
Q4	I3	17 :00
Q5	I4	17:15
Q1	I2	17:30

(a) Session logs



(b) *Query Reformulation* Network



(c) *Item Click* Network

Figure 4.1: Network Creation Process. Snapshots of four distinct session logs. Each entry in the log consists of a query, an item, the time of engagement among other data. (b) *Query Reformulation* network and (c) *Item Click* networks constructed from the session logs.

relationships between query-URL pairs. The mined information is then used for applications like related query recommendation, search quality improvement (via relevant URL retrieval) etc.

In context of E-Commerce search however, literature is sparse. There has been some work on analyzing E-Commerce search logs to study relationships between customers and products [148, 78]. Nevertheless, to the best of our knowledge, no formal study on the properties and utility of various query-query and query-item graph representations currently exists. Compared to web-search, the E-Commerce domain presents several unique characteristics, which make such a study interesting.

1. **Precise Intent:** Since E-Commerce search is a type of entity search, the notion of query intent is more precise compared to web search. In E-Commerce, intent can typically be represented by a well defined set of product attribute-value pairs expected by the query.
2. **Narrow Search Mission:** The goal of search is also narrow i.e. to buy a particular product, which makes sessions coherent. There are also clear task completion signals i.e. an item being added-to-cart or purchased.
3. **Category Hierarchy:** Products in an E-Commerce catalog are usually organized into a well defined category hierarchy, which can serve as useful ground truth for intent mining tasks.

These characteristics allow us to define better metrics around various graph mining tasks, and conduct large scale evaluations without using human input. A review of prior papers in web search domains suggests this is a significant problem, since evaluations are typically manual and consequently small in scale. Also owing to these differences, we use the term Customer Interaction Networks (CINs) as an umbrella term to refer to various graphs constructed using E-Commerce search logs, distinguishing them from their web-search counterparts.

In this paper, we present a formal study of the properties of various graphs constructed using E-Commerce search logs, with a focus on their utility toward mining query-query and query-product relationships. Our proposed graph mining based techniques provide us a complementary source for discovering query and product relationships, without directly using their textual content, making our techniques language independent. We begin by studying the properties of real-world customer interaction networks developed using Walmart.com's product search logs. We observe that CINs exhibit significantly different properties compared to other real world networks (e.g. WWW, social networks etc.), making it possible to mine intent relationships between queries, based purely on their structural information. We then leverage CINs for three different query relations mining tasks. Our main contributions are:

- **Empirical Study:** We study structural properties of four different CINs, namely *Query Reformulation*, *Item Click*, *Composite Click*, and *Cover* networks.

- **Graph Theoretic Problem Formulation:** We formulate intent based query clustering and critical query mining problems as formal Query Clustering and Critical Queries problems.
- **Algorithms:** We propose carefully designed efficient HubQExpansion and Critical-Queries algorithms to solve Query Clustering and Critical Queries problems respectively.

We have omitted proofs for some of the lemmas due to lack of space.

4.1 Related Work

Network Analysis. [50, 88] were among the first to study the macroscopic structure of large scale networks. Recently, Zhang et al. [264] studied the structure of expertise networks, Shen et al. [216] presented empirical study of E-commerce marketplace network, and Zlatic et al. [274] studied the network formed by hyperlinks among Wikipedia pages. Community detection is a well-studied problem in network setting [100, 171], including for bipartite networks [32] and heterogeneous networks [145]. Other important tasks related to our work include discovering influential nodes in networks [46, 129] and measuring node centrality [180, 45].

Query Graphs. Beeferman and Berger introduced click graphs [35] for clustering similar queries and URLs. Several subsequent works utilized click-graphs for other applications [93, 92, 73] like query-suggestion, document-search, relevance feedback and URL-annotation. Baeza-Yates [27] proposed various variants of query relation graphs including *Cover* graphs. Boldi et al. [43, 44] studied query-flow graphs and used them for related query suggestion. These are similar to our *Query Reformulation* networks (Section 4.2.1), except that they also use query content to decide the existence of query-query edges.

Query Relation mining. Many techniques have been employed in mining relations between queries. Query clustering has been exploited to group similar queries [35, 28, 251]. Some other works are based on association rules [90] and modeling users [273].

4.2 Data and Applications

4.2.1 Our Networks

In this work, we used session level customer interaction data collected over a year's period from Walmart.com. The collected data consists of information including query string,

clicked items, time of interaction etc. From these, we created four CINs, namely *Query Reformulation*, *Item Click*, *Composite Click*, and *Cover* networks (See Figure 9.1).

Query Reformulation Network. *Query Reformulation* network is a directed weighted network $G(Q, E, W)$, where each node $q \in Q$ is a query string. A directed edge (q_1, q_2) exists if query string q_2 was a consecutive reformulation of query string q_1 within a session. The weight $w(q_1, q_2) \in \mathrm{R}^+$ for edge (q_1, q_2) indicates frequency with which q_1 tends to get reformulated to q_2. To filter out noisy and insignificant data, we define some constraints. An edge (q_1, q_2) is added to the network $G(Q, E, W)$, only if the support of query q_1 is greater than δ_1 and the reformulation ratio from q_1 to q_2 is greater than δ_2 percentage. For our experiments both δ_1 and δ_1 were in the range $[0, 20]$[1]. Note that filtering out the noisy nodes and edges does not affect the performance of our algorithms as we are concerned only about significant reformulation relations. After clearing out insignificant edges and nodes, our final *Query Reformulation* network has 2.11 million nodes and 2.14 million edges.

Item Click Network. *Item Click* network is a bipartite weighted network $B(Q, I, E, W)$ where Q is the query partition and I is the item partition. A query node $q \in Q$ is a query string. An item node $i \in I$ is an item id. An edge $(q, i) \in E$ exits if a customer clicks on an item i after giving query q. The weight $w(q, i) \in \mathrm{R}^+$ for edge (q_1, q_2) indicates frequency with which i tends to get clicked for query q. Similar to *Query Reformulation* network, we filter out insignificant edges. Our final *Item Click* network has 5.4 million nodes and 18.4 million edges.

Composite Click Network. *Composite Click* network is a standard click network consisting of both query-to-query and query-to-item edges. We created *Composite Click* network by superimposing the *Query Reformulation* and *Item Click* networks. The resulting network has 6.3 million nodes and 20.5 million edges.

Cover Network. From our *Item Click* network, we inferred query-to-query *Cover* network. In the *Cover* network, two queries have an edge between them if they share an item neighbor in the *Item Click* network. Query to item click relations are clear indication of query intents. Hence, the edges in the the *Cover* network, inferred from click relations, naturally represent similar intents between two query nodes. These networks tend to get very dense. Hence we imposed additional threshold on edge-weight. The resulting graph has 785 thousand nodes and 71 million edges.

4.2.2 Applications

Since our CINs capture various facets of customer interaction data, they can be leveraged for various applications like query clustering, improving performance of queries with no engagement data, and so on. One can design methods based on query contents for these applications, however our goal here is to exploit the network structure to solve these problems.

[1]The exact values of the threshold is not disclosed due to confidentiality issues.

An advantage of leveraging the graph structure over a content-based approach is that graph based methods are language independent. Hence, our approach can be easily used for any E-commerce search system, regardless of the language it uses. Moreover, our methods are complementary to language/content-based approaches. Combining these two approaches is an interesting future work.

In this work, we focus on leveraging CINs for three different applications. Descriptions of the applications are as follows.

Intent Based Query Clustering: In E-commerce search, identification of query intent is crucial to returning relevant items. An intent of a query is a mapping to attribute-value pairs of the products. Ultimately, it is represented as a set of products.

Product Recommendation: In any E-commerce search system, one often encounters queries with no customer engagement data. In this application, we exploit query relations to recommend products for poorly performing queries.

Critical Queries: Critical queries are the queries which have the highest impact on the performance of other queries. In this application, we try to exploit structure of the *Query Reformulation* network to identify most critical queries. We formalize the notion of critical queries in a later section.

In the next sections, we first characterize various structural properties of our CINs. We then discuss how to exploit them for various applications.

4.3 Characterizing our Networks

It is well-known that most real networks like WWW, social networks, the Internet, buyer-seller networks, etc. [50, 88, 216] demonstrate specific regular structural properties. In this section, we investigate the structural properties of our CINs and show how they differ from other networks. These differences have a major implications for our applications.

4.3.1 Degree Distribution

Many real networks are scale free in nature, i.e, the in-degree and the out-degree follow power law distributions [50, 88]. The probability of a node having a degree θ in a scale-free networks is given by the probability density function $P(\theta) \propto \theta^{-\alpha}$. Another distribution that is prevalent in real networks is the log-normal distribution where $P(\theta) = \frac{1}{\sqrt{2\pi}\sigma\theta} e^{-(\ln\theta-\mu)^2/2\sigma^2}$ [163]. Both distributions are heavy tailed, i.e. they have (near) linear log density.

We first look at the degree distributions of the *Query Reformulation* network. The observed empirical pattern in the degree distribution of *Query Reformulation* network is summarized in the following observation.

Observation 1 ***Query-Query degree dist.*** *The in-degree distribution of the Query Reformulation network follows power law distribution with $\alpha = 2.41$, while the out-degree distribution follows log-normal distribution with $\mu = 0.12$ and $\sigma = 0.38$.*

The degree distribution plots for *Query Reformulation* network are presented in Figure 4.2. The in-degree follows power law distribution with multiple nodes having in-degree greater than 1000. On the other hand, the maximum out-degree is 9, which is negligible in comparison. Note that, our noise filtering process contributes in reducing the maximum value of out-degree in the *Query Reformulation* network to some extent. However, the exact value of the maximum out degree is much less than that warranted by our thresholds. This suggests that while it is probable that many queries get re-formulated into a single query consistently, it is not the case where one query repeatedly gets reformulated into many other queries. This observation is very different from other networks where both in and out degree tend to have similar power law distributions [50, 216]. We found the queries with highest in-degrees tend to be very general queries such as "sweatshirts", "tablets", "tv" etc. The in-neighbors of these queries tend to be more specialized queries such as "hooded fleece sweatshirts", "infant sweatshirts", "hp tablet", "htc tablet" etc.

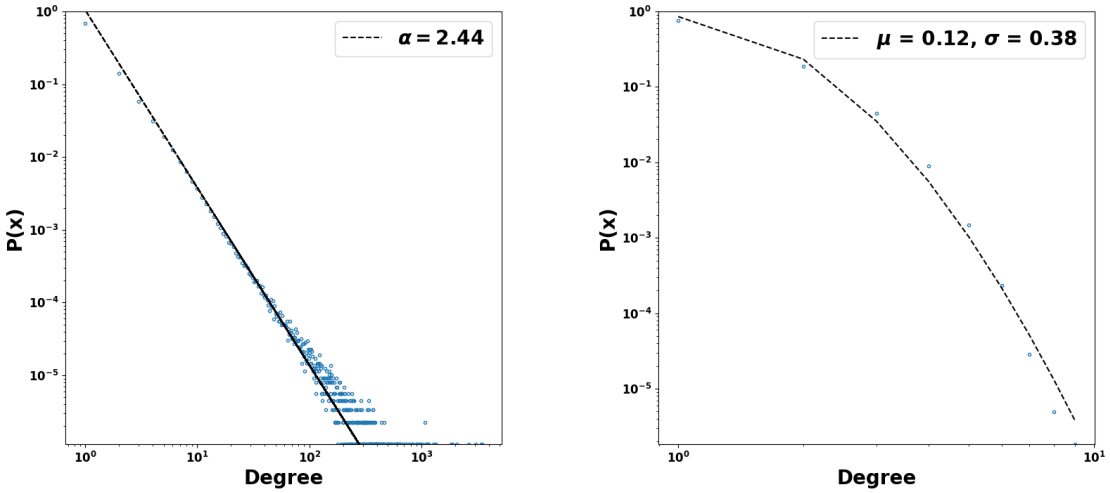
In the bipartite *Item Click* network $B(Q, I, E, W)$, we look at the degree distribution of the query partition Q and the item partition I individually. We observed that the degree distribution for both partitions follow log-normal distribution. Similarly, we also observed that the degree distribution for *Cover* network follows log-normal distribution while that for *Composite Click* network follows power law distribution.

In summary, we found that degree distributions for all of our CINs follow heavy tailed degree distributions. The heavy tailed degree distributions indicate that while there exist some popular queries which connect with many other queries and items, most queries connect only to a few queries and items. Hence, the networks (due to sparse connections only between relevant nodes) preserve the relationships between queries and items for the most part.

4.3.2 Assortativity and Degree Correlation

Degree assortativity, $r \in [-1, 1]$, is a measure of similarity between nodes and their neighbors in terms of degree [171]. Formally, degree assortativity is defined as the Pearson Correlation Coefficient of degrees between all pairs of connected nodes. The value $r = -1$ implies that the network is disassortative (negative correlation) and $r = 1$ implies that the network is assortative (positive correlation). Social networks are known to be assortative. However, other networks like protein-protein interaction network are known to be disassortative [170]. The observation regarding assortativity of our CINS networks is as follows:

Observation 2 ***Degree assortativity.*** *Query Reformulation, Item Click, and Composite Click networks are neither assortative nor disassortative, with $r = -0.02$, $r = -0.09$,*



(a) In-degree Distribution (b) Out-Degree Distribution

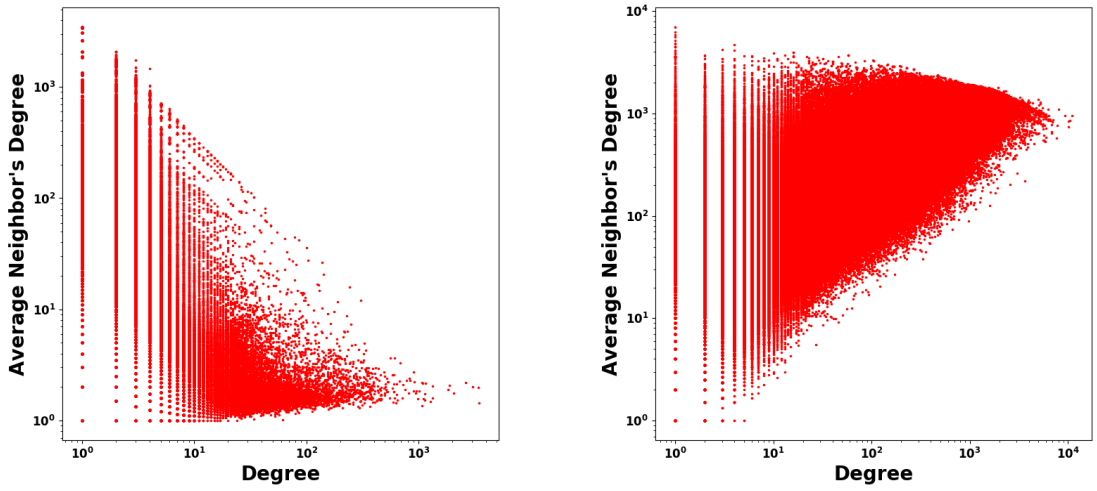
Figure 4.2: In and out degree distributions of *Query Reformulation* network.

and $r = -0.07$ *respectively, while the Cover network is assortative with* $r = 0.22$.

The assortativity plot for the *Query Reformulation* and *Cover* networks are shown in Figure 4.3. For *Query Reformulation* network, we observe that the neighbors of high degree nodes have very low degrees. On the other hand, high degree nodes connect to each other predominantly in the *Cover* network. For *Item Click* and *Composite Click* networks, we do not observe any asymmetrical pattern. The positive assortativity of the *Cover* network implies that it is ill-suited for query intent mining, as general influential queries which typically have distinct intents, tend to connect to each other. The degree distribution and assortativity of the *Query Reformulation* network suggests the dominance of star-like structures in the network. It highlights that unpopular queries are typically reformulated to related popular queries, capturing the intent of the queries.

4.3.3 Connected Components, Diameter, and Clustering

Many real directed networks are known to have the "bow-tie" structure with a giant strongly connected component (SCC)[50, 264]. It is reported that the WWW has SCC consisting of 27.7% of the nodes [50], while community expertise network for Java forum has SCC consisting of 12.3% of the nodes [264]. In our only directed network, the *Query Reformulation* network, we do not find the "bow-tie" structure, with just 300 out of 2.11 million nodes in the largest strongly connected component. The reason for absence of "bow-tie" structure can be attributed to customer behavior. It is unlikely that customers reformulate a query



(a) *Query Reformulation* *Cover*

Figure 4.3: Assortativity Plots (degree vs average neighbor's degree) for *Query Reformulation* and *Cover* networks.

with an specific intent to a query with drastically different intent repeatedly. Most significant reformulations are related, thus creating distinct partitions of graphs which are not reachable from each other in both directions. On the other hand, the web pages can arbitrarily link to one another in the WWW and people with different expertise may interact with each other in the Java Forum network, which leads to a formation of SCC in these networks.

Another common property exhibited by most real world networks is the "small world" phenomenon, commonly referred to as six degree of separation. Very large real networks like the WWW, social networks e.t.c are known to have small diameters [240, 50]. However, all three of *Query Reformulation*, *Item Click*, and *Composite Click* networks have relatively large diameters of 94, 37, and 36 respectively. This suggests that "weak links" are missing in these networks which implies customers do not typically search for unrelated queries one after another and do not click on arbitrary items for a given query significant number of times. On the other hand, the diameter of the *Cover* network is only 12, which suggests that *Cover* network does not consist of regions representing homogeneous intent.

Average Clustering Co-efficient, $ACC \in [0, 1]$, of a network measures how well the nodes are clustered together. The value of $ACC = 0$ indicates that the network is not clustered at all, whereas $ACC = 1$ indicates that the network is well clustered. For bipartite networks, clustering co-efficient is defined in terms of overlapping neighbors of nodes in the same partition [136]. We computed the average clustering co-efficient for all of our networks and observed that the *Query Reformulation*, *Item Click*, and *Composite Click* networks have very low clustering co-efficient of 0.05, 0.12, and 0.07, respectively, while the *Cover* network has

Table 4.1: Summary of properties of CINs. QQ stands for *Query Reformulation*, Qi for *Item Click*, QQI for *Composite Click* and C for *Cover* networks. ACC stands for average clustering co-efficient.

Properties	QQ	QI	QQI	C
degree	power-law log-normal	log-normal	power-law	log-normal
assortativity	none	none	none	positive
diameter	94	37	36	12
ACC	0.05	0.12	0.07	0.76

very high clustering co-efficient of 0.76. The clustering co-efficient gives further validation of previous implication that *Query Reformulation*, *Item Click*, and *Composite Click* network are suitable for query intent mining, while the *Cover* network is not.

4.3.4 Summary

In this section we explored various properties exhibited by our CINs. The properties of our networks indicate that they are different from common real world networks and that they preserve relevance between queries and items. Thus, our CINs can be leveraged for various query mining tasks. In the next three sections, we explore applications of CINs in query intent mining and product recommendation.

4.4 Application 1: Intent Based Query Clustering

In E-commerce search, identification of query intent is crucial to returning relevant items. However, in practice, one encounters with many queries with ambiguous intent due to very little engagement data. An approach to identify intent of such queries is to cluster them with other queries whose intent is known and leverage the general intent of the cluster to recommend product for queries with low engagement data. Clustering queries based on intent is known to be useful in many potential applications like query recommendation, categorization etc. in both web and E-commerce search [35, 28].

Since our *Query Reformulation* network captures the significant reformulation relations, we propose to exploit the *Query Reformulation* network to cluster the queries with same intent.

4.4.1 Problem Formulation

Recall that the *Query Reformulation* network is a query-to-query reformulation network. Hence neighboring queries in the *Query Reformulation* network are similar to each other. Therefore, intuitively a community in the *Query Reformulation* network is expected to consist of queries with similar intent. Hence the problem of intent based query clustering in the *Query Reformulation* network is well-founded. The problem can be stated as follows:

Informal Problem 1 QUERY CLUSTERING

GIVEN: *A Query Reformulation network $G(Q, E, W)$, and an integer $k \in \mathrm{Z}$.*

FIND: *A k partition of Q, such that each partition contains queries with the same intent.*

To formalize Informal Problem 1, two questions must be addressed (i) How is intent defined in terms of graph structure? (ii) How to measure 'closeness' between two queries in terms of intent?

To address the first question, we rely on the empirical study. As mentioned in Section 10.2, nodes with high in-degree tend to be general queries with broad intent like 'tv', 'phone', 'sweater' etc. Majority of specific queries reformulated to these general queries tend to have similar intents. Hence, these general queries with high in-degree nodes in the *Query Reformulation* network are good candidates to represent the intent. To address the second question, we look at the edge relation in the *Query Reformulation* network. Each edge in the *Query Reformulation* network represents significant reformulation. Therefore, shorter reformulation paths from one query to another is a good indication of similar intents and vice-versa. Hence both questions (i) and (ii) can be answered in terms of the graph structure.

Next, we formalize Informal Problem 1 leveraging two graph properties (i) high in-degree nodes and (ii) shortest paths. Given a *Query Reformulation* network and the number of distinct intents k, our goal is to discover k disjoint partitions $\{C_1, C_2, \ldots, C_k\}$. Since intents are well-represented by the high in-degree nodes, we formalize the problem by asking to find a set $\mathcal{S} = \{s_1, s_2, ..., s_k\}$ of such nodes and partitions $\mathcal{C} = \{C_1, C_2, ..., C_k\}$, such that C_i is the partition with the intent represented by s_i. Moreover, since the short reformulation path indicates closer intent between queries, we require nodes in C_i to have a short distance to s_i.

Let $\theta^i(v)$ be the in-degrees of v; $d(a, b)$ be the shortest hop distance between two nodes a and b; and $s(v)$ be the node in $\mathcal{S}$ such that both the nodes v and $s(v)$ belong to the same partition (i.e. $s(v)$ is the seed node of the community v belongs to). Now, our formal problem, purely in terms of network structure, can be stated as follows:

Problem 3 *Given a query reformulation network $G(Q, E)$ and an integer k, identify a set $\mathcal{S}^* = \{S_1, S_2, ..., S_k\}$ of the general query nodes and set of partitions $\mathcal{C}^* = \{C_1, C_2, ..., C_k\}$,*

such that $S_i \in C_i$ *and*

$$S^*, C^* = \arg\min_{S,C} J(S, C) = \arg\min_{S,C} \left[\left(\sum_{v \in V} d(v, s(v)) \right) \left(\sum_{s \in S} \frac{1}{\theta^i(s)} \right) \right]$$

4.4.2 Methods

Since our original problem requires partitions, traditional community detection methods are natural baselines for our problem. Hence we use an existing community detection method based on modularity [172], an overlapping community detection method and a heuristic specifically designed for Problem 3 as baseline methods. Brief descriptions are as follows:

- LOUVIAN: We used the popular LOUVIAN method to maximize modularity in *Query Reformulation* network [42].
- BIGCLAM: It is an overlapping community detection method based on bipartite affiliation model [259].
- LOUVIANSMALL: Most queries share intent with few other queries. Hence, we modified the LOUVIAN to generate smaller communities by defining threshold on the first stage of the LOUVIAN algorithm.
- STAR: Since *Query Reformulation* network is dominated by star-like structures, we generate star shaped communities by clustering high in-degree nodes with their neighbors. This approach is designed to choose high degree nodes as the community center. Hence, it is a heuristic for Problem 3.

We run LOUVIAN to cluster the *Cover* and the *Composite Click* networks as well. Since, the *Composite Click* network is a heterogeneous network, we also used modified version of LOUVIAN to maximize the composite modularity [145] defined on heterogeneous networks. We name this method COMLOUVIAN.

While traditional community detection methods are natural baselines for Problem 3, they would be sub-optimal as they do not directly optimize the given objective. Our main idea is to leverage the structural properties of the *Query Reformulation* network instead, to solve Problem 3. We exploit the following properties: (a) over half of the nodes in the *Query Reformulation* network lie outside the giant weakly connected component, (b) the assortativity plots (see Figure 4.3) shows that the high in-degree nodes are very unlikely to have an edge between them, and finally, (c) the *Query Reformulation* network has low clustering co-efficient and long diameter which indicates that the queries with distinct intents are well separated. Based on these observations, we propose our algorithm HUBQEXPANSION

(Hub-Query Expansion) for clustering queries with similar intents in E-Commerce *Query Reformulation* network.

Property (a) indicates that the significant number of queries exist outside the giant connected component, hence we cluster queries in each connected components. We distribute the number of communities to be found in each connected component proportionally to their size, i.e., for each connected component $G_i(Q_i, E_i, W_i)$ in G, the number of community to be found is set to $k_i = \frac{|Q_i|}{|Q|}$. Following the property (b), we assign k_i nodes with highest in-degree in the component G_i, to their own community. Assigning high degree nodes to to their own community is justified as they tend to be general queries and it is intuitive that general queries like 'tv' and 'sweater' have distinct intents. Finally, (c) suggests that queries with distinct intents are well separated. Hence, we expand the communities using breadth-first search. We continue community expansion until all the nodes in the connected component are assigned to a community. The complete pseudocode is in Algorithm 1.

The objective in Problem 3, involves two terms $\sum_{v \in V} [d(v, s(v))]$ and $\sum_{s \in S} \left[\frac{1}{\theta^i(s)}\right]$. Intuitively, Algorithm 1 tries to optimize the second term of the objective by assigning high in-degree nodes as the cluster centers and the first term by assigning nodes to the same community as the closest (shortest-path) cluster centers. Since we observe that the high in-degree nodes tend to have short paths to many queries and also are well-separated with each other, we expect the solution obtained from Algorithm 1 to minimize both terms in the objective and result in a good solution to Problem 1.

Algorithm 1 HubQExpansion

Require: *Query Reformulation* network $G(Q, E, W)$, number of communities k
Ensure: k disjoint partitions of Q

```
Partition P = ∅
for each connected component G_i(Q_i, E_i, W_i) in G do
    k_i = |Q_i| / |Q|
    Temp set S = ∅
    for node v in k_i nodes in Q_i with highest in-degree do
        S = S ∪ {v}
        Assign nodes in Q_i to nodes in S using BFS
    P = P ∪ S
return P
```

Lemma 1 *Algorithm 1 has linear time complexity of $O(m + n)$, where m is the number of edges and n is the number of nodes.*

Table 4.2: Performance of LOUVIAN on *Query Reformulation*, *Composite Click*, and *Cover* networks. The table shows AIH, AIS, and $F1$ based on categories. The performance of LOUVIAN on *Query Reformulation* network is the best.

Networks	AIH_{cat}	AIS_{cat}	$F1_{cat}$
Query Reformulation	0.26	0.11	**0.15**
Composite Click	0.07	0.31	0.11
Cover	0.05	0.54	0.09

Table 4.3: Performance of various methods for Query Clustering in *Query Reformulation* network. The table shows AIH, AIS, and $F1_{cat}$. The final objective value J is also shown. HUBQEXPANSION outperforms all the baselines.

Method	AIH_{cat}	AIS_{cat}	$F1_{cat}$	$J(\times 10^6)$
LOUVIAN	0.26	0.11	0.15	19.7
COMLOUVIAN	0.07	0.33	0.12	118.7
LOUVIANSMALL	0.39	0.08	0.13	0.73
STAR	0.38	0.12	0.18	3.01
BIGCLAM	0.14	0.21	0.17	17.7
HUBQEXPANSION	0.37	0.14	**0.20**	**0.54**

4.4.3 Experiments

Metrics. Measuring how well the methods minimize the objective in Problem 3 demonstrates their ability in solving the problem. However, it does not indicate how well the communities are clustered in terms of their intents. Since sets of relevant items were not available for most queries, we treat product category learned from an accurate tagger as the proxy for query intents. Intuitively, if two queries have associated items in common, they should also have product categories in common. Hence, product categories are good proxy for intent. We obtained categories for 267K queries, which we use to evaluate all the methods.

A measure of cluster goodness is the categorical homogeneity of each community. To that end, for a community C, we define its Community Intent Homogeneity CIH as the fraction of node pairs which share a category, i.e., $CIH(C) = 2 * \frac{\sum_{q_i,q_j \in C} \delta(PC(q_i),PC(q_j))}{|C| \times |C-1|}$, where $PC(q_i)$ represents the category associated with node q_i and $\delta(a,b) = 1$ if $a = b$, 0 otherwise. Note that for CIH, we only include the nodes for which category information is available. We then compute the Average Intent Homogeneity AIH_{cat} for a partition P as $AIH_{cat} = \frac{\sum_{C \in P} CIH(C)}{|P|}$. The AIH_{cat} score of 0 represents that the communities in the partition are heterogeneous, while the AIH_{cat} score of 1 represents that the communities are perfectly homogeneous.

AIH_{cat} has a drawback as the smaller communities tend to get higher score. Hence, to overcome this, we also measure the number of communities in which a category is represented. Ideally, we would want each category to be represented in a single community. Hence, we measure average inverse spread AIS_{cat} of a category as $\frac{1}{Spread}$, where *Spread* is defined as the average number of communities the categories are represented in. The AIS_{cat} score of 1 represents that each category is represented in a single community, while a score close to 0 represents that the categories are spread across communities. Finally, we compute $F1_{cat}$ as the harmonic mean of AIH_{cat} and AIS_{cat}.

Performance. First, we ran LOUVIAN on the *Query Reformulation*, the *Composite Click*, and the *Cover* networks. The results are summarized in Table 4.2. The results show that the clusters obtained from the *Query Reformulation* perform the best, indicating that it is the most suitable network for query clustering.

Next, we ran all the methods in the *Query Reformulation* network and COMLOUVIAN in the *Composite Click* network. First, we computed performance of the methods with respect to the objective of Problem 1. The result is presented in Table 4.3. As expected HUBQ-EXPANSION outperforms all the baselines in terms of the Problem 3 objective. Next, we computed the intent based metrics described above. The results are summarized in Table 4.3. As observed, HUBQEXPANSION outperforms all the baselines. LOUVIAN performs decently indicating that traditional community detection methods are indeed suitable for Problem 3. Poor performance of LOUVIANSMALL indicates that artificially creating smaller clusters prevents good clusters from forming. Naive STAR heuristic performs well due to the fact that communities in the *Query Reformulation* network are centered around the 'popular' queries. However, HUBQEXPANSION outperforms all the methods, since it exploits the unique structure of the *Query Reformulation* network to find communities with the same intent.

4.5 Application 2: Product Recommendation

Improving search quality for queries with no customer engagement data is a challenging task. In this section, we propose to leverage the *Composite Click* network to associate items with poorly performing queries and evaluate whether such a method does in fact improve search quality for queries with no engagement data.

4.5.1 Problem and Method

In this task, we explore whether product recommendations made based on the *Composite Click* network could help improve search quality for poorly performing queries. We used the current Walmart.com product search engine as the baseline, and identified poorly performing queries. The criteria for selection was queries in the lowest $10th$ percentile in terms of click

through rate, and a conversion rate, defined as ($\frac{\#Querieswithorders}{\#Queries}$), of 0. For variation, we used a random walk based method similar to [73], except that our network is directed (we treat query-to-item links as directed here), and there are no out going edges from the items, making them sink nodes. The unnormalized edge weights for our *Composite Click* network were computed as follows:

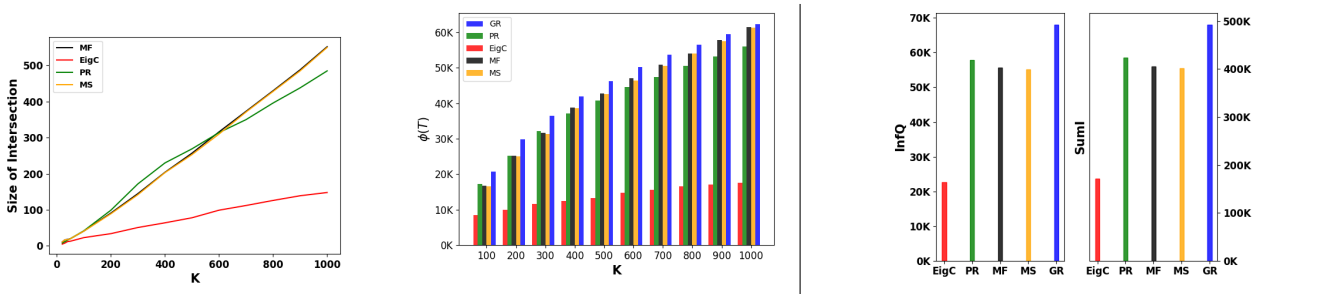
- **Query q_1 to q_2 edge:** $w(q_1, q_2) = \frac{c(q_1,q_2)}{c(q_1)}$
- **Query q to product i edge:** $w(q, i) = \frac{c(q,i)}{c(q)}$,

where $c(q_1, q_2)$ represents the number of times q_1 is formulated to q_2, $c(q_1)$ represents the number of times q_1 occurs, and $c(q, i)$ represents the number of times product i is clicked for query q. We further normalized the weight of each directed edge (q, x), where x may be a query or product, by the sum of all out-going edges from the base node q.

In order to make product recommendations for some query q, we started with a weight of 1 at node q, and spread it across the graph by executing random walk iterations. After several iterations (50 were usually enough), a portion of the weight settled on the product nodes. Top 5 highest weighted products were then used as recommendations for q. Our variation ranking method injected these recommended products into the top 10 search results for query q demoting some of the original results below 10th position.

4.5.2 Experiments

For evaluation, we identified a random sample of 136 poorly performing queries[2]. A dataset of query-product pairs was created by obtaining top 10 results for each query, from both control and variation. Expert E-Commerce analysts were then asked to assign a relevance rating between $0 - 4$ for each query document pair, 4 being extremely relevant and 0 being irrelevant. We observed that our recommendation based variation performed significantly better achieving a 34% improvement in average NDCG@10[112] (baseline NDCG10: 0.439, NDCG10: 0.588). Out of 136, 99 queries were improved, while 31 were degraded. From a practical point of view the degradations are not harmful, since the queries already have poor conversion rates. This observation was further backed up by our online A/B test evaluation which showed a statistically significant 5.8% lift in click through rate and 6.9% lift in conversion.



(a) Intersection (b) $\phi(\mathcal{T})$ (c) Item relevance based performance

Figure 4.4: (a) Intersection between top-k critical queries returned by Algorithm 2 and other baselines. (b) Our method outperforms all the baselines in terms of $\phi(\mathcal{T})$. (c) Our method performs the best based on usability metrics.

4.6 Application 3: Critical Queries

In the previous section (Section 4.5), we showed that the engagement data from one query can be leveraged to improve the performance of other related (via reformulation relation) queries. A natural question that arises in this setting is which queries have the highest cumulative impact on the performance of other related queries? We refer to these as the 'Critical Queries'.

Mining the critical queries is important for improving the overall performance of the search system. Typically in processes like manual curation, which aims at improving search quality by manually improving the search results, the queries that appear most frequently in the search log are selected. However, this yields a sub-optimal improvement in the overall performance as these most-frequent queries do not necessarily improve the performance of other queries. On the other hand, by definition, critical queries have the highest impact on the performance of the related queries. Hence, correctly identifying the critical queries for the curation process would yield a better improvement in performance. Note that the critical queries can be leveraged for many other tasks in E-commerce like measuring performance of a search system, identifying broad search categories and so on.

Since our *Query Reformulation* network captures the reformulation relation well, we propose to mine the critical queries by leveraging the *Query Reformulation* network.

4.6.1 Problem Formulation

To formalize the problem of choosing the most critical queries, we need to correctly model customer behavior during the query reformulation process. We model customer behavior in query reformulation as a discrete-time dynamical process that occurs over the *Query*

[2]Number of queries was obtained based on the available crowdsourcing budget.

Reformulation network. We call it the RANDOMIZED USER NAVIGATION (`RUN`) model.

`RUN` Model: In an E-Commerce search system, a customer (user) submits an arbitrary query to the search system. A list of items relevant to the query are displayed. The customer, depending on many factors like satisfaction with the search result, relevance of products etc. may decide to submit another query or exit the search system.

This process can actually be viewed as a discrete-time probabilistic dynamical process over the *Query Reformulation* network $G(Q, E, W)$. Given a current node v, we proceed as follows:

1. With probability p_t, the process terminates
2. With probability $1 - p_t$, we continue the process and jump from current node v to a query node u, such that $(v, u) \in E$, with probability $p_j = \frac{w(v,u)}{\sum_{(v,a)\in E} w(v,a)}$

The process starts from an arbitrary query node sampled from Q uniformly at random. Note that, an instance of `RUN` model produces a sequence of queries which we call 'reformulation logs'.

Now, let $\mathcal{T}$ be a set of nodes. We define $\phi(\mathcal{T})$ as the probability that an arbitrary instance of `RUN` model goes through at least one node in $\mathcal{T}$. Empirically, $\phi(\mathcal{T})$ is the fraction of times at least one node in $\mathcal{T}$ appears in reformulation log produced by repetitions of the `RUN` model.

Remark. Since both PAGERANK's Random Surfer model [180] and our `RUN` model simulate a random walker over a network, they appear to be similar. However, PAGERANK's Random Surfer model is distinct from our `RUN` model as it has no notion of a termination probability and the walker can teleport to any node in the network. Given enough time, every node $v \in Q$ is visited and hence, $\phi(\mathcal{T})$ for any set $\mathcal{T}$ is always 1 under the Random Surfer model—which is not the case for the `RUN` model. The `RUN` model is also distinct from the cascade style models (like IC [129]) as only a single node is visited at a time in the `RUN` model, whereas the 'contagion' spreading in the cascade models can infect multiple nodes at once (depending on who else was 'infected' in the previous time-step)

Having defined the `RUN` model and $\phi(\cdot)$, we can state our Critical Queries identification problem formally as follows:

Problem 4 CRITICAL QUERIES

GIVEN: *A Query Reformulation network* $G(Q, E, W)$*, and budget* $k \in \mathbb{Z}$.

FIND: *a set of nodes* $\mathcal{T}^* = \{q | q \in Q\}$*, such that* $|\mathcal{T}^*| = k$ *and*

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \phi(\mathcal{T})$$

.

4.6.2 Methods

Problem 4 is NP-hard (we can reduce from the SetCover problem; proof omitted due to space). Although it is challenging to solve Problem 4 optimally in an efficient manner, one can use various centrality measures or query logs based heuristics to identify critical queries. Some of these methods can be the following:

- MostFreq (MF): In this method, we select the queries that had the highest frequency from the same data from which the *Query Reformulation* network was created.
- SessionFreq (MS): In this method, we select the queries that appeared in most sessions.
- PageRank (PR): We pick the nodes with highest page rank [180] on the *Query Reformulation* network.
- EigCentrality (EigC): We pick the nodes with highest eigenvector centrality [45] on the *Query Reformulation* network.

None of methods mentioned above solve Problem 4 directly. Hence, we seek for a fast algorithm with a performance guarantee. It turns out that $\phi(\cdot)$ is sub-modular [129]. A function $f(\cdot)$, which maps a set to a real number, is sub-modular if it satisfies the diminishing return property i.e. $f(\mathcal{A} \cup \{v\}) - f(\mathcal{A}) \geq f(\mathcal{B} \cup \{v\}) - f(\mathcal{B})$, for any element v, and sets $\mathcal{A} \subset \mathcal{B}$. Next, we prove that $\phi(\cdot)$ is sub-modular and monotonous.

Lemma 2 *$\phi(\cdot)$ is sub-modular and monotonous.*

Proof 1 *In the* `RUN` *model, a customer at node q_i makes a decision to whether make a random jump to one of q_i's out-neighbors and which node to jump to. Suppose the customer makes the random decision before the process, i.e, decides on the number of* `RUN` *model iterations l, the starting nodes, and the node jumps beforehand. Let the set D be the set of outcomes of the random decisions. Note that given D, these k iterations of* `RUN` *are a deterministic processes, which produce l reformulation logs.*

Now, let $\mathcal{A}$ and $\mathcal{B}$ be two sets of nodes of Query Reformulation network and let $\mathcal{A} \subset \mathcal{B}$. Consider a node v such that $v \notin \mathcal{B}$. Also let $\phi_D(\mathcal{A})$ be the fraction of logs that go through any node is $\mathcal{A}$ given D. Now, $\phi_D(\mathcal{A} \cup \{v\}) - \phi_D(\{v\})$ is the fraction of logs that go though v, which does not already go through any node in $\mathcal{A}$. This value is definitively larger than or equal to $\phi_D(\mathcal{B} \cup \{v\}) - \phi_D(\{v\})$, since $\mathcal{A} \subset \mathcal{B}$. Hence, $\phi_D()$ is sub-modular.

Now, $\phi(\mathcal{T})$ equals $\sum_{Decision\ D} Prob(D)\phi_D(\mathcal{T})$.

Since, any linear combination of sub-modular functions is also sub-modular, $\phi(\cdot)$ is sub-modular. Since, $\phi(\cdot)$ is a non-decreasing function, it is also monotonous.

Speeding up. Due to Lemma 2, a simple greedy algorithm will give $1 - 1/e$ approximation [169]. However, such a method is expensive due to repetitive simulations. Leveraging the idea in [64], we propose sampling based greedy algorithm CRITICAL-QUERIES. First of all, we initialize set $\mathcal{T}$ to an empty set. We then sample a graph G' from G based on the number of iterations l and termination probability p_t. Given the deterministic network G', we compute $\phi_{G'}(\mathcal{T})$ and $\phi_{G'}(\{v\})$ for every v in G'. We repeat such process R times and choose v with the highest average gain in $\phi(\cdot)$. We repeat the entire process until $|\mathcal{T}| = k$. Note that there exist other techniques from related problems to speed up our algorithm [82, 46]. However, we chose this method for its simplicity. The complete pseudocode is presented in Algorithm 2.

Algorithm 2 CRITICAL-QUERIES (GR)

Require: *Query Reformulation* network $G(Q, E, W)$, termination probability p_t, number of iterations of `RUN` l, and budget k
Ensure: Best set of nodes $\mathcal{T}$

$\mathcal{T} = \emptyset$
for $i = 1$ to k **do**
  $g_v = 0$ for all $v \in V \setminus \mathcal{T}$
  for $i = 1$ to R **do**
    Sample G' based on p_t and l
    compute $\phi_{G'}(\mathcal{T})$
    for $v \in V \setminus \mathcal{T}$ **do**
      $g_v += \phi_{G'}(\{v\})$
    $g_v = g_v/R$ for all $v \in V \setminus \mathcal{T}$
    $\mathcal{T} = \mathcal{T} \cup \{\arg\max_v(g_v)\}$
return $\mathcal{T}$

As shown by the next two lemmas, Algorithm 2 gives a provable approximation guarantee and has near linear time complexity.

Lemma 3 *Algorithm 2 provides a* $(1 - 1/e)$ *approximation to Problem 4.*

Lemma 4 *The time complexity of Algorithm 2 is* $O(kR(n + m))$.

4.6.3 Experiments

Metrics. While the value of $\phi(\mathcal{T})$ for various methods indicates the quality of $\mathcal{T}$, it does not highlight usefulness of queries in $\mathcal{T}$. Hence we define two additional metrics to measure usability of $\mathcal{T}$.

To improve state of search system by curating the $\mathcal{T}$, the ancestor nodes in *Query Reformulation* network, i.e. queries leading up to $\mathcal{T}$ must actually be related to the queries in $\mathcal{T}$.

Hence, we measure whether the ancestor queries of $\mathcal{T}$ are actually relevant to it or not. To this end, we determined the set of items related to each query in $\mathcal{T}$ and computed the number of ancestor queries, $InfQ$, that had at least one common relevant items with the queries in $\mathcal{T}$. Formally, let $\mathcal{A}^d$ be the set of ancestors within distance d of all query q in $\mathcal{T}$. Let, the set of items relevant to set of queries $\mathcal{T}$, be $\mathcal{I}(\mathcal{T})$. Now, we calculate $InfQ$, influenced queries, as $InfQ = \sum_{q \in \mathcal{A}^5} \mathbb{1}(|\mathcal{I}(\mathcal{T}) \cap \mathcal{I}(\{q\})| \geq 1)$. Similarly, another metric of interest is how close the ancestors are related to the critical queries. To capture the notion of overall relation between queries and their ancestors, we also compute the size of set of items, $SumI$, which are relevant for both the nodes in $\mathcal{T}$ and their ancestors as $SumI = |\mathcal{I}(\mathcal{T}) \cap \mathcal{I}(\mathbf{A}^5)|$.

Performance. For Critical-Queries, we set p_t as 0.7 and l as the number of nodes in the network. We ran all the methods on the *Query Reformulation* network. First of all, we check the whether the set $\mathcal{T}$ returned by Critical-Queries is distinct from the ones returned by the baselines. In Figure 4.4 (a), we plot the size of intersection of $\mathcal{T}$ returned by various methods and $\mathcal{T}$ obtained from Critical-Queries against, k, the size of $\mathcal{T}$. At least 45 % of nodes returned by Critical-Queries are not present in sets returned by any other method. Hence, Critical-Queries returns the critical queries which other methods fail to discover.

The performance of all the methods in terms of $\phi(\mathcal{T})$ is shown in Figure 4.4 (b). As we can see, Critical-Queries consistently outperforms all the baselines for multiple values of k in terms of $\phi(\mathcal{T})$. PageRank (PR) and EigCentrality (EigC) and have poor performance as they tend to choose nodes which are close to each other and return a set of similar queries. The MostFreq (MF) and SessionFreq (MS) heuristics perform better than other baselines, however, they too suffers from the same problem especially for lower values of k. The results for $InfQ$ and $SumI$ are shown in Figure 4.4 (c). Our method has higher values for both $InfQ$ and $SumI$ compared to all the baselines. The results reveal that the queries we find using Critical-Queries are closely related to their ancestor queries showcasing their usability.

4.7 Conclusions

In this work, we studied various structural properties of CINs constructed from customer interaction with E-Commerce search engine. Our results show that these networks are significantly distinct from other real world networks. We also observed that the structural properties of CINs, the *Query Reformulation* and the *Composite Click* networks in particular, make them useful for mining query relations. We demonstrated usability of these networks, by leveraging them to cluster queries based on their intents, improve performance of poorly performing queries, and mine critical queries. To cluster queries based on intent, we proposed efficient HubQExpansion algorithm, carefully designed to exploit special structure of the *Query Reformulation* network. Similarly, we modeled user interactions in E-Commerce search system as the `RUN` model, formulated Critical Queries problem and proposed effi-

cient CRITICAL-QUERIES algorithm to identify critical queries. Our extensive experiments demonstrate that the *Query Reformulation* network is useful and our methods are successful in mining query relations.

Chapter 5

Temporal Network Summarization

Given a large time-varying network, can we get a smaller, nearly "equivalent" one? Networks are a common abstraction for many different problems in various domains. Further, propagation-based processes are very useful in modeling multiple situations of interest in real-life such as word-of-mouth viral marketing, epidemics like flu, malware spreading, information diffusion and more. Understanding the propagation process can help in eventually managing and controlling it for our benefit, like designing effective immunization policies. However, the large size of today's networks makes it very hard to analyze them. It is even more challenging considering that such networks evolve over time. Indeed, typical mining algorithms on dynamic networks are very slow.

One way to handle the scale is to get a summary: the idea is that the (smaller) summary can be analyzed instead of the original larger network. While summarization (and related problems) on static networks has been recently studied, surprisingly, getting a smaller representation of a temporal network has not received much attention (see related work). Since the size of temporal networks are orders of magnitude higher than static networks, their succinct representation is important from a data compression viewpoint too. In this paper, we study the problem of 'condensing' a temporal network to get one *smaller in size* which is nearly 'equivalent' with regards to propagation. Such a condensed network can be very helpful in downstream data mining tasks, such as 'sense-making', influence maximization, event detection, immunization and so on. Our contributions are:

- *Problem formulation:* Using spectral characterization of propagation processes, we formulate a novel and general Temporal Network Condensation problem.
- *Efficient Algorithm:* We design careful transformations and reductions to develop an effective, near-linear time algorithm NetCondense which is also easily parallelizable. It merges unimportant node and time-pairs to quickly shrink the network without much loss of information.
- *Extensive Experiments:* Finally, we conduct multiple experiments over large diverse

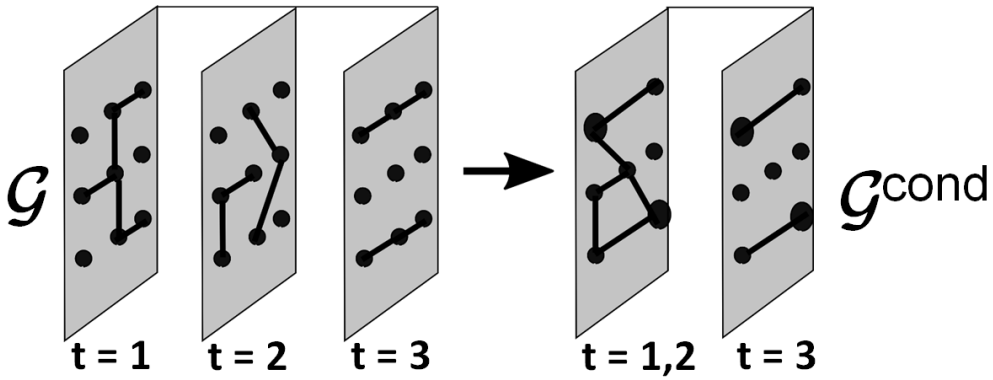


Figure 5.1: Condensing a Temporal Network

real datasets to show correctness, scalability, and utility of our algorithm and condensation in several tasks e.g. we show speed-ups of *48x* in influence maximization and *3.8x* in event detection over dynamic networks.

The rest of the paper is organized in the following way. We present required preliminaries in Section 7.1, followed by problem formulation in Section 10.2. We present our approach and discuss empirical results in Sections 5.3 and 5.4 respectively. We discuss relevant prior works in Section 10.4 and we conclude in Section 11.5.

5.1 Preliminaries

We give some preliminaries next. Notations used and their descriptions are summarized in Table 5.1.

Temporal Networks: We focus on the analysis of dynamic graphs as a series of individual snapshots. In this paper, we consider directed, weighted graphs $G = (V, E, W)$ where V is the set of nodes, E is the set of edges and W is the set of associated edge-weights $w(a, b) \in [0, 1]$. A *temporal network* $\mathcal{G}$ is a sequence of T graphs, i.e., $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$, such that the graph at time-stamp i is $G_i = (V, E_i, W_i)$. Without loss of generality, we assume every G_i in $\mathcal{G}$ has the same node-set V (as otherwise, if we have G_i with different V_i, just define $V = \cup_{i=1}^{T} V_i$). We also assume, in principle there is a path for any node to send information to any other node in $\mathcal{G}$ (ignoring time), as otherwise we can simply decompose. Our ideas can, however, be easily generalized to other types of dynamic graphs.

Propagation models: We primarily base our discussion on two fundamental discrete-time propagation/diffusion models: the SI [21] and IC models [129]. The SI model is a basic epidemiological model where each node can either be in 'Susceptible' or 'Infected' state. In a static graph, at each time-step, a node infected/active with the virus/contagion can

Table 5.1: Summary of symbols and descriptions

Symbol	Description
$\mathcal{G}$	Temporal Network
$\mathcal{G}^{\text{cond}}$	Condensed Temporal Network
G_i, $\mathbf{A}_i$	i$^{\text{th}}$ graph of $\mathcal{G}$ and adjacency matrix
$w_i(a,b)$	Edge-weight between nodes a and b in time-stamp i
V; E	Node-set; Edge-set
α_N	Target fraction for nodes
α_T	Target fraction for time-stamps
T	# of timestamps in Temporal Network
$F_{\mathcal{G}}$	Flattened Network of $\mathcal{G}$
$X_{\mathcal{G}}$	Average Flattened Network of $\mathcal{G}$
$\mathbf{S}_{\mathcal{G}}$	The system matrix of $\mathcal{G}$
$\mathbf{F}_{\mathcal{G}}$; $\mathbf{X}_{\mathcal{G}}$	The adjacency matrix of $F_{\mathcal{G}}$; $X_{\mathcal{G}}$
$\lambda_{\mathbf{S}}$	Largest eigenvalue of $\mathbf{S}_{\mathcal{G}}$
$\lambda_{\mathbf{F}}$; $\lambda_{\mathbf{X}}$	Largest eigenvalue of $\mathbf{F}_{\mathcal{G}}$; $\mathbf{X}_{\mathcal{G}}$
$\mathbf{A}$	Matrix (Bold capital letter)
$\mathbf{u}$, $\mathbf{v}$	Column Vectors (Bold small letter)

infect each of its 'susceptible' (healthy) neighbors independently with probability $w(a,b)$. Once the node is infected, it stays infected. SI is a special case of the general 'flu-like' SIS model, as the 'curing rate' (of recovering from the infected state) δ in SI is 0 while in SIS $\delta \in [0,1)$. In the popular IC (Independent Cascade) model nodes get exactly one chance to infect their healthy neighbors with probability $w(a,b)$; it is a special case of the general 'mumps-like' SIR (Susceptible-Infected-Removed) model, where nodes in 'Removed' state do not get re-infected, with $\delta = 1$.

We consider generalizations of the SI model to temporal networks [184], where an infected node can only infect its susceptible 'current' neighbors (as given by $\mathcal{G}$). Specifically, any node a which is in the infected state at the beginning of time i, tries to infect any of its susceptible neighbor b in G_i with probability $w_i(a,b)$, where $w_i(a,b)$ is the edge-weight for edge (a,b) in G_i. Note that the SI model on static graphs are special cases of those on temporal networks (with all $G_i \in \mathcal{G}$ identical).

5.2 Our Problem Formulation

Real temporal networks are usually gigantic in size. However, their skewed nature [3] (in terms of various distributions like degree, triangles etc.) implies the existence of many nodes/edges which are not important in propagation. Similarly, as changes are typically gradual, most of adjacent time-stamps are not drastically different [97]. There may also be time-periods with sparse connectivities which will not contribute much to propagation. Overall, these observations intuitively imply that it should be possible to get a smaller 'condensed' representation of $\mathcal{G}$ while preserving its diffusive characteristics, which is our task.

It is natural to condense as a result of only local 'merge' operations on node-pairs and time-pairs of $\mathcal{G}$—such that each application of an operation maintains the propagation property and shrinks $\mathcal{G}$. This will also ensure that successive applications of these operations 'summarize' $\mathcal{G}$ in a multi-step hierarchical fashion.

More specifically, merging a node-pair $\{a, b\}$ will merge nodes a and b into a new *super-node* say c, *in all* G_i in $\mathcal{G}$. Merging a time-pair $\{i, j\}$ will merge graphs G_i and G_j to create a new *super-time*, k, and associated graph G_k. However, allowing merge operations on every possible node-pair and time-pair results in loss of interpretability of the result. For example, it is meaningless to merge two nodes who belong to completely different communities or merge times which are five time-stamps apart. Therefore, we have to limit the merge operations in a natural and well-defined way. This also ensures that the resulting summary is useful for downstream applications. We allow a single node-merge only on node pairs $\{a, b\}$ such that $\{a, b\} \in E_i$ for *at least* one G_i, i.e. $\{a, b\}$ is in the unweighted 'union graph' $U_{\mathcal{G}}(V, E_u = \cup_i E_i)$. Similarly, we restrict a single time-merge to only adjacent time-stamps. Note that we can still apply multiple successive merges to merge multiple node-pairs/time-pairs. Our general problem is:

Informal Problem 2 *Given a temporal network* $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$ *with* $G_i = (V, E_i, W_i)$ *and target fractions* $\alpha_N \in (0, 1]$ *and* $\alpha_T \in (0, 1]$*, find a condensed temporal network* $\mathcal{G}^{\text{cond}} = \{G'_1, G'_2, \ldots, G'_{T'}\}$ *with* $G'_i = (V', E'_i, W'_i)$ *by repeatedly applying "local" merge operations on node-pairs and time-pairs such that (a)* $|V'| = (1 - \alpha_N)|V|$*; (b)* $T' = (1 - \alpha_T)T$*; and (c)* $\mathcal{G}^{\text{cond}}$ *approximates* $\mathcal{G}$ *w.r.t. propagation-based properties.*

5.2.1 Formulation framework

Formalizing Informal Problem 2 is challenging as we need to tackle the following two research questions: (Q1) Characterize and quantify the propagation-based property of a temporal network $\mathcal{G}$; (Q2) Define "local" merge operations.

In general, Q1 is difficult as the characterization should be scalable and concise. For Q2, the merges are local operations, and so intuitively they should be defined so that any local

diffusive changes caused by them is minimum. Using Q1 and Q2, we can formulate Informal Problem 2 as an optimization problem where the search space is all possible temporal networks with the desired size and which can be constructed via some sequence of repeated merges from $\mathcal{G}$.

5.2.2 Q1: Propagation-based property

One possible naive answer is to run some diffusion model on $\mathcal{G}$ and $\mathcal{G}^{\text{cond}}$ and see if the propagation is similar; but this is too expensive. Therefore, we want to find a tractable concise metric that can characterize and quantify propagation on a temporal network.

A major metric of interest in propagation on networks is the epidemic threshold which indicates whether the virus/contagion will quickly spread throughout the network (and cause an 'epidemic') or not, regardless of the initial conditions. Past works [94, 183] have studied epidemic thresholds for various epidemic models on static graphs. Recently, [184] show that in context of temporal networks and the SIS model, the threshold depends on the largest eigenvalue λ of the so-called system matrix of $\mathcal{G}$: an epidemic will not happen in $\mathcal{G}$ if $\lambda < 1$. The result in [184] was only for undirected graphs; however it can be easily extended to weighted directed $\mathcal{G}$ with a strongly connected union graph $U_{\mathcal{G}}$ (which just implies that in principle any node can infect any other node via a path, ignoring time; as otherwise we can just examine each connected component separately).

Definition 1 ***System Matrix:*** *For the SI model, the system matrix* $\mathbf{S}_{\mathcal{G}}$ *of a temporal network* $\mathcal{G} = \{G_1, G_2, ..., G_T\}$ *is defined as* $\mathbf{S}_{\mathcal{G}} = \prod_{i=1}^{T}(\mathbf{I} + \mathbf{A}_i)$.

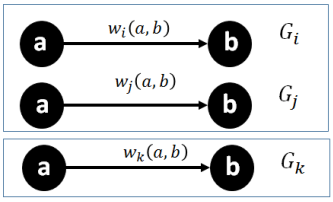
where $\mathbf{A}_t$ is the weighted adjacency matrix of G_t and $\mathbf{I}$ is the identity matrix. For the SI model, the rate of infection is governed by $\lambda_{\mathbf{S}}$, the largest eigenvalue of $\mathbf{S}_{\mathcal{G}}$. Preserving $\lambda_{\mathbf{S}}$ while condensing $\mathcal{G}$ to $\mathcal{G}^{\text{cond}}$ will imply that the rate of virus spreading out in $\mathcal{G}$ and $\mathcal{G}^{\text{cond}}$ will be preserved too. Therefore $\lambda_{\mathbf{S}}$ is a well motivated and meaningful metric to preserve during condensation.

5.2.3 Q2: Merge Definitions

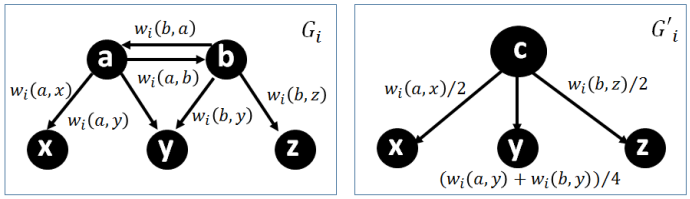
We define two operators: $\mu(\mathcal{G}, i, j)$ merges a time-pair $\{i, j\}$ in $\mathcal{G}$ to a super-time k in $\mathcal{G}^{\text{cond}}$; while $\zeta(\mathcal{G}, a, b)$ merges node-pair $\{a, b\}$ in all $G_i \in \mathcal{G}$ and results in a super-node c in $\mathcal{G}^{\text{cond}}$.

As stated earlier, we want to condense $\mathcal{G}$ by successive applications of μ and ζ. We also want them to preserve local changes in diffusion in the locality of merge operands. At the node level, the level where local merge operations are performed, the diffusion process is best characterized by the probability of infection. Hence, working from first principles, we design these operations to maintain the probabilities of infection before and after the merges in the

'locality of change' without worrying about the system matrix. For $\mu(\mathcal{G}, i, j)$, the 'locality of change' is G_i, G_j and the new G_k. Whereas, for $\zeta(\mathcal{G}, a, b)$, the 'locality of change' is the neighborhood of $\{a, b\}$ in all $G_i \in \mathcal{G}$.



(a) Time merge of a single edge



(b) Node Merge in a single time

Figure 5.2: (a) Example of merge operation on a single edge (a, b) when time-pair $\{i, j\}$ is merged to form super-time k. (b) Example of node-pair $\{a, b\}$ being merged in a single time i to form super-node c.

Time-pair Merge: Consider a merge $\mu(\mathcal{G}, i, j)$ between consecutive times i and j. Consider any edge (a, b) in G_i and G_j (note if $(a, b) \notin E_i$, then $w_i(a, b) = 0$) and assume that node a is infected and node b is susceptible in G_i (illustrated in Figure 5.2 (a)). Now, node a can infect node b in i via an edge in G_i, or in j via an edge in G_j. We want to maintain the local effects of propagation via the merged time-stamp G_k. Hence we need to readjust edge-weights in G_k such that it captures the probability a infects b in $\mathcal{G}$ (in i and j).

Lemma 5 *(Infection via i & j) Let* $\Pr(a \to b|G_i, G_j)$ *be the probability that a infects b in* $\mathcal{G}$ *in either time i or j, if it is infected in* G_i*. Then* $\Pr(a \to b|G_i, G_j) \approx [w_i(a, b) + w_j(a, b)]$*, upto a first order approximation.*

Proof 2 *In the SI model, for node a to infect node b in time pair* $\{i, j\}$*, either the infection occurs in* G_i *or in* G_j*, therefore,*

$$P(a \to b|G_k, G_j) = w_i(a, b) + (1 - w_i(a, b))w_j(a, b).$$

We have

$$P(a \to b|G_k, G_j) = w_i(a, b) + w_j(a, b) - w_i(a, b)w_j(a, b)$$

Now, ignoring the lower order terms, we have

$$P(a \to b|G_k, G_j) \approx w_i(a, b) + w_j(a, b).$$

Lemma 5 suggests that the condensed time-stamp k, after merging a time-pair $\{i, j\}$ should be $\mathbf{A}_k = \mathbf{A}_i + \mathbf{A}_j$. However, consider a $\mathcal{G}$ such that all G_i in $\mathcal{G}$ are the same. This is

effectively a static network: hence the time-merges should give the network G_i rather than $T \times G_i$. This discrepancy arises because for any single time-merge, as we reduce 'T' from 2 to 1, to maintain the final spread of the model, we have to increase the infectivity along each edge by a factor of 2 (intuitively speeding up the model [107]). Hence, the condensed network at time k should be $\mathbf{A}_k = \frac{\mathbf{A}_i + \mathbf{A}_j}{2}$ instead; while for the SI model, the rate of infection should be doubled for time k in the system matrix. Motivated by these considerations, we define a time-stamp merge as follows:

Definition 2 ***Time-Pair Merge*** *$\mu(\mathcal{G}, i, j)$. The merge operator $\mu(\mathcal{G}, i, j)$ returns a new time-stamp k with weighted adjacency matrix $\mathbf{A}_k = \frac{\mathbf{A}_i + \mathbf{A}_j}{2}$.*

Node-pair Merge: Similarly, in $\zeta(\mathcal{G}, a, b)$ we need to adjust the weights of the edges to maintain the local effects of diffusion between a and b and their neighbors. Note that when we merge two nodes, we need to merge them in all $G_i \in \mathcal{G}$.

Consider any time i. Suppose we merge $\{a, b\}$ in G_i to form *super-node* c in G'_i (note that $G'_i \in \mathcal{G}^{\text{cond}}$). Consider a node x such that $\{a, b\}$ and $\{a, x\}$ are neighbors in G_i (illustrated in Figure 5.2 (b)). When c is infected in G'_i, it is intuitive to imply that *either* node a or b is infected in G_i uniformly at random. Hence we need to update the edge-weight from c to x in G'_i, such that the new edge-weight is able to reflect the probability that either node a or b infects x in G_i.

Lemma 6 *(Probability of infecting out-neighbors) If either node a or node b is infected in G_i and they are merged to form a super-node c, then the first order approximation of probability of node c infecting its out-neighbors is given by:*

$$\Pr(c \to z | G_i) \approx \begin{cases} \frac{w_i(a,z)}{2} & \forall z \in Nb_i^o(a) \backslash Nb_i^o(b) \\ \frac{w_i(b,z)}{2} & \forall z \in Nb_i^o(b) \backslash Nb_i^o(a) \\ \frac{w_i(a,z) + w_i(b,z)}{4} & \forall z \in Nb_i^o(a) \cap Nb_i^o(b) \end{cases}$$

where, $Nb_i^o(v)$ is the set of out-neighbors of node v in time-stamp i. We can write down the corresponding probability $\Pr(z \to c | G_i)$ (for getting infected by in-neighbors) similarly.

Proof 3 *Note that $Nb_i^o(v)$ is the set of out-neighbors of node v at time-stamp i. When super-node c is infected in $G'_i \in \mathcal{G}^{\text{cond}}$ (the summary network), either node a or node b is infected in the underlying original network i.e, in $G_i \in \mathcal{G}$. Hence, for a node $z \in Nb_i^o(a) \backslash Nb_i^o(b)$, the probability of node c infecting z is,*

$$P(c \to z|G_i) = \frac{P(a \to z|G_i) + P(b \to a|G_{i-1})P(a \to z|G_i)}{2}$$

Hence, if a is infected, it infects z at time i directly. But for b, to infect z at time i, b has to infect a at time $i-1$, and then a infects b at time i. We rewrite the probabilities as defined by the edge-weights,

$$P(c \to z|G_i) = \frac{w_i(a,z) + w_{i-1}(b,z)w_i(a,z)}{2}$$

Ignoring lowering order terms, we can get,

$$P(c \to z|G_i) \approx \frac{w_i(a,z)}{2}$$

Similarly, we can prove other cases.

Motivated by Lemma 6, we define node-pair merge as:

Definition 3 ***Node-Pair merge*** $\zeta(\mathcal{G}, a, b)$. *The merge operator $\zeta(\mathcal{G}, a, b)$ merges a and b to form a new super-node c in all $G_i \in \mathcal{G}$, s.t. $w_i(c,z) = \Pr(c \to z|G_i)$ and $w_i(z,c) = \Pr(z \to c|G_i)$.*

Note: We use a first-order approximation of the infection probabilities in our merge definitions as the higher-order terms introduce non-linearity in the model. This in turn makes it more challenging to use matrix perturbation theory later [222]. As shown by our experiments, keepin just the first-order terms still leads to high quality summaries.

5.2.4 Problem Definition

We can now formally define our problem.

Problem 5 *(*Temporal Network Condensation *Problem (*TNC*)) Given a temporal network $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$ with strongly connected $U_\mathcal{G}$, $\alpha_N \in (0,1]$ and $\alpha_T \in (0,1]$, find a condensed temporal network $\mathcal{G}^{\text{cond}} = \{G'_1, G'_2, \ldots, G'_{T'}\}$ with $G'_i = (V', E'_i, W'_i)$ by repeated applications of $\mu(\mathcal{G}, \cdot, \cdot)$ and $\zeta(\mathcal{G}, \cdot, \cdot)$, such that $|V'| = (1-\alpha_N)|V|$; $T' = (1-\alpha_T)T$; and $\mathcal{G}^{\text{cond}}$ minimizes $|\lambda_{\boldsymbol{S}} - \lambda^{\text{cond}}_{\boldsymbol{S}}|$.*

Problem 5 is likely to be challenging as it is related to immunization problems [270]. In fact, a slight variation of the problem can be easily shown to be NP-complete by reduction form the Maximum Clique problem [125] (see Appendix). Additionally, Problem 5 naturally contains the GCP coarsening problem for a static network [186] as a special case: when $\mathcal{G} = \{G\}$, which itself is challenging.

5.3 Our Proposed Method

The naive algorithm is combinatorial. Even the greedy method which computes the next best merge operands will be $O(\alpha_N \cdot V^6)$, even without time-pair merges. In fact, even computing $\mathbf{S}_\mathcal{G}$ is inherently non-trivial due to matrix multiplications. It does not scale well for large temporal networks because $\mathbf{S}_\mathcal{G}$ gets denser as the number of time-stamps in $\mathcal{G}$ increases. Moreover, since $\mathbf{S}_\mathcal{G}$ is a dense matrix of size $|V|$ by $|V|$, it does not even fit in the main memory for large networks. Even if there was an algorithm for Problem 5 that could bypass computing $\mathbf{S}_\mathcal{G}$, $\lambda_\mathbf{S}$ still has to be computed to measure success. Therefore, even just measuring success for Problem 5, as is, seems hard.

5.3.1 Main idea

To solve the numerical and computational issues, our idea is to find an alternate representation of $\mathcal{G}$ such that the new representation has the same diffusive properties and avoids the issues of $\mathbf{S}_\mathcal{G}$. Then we develop an efficient sub-quadratic algorithm.

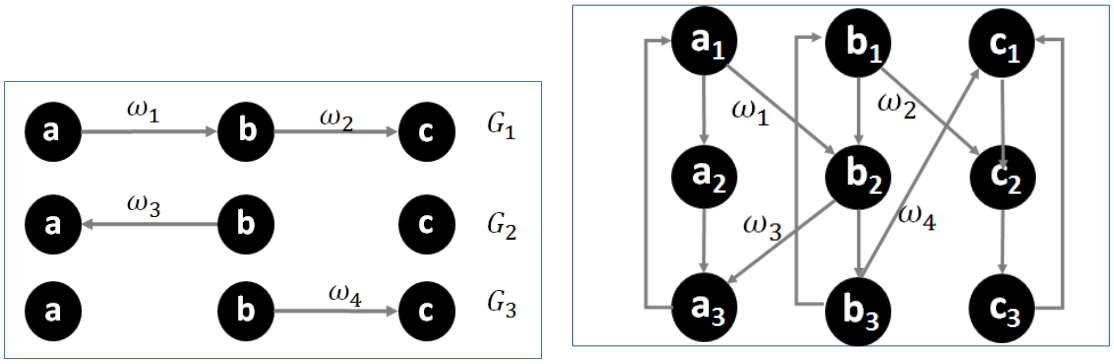
Our main idea is to look for a *static* network that is similar to $\mathcal{G}$ with respect to propagation. We do this in two steps. First we show how to construct a static flattened network $F_\mathcal{G}$, and show that it has similar diffusive properties as $\mathcal{G}$. We also show that eigenvalues of $\mathbf{S}_\mathcal{G}$ and the adjacency matrix $\mathbf{F}_\mathcal{G}$ of $F_\mathcal{G}$ are precisely related. Due to this, computing eigenvalues of $\mathbf{F}_\mathcal{G}$ too is difficult. Then in the second step, we derive a network from $F_\mathcal{G}$ whose largest eigenvalue is easier to compute and related to the largest eigenvalue of $\mathbf{F}_\mathcal{G}$. Using it we propose a new related problem, and solve it efficiently.

5.3.2 Step 1: An Alternate Static View

Our approach for getting a static version is to expand $\mathcal{G}$ and create *layers* of nodes, such that edges in $\mathcal{G}$ are captured by edges *between* the nodes in adjacent layers (see Figure 5.3). We call this the "flattened network" $F_\mathcal{G}$.

Definition 4 ***Flattened network.*** *$F_\mathcal{G}$ for $\mathcal{G}$ is defined as follows:*

- ***Layers****: $F_\mathcal{G}$ consists of $1, ..., T$ layers corresponding to T time-stamps in $\mathcal{G}$.*
- ***Nodes****: Each layer i has $|V|$ nodes (so $F_\mathcal{G}$ has $T|V|$ nodes overall). Node a in the temporal network $\mathcal{G}$ at time i is represented as a_i in layer i of $F_\mathcal{G}$.*
- ***Edges****: At each layer i, each node a_i has a direct edge to $a_{(i+1) \bmod T}$ in layer $(i+1) \bmod T$ with edge-weight 1. And for each time-stamp G_i in the temporal network $\mathcal{G}$, if there is a directed edge (a, b), then in $F_\mathcal{G}$, we add a direct edge from node a_i to node $b_{(i+1) \bmod T}$ with weight $w_i(a, b)$.*



(a) Temporal Network (b) Flattened Network

Figure 5.3: (a) $\mathcal{G}$, and (b) corresponding $F_{\mathcal{G}}$.

For the relationship between $\mathcal{G}$ and $F_{\mathcal{G}}$, consider the SI model running on $\mathcal{G}$ (Figure 5.3 (a)). Say node a is infected in G_1, which also means node a_1 is infected in $F_{\mathcal{G}}$ (Figure 5.3 (b)). Assume a infects b in G_1. So in the beginning of G_2, a and b are infected. Correspondingly in $F_{\mathcal{G}}$ node a_1 infects nodes a_2 and b_2. Now in G_2, no further infection occurs. So the same nodes a and b are infected in G_3. However, in $F_{\mathcal{G}}$ infection occurs between layers 2 and 3, which means a_2 infects a_3 and b_2 infects b_3. Propagation in $F_{\mathcal{G}}$ is different than in $\mathcal{G}$ as each 'time-stamped' node gets exactly one chance to infect others. Note that the propagation model on $F_{\mathcal{G}}$ we just described is the popular IC model. Hence, running the SI model in $\mathcal{G}$ should be "equivalent" to running the *IC model* in $F_{\mathcal{G}}$ in some sense.

We formalize this next. Assume we have the SI model on $\mathcal{G}$ and the IC model on $F_{\mathcal{G}}$ starting from the same node-set of size $I(0)$. Let $I_{SI}^{\mathcal{G}}(t)$ be the *expected* number of infected nodes at the end of time t. Similarly, let $I_{IC}^{F_{\mathcal{G}}}(T)$ be the expected number of infected nodes under the IC model till end of time T in $F_{\mathcal{G}}$. Note that $I_{IC}^{F_{\mathcal{G}}}(0) = I_{SI}^{F_{\mathcal{G}}}(0) = I(0)$. Then:

Lemma 7 *(Equivalence of propagation in $\mathcal{G}$ and $F_{\mathcal{G}}$) We have $\sum_{t=1}^{T} I_{SI}^{\mathcal{G}}(t) = I_{IC}^{F_{\mathcal{G}}}(T)$.*

Proof 4 *First we will show the following:*

$$\sum_{t=0}^{T-1} I_{SI}^{\mathcal{G}}(t) = I_{IC}^{F_{\mathcal{G}}}(T-1) \tag{5.1}$$

We prove this by induction over time-stamp $t = \{0, 1, ..., T-1\}$.

Base Case: *At $t = 0$, since the seed set is the same, the infections in both the model are same. Hence, $I_{SI}^{\mathcal{G}}(0) = I_{IC}^{F_{\mathcal{G}}}(0)$*

Inductive Step: *For inductive step, let the inductive hypothesis be that for time-stamp $0 < k < T-1$, $\sum_{t=0}^{k} I_{SI}^{\mathcal{G}}(t) = I_{IC}^{F_{\mathcal{G}}}(k)$.*

Let $\delta^{\mathcal{G}}_{SI}(k+1)$ be the number of new infection in the SI model in $\mathcal{G}$ at time $k+1$. The total number of infected nodes at time $k+1$ is $I^{\mathcal{G}}_{SI}(k) + \delta^{\mathcal{G}}_{SI}(k+1)$. Similarly, let $\delta^{F_{\mathcal{G}}}_{IC}(k+1)$ be the number of newly infected nodes at the time $k+1$. Since the number of $\delta^{\mathcal{G}}_{SI}(k+1)$ new nodes got infected in SI model in $\mathcal{G}$, the same number of nodes in the layer $k+2$ will get infected in $F_{\mathcal{G}}$. Moreover, all the nodes that are infected in layer $k+1$ at time k in $F_{\mathcal{G}}$ infect corresponding nodes in the next layer. Hence,

$$\delta^{F_{\mathcal{G}}}_{IC}(k+1) = \delta^{\mathcal{G}}_{SI}(k+1) + I^{\mathcal{G}}_{SI}(k)$$

Now, we have

$$\sum_{t=0}^{k+1} I^{\mathcal{G}}_{SI}(t) = \sum_{t=0}^{k} I^{\mathcal{G}}_{SI}(t) + I^{\mathcal{G}}_{SI}(k) + \delta^{\mathcal{G}}_{SI}(k+1)$$

By inductive hypothesis, we get,

$$\begin{aligned}\sum_{t=0}^{k+1} I^{\mathcal{G}}_{SI}(t) &= I^{F}_{IC}(k) + I^{\mathcal{G}}_{SI}(k) + \delta^{\mathcal{G}}_{SI}(k+1) \\ &= I^{F}_{IC}(k) + \delta^{F_{\mathcal{G}}}_{IC}(k+1) = I^{F}_{IC}(k+1)\end{aligned}$$

Now, at the time T, the infection in $\mathcal{G}$ occurs in time-stamp T, however, the infection in $F_{\mathcal{G}}$ occurs between layers T and 1. Recall that nodes are seeded in the layer 1 of $F_{\mathcal{G}}$ for IC model, hence they cannot get infected. Therefore, the difference in the cumulative sum of infection of SI and total infection in IC is $I^{F_{\mathcal{G}}}_{IC}(0)$. Therefore,

$$\sum_{t=0}^{T} I^{\mathcal{G}}_{SI}(t) = I^{F}_{IC}(T) + I^{F_{\mathcal{G}}}_{IC}(0)$$

Since $I^{F}_{IC}(0) = I^{\mathcal{G}}_{SI}(0)$, we have $\sum_{t=1}^{T} I^{\mathcal{G}}_{SI}(t) = I^{F}_{IC}(T)$.

That is, the cumulative expected infections for the SI model on $\mathcal{G}$ is the same as the infections after T for the IC model in $F_{\mathcal{G}}$. This suggests that the largest eigenvalues of $\mathbf{S}_{\mathcal{G}}$ and $\mathbf{F}_{\mathcal{G}}$ are closely related. Actually, we can prove a stronger statement that the spectra of $F_{\mathcal{G}}$ and $\mathcal{G}$ are closely related (Lemma 8).

Lemma 8 *(Eigen-equivalence of $\mathbf{S}_{\mathcal{G}}$ and $\mathbf{F}_{\mathcal{G}}$) We have $(\lambda_{\boldsymbol{F}})^T = \lambda_{\boldsymbol{S}}$. Furthermore, λ is an eigenvalue of $\mathbf{F}_{\mathcal{G}}$, iff λ^T is an eigenvalue of $\mathbf{S}_{\mathcal{G}}$.*

Proof 5 *According to the definition of* $\mathbf{F}_{\mathcal{G}}$*, we have*

$$\mathbf{F}_{\mathcal{G}} = \begin{pmatrix} 0 & \mathbf{I}+\mathbf{A}_1 & 0 & ... & 0 \\ 0 & 0 & \mathbf{I}+\mathbf{A}_2 & ... & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & ... & \mathbf{I}+\mathbf{A}_{T-1} \\ \mathbf{I}+\mathbf{A}_T & 0 & 0 & ... & 0 \end{pmatrix}$$

Here $\mathbf{A}_i$ *is the weighted adjacency matrix of* $G_i \in \mathcal{G}$ *and* $\mathbf{I}$ *is the identity matrix. Both have size* $|V| \times |V|$*. Now any eigenvalue* λ *and corresponding eigenvector* $\mathbf{x}$ *of* $\mathbf{F}_{\mathcal{G}}$ *satisfies the equation* $\mathbf{F}_{\mathcal{G}}\mathbf{x} = \lambda\mathbf{x}$*. We can actually decompose* $\mathbf{x}$ *as* $\mathbf{x} = [\mathbf{x}_1^{'}, \mathbf{x}_2^{'}, \dots \mathbf{x}_T^{'}]^{'}$*, where each* $\mathbf{x}_i$ *is a vector of size* $|V| \times 1$*. Hence, we get the following:*

$$\mathbf{F}_{\mathcal{G}} \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{T-1} \\ \mathbf{x}_T \end{pmatrix} = \lambda \begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{T-1} \\ \mathbf{x}_T \end{pmatrix}$$

From the above equation, we can get

$$\begin{aligned} (\mathbf{I}+\mathbf{A}_1)\mathbf{x}_2 &= \lambda\mathbf{x}_1, \\ (\mathbf{I}+\mathbf{A}_2)\mathbf{x}_3 &= \lambda\mathbf{x}_2, \\ &\vdots \\ (\mathbf{I}+\mathbf{A}_T\mathbf{x}_1) &= \lambda\mathbf{x}_T \end{aligned}$$

Multiplying the equations together,

$$\left[\prod_{i=1}^{T}(I+\mathbf{A}_i)\right]\mathbf{x}_1 = \lambda^T \cdot \mathbf{x}_1$$

Finally,

$$\mathbf{S}_{\mathcal{G}} \cdot \mathbf{x}_1 = \lambda^T \cdot \mathbf{x}_1$$

Hence λ^T *is the eigenvalue of* $\mathbf{S}_{\mathcal{G}}$ *if* λ *is the eigenvalue of* $\mathbf{F}_{\mathcal{G}}$*. Same argument in reverse proves the converse.*

Now, since $U_{\mathcal{G}}$ *is strongly-connected, we have* $|\lambda_{\boldsymbol{F}}| \geq |\lambda|$*, for any* λ *that is eigenvalue of* $\mathbf{F}_{\mathcal{G}}$*. And we also have, if* $|x| > |y|$ *then* $|x^k| > |y^k|$ *for any* $k > 1$*. Therefore there are not any* λ *such that* $|\lambda^T| > |\lambda_{\boldsymbol{F}}^T|$*. So,* λ_F^T *has to be the principal eigenvalue of* $S_{\mathcal{G}}$*.*

Lemma 8 implies that preserving $\lambda_{\mathbf{S}}$ in $\mathcal{G}$ is equivalent to preserving $\lambda_{\mathbf{F}}$ in $F_{\mathcal{G}}$. Therefore, Problem 5 can be re-written in terms of $\lambda_{\mathbf{F}}$ (of a static network) instead of $\lambda_{\mathbf{S}}$ (of a temporal one).

5.3.3 Step 2: A Well Conditioned Network

However $\lambda_{\mathbf{F}}$ is problematic too. The difficulty in computing $\lambda_{\mathbf{F}}$ arises because $\mathbf{F}_{\mathcal{G}}$ is ill-conditioned. Note that $\mathbf{F}_{\mathcal{G}}$ is a very sparse directed network. The sparsity causes the smallest eigenvalue to be very small. Hence, the condition number[68], defined as the ratio of the largest eigenvalue to the smallest eigenvalue in absolute value is high, implying that the matrix is ill-conditioned. The ill-conditioned matrices are unstable for numerical operations. Therefore modern packages take many iterations and the result may be imprecise. Intuitively, it is easy to understand that computing $\lambda_{\mathbf{F}}$ is difficult: as if it were not, computing $\lambda_{\mathbf{S}}$ itself would have been easy (just compute $\lambda_{\mathbf{F}}$ and raise it to the T-th power).

So we create a new static network that has a close relation with $F_{\mathcal{G}}$ and whose adjacency matrix is well-conditioned. To this end, we look at the *average* flattened network, $X_{\mathcal{G}}$, whose adjacency matrix is defined as $\mathbf{X}_{\mathcal{G}} = \frac{\mathbf{F}_{\mathcal{G}}+\mathbf{F}_{\mathcal{G}}{}'}{2}$, where $\mathbf{F}_{\mathcal{G}}{}'$ is the transpose of $\mathbf{F}_{\mathcal{G}}$. It is easy to see that trace of $\mathbf{X}_{\mathcal{G}}$ and $\mathbf{F}_{\mathcal{G}}$ are equal, which means that the sum of eigenvalues of $\mathbf{X}_{\mathcal{G}}$ and $\mathbf{F}_{\mathcal{G}}$ are equal. Moreover, we have the following:

Lemma 9 *(Eigenvalue relationship of $\mathbf{F}_{\mathcal{G}}$ and $\mathbf{X}_{\mathcal{G}}$) The largest eigenvalue of $\mathbf{F}_{\mathcal{G}}$, $\lambda_{\boldsymbol{F}}$, and the largest eigenvalue of $\mathbf{X}_{\mathcal{G}}$, $\lambda_{\boldsymbol{X}}$, are related as $\lambda_{\boldsymbol{F}} \leq \lambda_{\boldsymbol{X}}$.*

Proof 6 *First, according to the definition, $\mathbf{X}_{\mathcal{G}} = \frac{\mathbf{F}_{\mathcal{G}}+\mathbf{F}_{\mathcal{G}}{}'}{2}$. Let $\lambda(\mathbf{F}_{\mathcal{G}})$ be the spectrum of $\mathbf{F}_{\mathcal{G}}$ and $\lambda(\mathbf{X}_{\mathcal{G}})$ be spectrum of $\mathbf{X}_{\mathcal{G}}$. Let $\lambda_{\boldsymbol{X}}$ be the largest eigenvalue of $\mathbf{X}_{\mathcal{G}}$. Function* $\mathrm{Re}(c)$ *returns the real part of c.*

Now, $\lambda(\mathbf{F}_{\mathcal{G}})$ and $\lambda(\mathbf{X}_{\mathcal{G}})$ are related by the majorization relation [262]. i.e., $\mathrm{Re}(\lambda(\mathbf{F}_{\mathcal{G}})) \prec \lambda(\mathbf{X}_{\mathcal{G}})$*, which implies that any eigenvalue of $\mathbf{F}_{\mathcal{G}}$, $\lambda \in \lambda(\mathbf{F}_{\mathcal{G}})$, satisfies* $\mathrm{Re}(\lambda) \leq \lambda_{\boldsymbol{X}}$.

Since the union graph $U_{\mathcal{G}}$ is strongly connected, $F_{\mathcal{G}}$ is strongly connected. Hence, by Perron Frobenius theorem [95], the largest eigenvalue of $\mathbf{F}_{\mathcal{G}}$, $\lambda_{\boldsymbol{F}}$, is real and positive. Therefore, $\lambda_{\boldsymbol{F}} \leq \lambda_{\boldsymbol{X}}$.

Note that if $\lambda_{\mathbf{X}} < 1$, then $\lambda_{\mathbf{F}} < 1$. Moreover, if $\lambda_{\mathbf{F}} < 1$ then $\lambda_{\mathbf{S}} < 1$. Hence if there is no epidemic in $X_{\mathcal{G}}$, then there is no epidemic in $F_{\mathcal{G}}$ as well, which implies that the rate of spread in $\mathcal{G}$ is low. Hence, $X_{\mathcal{G}}$ is a good proxy static network for $F_{\mathcal{G}}$ and $\mathcal{G}$ and $\lambda_{\mathbf{X}}$ is a well-motivated quantity to preserve. Also we need only weak-connectedness of $U_{\mathcal{G}}$ for $\lambda_{\mathbf{X}}$ (and corresponding eigenvectors) to be real and positive (by the Perron-Frobenius theorem). Furthermore, $\mathbf{X}_{\mathcal{G}}$ is free of the problems faced by $\mathbf{F}_{\mathcal{G}}$ and $\mathbf{S}_{\mathcal{G}}$. Since $\mathbf{X}_{\mathcal{G}}$ is symmetric and denser than $\mathbf{F}_{\mathcal{G}}$, it is well-conditioned and more stable for numerical operations. Hence, its eigenvalue can be efficiently computed.

New problem: Considering all of the above, we re-formulate Problem 5 in terms of $\lambda_{\mathbf{X}}$. Since $\mathcal{G}$ and $X_{\mathcal{G}}$ are closely related networks, the merge definitions on $X_{\mathcal{G}}$ can be easily extended from those on $\mathcal{G}$.

Note that edges in one time-stamp of $\mathcal{G}$ are represented between *two* layers in $X_{\mathcal{G}}$ and edges in two consecutive time-stamps in $\mathcal{G}$ are represented in *three* consecutive layers in $X_{\mathcal{G}}$. Hence, merging a time-pair in $\mathcal{G}$ corresponds to merging three layers of $X_{\mathcal{G}}$.

A notable difference in $\mu(\mathcal{G}, ., .)$ and $\mu(X_{\mathcal{G}}, ., .)$ arises due to the difference in propagation models; we have the SI model in $\mathcal{G}$ whereas we the IC model in $X_{\mathcal{G}}$. Since a node gets only a single chance to infect its neighbors in the IC model, infectivity does not need re-scaling $X_{\mathcal{G}}$. Despite this difference, the merge definitions on $\mathcal{G}$ and $X_{\mathcal{G}}$ remain identical.

Let us assume we are merging time-stamps i and j in $\mathcal{G}$. For this, we need to look at the edges between layers i and j, and j and k, where k is layer following j. Now, merging time-stamps i and j in $\mathcal{G}$ corresponds to merging layers i and j in $X_{\mathcal{G}}$ and updating out-links and in-links in the new layers. Let $w_{i,j}(a, b)$ be the edge weight between any node a in layer i.

Definition 5 ***Time-Pair Merge*** $\mu(X_{\mathcal{G}}, i, j)$. *The merge operator $\mu(X_{\mathcal{G}}, i, j)$ results in a new layer m such that edge weight between any nodes a in layer m and b in layer k, $w_{m,k}(a, b)$ is defined as*

$$w_{m,k}(a, b) = \frac{w_{i,j}(a, b) + w_{j,k}(a, b)}{2}$$

Note for $h = i - 1 \mod T$, $w_{h,i}$ and $w_{h,m}$ are equal, since the time-stamp h in $\mathcal{G}$ does not change. And as $\mathbf{X}_{\mathcal{G}}$ is symmetric $w_{m,k}$ and $w_{k,m}$ are equal. Similarly, we extend node-pair merge definition in $\mathcal{G}$ as follows. As in $\mathcal{G}$, we merge node-pairs in all layers of $X_{\mathcal{G}}$.

Definition 6 ***Node-Pair Merge*** $\zeta(X_{\mathcal{G}}, a, b)$. *Let $Nb^o(v)$ denote the set of out-neighbors of a node v. Let $w_{i,j}(a, b)$ be edge weight from any node a in layer i to any node b at layer j. Then the merge operator $\zeta(X_{\mathcal{G}}, a, b)$ merges node pair a, b to form a super-node c, whose edges to out-neighbors are weighted as*

$$w_{i,j}(c, z) = \begin{cases} \frac{w_{i,j}(a, z)}{2} & \forall z \in Nb^o(a) \backslash Nb^i(b) \\ \frac{w_{i,j}(b, z)}{2} & \forall z \in Nb^o(b) \backslash Nb^i(a) \\ \frac{w_{i,j}(a, z) + w_{i,j}(b, z)}{4} & \forall z \in Nb^o(a) \cap Nb^i(b) \end{cases}$$

Finally, our problem can be re-formulated as following.

Problem 6 *Given $\mathcal{G}$ with weakly connected $U_{\mathcal{G}}$ over V, α_N and α_T, find $\mathcal{G}^{\text{cond}}$ by repeated application of $\mu(X_{\mathcal{G}}, ., .)$ and $\zeta(X_{\mathcal{G}}, ., .)$ such that $|V'| = (1 - \alpha_N)|V|$; $T' = (1 - \alpha_T)T$; and $\mathcal{G}^{\text{cond}}$ minimizes $|\lambda_{\boldsymbol{X}} - \lambda_{\boldsymbol{X}}^{\text{cond}}|$.*

5.3.4 NetCondense

In this section, we propose a fast top-k selection algorithm for Problem 6 called NetCondense, which only takes sub-quadratic time in the size of the input. Again, the obvious approach is combinatorial. Consider a greedy approach using Δ-Score.

Definition 7 ***Δ-Score.*** *$\Delta_{X_\mathcal{G}}(a,b) = |\lambda_{\boldsymbol{X}} - \lambda_{\boldsymbol{X}}^{\text{cond}}|$ where $\lambda_{\boldsymbol{X}}^{\text{cond}}$ is the largest eigenvalue of the new $\mathbf{X}_\mathcal{G}$ after merging a and b (node or time-pair).*

The greedy approach will successively choose those merge operands at each step which have the *lowest* Δ-Score. Doing this naively will lead to an expensive algorithm (due to repeated re-computations of $\lambda_{\mathbf{X}}$ for all possible time/node-pairs). Recall that we limit time-merges to adjacent time-pairs and node-merges to node-pairs with an edge in at least one $G_i \in \mathcal{G}$, i.e. edge in the union of all $G_i \in \mathcal{G}$, called the Union Graph $U_\mathcal{G}$. Now, computing Δ-Score simply for all edges $(a,b) \in U_\mathcal{G}$ is still expensive, as it requires computing eigenvalue of $\mathbf{X}_\mathcal{G}$ for each node-pair. Hence we *estimate* Δ-Score for node/time pairs instead using Matrix Perturbation Theory [222]. Let $\mathbf{v}$ be the eigenvector of $\mathbf{X}_\mathcal{G}$, corresponding to $\lambda_{\mathbf{X}}$. Let $\mathbf{v}(a_i)$ be the 'eigenscore' of node a_i in $\mathbf{X}_\mathcal{G}$. $\mathbf{X}_\mathcal{G}(a_i, b_i)$ is the entry in $\mathbf{X}_\mathcal{G}$ in the row a_i and the column b_i. Now we have the following lemmas.

Lemma 10 *(Δ-Score for time-pair) Let V_i = nodes in Layer i of $X_\mathcal{G}$. Now, for merge $\mu(X_\mathcal{G}, i, j)$ to form k,*

$$\Delta_{X_\mathcal{G}}(i,j) = \frac{-\lambda_{\boldsymbol{X}}(\sum_{i \in V_i, V_j} \mathbf{v}(i)^2) + \sum_{k \in V_k} \mathbf{v}(i)\mathbf{k}^{oT}\mathbf{v} + Y}{\mathbf{v}^T\mathbf{v} - \sum_{i \in V_i, V_j} \mathbf{v}(i)^2}$$

upto a first-order approximation, where $\eta_{(i,j)} = \mathbf{v}(i)\mathbf{v}(j)$, $Y = \sum_{i \in V_i, j \in V_j}(2 \cdot \eta_{(i,j)})\mathbf{X}_\mathcal{G}(i,j)$, and $\mathbf{k}^{oT}\mathbf{v} = \frac{1}{2}(\lambda_{\boldsymbol{X}}\mathbf{v}(i) + \lambda_{\boldsymbol{X}}\mathbf{v}(j) + \mathbf{v}(i) + \mathbf{v}(j))$.

Proof 7 *For convenience, we write $\lambda_{\boldsymbol{X}}$ as λ and $\mathbf{X}_\mathcal{G}$ as $\mathbf{X}$. Similarly, we write $\mathbf{v}(x_i)$ as $\mathbf{v}_i$. Now, according to the matrix perturbation theory, we have*

$$\Delta\lambda = \frac{\mathbf{v}^T\Delta\mathbf{X}\mathbf{v} + \mathbf{v}^T\Delta\mathbf{X}\Delta\mathbf{v}}{\mathbf{v}^T\mathbf{v} + \mathbf{v}^T\Delta\mathbf{v}} \tag{5.2}$$

When we merge a time-pair in $\mathbf{X}$, we essentially merge blocks corresponding to the time-stamps i and j in $\mathbf{X}$ to create new blocks corresponding to time-stamp k. Since we want to maintain the size of the matrix as the algorithm proceeds, we place layer k in layer i's place and set rows and columns of $\mathbf{X}_\mathcal{G}$ corresponding to layer j to be zero. Therefore, the change in $\mathbf{X}$ can be written as

$$\Delta \mathbf{X} = \sum_{i \in V_i, V_j} -(\mathbf{i}^i \mathbf{e}_i^T + \mathbf{e}_i \mathbf{i}^{oT}) + \sum_{k \in V_k} -(\mathbf{k}^i \mathbf{e}_k^T + \mathbf{e}_k \mathbf{k}^{oT}) \tag{5.3}$$

where $\mathbf{e}_a$ *is a column vector with* 1 *at position* a *and* 0 *elsewhere, and* $\mathbf{k}^i$ *and* $\mathbf{k}^{oT}$ *are* k*-th column and row vectors of* $\mathbf{X}$ *respectively. Similarly, the change in the right eigenvector can be written as:*

$$\Delta \mathbf{v} = \sum_{i \in V_i, V_j} -(\mathbf{v}_i \mathbf{e}_i) + \delta \tag{5.4}$$

As δ *is very small, we can ignore it. Note that* $\mathbf{v}^T \mathbf{e}_i = \mathbf{v}_i$*,* $\mathbf{v}^T \mathbf{i}^i = \lambda \mathbf{v}_i$ *and* $\mathbf{i}^{oT} \mathbf{v} = \lambda \mathbf{v}_i$*. Now, we can compute Eqn. 5.2 as follows:*

$$\mathbf{v}^T \Delta \mathbf{X} \mathbf{v} = \sum_{i \in V_i, V_j} -(\mathbf{v}_i \mathbf{v}^T \mathbf{i}^i + \mathbf{v}_i \mathbf{i}^{oT} \mathbf{v}) + \sum_{k \in V_k} (\mathbf{v}_i \mathbf{v}^T \mathbf{k}^i + \mathbf{v}_i \mathbf{k}^{oT} \mathbf{v}) \tag{5.5}$$

Further simplifying,

$$\mathbf{v}^T \Delta \mathbf{X} \mathbf{v} = \sum_{i \in V_i, V_j} -(2\lambda \mathbf{v}_i^2) + \sum_{k \in V_k} (\mathbf{v}_i \mathbf{v}^T \mathbf{k}^i + \mathbf{v}_i \mathbf{k}^{oT} \mathbf{v}) \tag{5.6}$$

And we have

$$\mathbf{v}^T \Delta \mathbf{v} = \mathbf{v} \sum_{i \in V_i, V_j} (-\mathbf{v}_i \mathbf{e}_i) = - \sum_{i \in V_i, V_j} \mathbf{v}_i^2 \tag{5.7}$$

Similarly,

$$\begin{aligned} \mathbf{v}^T \Delta \mathbf{X} \Delta \mathbf{v} = \\ \mathbf{v}^T [\sum_{i \in V_i, V_j} -(\mathbf{i}^i \mathbf{e}_i^T + \mathbf{e}_i \mathbf{i}^{oT}) + \sum_{k \in V_k} -(\mathbf{k}^i \mathbf{e}_k^T + \mathbf{e}_k \mathbf{k}^{oT})] \\ [\sum_{i \in V_i, V_j} -(\mathbf{v}_i \mathbf{e}_i)] \end{aligned} \tag{5.8}$$

Here we notice that there are edges between two layers in $X_{\mathcal{G}}$*, only if they are adjacent. Moreover, the edges are in both directions between the layers. Hence,*

$$\begin{aligned} \mathbf{v}^T \Delta \mathbf{X} \Delta \mathbf{v} = \sum_{i \in V_i, V_j} \lambda \mathbf{v}_i \mathbf{v}_j + \sum_{i \in V_i, j \in V_j} (\mathbf{v}_i \mathbf{v}_j + \mathbf{v}_j \mathbf{v}_i) \mathbf{X}(i,j) \\ - \sum_{k \in V_k} \mathbf{v}_i \mathbf{v}^T \mathbf{k}^i - \sum_{k \in V_k, j \in V_j} \mathbf{v}_i \mathbf{k}^{oT} \mathbf{v}_j \mathbf{e}_j. \end{aligned} \tag{5.9}$$

Since self loops have no impact on diffusion, we can write $\sum_{k\in V_k, j\in V_j} \mathbf{v}_i \mathbf{k}^{oT} \mathbf{v}_j \mathbf{e}_j = 0$. *Hence,*

$$\mathbf{v}^T \Delta\mathbf{X}\Delta\mathbf{v} = \sum_{i\in V_i, V_j} \lambda \mathbf{v}_i \mathbf{v}_j + \sum_{i\in V_i, j\in V_j} (2\mathbf{v}_i \mathbf{v}_j \mathbf{X}(i,j) - \sum_{k\in V_k} \mathbf{v}_i \mathbf{v}^T \mathbf{k}^i \quad (5.10)$$

Putting together, we have

$$\Delta\lambda = \frac{-\lambda \sum_{i\in V_i, V_j} \mathbf{v}_i^2 + \sum_{i\in V_i, j\in V_j} 2\mathbf{v}_i \mathbf{v}_j)\mathbf{X}(i,j) + \sum_{k\in V_k} \mathbf{v}_i \mathbf{k}^{oT} \mathbf{v}}{\mathbf{v}^T\mathbf{v} - \sum_{i\in V_i, V_j} \mathbf{v}_i^2} \quad (5.11)$$

Note that we merge the same node in different layers of $X_{\mathcal{G}}$ *corresponding to different time-stamps in* $\mathcal{G}$. *Now, Let* i *be a node in* t_i *and* j *the same node in* t_j, *and we merge them to get new node* k *in* t_k. *Notice that* i *and* j *cannot have common neighbors. Let* $Nb^o(v)$ *be the set of out-neighbors of node* v. *For brevity, let* $I = Nb^o(i)$ *and* $J = Nb^o(j)$. *We have the following,*

$$\begin{aligned} \mathbf{k}^{oT}\mathbf{v} &= \sum_{y\in I} \mathbf{v}_y \mathbf{k}_y^{oT} + \sum_{z\in J} \mathbf{v}_z \mathbf{k}_z^{oT} \quad (5.12) \\ &= \sum_{y\in I} \mathbf{v}_y \frac{1}{2} \mathbf{i}_y^{oT} + \sum_{z\in J} \mathbf{v}_z \frac{1}{2} \mathbf{j}_z^{oT} \end{aligned}$$

Using the definition of $\mathbf{c}^{oT}$ *from Time-Merge Definition,*

$$\mathbf{c}^{oT}\mathbf{v} = \sum_{y\in A} \mathbf{v}_y \frac{1}{2} \mathbf{a}_y^{oT} + \sum_{z\in B} \mathbf{v}_z \frac{1}{2} \mathbf{b}_z^{oT} \quad (5.13)$$

,

Now, let w *be the edge-weight between* i *and* j, $\lambda\mathbf{v}_i = \sum_{y\in I} \mathbf{v}_y \mathbf{i}_y^{oT} - \mathbf{v}_j w$ *therefore,* $\sum_{y\in I} \mathbf{v}_y \mathbf{i}_y^{oT} = \lambda\mathbf{v}_i + \mathbf{v}_j w$

Similarly, $\sum_{z\in B} \mathbf{v}_z \mathbf{j}_y^{oT} = \lambda\mathbf{v}_j + \mathbf{v}_i w$. *By construction, we have* $w = 1$ *in* $X_{\mathcal{G}}$. *Hence,*

$$\mathbf{k}^{oT}\mathbf{v} = \frac{1}{2}(\lambda_{\boldsymbol{X}}\mathbf{v}_i + \lambda_{\boldsymbol{X}}\mathbf{v}_j + \mathbf{v}_i + \mathbf{v}_j). \quad (5.14)$$

Lemma 11 *(*Δ*-Score for node-pair) Let* $V_a = \{a_1, a_2, \ldots, a_T\} \in X_{\mathcal{G}}$ *corresponding to node* a *in* $\mathcal{G}$. *For merge* $\zeta(X_{\mathcal{G}}, a, b)$ *to form* c,

$$\Delta_{X_{\mathcal{G}}}(a,b) = \frac{-\lambda_{\boldsymbol{X}}(\sum_{a\in V_a,V_b}\mathbf{v}(a)^2) + \sum_{c\in V_c}\mathbf{v}(a)\mathbf{c}^{oT}\mathbf{v} + Y}{\mathbf{v}^T\mathbf{v} - \sum_{a\in V_a,V_b}\mathbf{v}(a)^2}$$

upto a first-order approximation, where $\eta_{(a,b)} = \mathbf{v}(a)\mathbf{v}(b)$, $Y = \sum_{a\in V_a,b\in V_b}(2\eta_{(a,b)})\mathbf{X}_{\mathcal{G}}(a,b)$, *and* $\mathbf{c}^{oT}\mathbf{v} = \frac{1}{2}\lambda_{\boldsymbol{X}}(\mathbf{v}(a)+\mathbf{v}(b))$.

Proof 8 *Following the steps in Lemma 10, we have*

$$\Delta\lambda = \frac{-\lambda\sum_{i\in V_i,V_j}\mathbf{v}_i^2 + \sum_{i\in V_i,j\in V_j}(2\mathbf{v}_i\mathbf{v}_j)\mathbf{X}(i,j) + \sum_{k\in V_k}\mathbf{v}_i\mathbf{k}^{oT}\mathbf{v}}{\mathbf{v}^T\mathbf{v} - \sum_{i\in V_i,V_j}\mathbf{v}_i^2} \tag{5.15}$$

Note that we merge the same node in different layers of $X_{\mathcal{G}}$ *corresponding to different time-stamps in* $\mathcal{G}$. *Now, Let* i *be a node in* t_i *and* j *the same node in* t_j, *and we merge them to get new node* k *in* t_k. *Notice that* i *and* j *cannot have common neighbors. Let* $Nb^o(v)$ *be the set of out-neighbors of node* v. *For brevity, let* $I = Nb^o(i)$ *and* $J = Nb^o(j)$. *We have the following,*

$$\begin{aligned}\mathbf{k}^{oT}\mathbf{v} &= \sum_{y\in I}\mathbf{v}_y\mathbf{k}_y^{oT} + \sum_{z\in J}\mathbf{v}_z\mathbf{k}_z^{oT} \\ &= \sum_{y\in I}\mathbf{v}_y\frac{1}{2}\mathbf{i}_y^{oT} + \sum_{z\in J}\mathbf{v}_z\frac{1}{2}\mathbf{j}_z^{oT}\end{aligned} \tag{5.16}$$

Now, let w *be the edge-weight between* i *and* j, $\lambda\mathbf{v}_i = \sum_{y\in I}\mathbf{v}_y\mathbf{i}_y^{oT} - \mathbf{v}_j w$ *therefore,* $\sum_{y\in I}\mathbf{v}_y\mathbf{i}_y^{oT} = \lambda\mathbf{v}_i + \mathbf{v}_j w$.

Similarly, $\sum_{z\in B}\mathbf{v}_z\mathbf{j}_y^{oT} = \lambda\mathbf{v}_j + \mathbf{v}_i w$. *By construction, we have* $w = 1$ *in* $X_{\mathcal{G}}$. *Hence,*

$$\mathbf{k}^{oT}\mathbf{v} = \frac{1}{2}(\lambda_{\boldsymbol{X}}\mathbf{v}_i + \lambda_{\boldsymbol{X}}\mathbf{v}_j + \mathbf{v}_i + \mathbf{v}_j). \tag{5.17}$$

Lemma 12 NetCondense *has sub-quadratic time-complexity of* $O(TE_u + E\log E + \alpha_N\theta TV + \alpha_T E)$, *where* θ *is the maximum degree in any* $G_i \in G$ *and linear space-complexity of* $O(E+TV)$.

Proof 9 *Line 1 in* NetCondense *takes* $O(E)$ *time. To calculate the largest eigenvalue and corresponding eigenvector of* $\mathbf{X}_{\mathcal{G}}$ *using the Lanczos algorithm takes* $O(E)$ *time [238]. It takes* $O(TV+E)$ *time for Lines 2 and 3. It takes* $O(T)$ *time to calculate score for each node pair. Therefore, Lines 4 and 5 take* $O(TE_u)$. *Line 6 takes* $O(T\log T)$ *to sort* Δ*-Score for time-pairs and* $O(E\log E)$ *to sort* Δ*-Score for node-pairs in the worst case. Now, for Lines 8*

to 10, merging node-pairs require us to look at neighbors of nodes being merged at each time-stamp. Hence, it takes $O(T\theta)$ time and we require $\alpha_N V$ merges. Therefore, time-complexity for all node-merge is $O(\alpha_N \theta TV)$. Similarly, it takes $O(\alpha_T E)$ time for all time-merges.

Therefore, the time-complexity of NetCondense *is $O(TE_u + E \log E + \alpha_N \theta TV + \alpha_T E)$. Note that the complexity is* sub-quadratic *as $E_u \leq V^2$ (In our large real datasets, we found $E_u << V^2$).*

For NetCondense*, we need $O(E)$ space to store $X_{\mathcal{G}}$ and $\mathcal{G}^{\text{cond}}$. We also need $O(E_u)$ and $O(T)$ space to store scores for node-pairs and time-pairs respectively. To store eigenvectors of $\mathbf{X}_{\mathcal{G}}$, we require $O(TV)$ space. Therefore, total space-complexity of* NetCondense *is $O(E + TV)$.*

Lemma 13 *(Space Complexity)* NetCondense *has linear space-complexity of $O(E+TV)$.*

Proof 10 *For* NetCondense*, we need $O(E)$ space to store $X_{\mathcal{G}}$ and $\mathcal{G}^{\text{cond}}$. We also need $O(E_u)$ and $O(T)$ space to store scores for node-pairs and time-pairs respectively. To store eigenvectors of $\mathbf{X}_{\mathcal{G}}$, we require $O(TV)$ space. Therefore, total space-complexity of* NetCondense *is linear $O(E + TV)$.*

Algorithm 3 NetCondense

Require: Temporal graph $\mathcal{G}$, $0 < \alpha_N < 1$, $0 < \alpha_T < 1$
Ensure: Temporal graph $\mathcal{G}^{\text{cond}}(V', E', T')$
1: obtain $\mathbf{X}_{\mathcal{G}}$ using Definition 4.
2: **for** every adjacent time-pairs $\{i, j\}$ **do**
3: Calculate $\Delta_{X_{\mathcal{G}}}(i, j)$ using Lemma 10
4: **for** every node-pair $\{a, b\}$ in $U_{\mathcal{G}}$ **do**
5: Calculate $\Delta_{X_{\mathcal{G}}}(a, b)$ using Lemma 11
6: sort the lists of Δ-Score for time-pairs and node-pairs
7: $\mathcal{G}^{\text{cond}} = \mathcal{G}$
8: **while** $|V'| > \alpha_N \cdot |V|$ or $T' > \alpha_T \cdot T$ **do**
9: $(x, y) \leftarrow$ node-pair or time-pair with lowest Δ-Score
10: $\mathcal{G}^{\text{cond}} \leftarrow \mu(\mathcal{G}^{\text{cond}}, x, y)$ or $\zeta(\mathcal{G}^{\text{cond}}, x, y)$
11: return $\mathcal{G}^{\text{cond}}$

Parallelizability: We can easily parallelize NetCondense: once the eigenvector of $\mathbf{X}_{\mathcal{G}}$ is computed, Δ-Score for node-pairs and time-pairs (loops in Lines 3 and 5 in Algorithm 3) can be computed independent of each other in parallel. Similarly, μ and ζ operators (in Line 11) are also parallelizable.

5.4 Experiments

5.4.1 Experimental Setup

We briefly describe our set-up next. All experiments are conducted using a 4 Xeon E7-4850 CPU with 512GB 1066Mhz RAM. Our code is publicly available for academic purposes[1].

Datasets. We run NetCondense on a variety of real datasets (Table 5.4) of varying sizes from different domains such as social-interactions (`WorkPlace`, `School`, `Chess`), co-authorship (`Arxiv`, `DBLP`) and communication (`Enron`, `Wikipedia`, `WikiTalk`). They include weighted and both directed and undirected networks. Edge-weights are normalized to the range $[0, 1]$.

`WorkPlace`, and `School` are contact networks publicly available from SocioPatterns[2]. In both datasets, edges indicate that two people were in proximity in the given time-stamp. Weights represent the total time of interaction in each day.

`Enron` is a publicly available dataset[3]. It contains edges between core employees of the corporation aggregated over 44 weeks. Weights in `Enron` represent the count of emails.

`Chess` is a network between chess players. Edge-weight represents number of games played in the time-stamp.

`Arxiv` is a co-authorship network in scientific papers present in arXiv's High Energy Physics Phenomenology section. We aggregate this network yearly, where the weights are number of co-authored papers in the given year.

`ProsperLoan` is loan network among user of Prosper.com. We aggregate the loan interaction among users to define weights.

`Wikipedia` is an edit network among users of English Wikipedia. The edges represent that two users edited the same page and weights are the count of such events.

`WikiTalk` is a communication network among users of Spanish Wikipedia. Edge between nodes represent that users communicates with each other in the given time-stamp. Weight in this dataset is aggregated count of communication.

`DBLP` is coauthorship network from DBLP bibliography, where two authors have an edge between them if they have co-authored a paper in the given year. We define the weights for co-authorship network as the number of co-authored papers in the given year.

Baselines. Though there are no direct competitors, we adapt multiple methods to use as baselines.

[1] `http://people.cs.vt.edu/~bijaya/code/NetCondense.zip`
[2] http://www.sociopatterns.org/
[3] https://www.cs.cmu.edu/~/enron/

Table 5.2: Datasets Information.

Dataset	**Weight**	$\|V\|$	$\|E\|$	T
`WorkPlace`	Contact Hrs	92	1.5K	12 Days
`School`	Contact Hrs	182	4.2K	9 Days
`Enron`	# Emails	184	8.4K	44 Months
`Chess`	# Games	7.3K	62.4K	9 Years
`Arxiv`	# Papers	28K	**3.8M**	9 Years
`ProsperLoan`	# Loans	89K	**3.3M**	7 Years
`Wikipedia`	# Pages	118K	**2.1M**	10 Years
`WikiTalk`	# Messages	497K	**2.7M**	12 Years
`DBLP`	# Papers	**1.3M**	**18M**	25 Years

RANDOM: Uniformly randomly choose node-pairs and time-stamps to merge.

TENSOR: Here we pick merge operands based on the centrality given by tensor decomposition. $\mathcal{G}$ can be also seen as a tensor of size $|V| \times |V| \times T$. So we run PARAFAC decomposition [134] on $\mathcal{G}$ and choose the largest component to get three vectors $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ of size $|V|$, $|V|$, and T respectively. We compute pairwise centrality measure for node-pair $\{a, b\}$ as $\mathbf{x}(a) \cdot \mathbf{y}(b)$ and for time-pair $\{i, j\}$ as $\mathbf{z}(i) \cdot \mathbf{z}(j)$ and choose the top-K least central ones.

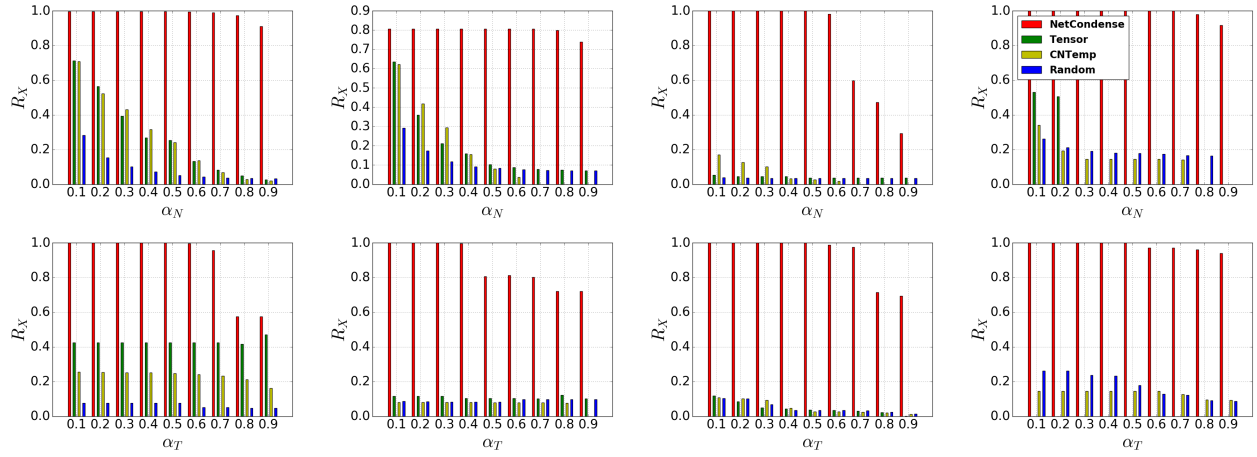
CNTEMP: We run Coarsenet [186] (a summarization method which preserves the diffusive property of a static graph) on $U_{\mathcal{G}}$ and repeat the summary to create $\mathcal{G}^{\text{cond}}$.

In RANDOM and TENSOR, we use our own merge definitions, hence the comparison is inherently unfair.

5.4.2 Perfomance of NetCondense: Effectiveness

We ran all the algorithms to get $\mathcal{G}^{\text{cond}}$ for different values of α_N and α_T, and measure $R_{\mathbf{X}} = \lambda_{\mathbf{X}}^{\text{cond}}/\lambda_{\mathbf{X}}$ to judge performance for Problem 6. See Figure 5.4. NETCONDENSE is able to preserve $\lambda_{\mathbf{X}}$ excellently (upto 80% even when the number of time-stamps and nodes are reduced by 50% and 70% respectively). On the other hand, the baselines perform much worse, and quickly degrade $\lambda_{\mathbf{X}}$. Note that TENSOR does not even finish within *7 days* for `DBLP` for larger α_N. RANDOM and TENSOR perform poorly even though they use the same merge definitions, showcasing the importance of right merges. In case of TENSOR, unexpectedly it tends to merge unimportant nodes with all nodes in their neighborhood even if they are "important"; so it is unable to preserve $\lambda_{\mathbf{X}}$. Finally CNTEMP performs badly as it does not use the full temporal nature of $\mathcal{G}$.

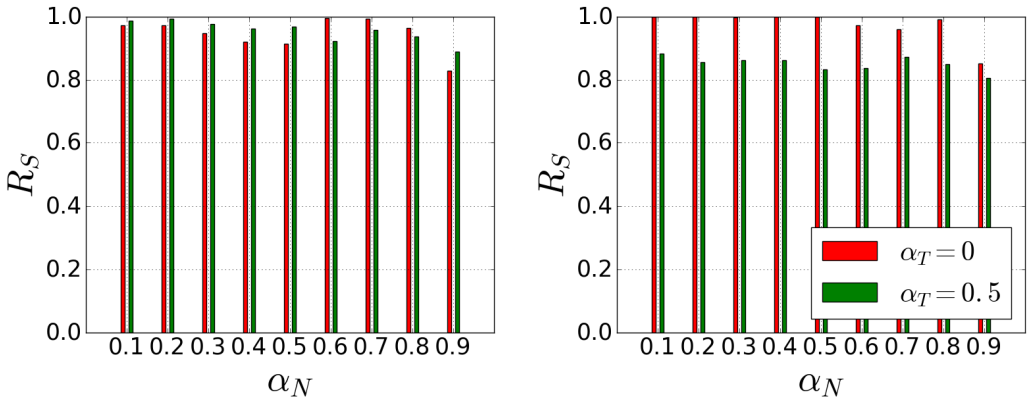
We also compare our performance for Problem 5, against an algorithm specifically designed for it. We use the simple greedy algorithm GREEDYSYS for Problem 5 (as the brute-force



(a) `Arxiv` (b) `ProsperLoan` (c)`WikiTalk` (d) `DBLP`

Figure 5.4: $R_{\mathbf{X}} = \lambda_{\mathbf{X}}^{\text{cond}}/\lambda_{\mathbf{X}}$ vs α_N (top row, $\alpha_T = 0.5$) and vs α_T (bottom row, $\alpha_N = 0.5$).

is too expensive): it greedily picks top node/time merges by actually re-computing $\lambda_{\mathbf{S}}$. We can run GreedySys only for small networks due to the $\mathbf{S}_{\mathcal{G}}$ issues we mentioned before. See Figure 5.5 ($\lambda_{\mathbf{S}}^{\text{M}}$ is $\lambda_{\mathbf{S}}^{\text{cond}}$ obtained from method M). NetCondense does almost as well as GreedySys, due to our careful transformations.



(a) `WorkPlace` (b) `School`

Figure 5.5: Plot of $R_{\mathbf{S}} = \lambda_{\mathbf{S}}^{\text{NetCondense}}/\lambda_{\mathbf{S}}^{\text{GreedySys}}$.

5.4.3 Application 1: Temporal Influence Maximization

In this section, we show how to apply our method to the well-known Influence Maximization problem on a temporal network (TempInfMax) [15]. Given a propagation model, TempInfMax aims to find a seed-set $\mathcal{S} \subseteq V$ at time 0, which maximizes the 'footprint' (expected

number of infected nodes) at time T. Solving it directly on large $\mathcal{G}$ can be very slow. Here we propose to use the much smaller $\mathcal{G}^{\text{cond}}$ as an approximation of $\mathcal{G}$, as it maintains the propagation-based properties well.

Specifically, we propose CondInf (Algorithm 4) to solve the TempInfMax problem on temporal networks. The idea is to get $\mathcal{G}^{\text{cond}}$ from NetCondense, solve TempInfMax problem on $\mathcal{G}^{\text{cond}}$, and map the results back to $\mathcal{G}$. Thanks to our well designed merging scheme that merges nodes with the similar diffusive property together, a simple random mapping is enough. To be specific, let the operator that maps node v from $\mathcal{G}^{\text{cond}}$ to $\mathcal{G}$ be $\zeta^{-1}(v)$. If v is a super-node then $\zeta^{-1}(v)$ returns a node sampled uniformly at random from v.

Algorithm 4 CondInf

Require: Temporal graph $\mathcal{G}$, $0 < \alpha_N < 1$, $0 < \alpha_T < 1$
Ensure: seed set $\mathcal{S}$ of top k seeds
1: $\mathcal{S} = \emptyset$
2: $\mathcal{G}^{\text{cond}} \leftarrow$ NetCondense ($\mathcal{G}$, α_N, α_T)
3: $k'_1, k'_2, ..., k'_S \leftarrow$ Run base TempInfMax on $\mathcal{G}^{\text{cond}}$
4: **for** every k'_i **do**
5: $\quad k_i \leftarrow \zeta^{-1}(k'_i)$; $\mathcal{S} \leftarrow \mathcal{S} \cup \{k_i\}$
6: return $\mathcal{S}$

We use two different base TempInfMax methods: ForwardInfluence [15] for the SI model and Greedy-OT [96] for the PersistentIC model. As our approach is general (our results can be easily extended to other models), and our actual algorithm/output is model-independent, we expect CondInf to perform well for both these methods. To calculate the footprint, we infect nodes in seed set $\mathcal{S}$ at time 0, and run the appropriate model till time T. We use footprints and running time as two measurements. We set $\alpha_T = 0.5$ and $\alpha_N = 0.5$ for all datasets for ForwardInfluence. Similarly, We set $\alpha_T = 0.5$ and $\alpha_N = 0.5$ for `School`, `Enron`, and `Chess`, $\alpha_N = 0.97$ for `Arxiv`, and $\alpha_N = 0.97$ for `Wikipedia` for Greedy-OT (as Greedy-OT is very slow). We show results for ForwardInfluence and Greedy-OT in Table 5.3. The results for Greedy-OT shows that it did not even finish for datasets larger than `Enron`. As we can see, our method performs almost as good as the base method on $\mathcal{G}$, while being *significantly* faster (upto 48 times), showcasing its usefulness.

5.4.4 Application 2: Event Detection

Event detection [188, 18] is an important problem in temporal networks. The problem seeks to identify time points at which there is a significant change in a temporal network. As snapshots in a temporal network $\mathcal{G}$ evolve, with new nodes and edges appearing and existing ones disappearing, it is important to ask if a snapshot of $\mathcal{G}$ at a given time differs significantly from earlier snapshots. Such time points signify intrusion, anomaly, failure, e.t.c depending upon the domain of the network. Formally, the event detection problem is defined as follows:

Table 5.3: Performance of CondInf (CI) with ForwardInfluence (FI) and Greedy-OT (GO) as base methods. σ_m and T_m are the footprint and running time for method m respectively. '-' means the method did not finish.

Dataset	σ_{FI}	σ_{CI}	T_{FI}	T_{CI}
School	130	121	14s	3s
Enron	110	107	18s	3s
Chess	1293	1257	36m	45s
Arxiv	23.7K	23.5K	3.7d	7.5h
Wikipedia	-	26.3K	-	7.1h

Dataset	σ_{GO}	σ_{CI}	T_{GO}	T_{CI}
School	135	128	15m	1.8m
Enron	119	114	9.8m	24s
Chess	-	2267	-	8.6m
Arxiv	-	357	-	2.2h
Wikipedia	-	4591	-	3.2h

Problem 7 *(Event Detection Problem (EDP)) Given a temporal network $\mathcal{G} = \{G_1, G_2, \ldots, G_T\}$, find a list $\mathcal{R}$ of time-stamps t, such that $1 \leq t \leq T$ and G_{t-1} differs significantly from G_t.*

As yet another application of NetCondense, in this section we show that summary $\mathcal{G}^{\text{cond}}$ of a temporal network $\mathcal{G}$ returned by NetCondense can be leveraged to speed up the event detection task. We show that one can actually solve the event detection task on $\mathcal{G}^{\text{cond}}$ instead of $\mathcal{G}$ and still obtain high quality results. Since, NetCondense groups only homogeneous nodes together and preserves important characteristics of the original network $\mathcal{G}$, we hypothesize that running SnapNETS [18] on $\mathcal{G}$ and $\mathcal{G}^{\text{cond}}$ should produce similar results. Moreover, due to the smaller size of $\mathcal{G}^{\text{cond}}$ running SnapNETS on $\mathcal{G}^{\text{cond}}$ is faster than running it on much larger $\mathcal{G}$. In our method, given a temporal network $\mathcal{G}$ and node reduction factor α_N (we set time reduction factor α_T to be 0), we obtain $\mathcal{G}^{\text{cond}}$ and solve the EDP problem on $\mathcal{G}^{\text{cond}}$. Specifically, we propose CondED (Algorithm 5) to solve the event detection problem.

Algorithm 5 CondED

Require: Temporal graph $\mathcal{G}$, $0 < \alpha_N < 1$
Ensure: List $\mathcal{R}$ of time-stamps
1: $\mathcal{G}^{\text{cond}} \leftarrow$ NetCondense ($\mathcal{G}$, α_N, 0)
2: $\mathcal{R} \leftarrow$ Run base Event Detection on $\mathcal{G}^{\text{cond}}$
3: return $\mathcal{R}$

For EDP we use SnapNETS as the base method. In addition to some of the datasets previously used, we run CondED on other datasets which are previously used for event detection [18]. These datasets are described below in detail and the summary is in Table 5.4.

AS Oregon-PA and AS Oregon-MIX are Autonomous Systems peering information network collected from the Oregon router views[4]. IranElection and Higgs are twitter networks, where the nodes are twitter user and edges indicate follower-followee realtionship. More details on these datasets are given in [18].

[4]http://www.topology.eecs.umich.edu/data.html

Table 5.4: Additional Datasets for EDP.

Dataset	$\|V\|$	$\|E\|$	T
Co-Occurence	202	2.8K	31 Days
AS Oregon-PA	633	1.08K	60 Units
AS Oregon-MIX	1899	3261	70 Units
IranElection	126K	**5.5M**	30 Days
Higgs	456K	**14.8M**	7 Days

Table 5.5: Performance of CondED. F1 stands for F1-Score. Speed-up is the ratio of time to run SnapNETS on $\mathcal{G}$ to the time to run SnapNETS on $\mathcal{G}^{\text{cond}}$.

	$\alpha_N = 0.3$		$\alpha_N = 0.7$	
Dataset	F1	Speed-Up	F1	Speed-Up
Co-Occurence	1	1.23	0.18	1.24
School	1	1.05	1	1.56
AS Oregon-PA	1	1.43	1	2.08
AS Oregon-MIX	1	1.22	1	2.83
IranElection	1	1.27	1	3.19
Higgs	0.66	1.17	1	3.79
Arxiv	1	1.09	1	2.27

Co-Occurence is a word co-occurrence network extracted from historical newspapers, published in January 1890, obtained from the library of congress[5]. Nodes in the network are the keywords and edges between two keywords indicate that they co-appear in a sentence in a newspaper published in a particular day. The edge-weights indicate the frequency with which two words co-appear.

To evaluate performance of CondED, we compare the list of time-stamps $\mathcal{R}^{\text{cond}}$ obtained by CondED with the list of time-stamps $\mathcal{R}$ obtained by SnapNETS on the original network $\mathcal{G}$. We treat the time-stamps discovered by SnapNETS as the ground truth and following the methodology in [18], we compute the F-1 score. We repeat the experiment with $\alpha_N = 0.3$ and $\alpha_N = 0.7$. The results are summarized in Table 5.5.

As shown in Table 5.5, CondED has very high F1-score for most datasets even when $\alpha_N = 0.7$. This suggests that the list of time-stamps $\mathcal{R}^{\text{cond}}$ returned by CondED matches the result from SnapNETS very closely. Moreover, the time taken for the base method to run in $\mathcal{G}^{\text{cond}}$ is upto 3.5 times faster than the time it takes to run on $\mathcal{G}$.

However, for Co-Occurence dataset, the F1-score for $\alpha_N = 0.7$ is a mere 0.18, despite having F1-score of 1 for $\alpha_N = 0.3$. Note that Co-Occurence is one of the smallest dataset that we have, hence very high α_N seems to deteriorate the structure of the network, which suggests

[5]http://chroniclingamerica.loc.gov

a different value of α_N is suitable for different networks in the EDP task.

5.4.5 Application 3: Understanding/Exploring Networks

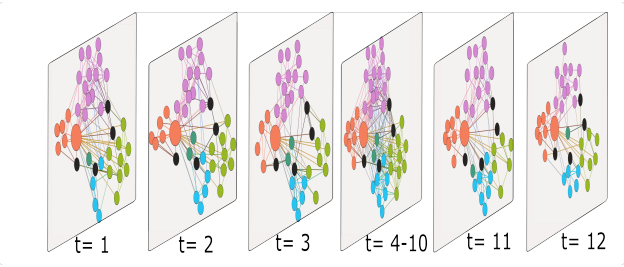


Figure 5.6: Condensed `WorkPlace` ($\alpha_N = 0.6$, $\alpha_T = 0.5$).

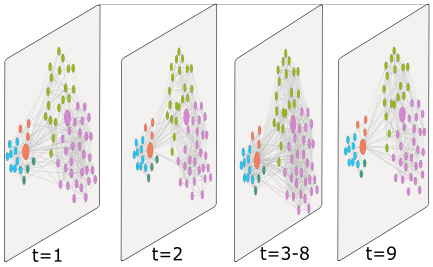


Figure 5.7: Condensed `School` ($\alpha_N = 0.5$ and $\alpha_T = 0.5$).

We can also use NetCondense for 'sense-making' of temporal datasets: it ensures that important nodes and times remain unmerged while super-nodes and super-times form coherent interpretable groups of nodes and time-stamps. This is not the case for the baselines e.g. Tensor merges important nodes, giving us heterogeneous super-nodes lacking interpretability.

`WorkPlace`: It is a social-contact network between employees of a company with five departments, where weights are normalized contact time. It has been used for vaccination studies [97]. In $\mathcal{G}^{\text{cond}}$ (see Figure 5.6), we find a super-node composed mostly of nodes from SRH (orange) and DSE (pink) departments, which were on floor 1 of the building while the rest were on floor 2. We also noticed that the proportion of contact times between individuals of SRH and DSE were high in all the time-stamps. In the same super-node, surprisingly, we find a node from DMCT (green) department on floor 2 who has a high contact with DSE nodes. It turns out s/he was labeled as a "wanderer" in [97].

Unmerged nodes in the $\mathcal{G}^{\text{cond}}$ had high degree in all T. For example, we found nodes 80, 150, 751, and 255 (colored black) remained unmerged even for $\alpha_N = 0.9$ suggesting their

importance. In fact, all these nodes were classified as "Linkers" whose temporal stability is crucial for epidemic spread [97]. The stability of "Linkers" mentioned in [97] suggests that these are important nodes in every time-stamp. The visualization of $\mathcal{G}^{\text{cond}}$ emphasizes that linkers connect consistently to nodes from multiple departments; which is not obvious in the original networks. We also examined the super-times, and found that the days in and around the weekend (where there is little activity) were merged together.

`School`: It is a socio-contact network between high school students from five different sections over several days [91]. We condensed `School` dataset with $\alpha_N = 0.5$ and $\alpha_T = 0.5$. The students in the dataset belong to five sections, students from sections "MP*1" (green) and "MP*2" (pink) were maths majors, and those from "PC" (blue) and "PC*" (orange) were Physics majors, and lastly, students from "PSI" (dark green) were engineering majors [91]. In $\mathcal{G}^{\text{cond}}$ (see Figure 5.7), we find a super-node containing nodes from MP*1 (green) and MP*2 (pink) sections (enlarged pink node) and another super-node (enlarged orange node) with nodes from remaining three sections PC (blue), PC* (orange), and PSI (dark green). Our result is supported by [91], which mentions that the five classes in the dataset can brodaly be divided into two categories of (MP*1 and MP*2) and (PC, PC*, and PSI). The groupings in the super-nodes are intuitive as it is clear from the visualization itself that the dataset can broadly be divided into two components (MP*1 and MP*2) and (PC, PC*, and PSI). We also see that unmerged nodes in each major connect densely every day. These connections are obscured by unimportant nodes and edges in the original network. This dense inter-connection of "important" nodes is obscured by unimportant nodes and edges in the original network. However, visualization of the condensed network emphasizes on these important structures and how they remain stable over time. We also noted that small super-nodes of size 2 or 3, were composed solely of male students, which can be attributed to the gender homophily exhibited only by male students in the dataset [91]. We noticed that weekends were merged together with surrounding days in `School`, while other days remained intact.

`Enron`: They are the email communication networks of employees of the Enron Corporation. In $\mathcal{G}^{\text{cond}}$ ($\alpha_N = 0.8, \alpha_T = 0.5$), we find that unmerged nodes are important nodes such as G. Whalley (President), K. Lay (CEO), and J. Skilling (CEO). Other unmerged nodes included Vice-Presidents and Managing Directors. We also found a star with Chief of Staff S. Kean in the center and important officials such as Whalley, Lay and J. Shankman (President) for six consecutive time-stamps. We also find a clique of various Vice-Presidents, L. Blair (Director), and S. Horton (President) for seven time-stamps in the same period. These structures appear only in consecutive time-stamps leading to when Enron declared bankruptcy. Sudden emergence, stability for over six/seven time-stamps, and sudden disappearance of these structures correctly suggests that a major event occurred during that time. We also note that time-stamps in 2001 were never merged, indicative of important and suspicious behavior.

To investigate the nature of nodes which get merged early, we look at the super-nodes at the early stage of NetCondense, we find two super-nodes with one Vice-President in each

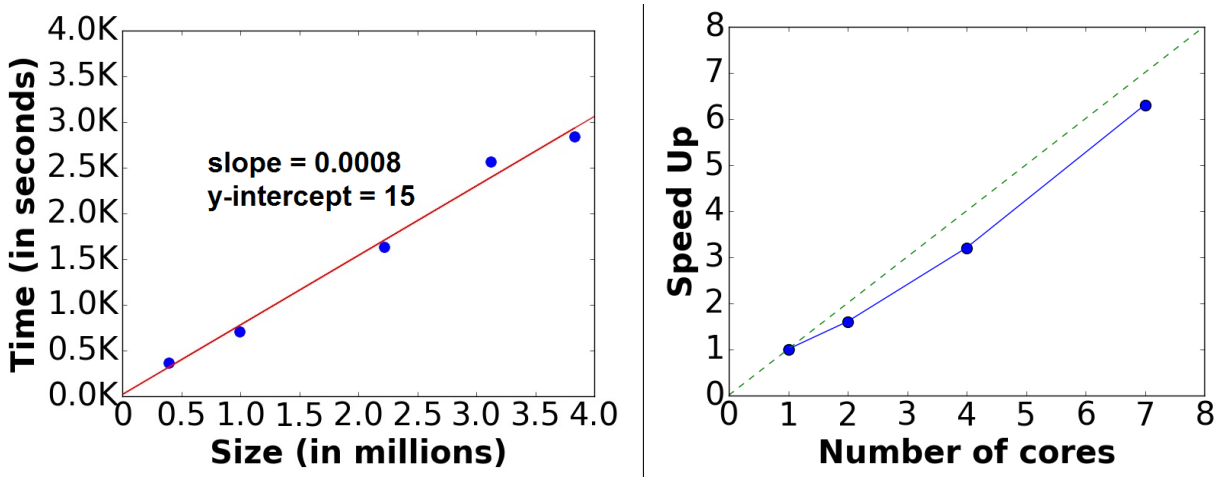
(Note that most other nodes are still unmerged). Both super-nodes were composed mostly of Managers, Traders, and Employees who reported directly or indirectly to the mentioned Vice-Presidents. We also look at unmerged time-stamps and notice that time-stamps in 2001 were never merged. Even though news broke out on October 2001, analysts were suspicious of practices in Enron Corporation from early 2001[6]. Therefore, our time-merges show that the events in early 2001 were important indicative of suspicious activities in Enron Corporation.

`DBLP`: These are co-authorship networks from DBLP-CS bibliography. This is an especially large dataset: hence exploration without any condensation is hard. In $\mathcal{G}^{\text{cond}}$ ($\alpha_N = 0.7, \alpha_T = 0.5$), we found that the unmerged nodes were very well-known researchers such as Philip S. Yu, Christos Faloutsos, Rakesh Aggarwal, and so on, whereas super-nodes grouped researchers who had only few publications. In super-nodes, we found researchers from the same institutions or countries (like Japanese Universities) or same or closely-related research fields (like Data Mining/Visualization). In larger super-nodes, we found that researchers were grouped together based on countries and broader fields. For example, we found separate super-nodes composed of scientists from Swedish, and Japanese Universities only. We also found super-nodes composed of researchers from closely related fields such as Data Mining and Information visualization. Small super nodes typically group researchers who have very few collaborations in the dataset, collaborated very little with the rest of the network and have edges among themselves in few time-stamps. Basically, these are researchers with very few publications. We also found a giant super-node of size $395,000$. An interesting observation is that famous researchers connect very weakly to the giant super-node. For example, Rakesh Aggarwal connects to the giant super-node in only two time-stamps with almost zero edge-weight. Whereas, less known researchers connect to the giant super-node with higher edge-weights. This suggests researchers exhibit homophily in co-authorship patterns, i.e. famous researchers collaborate with other famous researchers and non-famous researchers collaborate with other non-famous researchers. Few super-nodes that famous researchers connect to at different time-stamps, also shows how their research interest has evolved with time. For example, we find that Philip S. Yu connected to a super-node of IBM researchers who primarily worked in Database in early 1994 and to a super-node of data mining researchers in 2000.

5.4.6 Scalability and Parallelizability

Figure 5.8 (a) shows the runtime of NetCondense on the components of increasing size of `Arxiv`. NetCondense has subquadratic time complexity. In practice, it is *near-linear* w.r.t input size. Figure 5.8 (b) shows the *near-linear* run-time speed-up of parallel-NetCondense vs # cores on `Wikipedia`.

[6]http://www.webcitation.org/5tZ26rnac



(a) Scalability (b) Parallelizability

Figure 5.8: (a) Near-linear Scalability w.r.t. size; (b) Near-linear speed up w.r.t number of cores for parallelized implementation.

5.5 Related Work

Mining Dynamic Graphs. Dynamic graphs have gained a lot of interest recently (see [14] for a survey). Many graph mining tasks on static graphs have been introduced to dynamic graphs. For example, Tantipathananandh et.al studied community detection in dynamic graphs [233]. Similarly, link prediction [207], and event detection [188] have also been studied in dynamic graphs. Due to the increasing size, typically it is challenging to perform analysis on temporal networks and summarizing large temporal networks can be very useful.

Propagation. Cascade processes have been widely studied, including in epidemiology [21, 107], information diffusion [129], cyber-security [130] and product marketing [199]. Based on the models, several papers studied propagation related applications, such as propagation of memes [141] and so on. In addition, there are research works that focus on the epidemic threshold of static networks, which determines the conditions under which virus spreads throughout the network [30, 153, 130, 181, 94, 58, 183]. Recently [184] studied the threshold for temporal networks based on the system matrix. Examples of propagation-based optimization problems are influence maximization [129, 96, 15], and immunization [270]. Remotely related work deals with weak and strong ties over time for diffusion [127].

Graph Summarization. Here, we seek to find a compact representation of a large graph by leveraging global and local graph properties like local neighborhood structure [167, 237], node/edge attributes [265], action logs [187], eigenvalue of the adjacency matrix [186], and key subgraphs. It is also related to graph sparsification algorithms [155]. The goal is to either reduce storage and manipulation costs, or simplify structure. Summarizing temporal networks has not seen much work, except recent papers based on bits-storage-compression [144],

or extracting a list of recurrent sub-structures over time [212]. Unlike these, we are the first to focus on hierarchical condensation: using *structural* merges, giving a *smaller propagation-equivalent* temporal network. There has been some work on summarizing temporal graphs, including compression-based Liu et. al. [144] tackled the problem by compressing edge weights at each timestamps. However, their method does not merge either nodes or timestamps. Hence, it cannot get a summary with much less nodes and timestamps. Shah et. al. [212] proposed a temporal graph summarization method (TIME CRUNCH) [212], by extracting representative local structures across timestamp. However, TIME CRUNCH do not work for our problem, as our problem requires the summary to be a compact temporal graph that maintains the diffusion property, while TIME CRUNCH just outputs pieces of subgraphs. To summarize, none of the above papers focused on the temporal graph summarization problem w.r.t. diffusion.

5.6 Discussion and Conclusions

In this paper, we proposed a novel general TEMPORAL NETWORK CONDENSATION Problem using the fundamental so-called 'system matrix' and present an effective, near-linear and parallelizable algorithm NETCONDENSE. We leverage it to dramatically speed-up influence maximization and event detection algorithms on a variety of large temporal networks. We also leverage the summary network given by NETCONDENSE to visualize, explore, and understand multiple networks. As also shown by our experiments, it is useful to note that our method itself is model-agnostic and has wide-applicability, thanks to our carefully chosen metrics which can be easily generalized to other propagation models such as SIS, SIR, and so on.

There are multiple ideas to explore further. Condensing attributed temporal networks, and leveraging NETCONDENSE for other graph tasks such as role discovery, immunization, and link prediction are some examples. We leave these tasks for future works.

Chapter 6

Near-Optimal Network State Inference using Probes

Most network applications assume the network state is static and is known ahead of time. This is not true in practice, and networks are inferred by indirect measurements, e.g., as in the case of the Internet router/AS level graphs, which are constructed using trace-routes, e.g., [88], or biological networks, which are inferred by experimental correlations, e.g., [209]. Further, network elements can fail dynamically, or their state may change with time. For instance, links in the Internet router network or the transportation network can get congested or fail. Reconstructing the network topology dynamically and inferring network states in such settings are fundamental problems. Such problems have been studied as part of the area of "network tomography", especially in communication networks, e.g., [121, 254, 173]. Such networks are not publicly accessible, and indirect probes are the only means of obtaining information; examples of probes include queries of the activity states of selected nodes and end-to-end measurements of delays between selected pairs of nodes. These become very challenging problems, and all prior work in this direction in network tomography has been focused on simple models of *independent* link failures and delays, e.g., with exponentially distributed probabilities [173].

In many settings, such as disaster events in infrastructure networks, however, failures might be spatially correlated, as in [38, 13, 204]. For instance, in the model considered in [13], where authors study vulnerability of networks, the probability that a node j fails decays with the distance from a source s. This motivates the problem of inferring the network states under such spatial correlations, which is the focus of our paper. For example in a toy road network shown in Figure 6.1, given a partial probe of failed nodes (shown in red), can we infer other nodes which have failed as well (shown in blue)?

A closely related topic is the inference of the source of an infection and other missing infections in the case of epidemic spread on networks—these are typically modeled as SI/SIR processes (see [107, 154] for an introduction to epidemic models), where the infection spreads

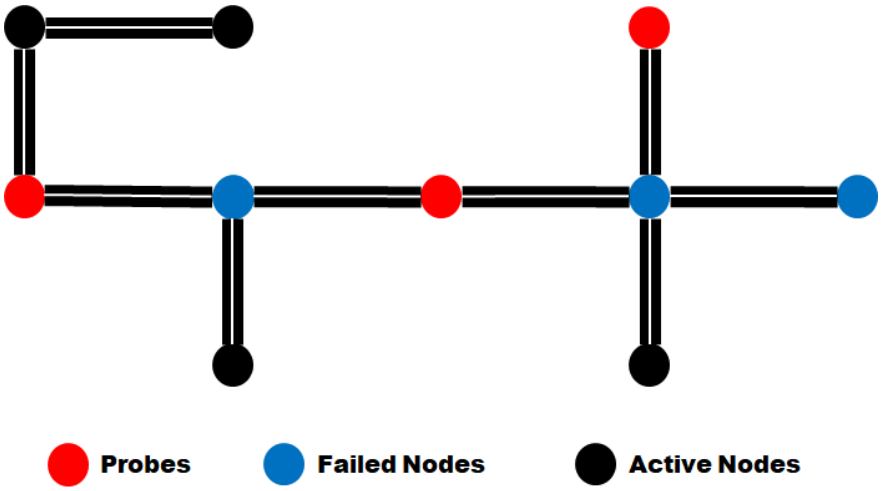


Figure 6.1: An example road network, where multiple intersections have failed. Given a partial probe of failed nodes (red), our method infers other nodes which have failed (blue).

from one node to its neighbors with some probability [185, 225, 211]. There has been some work in missing infection problems in this direction e.g. [185, 225] develop algorithms for the SI process. Intuitively, these link-based methods do not seem to directly work for our problem, due to the difference in propagation process of failures/infections and epidemiological processes. For example, if a node in a fully-connected clique is infected in an epidemic process, all the remaining nodes are highly likely to get infected; which is not necessarily the case in infrastructure networks, say road networks, where failures are more spatially correlated.

Our main contributions are summarized below.

1. We develop a novel robust formulation for the GraphStateInf problem using the Minimum Description Length (MDL) principle [104, 195], which takes correlated failures into account and aims to find the missing failures which best explain the given data. We present GraphMap, an algorithm for inferring the missing failed nodes, given a sample probe of the failures. We prove that the MDL cost of the solution computed by GraphMap is within an additive approximation of the minimum cost MDL solution. Typically, approaches using MDL are based on heuristics and getting bounds is non trivial as MDL cost functions are not convex. To the best of our knowledge, our algorithm is the first to obtain rigorous bounds on the objective value among MDL based approaches for network inference.

2. We evaluate our results on different kinds of synthetic and real datasets, namely, one week's worth of traffic status and incident reports from WAZE for the city of Boston, and electric disturbance events in the power grid. We study the precision, recall and F1-score for GraphMap, compared with a baseline. We observe that our algorithm

is quite effective in inferring unknown/missing failures in the network, and has lower MDL cost than the baseline.

The rest of the paper is organized in the following way: we formulate our problem in Section 6.1 and propose our methods in Section 6.2. In Section 6.3, we describe our datasets in detail and show experimental results. We then present the related work and conclusions in Sections 10.4 and 7.5 respectively.

6.1 Our Problem Formulation

We are given an undirected graph $G(V, E)$ representing an infrastructure network. We assume there is an initial failure at a node, referred to as the seed node, which causes other nodes to fail. Further, a subset $\mathcal{Q} \subseteq I$ of the actual failed nodes are assumed to be known. The objective in the GraphStateInf problem is to infer all the missing failures.

6.1.1 Failure Model

Next, we will discuss the failure model we use to describe the failures in the given network G. This model is motivated by the geographically correlated failure model introduced by Agarwal et al [13], to capture failures in infrastructure networks due to large scale disasters. In such events, there is an initial localized failure, which causes other nodes to fail with some probability that decays with the distance from the source.

Following [13], we assume there is an initial single 'seed' node s and all the failures I in G are caused due to the influence of that seed node. We assume a discrete probability distribution function $p_s : V \to [0, 1]$ that gives the probability of each node $v \in V$ being a seed and conditional failure probability distribution function $F : V \times V \to [0, 1]$ that gives the failure probability of a node $v \in V$ given a seed node s. Note that the $p_s(v)$ is the probability of v being the only seed, i.e, $\sum_{v \in V} p_s(v) = 1$. These probability distributions are precomputed from historically observed failures. We assume that the conditional failure probabilities given by F are independent i.e., for-all $v_1, v_2 \in V$ and $v_1 \neq v_2$,

$$F(v_1 \cap v_2 \mid s) = F(v_1 \mid s)F(v_2 \mid s) \tag{6.1}$$

6.1.2 Probes

Based on our model given above, we assume that some seed failed causing multiple correlated failures across the network G. The final set of true failures is represented by $I \subseteq V$. Further, we are also given a set of failed nodes represented by $\mathcal{Q} \subseteq I$, which we will refer to as probes

in rest of this paper. In reality, the probes $\mathcal{Q}$ are failed nodes that are observed, and this is typically a random process. For ease of modeling, we assume that the probes are sampled uniformly at random from the true failure set I with probability γ.

6.1.3 MDL

We formulate our problem using the Minimum Description Length (MDL) principle [104]. We will use two-part MDL, or the sender-receiver framework. Our goal here is to transmit the given set of probes $\mathcal{Q}$ from sender to receiver by assuming that both of them know the layout of the network G. We do this by identifying the model that best describes the given data in terms of a formal objective or cost function. This cost function consists of two parts:

1. Model cost that signifies the complexity of the selected model that explains the failures in the network; and
2. Data cost that represents the cost of observing the given probe data $\mathcal{Q}$ given the model.

More formally, given a set of models $\mathcal{M}$, MDL identifies the best model M^* as the model that minimizes $\mathcal{L}(M)+\mathcal{L}(\mathcal{D}\,|\,M)$, in which $\mathcal{L}(M)$ is the model-cost (length in bits to describe model M), and $\mathcal{L}(\mathcal{D}\,|\,M)$ is the data-cost (the length in bits to describe the data using M). Note that the data we need to describe in our situation is the probes set $\mathcal{Q}$ (and not the true failures set I). Next we describe the model space and the model and data cost, which we will optimize.

6.1.4 Model Space and Cost

Model Space: The most natural model for our problem would have been $\mathcal{M} = (s, I)$ (the source $s \in V$ and the full failure set $I \subseteq V$), as it directly mimics the generative process of the failure model. However, this model has several disadvantages. Firstly, note that this model space is intuitively 'fragile': small changes in I or the source s can have vastly different costs. Hence due to data sparsity, we expect it would be very hard to learn the true source which generated the failures—indeed, in our experiments, we find that it was not robustly learning the true source. As a result, we also found that the solutions with minimum MDL cost were finding very few missing failed nodes (i.e. $I - Q$), leading to a very low recall. How to design a better model space for our problem? We observe that this model intuitively tries to explain 'more' than what is needed. Note that while our original goal was to map the missing *failures* only, this approach tries to explain the source as well as the set of failures. Hence we adopt a different approach, where we try to marginalize over the seeds, and focus only on the failures. This makes our model space more robust too. This motivates our proposed model, which consists of three components, namely, $\mathcal{M} = (|\mathcal{Q}|, |I|, I)$. In other

words, we send the size of probes $\mathcal{Q}$, the size of true failure set I, and then identify the set itself. After sending the model, we will then identify the actual probes set Q as the data.

Model cost: The MDL model cost, $\mathcal{L}(|\mathcal{Q}|, |I|, I)$ has three components

$$\mathcal{L}(|\mathcal{Q}|, |I|, I) = \mathcal{L}(|\mathcal{Q}|) + \mathcal{L}\big(|I| \;\big|\; |\mathcal{Q}|\big) + \mathcal{L}\big(I \;\big|\; |\mathcal{Q}|, |I|\big).$$

We derive these below. We have $\mathcal{L}(|\mathcal{Q}|) = -\log\Big(Pr(|\mathcal{Q}|)\Big)$, by using the *Shannon-Fano* code to encode $|\mathcal{Q}|$. Similarly we have:

$$\begin{aligned}\mathcal{L}\big(|I| \;\big|\; |\mathcal{Q}|\big) &= -\log\Big(Pr(|I| \;\big|\; |\mathcal{Q}|)\Big) \\ &= -\log\Big(\frac{Pr\big(|\mathcal{Q}| \;\big|\; |I|\big)Pr(|I|)}{Pr(|\mathcal{Q}|)}\Big)\end{aligned} \tag{6.2}$$

From the sampling assumption for $\mathcal{Q}$, we can get:

$$Pr\big(|\mathcal{Q}| \;\big|\; |I|\big) = \binom{|I|}{|\mathcal{Q}|}\gamma^{|\mathcal{Q}|}(1-\gamma)^{|I\setminus\mathcal{Q}|} \tag{6.3}$$

Also observe that:

$$\begin{aligned}\mathcal{L}\big(I \;\big|\; |\mathcal{Q}|, |I|\big) &= -\log\Big(Pr\big(I \;\big|\; |\mathcal{Q}|, |I|\big)\Big) \\ &= -\log\Big(Pr\big(I \;\big|\; |I|\big)\Big) = -\log\Big(\frac{Pr(I)}{Pr(|I|)}\Big)\end{aligned} \tag{6.4}$$

Combining all of the above, the complete model cost is:

$$\begin{aligned}&\mathcal{L}(|\mathcal{Q}|, |I|, I) \\ =&\mathcal{L}(|\mathcal{Q}|) + \mathcal{L}\big(|I| \;\big|\; |\mathcal{Q}|\big) + \mathcal{L}\big(I \;\big|\; |\mathcal{Q}|, |I|\big) \\ =& -\log\Big(Pr(|\mathcal{Q}|)\Big) - \log\Big(\frac{Pr\big(|\mathcal{Q}| \;\big|\; |I|\big)Pr(|I|)}{Pr(|\mathcal{Q}|)}\Big) \\ &- \log\Big(\frac{Pr(I)}{Pr(|I|)}\Big) \\ =& -\log\Big(Pr\big(|\mathcal{Q}| \;\big|\; |I|\big)\Big) - \log\Big(Pr(|I|)\Big) - \log\Big(\frac{Pr(I)}{Pr(|I|)}\Big) \\ =& -\log\Big(\binom{|I|}{|\mathcal{Q}|}\gamma^{|\mathcal{Q}|}(1-\gamma)^{|I\setminus\mathcal{Q}|}\Big) \\ &- \log\Big(\sum_{s\in V} Pr(I \mid s)p(s)\Big) \\ =& -\log\binom{|I|}{|\mathcal{Q}|} - |\mathcal{Q}|\log(\gamma) - (|I| - |\mathcal{Q}|)\log(1-\gamma) \\ &- \log\Big(\sum_{s\in V} p_s(s)\prod_{v\in I}F(v \mid s)\prod_{v'\notin I}\Big(1 - F(v' \mid s)\Big)\Big)\end{aligned} \tag{6.5}$$

6.1.5 Data Cost

Now, we need to describe the given input probes $\mathcal{Q}$ in terms of the model. Given model $\mathcal{M} = (|\mathcal{Q}|, |I|, I)$, describing $\mathcal{Q}$ is the same as specifying the adjustments that needs to be applied to the failure set I in the model to reach $\mathcal{Q}$, which can be done by describing the following sets:

1. Unobserved failures i.e., $\mathcal{Q}^+ = I \setminus \mathcal{Q}$
2. Observation errors i.e., $\mathcal{Q}^- = \mathcal{Q} \setminus I$

In this paper, we assume that there are no observation errors, i.e., $\mathcal{Q}^- = \emptyset$ (as $\mathcal{Q} \subseteq I$). According to the sampling assumption we have, $\mathcal{Q}$ is sampled uniformly at random from I with probability γ. This implies that $\mathcal{Q}^+ = I \setminus \mathcal{Q}$ is sampled from I with uniform probability $(1-\gamma)$. Hence we can compute the probability of seeing a set $\mathcal{Q}^+$ when sampled from I as follows

$$Pr(\mathcal{Q}^+|I) = \gamma^{|\mathcal{Q}|}(1-\gamma)^{|\mathcal{Q}^+|} \tag{6.6}$$

Now, using this probability distribution of observing the set $\mathcal{Q}^+$ given the failure set I we can compute the optimal number of bits required to transmit $\mathcal{Q}^+$ encoded in terms of model M as follows:

$$\begin{aligned} \mathcal{L}(\mathcal{Q}^+|I) &= -\log\Big(\gamma^{|\mathcal{Q}|}(1-\gamma)^{|\mathcal{Q}^+|}\Big) \\ &= -|\mathcal{Q}|\log(\gamma) - (|I| - |\mathcal{Q}|)\log(1-\gamma) \end{aligned} \tag{6.7}$$

6.1.6 Our Formal Problem

Putting it all together, we can state our formal problem GraphStateInf as following:

GraphStateInf: *Given an undirected graph $G(V, E)$, where node failures taken place in the network as per the model described in Section-6.1.1, and a set of observed failures $\mathcal{Q} \subseteq V$, which are sampled independently from the true failure set I^* with a uniform probability γ, find the complete set of failures $I \subseteq V$ by minimizing the MDL cost function $\mathcal{L}\big(|\mathcal{Q}|, |I|, I, \mathcal{Q}\big)$ given by*

$$\begin{aligned} &\mathcal{L}\big(|\mathcal{Q}|, |I|, I, \mathcal{Q}\big) \\ &= \mathcal{L}(|\mathcal{Q}|) + \mathcal{L}\big(|I| \,\big|\, |\mathcal{Q}|\big) + \mathcal{L}\big(I \,\big|\, |\mathcal{Q}|, |I|\big) + \mathcal{L}\big(\mathcal{Q} \,\big|\, |\mathcal{Q}|, |I|, I\big) \\ &= -\log\Big(\sum_{s\in V} p_s(s) \prod_{v\in I} F(v \mid s) \prod_{v'\notin I} \Big(1 - F(v' \mid s)\Big)\Big) \\ &\quad - 2|\mathcal{Q}|\log(\gamma) - 2(|I| - |\mathcal{Q}|)\log(1-\gamma) - \log\binom{|I|}{|\mathcal{Q}|} \end{aligned} \tag{6.8}$$

where $p_s(s)$ is the seed probability of s and $F(v \mid s)$ is the failure probability of node v given seed node s.

6.2 Proposed Methods

Clearly the search space for the problem is large, and there exists no trivial structure for fast search. Indeed, typically MDL-based optimization problems are very challenging. We now describe two approaches for finding solutions with low MDL cost. The first, LocalSearch, incrementally adds a node that gives the most reduction in MDL cost, till no further improvements occur. The second, GraphMap, guesses the size k of the optimal solution, and greedily picks the k nodes that would minimize the cost. We show that the cost of the solution produced by GraphMap is within an additive factor of the optimum.

6.2.1 Algorithm LocalSearch

Here we discuss an algorithm based on a greedy local search approach which is popularly used in many MDL optimizations. Our algorithm LocalSearch works as follows: we initialize $\hat{I}$ to $\mathcal{Q}$. Then for each node v in $V \setminus \hat{I}$, we compute marginal change in the MDL cost caused by adding v to $\hat{I}$. We add the node u which results in maximum decrease in the MDL cost to $\hat{I}$. We repeat the process until the MDL cost cannot be reduced further. The complete pseudocode is given in Algorithm 8. Although intuitive and natural, it is hard to get provable guarantees for this algorithm.

Algorithm 6 Algorithm LocalSearch

Input: Instance $(V, \mathcal{Q}, p, F, \gamma)$
Output: Solution $\hat{I}$ that minimizes $\mathcal{L}(|\mathcal{Q}|, |\hat{I}|, \hat{I}, \mathcal{Q})$
$\hat{I} \leftarrow \mathcal{Q}$
while $\exists\, v \in V \setminus \hat{I} : \mathcal{L}(|\mathcal{Q}|, |\hat{I}|, \hat{I}, \mathcal{Q}) - \mathcal{L}(|\mathcal{Q}|, |\hat{I}| + 1, \hat{I} \cup \{v\}, \mathcal{Q}) > 0$ **do**
    $u \leftarrow \arg\max_{v \in V \setminus \hat{I}} \mathcal{L}(|\mathcal{Q}|, |\hat{I}|, \hat{I}, \mathcal{Q}) - \mathcal{L}(|\mathcal{Q}|, |\hat{I}| + 1, I \cup \{v\}, \mathcal{Q})$
    $\hat{I} \leftarrow \hat{I} \cup \{u\}$
Return $\hat{I}$

6.2.2 Algorithm GraphMap

In this section we propose an efficient algorithm for finding a failure set I which also provides an additive approximation guarantee on the MDL cost of the solution, thereby ensuring that our nodes-set is of high-quality.

First, let $A = -\log \binom{|I|}{|\mathcal{Q}|} - 2|\mathcal{Q}| \log(\gamma)$. We rewrite the MDL cost function in the following

manner: small

$$\begin{aligned}
&\mathcal{L}\big(|\mathcal{Q}|, |I|, I, \mathcal{Q}\big) \\
=&A - \log\Big(\sum_{s\in V} p_s(s) \prod_{v\in I} F(v \mid s) \prod_{v'\notin I} \Big(1 - F(v' \mid s)\Big)\Big) \\
&- 2(|I| - |\mathcal{Q}|)\log(1-\gamma) \\
=&A - \log\Big(\sum_{s\in V} p_s(s) \prod_{v\in V} \Big(1 - F(v' \mid s)\Big) \\
&\qquad \prod_{v\in I} \frac{F(v \mid s)}{(1 - F(v \mid s))}\Big) - \log\,(1-\gamma)^{2(|I|-|\mathcal{Q}|)} \\
=&A - \log\Big(\sum_{s\in V} p_s(s) \prod_{v\in V} \Big(1 - F(v' \mid s)\Big)(1-\gamma)^{-2|\mathcal{Q}|} \\
&\qquad \prod_{v\in I} \frac{F(v \mid s)(1-\gamma)^{2|I|}}{(1 - F(v \mid s))}\Big) \\
=&A - \log\Big(\sum_{s\in V} g(s) \prod_{v\in I} f(s,v)\Big), \qquad (6.9)
\end{aligned}$$

where $g(s) = (1-\gamma)^{-2|\mathcal{Q}|} p_s(s) \prod_{v\in V} \Big(1 - F(v \mid s)\Big)$ and $f(s,v) = \frac{F(v \mid s)(1-\gamma)^2}{1-F(v \mid s)}$. Therefore, the problem reduces to finding a set $\hat{I}$ such that

$$\begin{aligned}
\hat{I} = \arg\min_{I} \Big\{ &- \log \binom{|I|}{|\mathcal{Q}|} + 2|\mathcal{Q}|\lambda_1 \\
&- \log\Big(\sum_{s\in V} g(s) \prod_{v\in I} f(s,v)\Big)\Big\}. \qquad (6.10)
\end{aligned}$$

In GraphMap, we first compute quantities $f(s,v)$ for each $s, v \in V$ and $g(s)$ for each $s \in V$. Then for each $s \in V$, we sort nodes $v \in V$ by $f(s,v)$. The main idea in our algorithm is to use the quantity $f(s,v) = \frac{F(v \mid s)(1-\gamma)^2}{1-F(v \mid s)}$ defined above as the 'weight' for each pair (s,v). We guess the size of the solution $|I_s|$, if the source were to be s. Then we compute, $\phi(s, I_s)$. For each possible size k of the failure set, we compute the cost $h(k)$. Based on pre-computed $\phi(s, I_s)$ and $h(k)$, we compute $\alpha(s,k)$. Finally, we pick the set of $|I_s|$ nodes which maximizes $\alpha(s,k)$. The complete pseudocode is presented in Algorithm 7 and analyzed in Theorem 1.

Theorem 1 *Let I^* be the set minimizing the MDL cost, and let I denote the solution computed by Algorithm* GraphMap. *Then, $\mathcal{L}(|\mathcal{Q}|, |I|, I, \mathcal{Q}) \leq \mathcal{L}(|\mathcal{Q}|, |I^*|, I^*, \mathcal{Q}) + \log(n)$, where n is the number of seed nodes.*

Proof 11 *Recall the definitions of $g(s)$, $f(s,v)$ and A above. Then,*

$$\mathcal{L}(|\mathcal{Q}|, k, I, \mathcal{Q}) = -\log\Big(\sum_{s\in V} \phi(s, I)\Big) + A \quad ,$$

Algorithm 7 Algorithm GraphMap

Input: Instance $(V, \mathcal{Q}, p, F, \gamma)$
Output: Solution $\hat{I}$ that minimizes $\mathcal{L}(|\mathcal{Q}|, |\hat{I}|, \hat{I}, \mathcal{Q})$
For each $s, v \in V$, compute $f(s, v)$
For each $s \in V$, compute $g(s)$
for each $s \in V$ **do**
    Order the nodes $v_1^s, \ldots, v_{n-|\mathcal{Q}|}^s$ such that $f(s, v_1^s) \geq f(s, v_2^s) \geq \ldots$
    ▷ The set $I_s(k) = \{v_1, \ldots, v_k\} \cup \mathcal{Q}$ will be considered
    Compute $\phi(s, \emptyset) = g(s) \prod_{v \in \mathcal{Q}} f(s, v)$
    for $k = 1$ to $n - |\mathcal{Q}|$ **do**
        Compute $\phi(s, I_s(k)) = \phi(s, I_s(k-1)) f(s, v_k)$
        ▷ $\phi(s, I_s(k)) = g(s) \prod_{v \in I_s(k)} f(s, v)$
$h(0) = 1$
for $k = 1$ to $|V| - |\mathcal{Q}|$ **do**
    $h(k) = h(k-1)(k + |\mathcal{Q}|)/k$
for $s \in V$ **do**
    for $k = 1$ to $|V| - |\mathcal{Q}|$ **do**
        Compute $\alpha(s, k) = -\log \phi(s, I_s(k)) - \log h(k)$
Return $I_s(k)$ which optimizes $\alpha(s, k)$

where $\phi(s, I) = g(s) \prod_{v \in I} f(s, v)$*. Note that* $\phi(s, I)$ *is maximized for the set* $I_s(k)$ *defined in Algorithm* GraphMap*, since this consists of the set of top* $k - |\mathcal{Q}|$ *nodes in* $V \setminus \mathcal{Q}$*, with respect to the quantity* $f(s, v)$*, along with all nodes in* $\mathcal{Q}$*. Therefore, we have*

$$\phi(s, I_s(k)) \geq \phi(s, I^*)$$

Adding over all possible seed nodes, we have

$$\sum_{s \in V} \phi(s, I_s(|I^*|)) \geq \sum_{s \in V} \phi(s, I^*)$$

which implies for some seed $\hat{s}$*, we have*

$$\begin{aligned} \phi(\hat{s}, I_{\hat{s}}(k)) &\geq \frac{1}{n} \sum_{s \in V} \phi(s, I^*) \\ \Rightarrow -\log\Big(\phi(\hat{s}, I_{\hat{s}}(k))\Big) &\leq -\log\Big(\frac{1}{n} \sum_{s \in V} \phi(s, I^*)\Big) \\ &= -\log\Big(\sum_{s \in V} \phi(s, I^*)\Big) + \log(n) \end{aligned}$$

This, in turn, implies

$$\begin{aligned}\mathcal{L}(|\mathcal{Q}|, k, I_{\hat{s}}(k), \mathcal{Q}) &= -\log\Big(\sum_{s\in V}\phi(s, I_{\hat{s}}(k))\Big) + A \\ &\leq -\log\Big(\phi(\hat{s}, I_{\hat{s}}(k))\Big) + A \\ &\leq -\log\Big(\sum_{s\in V}\phi(s, I^*)\Big) + \log(n) + A \\ &\leq \mathcal{L}(|\mathcal{Q}|, k, I^*, \mathcal{Q}) + \log(n),\end{aligned}$$

where the first inequality follows because $\phi(s, I) \geq 0 \;\forall\; s, I$*, so that* $\sum_{s\in V}\phi(s, I_{\hat{s}}(k)) \geq \phi(\hat{s}, I_{\hat{s}}(k))$*. Since Algorithm 7 searches over all possible solution sizes* k*, the theorem follows.*

Lemma 14 *Algorithm* GraphMap *runs in* $O(|V|^2 \log |V|)$ *time.*

Proof 12 *The quantities* $g(s)$ *and* $f(s, v)$ *defined earlier in (6.9) can be computed for all* s, v *in* $O(|V|^2)$ *time, which is done in lines 1 and 2. The sort step in line 4 takes* $O(|V| \log |V|)$ *time. The inner for loop in lines 7-10 computes* $\phi(s, I_s(k))$ *defined above, and takes* $O(|V|)$ *time. Therefore, the for loop in the lines 3–11 takes* $O(|V|^2 \log |V|)$ *time. The remainder of the steps take* $O(|V|^2)$ *time.*

6.3 Experiments

6.3.1 Setup

We briefly describe our setup next. All the experiments were conducted on a hybrid cluster with over 2500 nodes and 28 TB of RAM. GraphMap takes roughly 30 minutes to complete for any setting in a single node for our JAM dataset (See Section 6.3.2), which is very practical. Our code is publicly available for academic purposes[1].

6.3.2 Datasets

We evaluate performance of our algorithms on various synthetic and real networks. We discuss our datasets in detail next.

Synthetic Dataset. We created a simple 60×60 grid where each cell is considered as a node in a road network, leading to 3600 nodes. We assumed an uniform seed probability

[1]Code and data at: `http://tiny.cc/GraphMap`

distribution across all nodes. We computed conditional failure probabilities (PlainCF) between pair of nodes (s, v) based on Geographically Correlated Failure (GCF) Model [13] i.e., if s is the seed node then

$$F(v \mid s) = 1 - d(s, v) \tag{6.11}$$

where $d(s, v)$ is a distance function $d : V \times V \to [0, 1]$. In our case, $d(\cdot, \cdot)$ is the Manhattan distance between the nodes normalized by the maximum distance. We will refer to this set of conditional failure probabilities as GCF.

Real Datasets. We created three datasets from real world node failure logs in transportation and power-grid networks. We use failure logs in road networks from WAZE alerts data, which is publicly available on the City of Boston's website[2]. These alerts have been reported by users of WAZE[3] between Monday 23$^{\text{rd}}$ February, 2015 and Sunday 1$^{\text{st}}$ March, 2015. WAZE is a popular crowd sourced application commonly used for navigation. The alerts in the dataset are spatially distributed across Boston, Cambridge, and Brook-line regions of Massachusetts. The alerts include different types failures such as traffic jam, extreme weather, accidents, and road closures. Additionally, the latitude and longitude of the affected locations, and start and end time of the alert is also given. From these alerts, we created two datasets based on traffic jam (JAM) and extreme weather (WEATHER).

Similarly, we use a list of Electric disturbance events from Energy.gov[4]—this list includes reported events of electric emergencies and disturbance in power supply from 2002 to 2015. Each event log contains information regarding date and time of the beginning and restoration of the event, geographical areas affected by the event, number of customer affected, and so on. We created POWER-GRID dataset from the log of electric emergencies and disturbances.

Network Creation. As discussed in Section 6.1.1, we need to define the seed probability and pair-wise conditional failure probability distributions over all nodes in the network. This is done in the following manner. For WAZE alert data, we have partitioned the complete geographical region occupied by these failures by using a 119×78 grid as shown in Figure 6.2a, where each cell is 0.00166° square and acts as a node in our virtual road network. For the POWER-GRID data, each location referred in the dataset acts as a node.

Seed Probability. Let $n_v = 1$ denote if node v is failed, 0 otherwise as discussed above, and let $N = \sum_v n_v$ denote the total number of failures across all nodes. We define the seed probabilities as:

$$p_s(v) = \frac{n_v}{N} \tag{6.12}$$

Conditional Failure Probabilities. We construct a Binary Failure State Time Series (BinTS) for a span of 7 days, by using the temporal information that is available from WAZE alerts data. This time series gives a binary (0 or 1) value for each time step which represents the

[2]https://data.cityofboston.gov/
[3]https://www.waze.com/
[4]https://www.oe.netl.doe.gov/

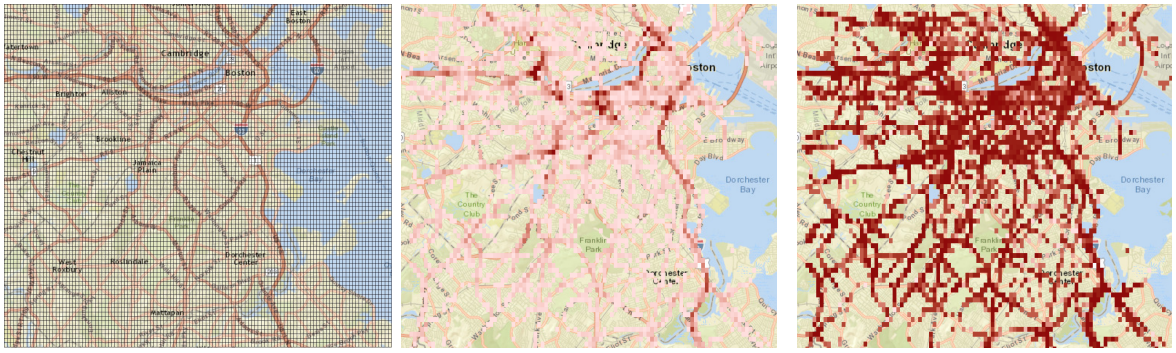
failure state of the respective node i.e., $BinTS_v(t) = 1$ implies that there is at-least one failure in v at time t. Using BinTS we were able to compute the pair-wise conditional failure probabilities for our datasets in the following manner. For two nodes v_1 and v_2, we define the Plain Conditional Failure Probability (PlainCF) of v_1 given v_2 as the ratio between number of time steps in which both v_1 and v_2 are failed (i.e. with value 1 in BinTS) to the number of time steps in which only v_2 is failed.

$$PlainCF(v_1 \mid v_2) = \frac{|\{t|\forall t, BinTS_{v_1}(t) = 1 \& BinTS_{v_2}(t) = 1\}|}{|\{t|\forall t, BinTS_{v_2}(t) = 1\}|}$$

We study two more synthetic conditional failure probabilities, named URandCF and NRandCF, for each dataset; these are defined in the following manner: URandCF is an arbitrary sample from a uniform distribution and NRandCF is an arbitrary sample from a normal distribution $(0.1 \times \mathcal{N}(5, 1))$ over the values $[0, 1]$. We follow the same procedure to generate conditional probability failures for the POWER-GRID dataset.

Descriptions. Following the above steps, we finally get the three different datasets below[1].

JAM: This is a dataset that we created from WAZE alerts data using data of failure type JAM. The resultant dataset consists of a road network with 2650 nodes along with seed probability distribution and conditional failure probabilities as discussed above. Figure 6.2b shows the spatial distribution of the seed probabilities and Figure 6.2c shows distribution of conditional failure probabilities for a randomly chosen seed.



(a): Partitions of Boston region occupied by WAZE alerts.

(b): Spatial distribution of seed probabilities.

(c): Distribution of PlainCF for a random seed node.

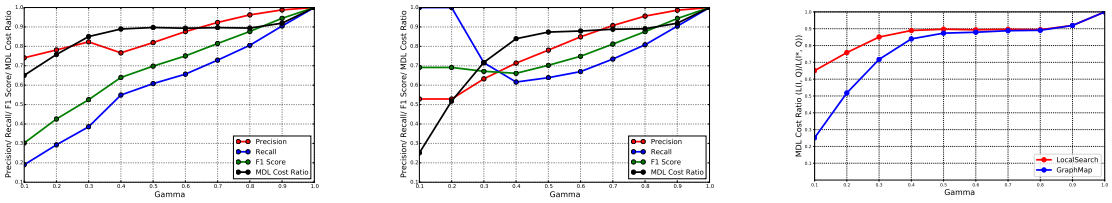
Figure 6.2: JAM dataset created from WAZE alerts data of Boston, Cambridge, Brookline regions. The darker regions indicate higher probability of failure. Note that the seed probability is higher closer to the city.

WEATHER: Similar to the JAM dataset this dataset is created by using WEATHERHAZARD failure data from WAZE alerts data. The resultant dataset consists of a road network with 1520 nodes along with seed probability distribution and conditional failure probabilities computed as discussed above.

POWER-GRID: As mentioned earlier, we created this dataset from the log of electrical emergencies and disturbances. We filtered out the events in the log which were not related to loss of electric service. For this dataset, which consists of 24 nodes, we computed failure likelihood and conditional failure probabilities are before.

6.3.3 Performance evaluation

In this section we discuss the performance of our algorithms against various datasets that are described earlier across various values of $\gamma \in [0.1, 1.0]$. We examine the precision, recall and F1-score for GraphMap, compared with LocalSearch. We observe that our MDL based approach does indeed allow us to infer unknown/missing failures in the network using the probes. The specific MDL formulation we consider in Section 6.1.6, which includes $|I|$ in the model seems to perform much better than other natural MDL formulations.

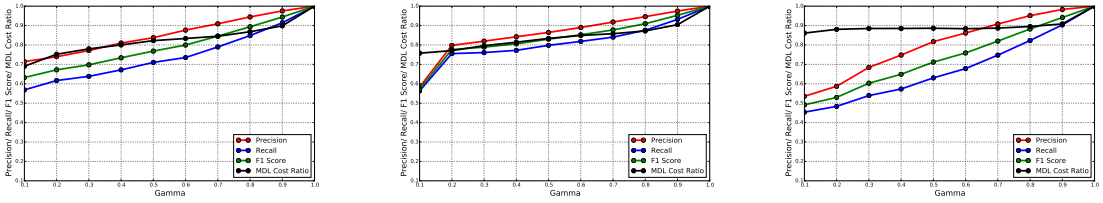


(a) (b) (c)

Figure 6.3: Performance of the (a) LocalSearch and (b) GraphMap on the JAM dataset with PlainCF probabilities. The MDL costs of the solutions is shown in (c).

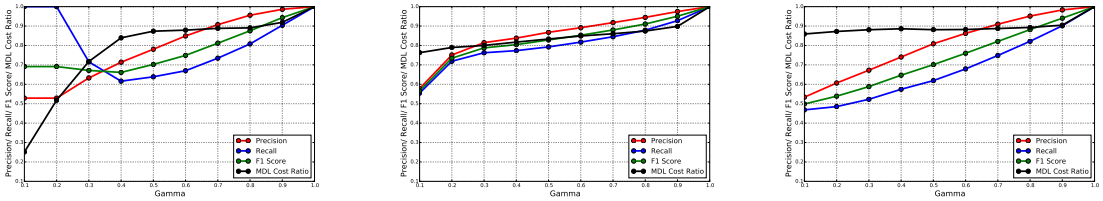
Comparison of GraphMap with LocalSearch. Figure 6.3 presents a comparison of the trends in performance of both algorithms on the JAM dataset with PlainCF probabilities across $\gamma \in [0.1, 1.0]$ and MDL cost of their respective solutions. For both algorithms, the performance varies with the sampling rate, γ. We find that the solution computed by GraphMap has lower MDL cost, compared to the baseline. One interesting observation from Figure 6.3b is that the recall for GraphMap decays with γ till 0.4 and then increases. GraphMap has higher F1-score compared to LocalSearch, for most values of γ. In the rest of our evaluation, we only consider GraphMap.

Performance of GraphMap for different datasets. Figures 6.4, 6.5, 6.6 and 6.7 present the performance of Algorithm GraphMap for all the datasets, and for the three different ways of defining conditional probabilities. Across all these results, on average we are able to find 80% of the failed nodes with an average precision of 79% across various values of $\gamma \in [0.1, 1.0]$. In other words, we are successful in inferring a reasonable fraction of unknown/missing failures in the network from partial set of observations with a reasonable precision.



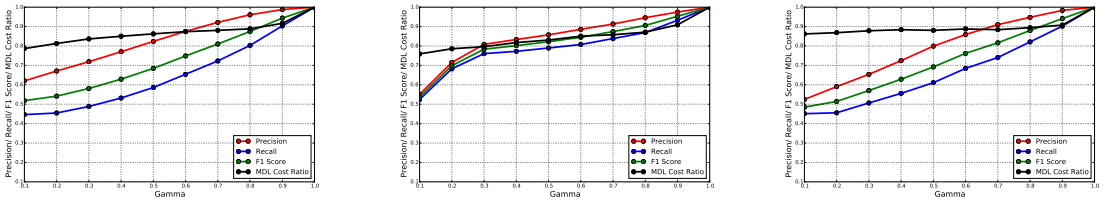
(a) (b) (c)

Figure 6.4: Performance of GraphMap on the Synthetic Grid Dataset with (a) GCF, (b) URandCF, and (c) NRandCF conditional probabilities.



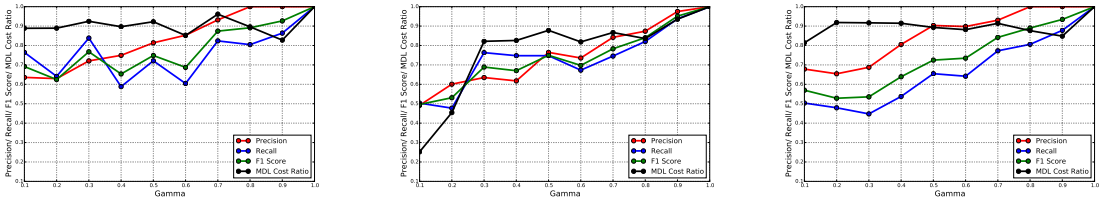
(a) (b) (c)

Figure 6.5: Performance of GraphMap on the JAM Dataset with (a) PlainCF, (b) URandCF, and (c) NRandCF conditional probabilities.



(a) (b) (c)

Figure 6.6: Performance of GraphMap on the WEATHER Dataset with (a) PlainCF, (b) URandCF, and (c) NRandCF conditional probabilities.



(a) (b) (c)

Figure 6.7: Performance of GraphMap on the Power Grid Dataset with (a) PlainCF, (b) URandCF, and (c) NRandCF conditional probabilities.

6.4 Related Work

Some of the different areas related to our work include network and state inference in communication networks, reconstructing networks from cascades and inferring missing infections in the case of epidemics. We briefly discuss these below.

The area of network tomography involves inferring link states, such as delays and failures, in the Internet and other communication networks; see, e.g., [121, 254, 173, 20, 85]. Probes such as end-to-end delays are the only measurements that are available in such networks. At an abstract level, the problem here involves solving for the link delay vector $\mathbf{x}$, given the measured delays across the probes. This becomes a very challenging problem and it is typically assumed that link characteristics such as delays are modeled as *independent* random variables with known distributions, but potentially unknown parameters. Xia et al. [254] solve this assuming link delays are exponentially distributed. Ni et al. [173] study different kinds of probing models, including multicast probes which can give estimates on a tree, and develop methods for inferring the topology in dynamic networks. There has also been work on designing probes to infer part of the network structure, as in [20].

Another related topic is the inference of the source of an infection and other missing infections, in the case of epidemic spread on networks. Epidemics are modeled as stochastic processes, e.g., SI/SIR, in which the infection spreads from an infected node to its susceptible neighbors. Usually, only partial information about the infections is known, and some of the problems that have been studied include identifying the source of an infection and finding other missing nodes [185, 202, 225, 211, 210]. The Minimum Description Length (MDL) principle [104, 195] has been successfully used in [185, 225] for these problems, whereas [211] develop an MLE method. As discussed before, these methods do not give rigorous approximation algorithms or give it only for special graphs (like k-regular trees). In contrast, we give an additive approximation for our MDL formulation for any graph.

A class of failure models, different from the SI/SIR type of epidemics, has been studied extensively, motivated by settings such as disaster events, e.g., [38, 13, 204]. These studies assume an initial failure, and subsequent failures whose probability is correlated with the source. For instance, in [13], the probability $p(j|i)$ that node j fails, given that i is the source is a function of the distance from i to j, with the probabilities decaying with the distance. Our work is motivated by these models.

6.5 Conclusions

Our results show that an MDL based approach is quite useful in the problem of inferring missing failures in settings with correlated failures. This motivates its use in other inference problems with partial information. We have considered the simplest notion of a probe here—information about specific nodes which have failed. Extending our work to other kinds of probes (like connectivity queries) is an interesting and natural problem. Inferring the state of the network using such probes, and supporting additional queries are interesting problems. Further we have assumed there are no observational errors—designing robust and provable algorithms in face of errors is also interesting future work.

Chapter 7

Network State Inference using Connectivity Queries

Inferring network states from partial observations is an important problem with many applications, such as in epidemics, social networks and Internet/AS graphs. Consider critical infrastructure networks like water or electric networks — their rapid functional restoration is a priority right after a disaster such as an earthquake [101]. Hence, it is crucial to quickly assess damage on the network to effectively plan recovery actions. To this end, real time monitoring systems installed in these networks (e.g. SCADA systems) inform decision makers about the functional state of a component ('failed' or not). However, sensors usually installed in a few components, which entail an issue of partial observability.

Inferring complete states can quickly become a very challenging problem. Indeed, in network tomography, all prior work has focused on independent failures [121]. In epidemics, even for simple infection models, inferring missing infections and source detection can become intractable [211]. Recently, there has been some progress on reconstructing states under correlated failures (e.g. spatial correlation naturally fits critical infrastructure networks) [6]. However, uniformly all past work look into so-called *point queries*, i.e. one is given some direct sample of failed components.

On the other hand, *connectivity queries* also arise naturally. Many failures in power grids, for example, spread through path disconnections [63]. Also, among the information provided by SCADA systems, we have whether or not there is supply of utilities on a specific demand node. These demand nodes are supplied by identifiable supply nodes, thus, a stop on supply to a demand node can be mapped to disconnectivity among itself and its supply nodes. Similarly, the continuous supply on a demand node indicates that there is at least one connected path to its supply nodes (see Figure 7.1). Hence failures are not partially observed in this setting — instead, the *effects* of failures are observed partially in terms of connectivity via probes.

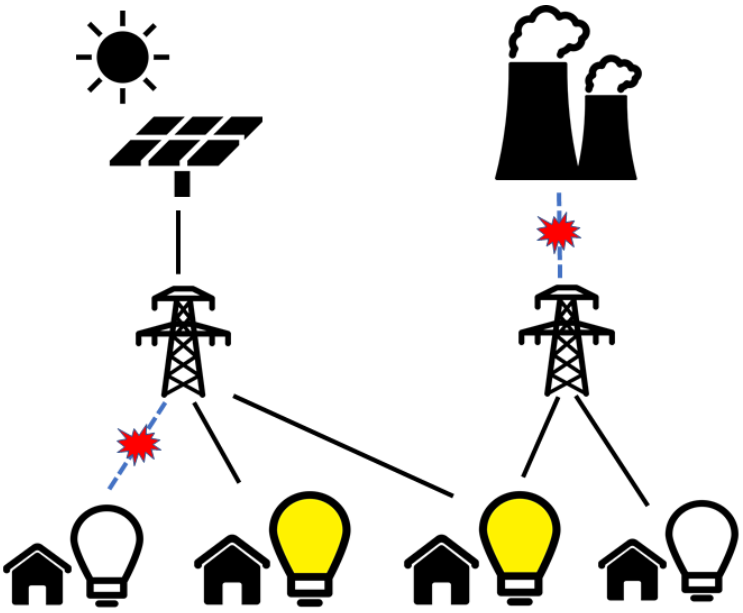


Figure 7.1: An example infrastructure network where some edges have failed due to a disaster. The demand nodes (houses) connected to the supply nodes (power generators) are serviced, while the ones which get disconnected are no longer serviced.

Indeed, after a natural disaster, connectivity queries are *more practical* to obtain than point queries. Sensors are still subject to failure as they are highly interdependent to other systems that may fail in a disaster scenario [179]. Moreover often, failure localization is assisted with on-site inspection by personnel or robots, remote sensing, and others [156], which prolongs time and use of resources. On the other hand, information about supplied demand and customer service (a natural source of connectivity queries) is often relatively accessible—as customers nowadays have multiple ways of reporting issues, including even social media [158]—and has been proven to be useful to localize failures [116].

Motivated by above, in this paper, we investigate the usefulness of connectivity queries in inferring network failures. We propose to the leverage the powerful Minimum Description Length (MDL) [104], to find out both the number of failures and their identity. MDL is especially suited for this task, as it gives us a principled way to formulate our problem. However we find that, both theoretically and empirically, using only connectivity queries is very challenging. Hence next we propose using so-called 'joint queries' where we rely primarily only on connectivity queries but also use a small sample of point queries to boost performance significantly. Our main contributions are:

1. *Using connectivity queries.* We propose problem NetPathState to leverage connectivity queries for inferring network states. We formulate it with MDL and propose an effective algorithm PathMap to solve it. Hardness and empirical results demonstrate this problem is challenging and connectivity queries are not enough.
2. *Using joint queries.* Motivated by the above results, we propose adding a small set of

point queries and formulate JointNetPathState. Our algorithm JointPathMap shows empirically that this small probe set helps considerably.

3. *Extensive experiments on real datasets.* We use domain-based datasets from civil engineers of three real network topologies affected by an earthquake. Our approaches, PathMap and JointPathMap, consistently outperform our baseline and enable inference even on large networks. They also demonstrate the usefulness of adding a small sample of point queries to our data.

To the best of our knowledge, our work is the first to study connectivity queries for network state inference.

7.1 Preliminaries

In this section we describe our setup and failure model.

Context and Setup. We are given an undirected graph $G(V, E)$ that represents an infrastructure network. Links in E represent transmission systems and set V contains three disjoint sets of nodes: supply nodes V_S, demand nodes V_D, and transshipment nodes V_T. Thus, $V = V_S \cup V_D \cup V_T$. Supply nodes are where flow is originated, demand nodes are where flow is received and distributed to final users, and transshipment nodes are where flow is transferred to other link(s). We assume supply and demand nodes are related by many-to-many relationships (i.e. one supply node can service many demand nodes and vice versa) as it is observed in electric and water networks.

In addition, we are given a set $\mathcal{O}$ containing disaster scenarios that characterize the most likely disaster hazards. For example, in case of earthquakes, they can be described by a magnitude and a geographical epicenter; even though there is a myriad of possibilities for this, typically there is a finite set of earthquake scenarios (our set $\mathcal{O}$) that can characterize most of the seismic hazard [2]. Once a scenario is characterized, the effects on the infrastructure network are driven by soil and structural properties in a stochastic manner.

Failure Model. At an abstract level, given a disaster scenario $o \in \mathcal{O}$, some set of edges $I \subseteq E$ fail[1]. We call the remaining edges *alive*. We describe the failure process using a 'geographically correlated' failure model, which has been extensively used in prior work both in the computer science [13] and civil engineering communities [101].

Let $p(o)$ be the probability of a disaster scenario to happen. Clearly, $\sum_{o \in \mathcal{O}} p(o) = 1$. Then, we have a *failure probability distribution* function $F : E \rightarrow [0, 1]$ that gives the failure probability of an edge $e \in E$. We assume all edge failures depend only on the disaster's impact (failures are independent from each other) i.e. for any $e_1, e_2 \in V$, $F(e_1 \cap e_2|o) = F(e_1|o)F(e_2|o)$.

[1]Extending our results to node failures is interesting future work.

7.2 Connectivity Queries Formulation and Approach

Here we systematically formulate our problem based on network connectivity. We use the MDL principle for our formulation.

Connectivity Probes. An operator can hope to, at best, get a partial view of the state of the network — she does not know the true set of failed edges I. Rather, the operator performs so-called "connectivity queries" on the network, as described next, to gather some information on the status.

When failures happens, they directly effect the *serviceability* of the demand nodes in the network. Demand nodes require an active path to a nearby supply node to be serviced. Thus, due to failures because of the disaster, some demand nodes become non-serviced. Connectivity queries (coming from sensors and/or customer reports) serve as indirect observers for the network states. We define a 'sensor' set $\mathcal{S} \subseteq V_D$ as the set of demand nodes which are still serviced after removing failure edge set I from G. Then, it is easy to see that the operator will finally receive only a *connectivity probe* set $\mathcal{Q}_c \subseteq \mathcal{S}$. We can assume that it is chosen uniformly at random (with some given probability say γ_c), from $\mathcal{S}$. Our problem can be roughly stated as: After a disaster, given G, F's, γ_c and probe demand node set $\mathcal{Q}_c$, infer the true set of failed *edges* I in G.

MDL. Note that we do not know apriori how many failures are present in the system. For example, it may happen that $\mathcal{Q}_c$ results from no failures in the system or when all demand nodes except $\mathcal{Q}_c$ are disconnected. Hence, intuitively, to formulate the problem above, we need a model selection framework which finds this automatically in a principled fashion. To this end, we propose to use the information-theoretic Minimum Description Length (MDL) principle [104]. MDL is a practical version of Kolmogorov complexity and both embrace the idea of 'induction by compression'. We use two-part MDL, also known as the sender-receiver framework as we are specifically interested in the model. Our goal is to transmit the given set of probes $\mathcal{Q}$ (the 'data') from sender to receiver in the least cost (in bits), by assuming that both of them know the layout of the network G, γ_c, $\mathcal{O}$ and $F's$. The MDL cost function consists of two parts: (a) Model cost includes the complexity of the selected model that explains the state of the network; and (b) Data cost that represents the cost of sending the given probe data $\mathcal{Q}$ given the model chosen. Formally, given a set of models $\mathcal{M}$, MDL claims the best model M^* is the one that minimizes $\mathcal{L}(M) + \mathcal{L}(\mathcal{D} \mid M)$, in which $\mathcal{L}(M)$ is the model-cost (length in bits to describe model M), and $\mathcal{L}(\mathcal{D} \mid M)$ is the data-cost (the length in bits to describe $\mathcal{D}$ using M).

Model Space and Cost. The first step in formulating our problem is to identify the right model. Our failure process states that the failure set I is induced due to a particular disaster scenario $o \in \mathcal{O}$. Hence, we include both o and I in the model. Note that related works [6] disregard o and only include I in the model (by normalizing over different disaster scenarios). However, in real networks different disaster scenarios induce failures in totally different regions of the network. Hence, normalizing over different scenarios and picking the

most likely failure set I over all possible disasters does not work well. Therefore, both o and I need to be included in the model.

We want to primarily infer only I, but this set by itself is not enough to be the model because there is no direct relationship between I and Q_c; thus, it would not be useful to explain the data. Therefore, we use all three of o, I and $\mathcal{S}$ as our model: $\mathcal{M} = (o, \mathcal{S}, I)$. In other words, we first send the disaster scenario o, then send the true sensor set $\mathcal{S}$, followed by the true failed components I. After sending the model, we will then identify the actual connectivity probes set Q_c as the data.

The MDL model cost, $\mathcal{L}(o, \mathcal{S}, I)$ has three components $\mathcal{L}(o, \mathcal{S}, I) = \mathcal{L}(o) + \mathcal{L}(\mathcal{S}|o) + \mathcal{L}(I|\mathcal{S}, o)$ where $\mathcal{L}(o) = -\log(Pr(o))$, $\mathcal{L}(\mathcal{S} \mid o) = -\log(Pr(\mathcal{S} \mid o))$, and $\mathcal{L}(I \mid \mathcal{S}, o) = -\log(Pr(I \mid \mathcal{S}, o))$. Thus,

$$
\begin{aligned}
&\mathcal{L}(o) + \mathcal{L}(\mathcal{S} \mid o) + \mathcal{L}\big(I \bigm| \mathcal{S}, o\big) \\
&= -\log(Pr(o)) - \log\big(Pr(\mathcal{S} \mid o)\big) - \log\big(Pr(I \mid \mathcal{S}, o)\big) \\
&= -\log(p(o)) - \log\big(Pr(\mathcal{S} \mid o)\big) - \log\left(\frac{Pr(\mathcal{S} \mid I, o) Pr(I \mid o)}{Pr(\mathcal{S} \mid o)}\right) \\
&= -\log(p(o)) - \log\big(Pr(\mathcal{S} \mid I, o) Pr(I \mid o)\big)
\end{aligned}
$$

where $Pr(o) = p(o)$. Note that given I, $\mathcal{S}$ does not depend on o. Hence we have, $Pr(\mathcal{S}|I, o) = Pr(\mathcal{S} \mid I)$. Note that we can think of $Pr(\mathcal{S} \mid I)$ as a 'feasibility function' $h(G, \mathcal{S}, I)$ (which is 0 if $\mathcal{S}$ is not possible after removing I from G, and 1 otherwise). Similarly, recall that we assume that edge failures depend only the disaster's impact and are independent from each other.

Using all of the above in combination with our failure model, the final model cost is

$$
\begin{aligned}
&\mathcal{L}(o) + \mathcal{L}(\mathcal{S} \mid o) + \mathcal{L}\big(I \bigm| \mathcal{S}, o\big) && (7.1)\\
&= -\log(p(o)) - \log\big(Pr(\mathcal{S} \mid I) Pr(I \mid o)\big) && (7.2)\\
&= -\log(p(o)) - \log(h(G, \mathcal{S}, I) \prod_{e \in I} F(e|o) \prod_{e \in Esetminus I} (1 - F(e|o))) && (7.3)
\end{aligned}
$$

Data Cost. An algorithm based on only model cost would add edges to the solution without considering Q_c; hence we need the data cost as well. For the data cost, we send first $|\mathcal{Q}_c|$ and then the set $\mathcal{Q}_c$ itself. To send the size (as $\mathcal{Q}_c$ is sampled uniformly from $\mathcal{S}$ with probability γ_c): $\mathcal{L}(|\mathcal{Q}_c||\mathcal{S}, I, o) = -\log(Pr(|\mathcal{Q}_c||\mathcal{S}, I, o)) = -\log(\binom{|\mathcal{S}|}{|\mathcal{Q}_c|} \gamma_c^{|\mathcal{Q}_c|} (1 - \gamma_c)^{|\mathcal{S} setminus \mathcal{Q}_c|})$. Then the cost of sending $\mathcal{Q}_c$ in terms of previously sent information ($\mathcal{S}$ and $|\mathcal{Q}_c|$) is: $\mathcal{L}(\mathcal{Q}_c \bigm| \mathcal{S}, I, |\mathcal{Q}_c|) = -\log(Pr(\mathcal{Q}_c \bigm| \mathcal{S}, I, |\mathcal{Q}_c|)) = -\log(\gamma_c^{|\mathcal{Q}_c|} (1 - \gamma_c)^{|\mathcal{S} setminus \mathcal{Q}_c|})$. So the total data cost is:

$$\mathcal{L}\big(|\mathcal{Q}_c| \,\big|\, \mathcal{S}, I\big) + \mathcal{L}(\mathcal{Q}_c \,|\, \mathcal{S}, I, |\mathcal{Q}_c|) = \\ -\log \binom{|\mathcal{S}|}{|\mathcal{Q}_c|} - 2|\mathcal{Q}_c|\log(\gamma_c) - 2(|\mathcal{S}| - |\mathcal{Q}_c|)\log(1-\gamma_c) \tag{7.4}$$

Note that maximizing data cost by itself (equivalent to Maximum Likelhihood Estimation) does not provide a principle way to infer the size of I; therefore, it needs to be accompanied with the model cost.

Formal Problem statement. We can now state our problem formally as:

Problem 8 NetPathState*: Given an undirected graph $G(V, E)$ where $V = V_S \cup V_D \cup V_T$, a many-to-many relation mapping source nodes V_S to demand nodes V_D, a set of observed serviced nodes $\mathcal{Q}_c$ sampled uniformly at random with probability γ_c from $\mathcal{S}^*$, find the complete set of serviced nodes $\mathcal{S}^* \subseteq V_D$ and failed edges $I^* \subseteq E$ (that failed as per the failure model described in Section 7.1), by minimizing the MDL cost function $\mathcal{L}\big(o, I, \mathcal{S}, |\mathcal{Q}_c|, \mathcal{Q}_c\big) = \mathcal{L}(o) + \mathcal{L}\big(\mathcal{S} \mid o\big) + \mathcal{L}\big(I \,\big|\, \mathcal{S}, o\big) + \mathcal{L}\big(|\mathcal{Q}_c| \,\big|\, \mathcal{S}, I, o\big) + \mathcal{L}\big(\mathcal{Q}_c \,\big|\, \mathcal{S}, I, |\mathcal{Q}_c|, o\big)$ i.e.*

$$\langle \mathcal{S}^*, I^* \rangle = \arg\min_{\mathcal{S}, I} \Big\{ \text{Eq. (7.3)} + \text{Eq. (7.4)} \Big\}. \tag{7.5}$$

Algorithm for NetPathState Problem. Solving Problem 8 is very challenging. First of all, the search space of the solution is exponentially large as it involves searching over all possible sets of serviced and failed nodes. Moreover, since the objective is a function of two interdependent sets, designing any algorithm with performance guarantee is of a major challenge. Formally, solving Problem 8 is NP-hard to approximate within an $\Omega(\sqrt{\log\log|V|})$ factor, as we discuss below.

Lemma 15 *Problem 8 is NP-hard to approximate within an $\Omega(\sqrt{\log\log|V|})$ factor, in general.*

Proof. (Sketch) Our proof is by a reduction from the multicut problem, which is defined in the following manner: we are given an instance $H = (V_H, E_H)$, and a set of source-sink pairs $(s_1, t_1), \ldots, (s_k, t_k)$. The goal is to pick the smallest subset $E' \subseteq E$ of edges, such that all the source-sink pairs are disconnected in $G[E - E']$, which was shown by [60] to be NP-hard to approximate within an $\Omega(\sqrt{\log\log|V_H|})$ factor, under standard complexity theoretic assumptions. We reduce this to an instance of Problem 8 in the following manner. We add a new source s, and add m' parallel edges (s, s_i) for each i (this can be changed to non-parallel edges, by adding a new node on each such edge). Finally, we add an additional demand node t_{k+1}, and add the path from s to t_{k+1}, which gives us the graph G. We set $V_D = \{t_1, \ldots, t_{k+1}\}$, and $S = \mathcal{Q}_c = \{t_{k+1}\}$, with $\gamma_c = 1$. Let M be number of edges in G. We set $F(e|o) = 1/2^M$ for all edges e. Observe that any feasible solution must make all the

terminals $\{t_1, \ldots, t_k\}$ disconnected from s. Next, the cost of any feasible solution $I \subset E$ is $\leq M|I| + 1$, because of the choice of $F(\cdot)$. Finally, due to the parallel edges (s, s_i), none of those edges will be picked in the solution. This implies, the solution I picked for this instance is also a multicut in H. Since the cost of the solution remains the same, within a fixed multiplicative factor, the lemma follows. ■

We describe our algorithm PathMap for Problem 8 next. It is a greedy heuristic (a strategy commonly leveraged for MDL optimization) which takes myopic optimal steps to minimize the MDL cost. It starts by initializing the failure set $\hat{I}$ to the empty set. At each iteration, an edge $e \in E$, whose addition decreases the MDL cost the most, is added to the failure set $\hat{I}$. The process is repeated until no such edge can be found. The complete pseudocode is in Algorithm 8.

Algorithm 8 PathMap

Input: Instance $(G, V, \mathcal{Q}_c, F, \gamma_c)$
Output: Solution $\hat{I}$ as per Eq. (11.1)
for $o \in \mathcal{O}$ **do**
    $\hat{I}(o) \leftarrow \emptyset$, $\hat{\mathcal{S}}(o) \leftarrow V_D$
    $\alpha(o) \leftarrow \mathcal{L}\big(o, \hat{I}, \hat{\mathcal{S}}, |\mathcal{Q}_c|, \mathcal{Q}_c\big), \alpha' \leftarrow -\infty$
    while $\exists\, e \in Esetminus\hat{I} : \alpha(o) - \alpha' > 0$ **do**
        $\{\mathcal{S}' \leftarrow$ fail $\hat{I} \cup \{u\}$ in $G\}$
        $\alpha' \leftarrow \mathcal{L}\big(o, \hat{I} \cup \{u\}, \mathcal{S}', |\mathcal{Q}_c|, \mathcal{Q}_c\big)$
        $e \leftarrow \underset{e \in Esetminus\hat{I}}{\arg\max}\ \alpha(o) - \alpha'$
        $\hat{I}(o) \leftarrow \hat{I}(o) \cup \{e\}$
        $\hat{\mathcal{S}}(o) \leftarrow \mathcal{S}'$, $\alpha(o) \leftarrow \alpha'$
Return $\hat{I}(o)$ for o that minimizes $\alpha(o)$

Note that the feasibility function contained in the MDL cost will ensure that the infeasible solutions are not selected. This is because the term $h(G, S, I) = 0$ for infeasible solutions. Hence, the MDL cost for such solutions will be infinite. Hence, the algorithm always ensures that $\mathcal{Q}_c \subset \hat{\mathcal{S}}$. sectionJoint Queries Formulation and Approach NetPathState Problem asks to infer the culprit disaster scenario o and the resulting edge failure set I given the connectivity probes $\mathcal{Q}_c$. However, if the demand nodes have multiple paths to the supply nodes (i.e., the network is redundant) the connectivity probes Q_c fail to provide enough information to infer I and o correctly. For example, consider a redundant toy network in Figure 7.2. Each demand node V_D (red circle) is connected to two supply nodes V_S (black square). The failed edges in the network are in blue dashes. Note that in this toy network, despite 50% edges having failed, none of the demand nodes get disconnected from the supply nodes. Hence, in such situations the connectivity probes are not sufficient to infer the failures. It turns out that such a phenomenon is quite common in real critical infrastructure networks. We empirically demonstrate it in Section 7.3. As the failures and serviceability are not interdependent in such networks, the connectivity probes do not relay

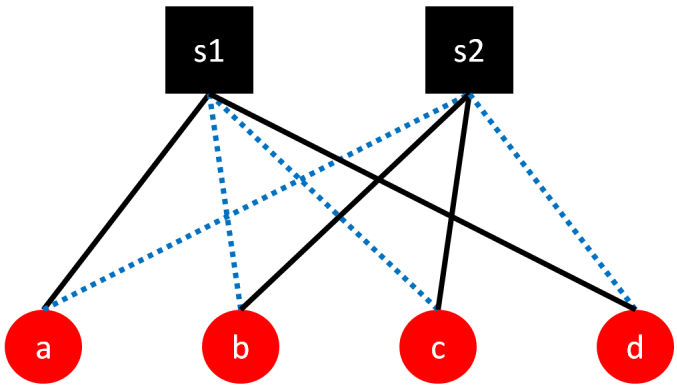


Figure 7.2: In networks where demand nodes $v \in V_D$ (red circle nodes) have multiple connections to the supply nodes $v \in V_S$ (black square), the failed edge (blue dashes) do not effect connectivity, as each demand node is still connected to a supply node despite multiple failures.

any information regarding the failures; making it near impossible to infer the failures given the connectivity probes. The problem is compounded by the fact that the number of failure sets which do not effect the serviceability is exponential in terms of number of such edges. Hence, even when the MDL objective is optimized, not all the failure edges can be inferred. As we show in our experiments, our algorithm PathMap, optimizes the MDL score very well and returns failure set I with high precision. However, the recall is low as it fails to identify the edges which have no effect on the serviceability. The same observation applies to cases, when more than one disaster scenario are present.

Our Idea: Connectivity queries are more practical to obtain; however, we observe that there will be cases where connectivity probes are not sufficient to infer failures. Therefore, a small sample of point probes of the failures may be helpful in boosting the power of connectivity probes as failures are geographically correlated. Based on this observation, we formulate another MDL based problem asking to infer the true failure set of edges I as well as the true sensor set $\mathcal{S}$ given both the connectivity probes and point queries of failures, the latter also referred as failure probes.

Joint Probes. In the previous setting, we were given the connectivity probe set $\mathcal{Q}_c \subseteq \mathcal{S}$ along with G, $F's$, and γ_c. Here, we assume that a small set point queries of failures $\mathcal{Q}_I$ is also available. In reality, the set q_i is the just the set of edges observed to be failed; hence, $\mathcal{Q}_I \in I$. For the ease of modeling, we assume that edges $e \in \mathcal{Q}_I$ are sampled uniformly at random with probability γ_I. In the rest of the paper, we refer to $\mathcal{Q}_I$ as the failure probe set and refer to both $\mathcal{Q}_I$ and $\mathcal{Q}_c$ together as the joint probes.

Before entering to the formulation details, it is worth observing that the algorithm to minimize the MDL objective of the point query formulation of [6] can be approximated within an additive $O(\log n)$ factor. Additive approximations are, in general, easier than multiplica-

tive approximation guarantees. Therefore, Lemma 15 implies Problem 8 is a much harder problem than the point query version, from a complexity standpoint.

Model Cost. Since only a new set of data is being added, the model cost remains unchanged from Problem 8.

Data Cost. The data here consists of the joint probes, including both the connectivity probes $\mathcal{Q}_c$ and the failure probes $\mathcal{Q}_I$. As in the previous setting, we include the size of $\mathcal{Q}_c$, $|\mathcal{Q}_c|$ in the data (described in Eq. (4)). In this new MDL framework, given the model is already transmitted, we know $|I|$, thus, we can use this information to send $\hat{q}_i$. Hence, our data cost consists of four terms.

$$\begin{aligned}&\mathcal{L}(|\mathcal{Q}_I|, \mathcal{Q}_I, |\mathcal{Q}_c|, \mathcal{Q}_c \;\big|\; I, \mathcal{S}, o)\\ &= \mathcal{L}(|\mathcal{Q}_I| \mid I) + \mathcal{L}(\mathcal{Q}_I \mid |\mathcal{Q}_I|, I)\\ &+ \mathcal{L}(|\mathcal{Q}_c| \mid \mathcal{S}) + \mathcal{L}(\mathcal{Q}_c \mid |\mathcal{Q}_c|, \mathcal{S})\end{aligned}$$

Following same the pattern of our previous derivations, the data cost is small

$$\begin{aligned}&= -\log \binom{|I|}{|\mathcal{Q}_I|} - 2|\mathcal{Q}_I| \log(\gamma_I) - 2(|I| - |\mathcal{Q}_I|) \log(1 - \gamma_I)\\ &- \log \binom{|\mathcal{S}|}{|\mathcal{Q}_c|} - 2|\mathcal{Q}_c| \log(\gamma_c) - 2(|\mathcal{S}| - |\mathcal{Q}_c|) \log(1 - \gamma_c) \qquad (7.6)\end{aligned}$$

Formal Problem Statement. We can now state our second problem as follows:

Problem 9 JointNetPathState*: Given an undirected graph $G(V, E)$ where $V = V_S \cup V_D \cup V_T$, a many-to-many relation mapping source nodes V_S to demand nodes V_D, a set of observed serviced nodes $\mathcal{Q}$ sampled uniformly at random with probability γ from $\mathcal{S}^*$, find the complete set of serviced nodes $\mathcal{S}^* \subseteq V_D$ and failed components $I^* \subseteq E$ (that failed as per the failure model described in Section 7.1), by minimizing the MDL cost function small*

$$\langle \mathcal{S}^*, I^* \rangle = \arg\min_{\mathcal{S}, I} \Big\{ \text{Eq. (7.3)} + \text{Eq. (7.6)} \Big\}. \qquad (7.7)$$

Algorithm for JointNetPathState Problem. Solving Problem 9 has all the challenges of Problem 8 like exponentially large search space and interdependence of two sets, namely I and $\mathcal{S}$. We can modify PathMap, described in Algorithm 8, in the following manner: initialization of $\hat{I}(o)$ is now $\mathcal{Q}_I$ (in line 5), and both α functions are now changed to our new MDL cost function. Let us call this algorithm JointPathMap.

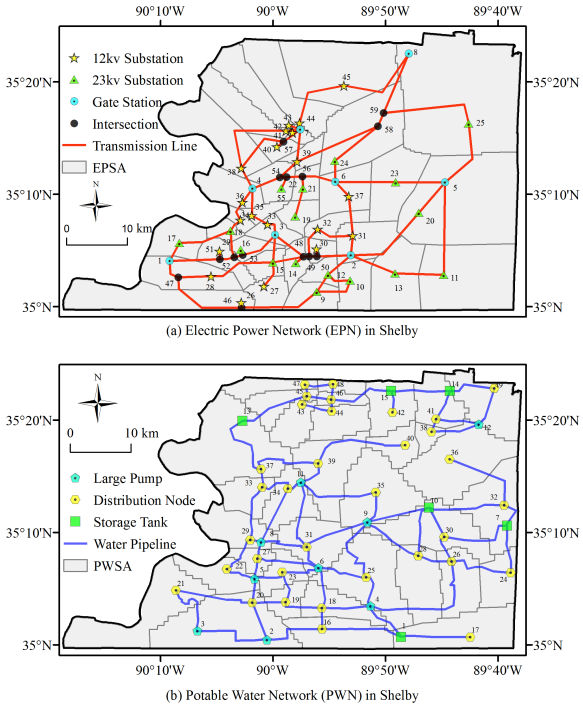


Figure 7.3: Skeletonized power and water network topologies in Shelby County, Tennessee: (a) electric power network (EPN); (b) potable-water network (PWN).

7.3 Experiments

subsectionSetup We run our algorithms on a Linux machine with 8 cores with 3.40 GHz and 16GB of RAM[2]. PathMap takes roughly 2 seconds to output a solution for EPN and PWN networks, and roughly 500 seconds for GridKit (see Section 7.3 for network specifications). Results presented are the average over 30 different *damage scenarios* simulated as per civil engineering standards, each originated from a single *scenario earthquake* $o \in \mathcal{O}$ (a magnitude and an epicenter) which was randomly selected with probability $p(o)$. From the resulting I and $\mathcal{S}$ of each damage scenario, we query (sample) connectivity or joint queries. For each $\gamma_{\mathcal{S}}$, we get two sets $\mathcal{Q}_c$ to assess the robustness of our results over sample randomness. The solutions of these two $\mathcal{Q}_c$ are also aggregated.

subsectionDatasets We evaluate the performance of our algorithms on the topologies of three real infrastructure networks. For each network, we follow procedures as described in civil engineering literature [2] with structural parameters from [89] to generate multiple geographical correlated failure probabilities for nodes. Edge failure probabilities are calculated from these counterparts. Specifically:

EPN & PWN. We use two utility networks for Shelby County, TN, USA: electric power

[2]We will release our code for research purposes.

network (EPN) and potable water network (PWN), with $|V_{EPN}| = 59$, $|E_{EPN}| = 73$, $|V_{PWN}| = 49$, $|E_{PWN}| = 71$, whose topologies are depicted in Figure 7.3 and are publicly available [267]. We adopt the 10 scenario earthquakes that characterize the seismic hazard in the area identified in [2]. This means that once an earthquake happens, its effects on structures can be characterized by one of these 10 scenario earthquakes. Each of the scenario earthquake induce different failure probability-sets (F's) for the network[3].

GridKit. We use the high-voltage power grid for Western North America publicly available at [253]. From it, we extract the subgraph corresponding to Northern California. It contains $|V_{GK}| = 415$ and $|E_{GK}| = 543$. We generate 50 scenario earthquakes, each of them as follows: epicenter and magnitude are selected randomly, the first from all node locations, and the second from the four values considered in [2].

Furthermore, we consider the following two factual observations. (1) Even though the network topology allows it, in practice, source nodes cannot service *any* demand node, but only the ones close to them that its generative capacity allows. To model this physical constraint, we restrict serviceability of a demand node to the case when there is at least one path of length L to any source. In our experiments, we use $L_{EPN} = 3$, $L_{PWN} = 2$, and $L_{GK} = 4$. (2) In practice, after an earthquake occurs, an estimate of its magnitude and epicenter is released within minutes, but this information is then corrected several times over the next hours. Therefore, for this kind of natural disaster, it is preferrable to handle uncertainty by considering multiple earthquake scenarios as we did in our formulation. However, once the estimation about magnitude/epicenter is reliable, we can add this extra information to our problem by reducing $\mathcal{O}$ to a single o (i.e. $|\mathcal{O}| = 1$), thus $p(o) = 1$. For our experiments in EPN and PWN, we use one of the hardest single scenario earthquakes (number of failures is higher than average across all scenarios).

subsectionBaselines and Metrics We use ModelCost as baseline, an algorithm that greedily minimizes model cost in Eq. (7.3), to demonstrate the importance of considering the data cost. To evaluate our algorithms, we use precision $= |\hat{I} \cap I|/|\hat{I}|$, recall $= |\hat{I} \cap I|/|I|$, and F1-score $=$ precision $\times$ recall/(precision $+$ recall) with the following limit cases: precision is 1 when $\hat{I} = I = \emptyset$ (we correctly identified no failures), and recall is 1 when $|I| = 0$ (there are no failures missed in our $\hat{I}$). In addition, we use the MDL ratio between the solution of the algorithm and the ground truth (the actual failure set). If this ratio is less than 1, it means our solution has lower MDL cost than ground truth.

subsectionResults In Section 7.2, we discussed theoretical and practical limitations of any algorithm for Problem 8, which are empirically corroborated in the results of this section. Even though connectivity queries are more practical to obtain, sometimes we are forced to add a *small* sample of point queries (Problem 9) to obtain an acceptable performance. Our experiments show the positive effect of adding this piece of information.

Connectivity Queries. The goal of Problem 8 is to minimize the MDL cost function.

[3]More information about the failure generation procedure can be found in our supplement.

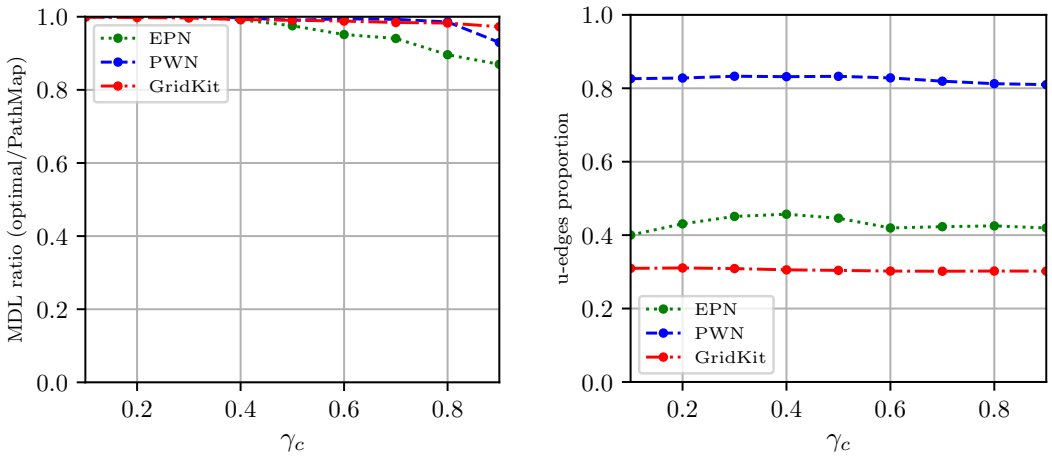


Figure 7.4: Left: MDL deviation from optimal for PathMap (as percentage) across γ_c. Right: u-edges proportion across γ_c.

PathMap consistently optimizes MDL with a deviation not greater than 15% as (Figure 7.4). Minimizing MDL proves to be meaningful because we are correctly identifying failures as demostrated by our precision (Figure 7.5); however, the low recall indicates that, while our inferred failures are mostly correct, they are just a fraction of the total failure set. As mentioned in section 7.2, depending on the network topology, some of the failures are not reflected in the connectivity queries, for which it would be very hard to infer by only using $\mathcal{Q}_c$.

To get a sense of how many of undiscoverable edges (u-edges) are present in ground truth failure set I, we add as many edges from I to $\hat{I}$ as long serviceability is not affected (i.e. $\hat{\mathcal{S}}$ does not change). The proportion of existing u-edges in I presented in Figure 7.4 is surpringly high, especially for the PWN, which is at the same time the hardest to solve for PathMap. This indicates there are many failure sets that can produce the same serviced set $\mathcal{S}$; consequently, solely using our connectivity probes $\mathcal{Q}_c$ is clearly not sufficient. In other words, there is no principle way to find u-edges as their failures are not reflected in our data. This is also confirmed by inspecting the network topologies in Figure 7.3 and examining our failure sets, where we found cases where multiple edges failed, but no or very few demand nodes were left non-serviced. In conclusion, u-edges pose a fundamental issue to Problem 8 for our datasets (principled simulations in real networks).

Joint Queries. In Figure 7.6, we observe that JointPathMap is a great choice to optimize our MDL at low computational cost. Furthermore, our performance improves significantly (EPN and PWN) by only adding a sample of 20% of point queries. The high precision obtained ensures our restoration resources are used efficiently. A higher recall allow us to reveal more failures to recover serviceability promptly. And F1-score captures the interplay

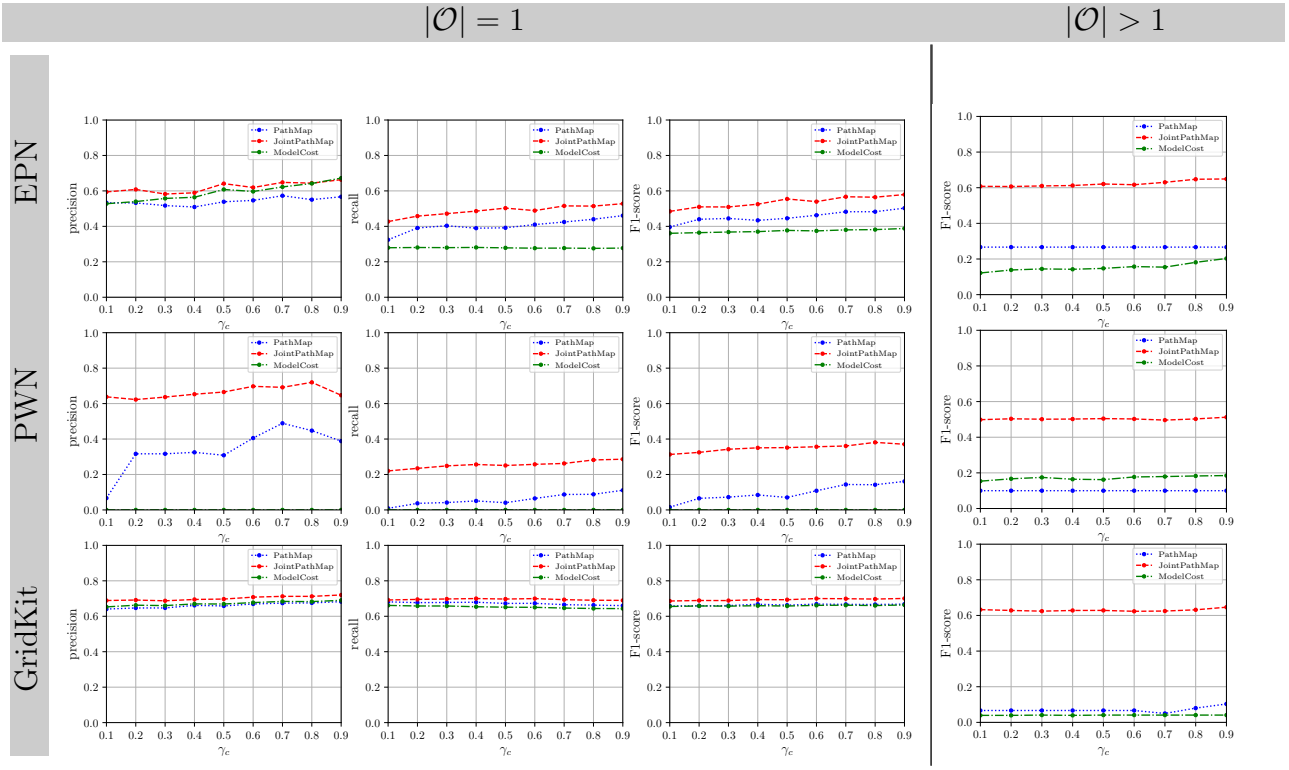


Figure 7.5: Datasets from top to bottom row: EPN, PWN, and GridKit. Left: precision, recall and F1-score for PathMap, JointPathMap ($\gamma_I = 0.2$) and ModelCost for $|\mathcal{O}| = 1$. ModelCost returns empty solutions ($\hat{I} = \emptyset$) for WPN because all F's are below 0.5 (i.e. are more likely to remain functional), but the expected failure set size is ≈ 10. Note the greatest improvement due to the small addition of point queries happens for PWN. Right: F1-score for PathMap, JointPathMap ($\gamma_I = 0.3$) and ModelCost for $|\mathcal{O}| > 1$.

between precision and recall at the solution level. In addition, we can see in Figure 7.6 the effect of different sample sizes γ_I in recall: each increment has a different effect.

For multiple scenario earthquakes, the algorithm has to infer first the culprit scenario o and then the complete failure set I. In a small network, some of scenario earthquakes are similar and missing the right one may not drastically affect further inference; however, in large networks, scenarios earthquakes are completely different (mainly in epicenter), therefore, without a proper inference of o it is very hard to make a good inference of I. This explains the results for PathMap in the rightmost column of Figure 7.5. Adding a small sample of failure probes helps to properly identify o, and gives a large improvement in F1-score.

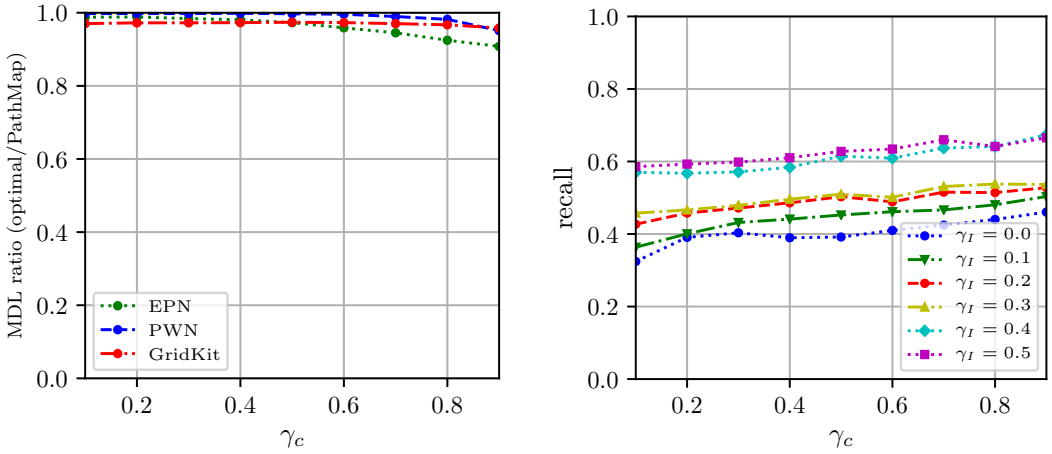


Figure 7.6: Left: MDL deviation from optimal for JointPathMap (as percentage) across γ_c. Right: The effect of increasing γ_I (i.e. increase the sample of failure probes) in recall for EPN dataset.

7.4 Related Work

In short, unlike past work, we tackle a more challenging problem of inferring network states given only *connectivity* probes and present near-optimal algorithms using MDL.

Critical Infrastructure Networks: Data mining critical infrastructure [227] has been gaining much research interest in recent times. These include works on modeling [179, 53], dependency inference [61, 84], vulnerability analysis [23] and finding hotspots [63]. A more closely related line of work studies failure inference on infrastructure networks. [22] proposed an algorithm to infer congested intersections given the road networks and usage data. Similarly [6] proposed a near-optimal MDL-based approach to infer failures, but given point probes.

Network Tomography: This involves inferring the state of the communication network such as delays and link failures given the end-to-end probes [254]. [121] proposed a heuristic to infer approximate topology based on end-to-end measurements while [20] study inferring topology of sparse networks.

Epidemic Inference: Another closely related field is reverse engineering an epidemic outbreak given partially observed infections. [211] propose a MLE based approach, [185, 225] leverage a MDL approach and [202] use Steiner trees to detect missing infections and identify sources.

7.5 Conclusions

We investigated the problem of inferring network states given connectivity queries, a piece of information more practical to obtain than a direct sample of network states (point queries). Theoretical and empirical observations indicate that redundancies, common in infrastructure networks, cause a significant proportion of states to be undiscoverable. Therefore, we propose adding to the connectivity queries a small sample of point queries to exploit relationships among network states, which proved to be valuable for boosting the power of connectivity queries. We formulated both novel problems with MDL and developed greedy heuristics that consistently optimizes MDL cost close to optimal in extensive experiments. These experiments are performance in real networks and principled simulations. Future work can focus on extending our results for node failures and also for noisy probe data.

Part II

Domain-aware representation learning

Chapter 8

Deep Learning for Task-based Network Summarization

Networks are common and occur in many different domains such as social networks, protein-protein interaction, communication networks and so on. Understanding these networks is essential for many tasks, such as viral marketing, community detection, anomaly identification and finding different patterns. One approach is to construct *network summaries* by merging (grouping) nodes and edges into super-nodes and super-edges.

Moreover, *task-based* summaries can help in solving various problems. For instance, identifying epidemiologically relevant anomalies in dynamic human-mobility networks [97]. There are ways to identify anomalies on a network, but in a dynamic network, it is not straightforward how to use the information of previous snapshots to detect anomalies effectively and efficiently. On the other hand, if we learn a function that summarizes each snapshot of the network and highlights the anomalies on them, it would make it easier to detect anomalies on the dynamic graph. So, given a task-based summary for the task, we can potentially discover epidemiologically relevant anomalies in dynamic human-mobility networks. Fig. 8.1 shows the summary networks using a learned summarization function for three different snapshots of such networks in the context of interactions between different departments of a company; the anomalous nodes are shown in black; as discussed in experiments later, the summaries consistently identify anomalies correctly matching the given ground-truth.

Similarly, task-based summaries can help in sense-making and visualizing hidden patterns in networks, e.g., for identifying bridges in different types of networks. Finally, task-based summaries can also help in getting a better quality solution for tasks with known algorithms [146]. Usually, past work has looked into constructing summaries preserving important characteristics of the network (e.g., structural characteristics).

Based on previous work, we make three observations. (1) Different tasks give rise to different network summaries. For example, the desired summary for the community detection task

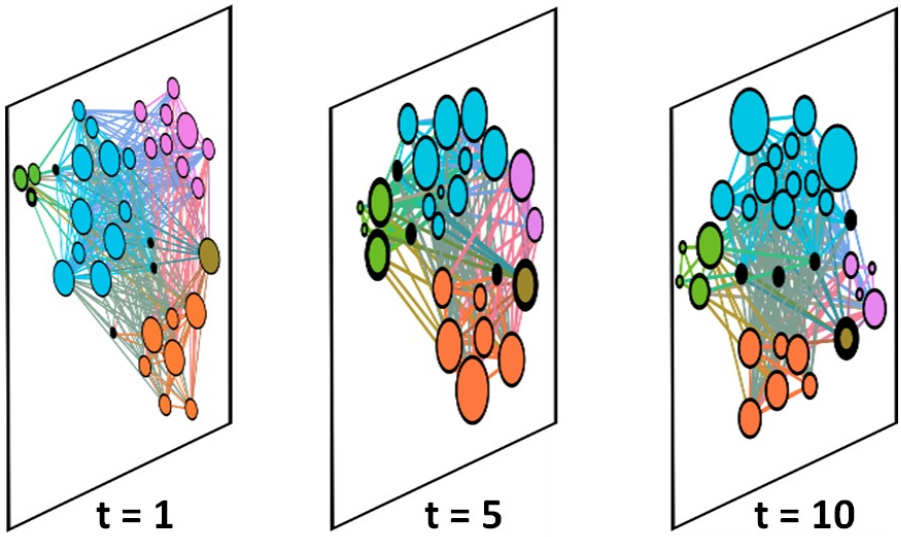


Figure 8.1: Summary graph of three snapshots of Work-Place data. Black nodes are anomalies.

(which tries to find cohesive regions) is very different from the summary for a task like influence maximization (which seeks to find central nodes). (2) There are tasks with no summarization algorithms such as the anomaly detection problem discussed above, traveling salesman problem, etc. Also, new tasks are getting defined on networks which may make it impractical to develop specific algorithms for each of them. (3) In real-world, we usually solve the same task repeatedly on similar networks that belong to the same distribution. For example, generating layouts for similar graphs is helpful in context of protein networks [19], network-traffic data [263] etc. Similarly, an advertiser may target the same social network repeatedly with probabilistic influence patterns [76]. Samples of networks from the same distribution are frequently observed in intelligence analysis [69], epidemiology [9] and many other settings.

Hence, leveraging these three observations, in such scenarios, the inherent similarity between networks and the variety of tasks opens a space for *learning* task-based network summaries for networks coming from a distribution. This motivates the following informal learning problem:

Problem: Task-based Summaries. Given a problem $Prob$, and a distribution D of network instances, can we learn *how* to generate meaningful network summaries that generalizes to unseen instances from D?

Leveraging recent advances in reinforcement learning and deep learning [37, 164], we develop NetGist, a flexible approach which automatically learns *how* to generate a summary for a given set of tasks. To the best of our knowledge, past work has not looked into this novel problem. Our contributions in this paper are, (I) Problem formulation (II) Designing effective

learner and (III) Extensive Experiments.

The rest of the paper is organized in the usual way.

8.1 Problem Formulation

Following prior work in network summarization [186, 80, 218], in a summary, it is natural to *group* ('merge') similar nodes to construct a graph of 'super-nodes' and 'super-edges' with a smaller size than the original network. In this way, each super-node represents homogeneous regions [80] in some sense and the connection between super-nodes highlights the overall structure of the network and important regions of the original graph [186]. Intuitively, our goal is to find a good network summary which guides us to find the solution for a given task.

Tasks: The first question is what kind of tasks we want to handle? As our first step towards learning task-based summaries for networks, we choose to solve a set of problems we call **G**raph **O**ptimization **P**roblems (GOP) which includes many popular graph mining problems. Suppose we have a graph $G(V, E)$ where V and E are sets of nodes and edges of the network. Also, assume Θ is the set of parameters of the given task. In a problem $prob \in$ GOP, the goal is to select $O \subseteq U$ (i.e. a set of objects O from the universe U of objects determined by the problem), that maximizes some quality function $F_{prob}(O; G, \Theta)$. For all GOP problems, U is some set of objects defined over the graph G (e.g. nodes, edges, subgraphs, etc.). Note F_{prob} maps a sets of objects to a real value based on the given task *prob* and its parameters Θ on the graph G. We formally define the set of graph optimization problems as follows:

Definition 8 *Graph Optimization Problem (*GOP*)*

Given *a graph $G(V, E)$, a set of input parameters Θ and a quality function $F_{prob}(O; G, \Theta) \in \mathbb{R}$ and set U,*

Find *the best set of objects O^* such that,*

$$O^* = \arg\max_{O \subseteq U} F_{prob}(O; G, \Theta) \tag{8.1}$$

Note, many combinatorial problems such as Minimum Steiner Tree, Maximum Clique, etc. [122] that defined over graphs can naturally be written as a GOP problem. So, the set of GOP problems is broad and includes many problems of interest. We use two common GOP problems to showcase our method: Influence maximization and Community detection.

What is a good summary? To generate any summary we need to (Q 1) Evaluate the structure of the summary graph; and (Q 2) Identify homogeneous regions. Our main insight is that both of these issues are *task-dependent*. We show the summary for the Influence maximization problem in figure (b). It shows the given tasks guide us to both evaluate the structure and characterize homogeneous regions in the summary graph.

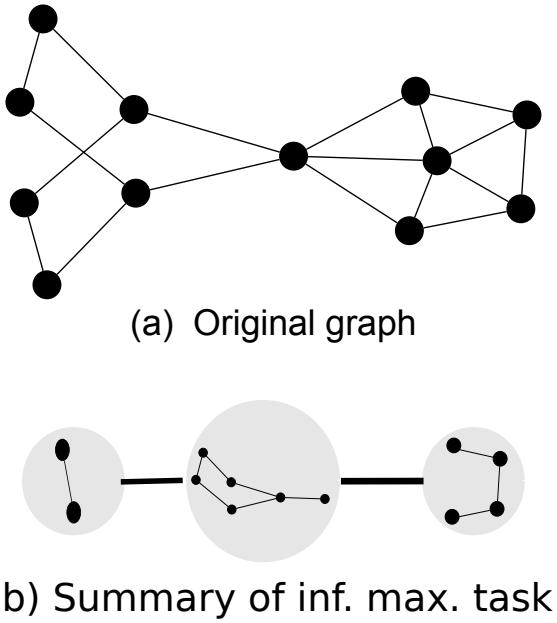


Figure 8.2: Influence maximization task based summary of a toy example.

The summary highlights how different super-nodes (regions) are connected to each other based on the given task and highlight the central region of the graph (Q1). Also, each super-node in the summary shows a region in the graph which contains nodes with the same role (effect on the solution) based on the given task. So, the central node in the graph is central in the central region as well (Q 2). Note that these properties also lead to visually appealing summaries, as they highlight the structural characteristics of the graph important to the given task.

Measuring Summary Quality is challenging as the answers are inter-dependent. Defining what characterizes a homogeneous region necessarily impacts the right measure to evaluate the overall structure (and vice versa). Our idea is to identify properties and develop a procedure which answers both questions simultaneously.

We first formalize the notions of a 'good summary'. Let $G^*(V^*, E^*)$ be an ideal summary of the original network $G(V, E)$ for an arbitrary GOP problem *prob* with parameters Θ. Also let O^* be the optimal solution for *prob* on G. Then we want the following properties to hold for G^*.

Property 1 *Let* $O'^* = \arg\max F_{prob}(O', G^*, \Theta^*)$. *Then,* $\forall_{o \in O^*}, \exists H(o) = x \in O'^*$.

Property 1 states that there exists a mapping H which maps all objects in the optimal

solution for *prob* on G to an element of the optimal solution for the *prob* on G'. This property indicates that G and G' have structural similarity w.r.t. *prob*.

Property 2 *For any two pairs of nodes v_i and v_j in an arbitrary super-node x in the summary,*

$$F_{prob}(\{v_i\}, G, \Theta_x) > F_{prob}(\{v_j\}, G, \Theta_x) \implies$$
$$F_{prob}(\{v_i\}, x, \Theta) > F_{prob}(\{v_j\}, x, \Theta)$$

Property 2 implies that the order of role of nodes inside each super-node, is preserved. It means the nodes in each super-node have a homogeneous role in determining the solution and they lead to the right solution as their order is preserved.

These two properties indicate if an ideal summary G^* for G w.r.t. to GOP problem *prob* exists, we can reconstruct the optimal solution to *prob*. So, to determine how far a summary is from G^*, we propose a divide-and-conquer Algorithm 9 to measure the quality of any summary G' of G: first solve *prob* on G' and recursively solve for *prob* on each solution obtained in the first step. And intuitively the quality of the solution tells us how good the summary is. Lemma 16 indicates that the Divide and Conquer framework we propose returns optimal score for the summary network that satisfies both the properties.

Lemma 16 *Algorithm 9 on a summary network G' returns the optimal solution O^* and optimal score $F_{prob}(O'^*; G', \Theta)$ if G' satisfies both Properties 1 and 2.*

Note that validating whether a summary satisfies Property 2 is computationally expensive as it involves repeatedly measuring F_{prob}. Our approach avoids this step and still manages to output the ideal solution and the corresponding optimal score F_{prob} with respect to *prob* if the summary network satisfies both the properties.

Problem definition: We are now ready to define our problem formally. Suppose we have a network $G(V, E)$ where V and E are sets of nodes and edges of the network respectively. Also, suppose we are given a graph optimization problem *prob* $\in$ GOP (Def. 8) which asks to find the best set of objects O^*. The parameters of *prob* and the graph G are drawn from a probability distribution D (i.e. $(G(V, E), \Theta) \sim D$). For example, D can be the set of probabilistic influence patterns (as discussed in the introduction). We want a summary network $G'(V', E')$ for any graph $G(V, E)$ drawn from D such that $|V'| = \alpha \cdot |V|$, where α is the 'reduction factor'. It is expensive to generate the high-quality summary for all the graphs in D. Hence, our idea is to learn the *process* of learning the summaries instead of merely the final summary graphs. Our goal is to learn a graph summarization process such that solving the given problem on G' using our divide-and-conquer strategy (Algorithm 9) results in a set O'^* such that $F_{prob}(O'^*; G, \Theta)$ is similar to the $F_{prob}(O^*; G, \Theta)$. Formally:

Algorithm 9 Divide and Conquer

Require: $G(V, E)$, $G'(V', E')$, $prob \in$ GOP
1: $O'^* \leftarrow \emptyset$
2: //Divide
3: $\Phi^* \leftarrow \arg\max F_{prob}(\Phi; G', \Theta)$
4: // Conquer
5: **for** $\phi \in \Phi^*$ **do**
6: $G_\phi \leftarrow$ subgraph of ϕ in G
7: $\Theta_\phi \leftarrow$ sub-parameters that related to ϕ
8: $o^*_\phi = \arg\max F_{prob}(o_\phi; G_\phi, \Theta_\phi)$
9: $O'^* \leftarrow O'^* \cup o'^*_\phi$
10: return O'^* and $F_{prob}(O'^*; G, \Theta)$

Problem 10 *Task-Based Network Summarization (TBNS)*

Given *a graph optimization problem (prob* $\in$ GOP*), a distribution of problem instances D, and a reduction factor $0 < \alpha \leq 1$.*

Learn *a graph summarization process such that for any $(G(V, E), \Theta) \sim D$,*

$$\max \rho = \mathbb{E}_{(G,\Theta)\sim D}\left[\frac{F_{prob}(O'^*; G, \Theta)}{F_{prob}(O^*; G, \Theta)}\right] \tag{8.2}$$

where, O'^* is the solution of the given problem prob on the summary graph G' using Alg. 9.

Comments: Generalizing a learned model to unseen instances is important from the learning perspective. A good model which is trained on a particular 'training' data is expected to do reasonably well in the 'test' data drawn from the same distribution [160]. Hence, our ultimate goal is to maximize the expected ratio ρ over all the instances (including the unseen ones) of (G, Θ) drawn from D. Note that in some real-world scenarios, D can be a single (G, Θ).

8.2 Our Method

Next, we show how to solve our TBNS problem. We want to precisely learn the steps to be taken for the summarization process, so that we can re-apply the same approach on other graphs from the same distribution. A basic summarization step we take is some sort of a 'merge' operation. The merge operation basically groups nodes in the original network and results in a so-called 'super-node' in the summary network. The question at hand is that once the summarization step is decided, how do we measure the 'goodness' of each step and how do we obtain a quality summary? Reinforcement Learning (RL)—more specifically Q-learning [226]—is a natural solution to the above questions. In general, RL methods learn to

take an '*action*' in an '*environment*' to maximize cumulative '*reward*' based on a '*transition*' function.

Recently, the success of deep reinforcement learning [164], where convolutional neural networks are used to learn a representation of 'states', in solving various AI tasks has gained much attention. It is known to perform better and converge faster than the traditional reinforcement learning. Hence, we propose to leverage a deep reinforcement learner to solve our TBNS problem.

8.2.1 Overview of NetGist framework

Algorithm 10 Overview of NetGist

Require: $prob \in$ GOP, α, D

Randomly Initialize the deep Q-learning parameters
for $i = 1$ **To** num of samples **do**
    Sample $G(V, E)$ and Θ drawn from D
    // learning how to summarize
    for episode=1 **to** T **do**
        $G'(V', E') \leftarrow G(V, E)$
        while $|V'| > \alpha \cdot |V|$ **do**
            Merge $G'(V', E')$ // (See Section 8.2.2)
        // Evaluate
        $O', F_{prob} \leftarrow$ DivideAndConquer$(G, G', prob)$ Alg. 9
        // Optimize
        Update the deep Q-learning parameters to yield better summary (see Section 8.2.4)
Return the trained model (which solves the TBNS problem)

We now give an overview of our NetGist approach (Algorithm 10). Overall, in our deep reinforcement learning paradigm, the initial 'state' is the network $G(V, E)$ drawn from D and each 'action' the learner takes reduces the size of original network G to produce a new 'state', a summary network $G'(V', E')$. The learner repeatedly takes actions till the summary network $G'(V', E')$ is of desired size. Next, we evaluate the *prob* on G' using Algorithm 9. Based on the quality of the solution, the learner's parameters are updated. The learner repeats the process on more samples from D until it learns to summarize any $(G, \Theta) \sim D$ to G', such that ρ is maximized. To formalize our framework, we need to answer some questions and make design choices. These question are: **Q1** How to define the universe of meaningful actions that summarizes any network $G(V, E)$ drawn from D into $G'(V', E')$? **Q2** What is the universe of all possible 'states'? and **Q3** What are the reward, policy, transition functions and the overall learning procedure? We answer these questions next.

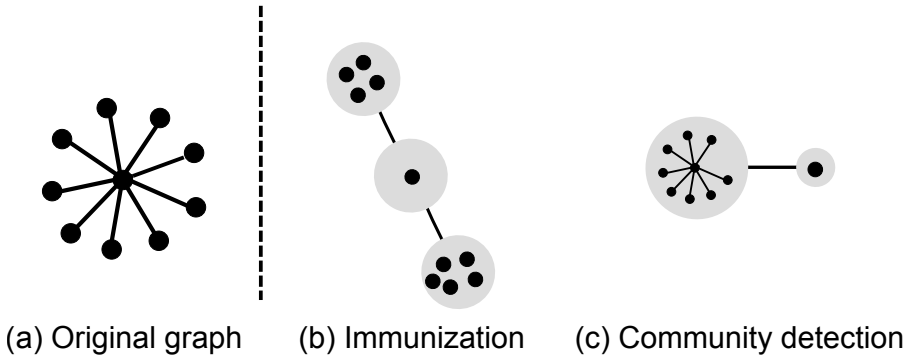


Figure 8.3: Task specific summaries of a simple star graph.

8.2.2 Q1. Universe of Actions

An action a taken at state s basically gives us the next state s'. Since our initial state is a network G from D and our goal is to summarize it, each action a should reduce the size of the network G. As briefly indicated above, we basically define our action as a merge operation. Specifically, a merge operation on a node pair $\{a, b\}$ in the network G would result in a new *super-node* c in G'. Moreover, new edges (v, c) for all nodes v which are neighbors of either nodes being merged would be added to G'. It is important to note that merging different types of nodes is useful for different graph optimization problems. For example, for tasks like community detection, it may be more meaningful to merge nodes which have an edge connecting them to ensure that densely connected nodes are grouped together (Fig. below (c)). In contrast, for task like immunization, it may be more meaningful to merge nodes around a central node to ensure that the central node, which is more likely to be in the solution, remains central in the summary network (Fig. below (b)). Therefore, we allow merging nodes that are any distance apart. To formalize this intuition, we define a k^{th}-order merge as follows:

Definition 9 *(k^{th}-Order Node Pair Merge) A k^{th}-Order node pair merge operation merges nodes a and b into a new node c, such that $a \in V$, $b \in V$, and $d(a, b) = k$. We add new edge (c, k) for all the nodes $k \in NB(a) \cup NB(b)$.*

In a single action a, we first decide on the order k of the merge operation. Then we merge all possible node pairs, which can be merged with a k^{th}-order merge. Formally, our universe of actions is defined as follows:

Definition 10 ***Universe of Actions*** *The universe of actions A in NetGist is a set of all possible k^{th}-order merges.*

Note that our single action is a set of k^{th}-order merge operations, for a fixed order k. Hence, we usually merge more than a single pair of nodes. This ensures that our universe of actions

is broad enough to incorporate meaningful merge operations for different tasks and yet restrictive enough to be tractable for learning purposes.

Lemma 17 *k^{th}-Order node pair merge is order sensitive. (Proof omitted for the lack of space)*

Given the lemma 17 the order of taken actions is essential in generating the summaries. Hence, we should learn the sequence of actions as well as the set of actions to take.

8.2.3 Q2. Universe of States

The initial state in our RL framework is any network $G(V, E)$ drawn from D and each action the learner takes, results in a summary network $G'(V', E')$. Recall that our one action is a set of k^{th}-order merge operations. Any state s other than the initial state is a result of one or more actions on the initial state, the network G. Hence the universe of states, which includes the original network as well, is defined as follows:

Definition 11 ***Universe of States*** *The universe of states S in NetGist is a set of networks $G'(V', E')$, such that G' is a result of zero or more merges on the network G from D.*

Note that our universe of states does not include all possible summary networks of network G from D. This is because our action merges multiple nodes at once. This reduces the size of the universe of states, which in turn helps in learning. However, as discussed earlier, this set is still large enough to explore meaningful summaries across the search space.

8.2.4 Q3. Reward, Policy, Transition and the Learning Procedure

Having described the states and actions earlier, now we can describe our method to learn meaningful summary.

Reinforcement learning.

Our idea is to use deep Q-learning as it is an off-policy RL which known to be more sample efficient than the policy gradient methods [226].

We define the reward to be -1 for a state s, unless it is a terminal state. A terminal state in our case is a summary network $G'(V', E')$ which has the desired size. Formally, we define our reward function as follows:

$$r(s,a) = \begin{cases} \max\limits_{O \subseteq U} F_{prob}(O; s_{next}, \Theta) & \text{if } s_{next} \text{ is a terminal state} \\ -1 & \text{otherwise} \end{cases} \tag{8.3}$$

In Eq. 8.3 we compute $F_{prob}(O; s_{next}, \Theta)$ on the summary s_{next} with our divide and conquer framework in Alg. 9.

Our goal is to learn optimal Q-value(s, a) which estimates the cumulative reward after taking action a at state s [226] and select actions that result in the highest cumulative reward.

Learning algorithm

Next, we elaborate on the whole learning pipeline. Note that our pipeline learns the summarization procedure for graphs in D such that it is generalizable to the unseen graphs. First we define how to estimate Q-value(s, a), using the Q-learning algorithm. In order to have an end-to-end framework to learn the optimal Q-value(s, a) we combine CNNs [102] with Q-learning as often done in literature. CNN helps us to have a compact representation of each state s in the universe of states.

In our deep RL framework the input state s is fed into the CNN layer. The CNN outputs a feature representation of the state, which is then fed into a fully connected Feed Forward Network, FFN. The output layer of the FFN is d-dimensional, where d is the number of possible actions. Each i^{th} entry in the output vector represents the predicted Q-value function Q-value(s, a_i) for the state s and the action a_i.

Lemma 18 *Time complexity of NetGist is $O(Itr \cdot VE \log V)$, where Itr is the number of iterations.*

8.3 Empirical Study

We design various experiments to evaluate NetGist. Our code is available for research purposes[1]. We set $\alpha = 0.3$ for all our experiments. However, we examined the robustness of NetGist to changes in α and Θ as parameters of the TBNS problem and found it to be stable over wide ranges.

Data. We collected a number of datasets from various domains for our experiments. See Table 8.1 for details.

Baselines. We compare the performance of NetGist with other summarization techniques: (I) Random selects random node-pairs and merges them. (II) CoarseNET [186] summarizes

[1] http://github.com/SorourAmiri/NetGist

		Dataset	#Nodes	#Edges	Description
Synthetic	1	ER [31]	20-2000	50-10000	Erods-Reyni graph
	2	BA [31]	20-2000	40-4000	Barabasi-Albert preferential attachment graph
	3	Block Model [31]	20-2000	40-30000	Block models graph
Real-world	4	Karate Club	34	78	A social network among members of a karate club
	5	Work-Place [1]	92	755	mobility networks
	6	High-School [1]	327	5,818	mobility networks
	7	Political Blogs	1,490	16,783	Network of hyperlinks between web-blogs on US politics
	8	Facebook [140]	4,039	88,234	A social network
	9	As-Oregon [140]	7,000	15,743	An autonomous systems peering information network

Table 8.1: Datasets details.

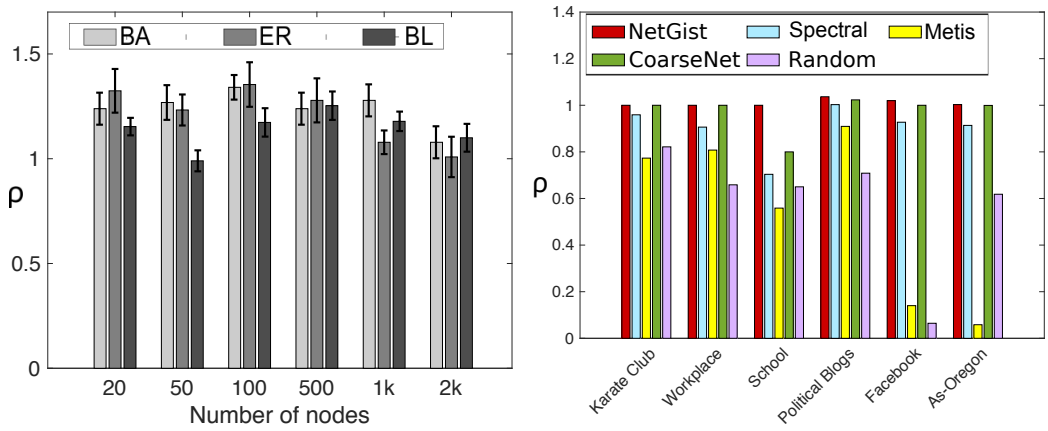
the network while ensuring that the propagation properties are preserved. (III) METIS [128] and Spectral [247] partition the network G while ensuring the summary is useful for the Community detection task. We take each partition as the super-node of the summary and connect two super-nodes if there is at least one edge between the nodes in them.

8.3.1 Q1. Quality of NetGist on distribution D

Here we evaluate the quality of NetGist on various networks and distributions and show a sample visualization.

Synthetic and real-world networks: We use popular synthetic networks (Erdos-Renyi (ER), Barabasi-Albert (BA) and Block Models) and different real datasets to evaluate NetGist on distribution D. For synthetic graphs we trained NetGist on 80 randomly sampled networks from a distribution and test on 20 previously unseen networks from the same distribution. The Avg. ρ and s.t.d. on the test with respect to Influence maximization tasks are shown in Fig. 8.4. Also, Fig. 8.4b shows the result on real-world datasets. In overall, NetGist

generalizes well and gets high ρ for all the network distributions with various sizes and even outperforms baseline methods specifically designed for the task, implying NetGist summaries can help solve GOP tasks. (Community detection task is omitted due to lack of space).



(a): Influence maximization (b): Influence maximization

Figure 8.4: (a) NetGist obtains high ρ consistently indicating that it learns well how to summarize networks for a given task. NetGist outperforms all the baselines and performs consistently for all the tasks and datasets. Note, the quality of NetGist solutions is equal or even better than the algorithms that are designed for the given task across all datasets.

Sample Visualization: Here we visualize our learnt summaries from the well-known Karate Club network (Figure 8.5) to demonstrate the difference between various tasks. The Influence maximization task-based summary consists of two super-nodes which have considerably higher degree than the rest of the network. These two super-nodes contain the optimal solution, while the rest of the super-nodes consists of homogeneous nodes with low influence. For the Community detection task as well, we see that two super-nodes in the summary highlight the homogeneous regions well (which match closely with the ground truth communities in the network). We show more such examples in the appendix.

8.3.2 Q3. Detecting anomalies on mobility graphs

Here we leverage NetGist for the application of discovering epidemiologically relevant anomalies in the Work-Place networks, which are based on a mobility log of employees in a company with five departments. The dataset has been studied before for vaccination problems [97]. Due to variation in organizational roles and mobility patterns, some nodes may play anomalous roles in any epidemic outbreak. Such nodes are difficult to mine due to the temporal nature of data and as different nodes could be anomalous at different times.

Our main idea here is to leverage the inherent uncertainty in the data (the fact that any two people interacting at a given time is probabilistic) to learn the summarization process specific

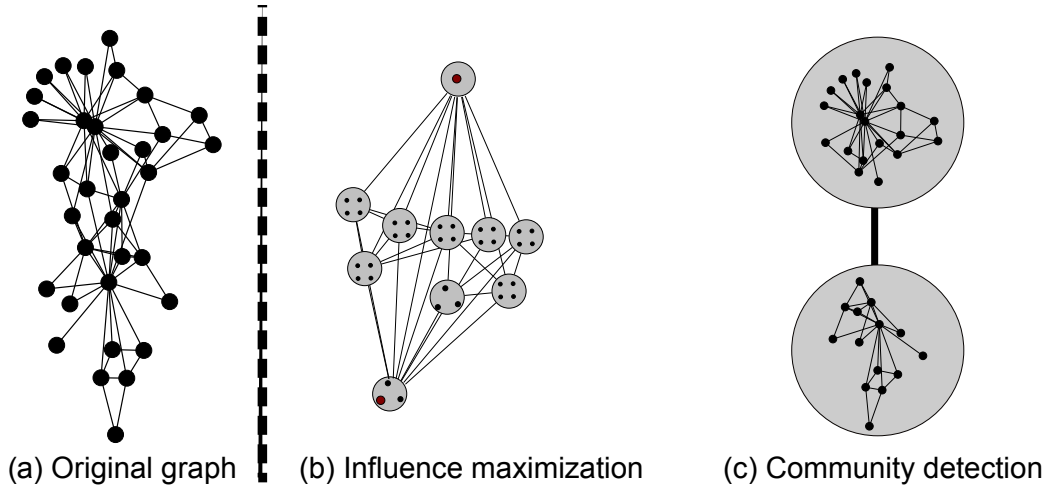


Figure 8.5: Karate Club. (a) Original graph (b and c) Summary graphs of two GOP tasks.

to the epidemiologically related influence maximization task. Once trained, we summarize the contact networks generated from the mobility log by applying the learned policy. Finally, we use the summaries to identify the anomalous nodes. Note that independently summarizing each network snapshot will not be useful as it discards the relationship between the snapshots (as we observe in our experiments as well).

NetGist runs in less than 0.5s per iteration. In Fig. 8.1, we show the learnt summary networks for three different snapshots. Color represents different departments and the size of the 'super-nodes' is proportinal to the number of nodes inside them. First note that all the departments are clearly visible in the summaries. This highlights that NetGist is able to capture the temporal stability of the departments in the dataset. Secondly, we take the singleton (unmerged) nodes connecting different departments as the anomalous nodes (shown in black). Interestingly the nodes we identify as anomalous (80, 131, 751, 222, 134) have also been labelled as the 'linkers' in [97]. We consistently find some of the linkers in most time-stamps highlighting the stability of linkers over time as discussed in [97]. On the other hand, we also find different nodes as linkers in different timestamps highlighting that different nodes play anomalous roles at different times. Hence, the visualization of the summaries helps in sense-making of the anomalous nodes and the role they play, which is not obvious from the original networks.

8.4 Related Work

We briefly discuss related work from multiple areas here.

Network Summarization. There are many summarization methods in many domains [146] for community detection problem [128], diffusion-related problem [186] and network sparsi-

fication [155]. We add a new direction to this line of work, by aiming to *learn task-based* summaries automatically instead of *designing* a different algorithm for each one separately.

Deep Learning for Graphs. Researchers have exploited deep learning for various graph mining tasks like node embedding [103], graph classification [174], and graph kernels [258] and structural data embedding [75]. As far as we know, we are the first to leverage deep learning for network summarization.

Learning to learn on Graphs. The field is seeing increasing recent interest. Methods include RNN based meta heuristic for shortest paths [34], neural networks for quadratic problems over graphs [175], reinforcement learning for the traveling salesman problem [37], RL to solve combinatorial problems [76], and so on. In contrast, we present deep reinforcement learning for network summarization in task dependent manner.

8.5 Conclusions

In this paper, we generalize over multiple threads of prior work and propose a novel Task-Based Network Summarization (TBNS) problem to automatically learn how to generate task specific network summaries. We proposed an effective method NetGist by leveraging the deep Q-learning framework. As shown by our experiments on both real and synthetic data, NetGist is able to learn meaningful summaries for various tasks and generalize them to the unseen instances. Also, NetGist helps in complex sense-making and anomaly-detection in temporal networks. We can explore using this framework for even more tasks and applications that can be transformed to GOP. Generalizing the set GOP (like to include graph alignment) and speeding up NetGist by leveraging deep network embedding would also be fruitful.

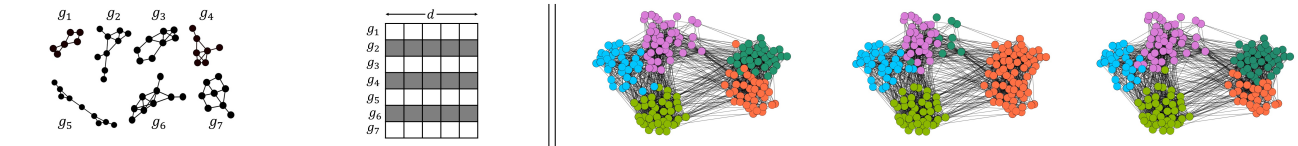
Chapter 9

Sub2Vec: Deep Representation Learning for Subgraphs

Graphs are a natural abstraction for representing relational data from multiple domains such as social networks, protein-protein interaction networks, the World Wide Web, and so on. Analysis of such networks include classification [39], detecting communities [100, 42], and so on. Many of these tasks can be solved using machine learning algorithms. Unfortunately, since most machine learning algorithms require data to be represented as features, applying them to graphs is challenging due to their high dimensionality and structure. In this context, learning discriminative feature representation of subgraphs can help in leveraging existing machine learning algorithms more widely on graph data.

Apart from classical dimensionality reduction techniques (see related work), recent works [182, 103, 248, 231] have explored various ways of learning feature representation of nodes in networks exploiting relationships to vector representations in NLP (like word2vec [161]). However, application of such methods are limited to binary and multi-class node classification and edge-prediction. It is not clear how one can exploit these methods for tasks like community detection which are inherently based on subgraphs and node embeddings result in loss of information of the subgraph structure. Embedding of subgraphs or neighborhoods themselves seem to be better suited for these tasks. Surprisingly, learning feature representation of networks themselves (subgraphs and graphs) has not gained much attention. Here, we address this gap by studying the problem of learning distributed representations of subgraphs in a low dimensional continuous vector space. Figure 9.1(a-b) gives an illustration of our framework. Given a set of subgraphs (Figure 9.1 (a)), we learn a low-dimensional feature representation of each subgraph (Figure 9.1(b)).

As shown later, the embeddings of the subgraphs enable us to apply off-the-shelf machine learning algorithms directly to solve subgraph mining tasks. For example, for community detection, we can first embed the ego-nets of each node using sub2vec, and then apply clustering algorithms like k-means on the embeddings (see Section 9.3.1 later for more details).



(a) Input subgraphs (b) Embeddings (c) Ground Truth (d) `Node2Vec` (e) `Sub2Vec`

Figure 9.1: (a) and (b) An overview of our `Sub2Vec`. Our input is a set of subgraphs $\mathcal{S}$. `Sub2Vec` learns d dimensional feature embedding of each subgraph. (c)-(e) Leveraging embeddings learned by `Sub2Vec` for community detection. (c) Communities in `School` network (different colors represent different communities). (d) Communities discovered via `Node2Vec` deviates from the ground truth, (e) while those discovered via `Sub2Vec` closely matches the ground truth.

Figure 9.1(c-e) shows a visualization of ground-truth communities in a network (c), communities found by using just node embeddings (d), and those found by our method `Sub2Vec` (e). Clearly our result matches the ground-truth well while the other is far from it. Our contributions are:

- We identify two intuitive properties of subgraphs (Neighborhood and Structural). Then we formulate two novel *Subgraph Embedding* problems, and propose `Sub2Vec`, a scalable subgraph embedding framework to learn features for arbitrary subgraphs that maintains the properties.
- We conduct multiple experiments over diverse datasets to show correctness, scalability, and utility of `Sub2Vec` in several tasks. We get upto a gain of 123.5% in community detection and upto 33.3% in graph classification compared to closest competitors.

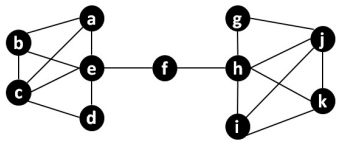
9.1 Problem Formulation

We begin with the setting of our problem. Let $G(V, E)$ be a graph where V is the vertex set and E is the associated edge-set (we assume unweighted undirected graphs here, but our framework can be easily extended to weighted and/or directed graphs as well). A graph $g_i(v_i, e_i)$ is said to be a subgraph of a larger graph $G(V, E)$ if $v_i \subseteq V$ and $e_i \subseteq E$. For simplicity, we write $g_i(v_i, e_i)$ as g_i. As input, we require a set of subgraphs $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$, typically extracted from the same graph $G(V, E)$. Our goal is to embed each subgraph $g_i \in \mathcal{S}$ into d-dimensional feature space $\mathbb{R}^d$, where $d << |V|$.

Main Idea: Intuitively, our goal is to learn a feature representation of each subgraph $g_i \in \mathcal{S}$ such that the likelihood of preserving certain *properties* of each subgraph, defined in the network setting, is maximized in the latent feature space. In this work, we provide a framework to preserve two different properties—namely *Neighborhood* and *Structural*—

properties of subgraphs.

Neighborhood Property: Intuitively, the Neighborhood property of a subgraph captures the neighborhood information within a subgraph itself for each node in it. For illustration, consider the following example. In the figure below, let g_1 be the subgraph induced by nodes $\{a, e, c, d\}$. The Neighborhood property of g_1 should be able to capture the information that the nodes a, c are in the neighborhood of node e, that nodes d, e are in the neighborhood of node c. To capture the neighborhood information of all the nodes in a given subgraph, we consider paths annotated by ids of the nodes. We refer to such paths as the *Id-paths* and define the Neighborhood property of a subgraph g_i as the set of all Id-paths in g_i.



The Id-paths capture the neighborhood information in subgraphs and each succession of nodes in Id-paths reveals how the neighborhood in the subgraph is evolving. For example in g_1 described above, the id-path $a \rightarrow c \rightarrow d$ shows that nodes a and c are neighbors of each other. Moreover, this path along with $a \rightarrow e \rightarrow d$ indicate that nodes a and d are in neighborhood of each other (despite not being direct neighbors). Hence, the set of all Id-paths captures important connectivity information of the subgraph.

Structural Property: The Structural property of a subgraph captures the *overall* structure of the subgraph as opposed to just the local connectivity information as captured by the Neighborhood property. Several prior works have leveraged degree of nodes and their neighbors to capture structural information in network representation learning [193, 217]. While degree of a node captures its local structural information within a subgraph, it fails in characterizing the similarity between the structures of two nodes in different subgraphs. Note that the nodes of two subgraphs with the same structure but of different sizes will have different degrees. For example, nodes in clique of size 10 have degree of 9, whereas nodes in clique of size 6 have degree 5. Therefore this suggests that instead, the *ratio* of degree to the size of the subgraph, of a node and its neighbors better identifies subgraph structure. Hence we rely on paths in g_i annotated by the ratio of node degrees to the subgraph size. We refer to the set of all such paths as Degree-paths. Degree-paths capture the structure by tracking how the density of edges changes in a neighborhood. While our method is simple, it is effective as shown by the results. One can build upon our framework by leveraging other techniques like rooted subgraphs [166] and predefined motifs [194].

As an example, consider the subgraph g_2 induced by nodes $\{a, b, c, e\}$ and subgraph g_3 induced by nodes $\{f, h, i, g\}$ in the graph shown above. As it is a clique, the ratio of degree to the size of the subgraph for each node in g_2 is 0.75. Hence any Degree-paths of length 3 in g_2 is $0.75 \rightarrow 0.75 \rightarrow 0.75$. Similarly, g_3 is a star and a Degree-path in g_3 from i to g is $0.25 \rightarrow 0.75 \rightarrow 0.25$. The consistent high values in the paths in cliques show that each node in the path is densely connected to the rest of the graph, while the fluctuation in values in stars show that the two spokes in the path are sparsely connected to the rest of the network while the center is densely connected. In practice, since we cannot treat each real value

distinctly, we generate labels for each node from a fixed alphabet (see Section 10.2).

Our Problems: Having defined the Neighborhood and Structural properties of subgraphs, we want to learn vector representations in $\mathbb{R}^d$, such that the likelihood of preserving these properties in the feature space is maximized. Formally the two versions of our Subgraph Embedding problem are:

Problem 11 *Given a graph $G(V, E)$, d, and set of $\mathcal{S}$ subgraphs (of G) $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$, learn an embedding function $f : g_i \rightarrow \mathbf{y_i} \in \mathbb{R}^d$ such that the Neighborhood property of each $g_i \in \mathcal{S}$ is preserved.*

Problem 12 *Given a graph $G(V, E)$, d, and set of $\mathcal{S}$ subgraphs (of G) $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$, learn an embedding function $f : g_i \rightarrow \mathbf{y_i} \in \mathbb{R}^d$ such that the Structural property of each $g_i \in \mathcal{S}$ is preserved.*

9.2 Our Methods

A common framework leveraged by most prior works in network embedding is to exploit Word2vec [161] to learn feature representation of nodes in the network. Word2vec learns similar feature representations for words which co-appear frequently in the same context. Network embedding methods, such as DeepWalk [182] and Node2vec [103], generate 'context' around each node based on random walks and embed nodes using Word2vec. These embeddings are known to preserve various node properties. However, such methods lack the global view of subgraphs, hence they are inherently unable to preserve the properties of entire subgraphs and fail in solving our problems.

9.2.1 Overview

A major challenge in solving our problems is to design an architecture which has global view of subgraphs and is able to capture similarities and differences between the properties of entire subgraphs. Our idea to overcome this challenge is to leverage the Paragraph2vec models for our subgraph embedding problems. Paragraph2vec [137] models learn latent representation of entire paragraphs while maximizing similarity between paragraphs which have similar word co-occurrences. Note that these models have the global view of entire paragraphs. Intuitively, such a model is suitable for solving Problems 11 and 12. Thus, we extend Paragraph2vec to learn subgraph embedding while preserving distance between subgraphs that have similar 'node co-occurrences'. We extend both Paragraph2vec models (PV-DBOW and PV-DM). We call our models *Distributed Bag of Nodes version of Subgraph Vector* (`Sub2Vec`-DBON) and *Distributed Memory version of Subgraph Vector* (`Sub2Vec`-DM) respectively. We discuss `Sub2Vec`-DM and `Sub2Vec`-DBON in detail in Subsections 9.2.3 and 9.2.4.

In addition, another challenge is to generate meaningful context of 'node co-occurrences' which preserve Neighborhood and Structural properties of subgraphs. We tackle this challenge by using our Id-paths and Degree-paths. As discussed earlier, Id-paths and Degree-paths capture the Neighborhood and Structural property respectively. We discuss on efficiently generating samples of Id-paths and Degree-paths next.

9.2.2 Subgraph Truncated Random Walks

Both Neighborhood and Structural properties of subgraphs require us to enlist paths in the subgraph (Id-paths for Neighborhood and Degree-paths for Structural). Since there are an unbounded number of paths, it is not feasible to enumerate all of them. Hence, we resort to random walks to generate samples of the paths efficiently. Next we describe the random walks for Id-paths and Degree-paths respectively. Note that the random walks are performed *inside* each subgraph g_i separately, and not on the entire graph G.

Id-paths: Our random walk for Id-paths in subgraph $g_i(v_i, e_i)$ starts from a node $n_1 \in v_i$ chosen uniformly at random. We choose a neighbor of n_1 uniformly at random as the next node to visit in the random walk. Specifically, if i^{th} node in the random walk is n_i, then the j^th node in the random walk is a node n_j, such that $(n_i, n_j) \in e_i$, chosen uniformly at random among such nodes.

We generate random walk of fixed length l for each subgraph g_i in the input set of subgraphs $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$. At the end of process, for each subgraph $g_i \in \mathcal{S}$, we obtain a random walk of length l, annotated by the ids of the nodes.

Degree-paths: As mentioned in Section 11.1, we generate labels for each node based on the ratio of degree of a node to the number of nodes in the subgraph for Degree-paths. We formally define the label generation for nodes next.

Let θ_a be the degree of the node n_a. Given a subgraph $g(v_i, e_i)$, we get the ratio $\gamma_a = \frac{\theta_a}{|v_i|}$ for each node $n_a \in v_i$. Note that $\gamma \in [0, 1]$. Roughly, we define various bins of range of values and associate a character with it (e.g. [0,0.2) is 'a', [0.2,0.5) is 'b' and so on). As real networks have very skewed degree distributions, we use logarithmic binning [162] to determine bin-widths. Formally, let Σ be a finite alphabet over distinct characters; and we define a many-to-one relabeling function $\tau : [0, 1] \rightarrow \Sigma$.

After generating the labels for each node, we follow the same procedure as in Id-paths, and do a random walk on each subgraph. The only difference is that we generate a sequence of characters $\alpha \in \Sigma$ for Degree-paths as opposed to node-ids for Id-paths.

9.2.3 Sub2Vec-DM

In the Sub2Vec-DM model, assuming we want to preserve Neighborhood property of subgraphs, we seek to predict a node-id u that appears in an Id-path, given other node-ids that co-occur with u in the path, and the subgraph that u belongs to. By co-occurrence, we mean that two ids co-appear in a sliding window of a fixed length w, i.e, they appear within distance w of each other. Consider a subgraph g_1 (a subgraph induced by nodes $\{a, b, c, e\}$) in the toy graph shown earlier. Suppose the random walk simulation of Id-paths in g_1 returns $a \rightarrow b \rightarrow c$, and we consider $w = 3$, then the model asks to predict node-id c given subgraph g_1, and the node's 2 predecessors (ids a and b), i.e., $\Pr(c|g_1, \{a, b\})$. Note that if our goal is to preserve the Structural property, we use Degree-paths instead.

Here we give a formal formulation of Sub2Vec-DM. Let $V = \bigcup_i v_i$ be the union of node-set of all the subgraphs. Let $\mathbf{W}_1$ be a $|\mathcal{S}| \times d$ matrix, where each row $\mathbf{W}_1(i)$ represents the embedding of a subgraph $g_i \in \mathcal{S}$. Similarly, let $\mathbf{W}_2$ be a $|V| \times d$ matrix, where each column $\mathbf{W}_2(n)$ is the vector representation of node $n \in V'$. Let the set of node-ids that appear within distance w of a node a be θ_a.

In Sub2Vec-DM, we predict a node-id a given θ_a and the subgraph g_i, from which a and θ_a are drawn. Formally, the objective of Sub2Vec-DM is to maximize:

$$\max \sum_{g_i \in \mathcal{S}} \sum_{a \in g_i} \log(\Pr(a|\mathbf{W}_2(\theta_a), \mathbf{W}_1(i))),$$

where $\Pr(a|\mathbf{W}_2(\theta_a), \mathbf{W}_1(i))$ is the probability of predicting node a given the vector representations of θ_a and g_i. It is defined using the softmax function:

$$\Pr(a|\mathbf{W}_2(\theta_a), \mathbf{W}_1(i)) = \frac{e^{\mathbf{W_3}(a) \cdot h(\mathbf{W}_2(\theta_a), \mathbf{W}_1(i))}}{\sum_{v \in V} e^{\mathbf{W_3}(v) \cdot h(\mathbf{W}_2(\theta_a), \mathbf{W}_1(g_i))}} \tag{9.1}$$

where matrix $\mathbf{W_3}$ is a softmax parameter and $h(\mathbf{x}, \mathbf{y})$ is average or concatanation of vectors $\mathbf{x}$ and $\mathbf{y}$ [137]. In practice, to compute Equation 9.1, we use negative sampling or hierarchical softmax [161].

9.2.4 Sub2Vec-DBON

In the Sub2Vec-DBON model, we want to predict a set θ of co-occurring node-ids in an Id-path sampled from subgraph g_i, given only the sugraph g_i. Note that Sub2Vec-DBON does not explicitly rely on the embeddings of node-ids as in Sub2Vec-DM. As shown in Section 9.2.3, the 'co-occurrence' means that two ids co-appear in a sliding window of a fixed length w. For example, consider the same example as in Section 9.2.3: the subgraph g_1 and the node sequence $a \rightarrow b \rightarrow c$ generated by random walks. Now in the Sub2Vec-DBON model, for $w = 3$, the goal is to predict the set $\{a, b, c\}$ given the subgraph g_1. This model

is parallel to the popular skip-gram model. The matrices $\mathbf{W}_1$ and $\mathbf{W}_2$ are the same as in Section 9.2.3.

Formally, given a subgraph g_i, and the θ drawn from g_i, the objective of `Sub2Vec`-DBON is the following:

$$\max \sum_{g_i \in \mathcal{S}} \sum_{\theta \in g_i} \log(\Pr(\theta | \mathbf{W}_1(i)), \tag{9.2}$$

where $\Pr(\theta|\mathbf{W}_1(i))$ is a softmax function, i.e.,

$$\Pr(\theta|\mathbf{W}_1(i)) = \frac{e^{\mathbf{W}_2(\theta).\mathbf{W}_1(i)}}{\sum_{\theta' \in G} e^{\mathbf{W}_2(\theta').\mathbf{W}_1(i)}},$$

Since computing Equation 9.2 involves summation over all possible sets of co-occuring nodes, we use approximation techniques such as negative sampling [161].

9.2.5 Algorithm

Algorithm 11 `Sub2Vec`

Require: Graph G, subgraph set $\mathcal{S} = \{g_1, g_2, \ldots, g_n\}$, length of the context window w, dimension d

```
walkSet = {}
for each g_i in s do
    walk = RandomWalk (g_i)
    walkSet[g_i] = walk
f = StochasticGradientDescent(walkSet, d, w)
return f
```

Our algorithm `Sub2Vec` works as follows: we first generate the samples of Id-paths/Degree-paths in each subgraph by running random walks. Then we optimize the SV-DBON/ SV-DM objectives using the stochastic gradient descent (SGD) method [47] by leveraging the random walks. We used the Gensim package for implementation [189]. The complete pseudocode is presented in Algorithm 11.

9.3 Experiments

We leverage `Sub2Vec`[1] for two applications, namely Community Detection and Graph Classification, and perform a case-study on a real subgraph data. All experiments are conducted using a 4 Xeon E7-4850 CPU with 512GB 1066Mhz RAM. We set the length of the random

[1]Code in Python available at: https://goo.gl/Ef4q8g

walk as 1000 and following literature [103], we set dimension of the embedding as 128 unless mentioned otherwise for both parameters.

9.3.1 Community Detection

Setup. Here we show how to leverage `Sub2Vec` for the well-known community detection problem [100, 42]. A community in a network is a coherent group of nodes which are densely connected among themselves and sparsely connected with the rest of the network. One expects many nodes within the same community to have similar neighborhoods. Hence, we can use `Sub2Vec` to embed subgraphs while preserving the Neighborhood property and cluster the embeddings to detect communities.

Approach. We propose to use `Sub2Vec` for community detection by embedding the surrounding neighborhood of each node. First, we extract the neighborhood C_v of each node $v \in V$ from the input graph $G(V, E)$. For each node v, we extract its neighborhood C_v only once. Hence, we get a set $\mathcal{S} = \{C_v | v \in V\}$ of $|V|$ neighborhoods are extracted from G. Since each C_v is a subgraph, we can leverage `Sub2Vec` to embed each $C_v \in \mathcal{S}$. The idea is that similar C_vs will be embedded together, which can then be clustered to detect communities. We use a clustering algorithm (K-Means) to cluster the feature vectors $f(C_v)$ of each C_v. For datasets with overlapping communities (like `Youtube`), we use the Neo-Kmeans algorithm [252] to obtain overlapping clusters. Cluster membership of $f(C_v)$ determines the community membership of node v. The complete pseudocode is in Algorithm 12.

In Algorithm 12, we define the neighborhood of each node to be its ego-network for dense networks (`School` and `Work`) and its 2-hop ego-network for others. The ego-network of a node is the subgraph induced by the node and its neighbors. The 2-hop ego-network is the subgraph induced by the node, its neighbors, and neighbors' neighbors.

Algorithm 12 Community Detection using `Sub2Vec`

Require: A network $G(V, E)$, `Sub2Vec` parameters, k number of communities
1: neighborhoodSet = {}
2: **for each** v in V **do**
3: neighborhoodSet = neighborhoodSet $\cup$ neighbordhood of v in G.
4: vecs = `Sub2Vec` (neighborhoodSet, w, d)
5: clusters = Clustering(vecs, k)
6: return clusters

Datasets. We use multiple real world datasets from multiple domains like social-interactions, co-authorship, social networks and so on of varying sizes. See Table 9.1.

Baselines. We compare `Sub2Vec` with various traditional community detection algorithms and network embedding based methods. `Newman` [100] is a well-known community detection algorithm based on betweenness. `Louvian` [42] is a popular greedy optimization method.

Table 9.1: Information on Datasets for Community Detection (Left) and Graph Classification (Right). # com denotes the number of ground truth communities in each dataset. # nodes denotes the average number of nodes in each graph classification dataset and # cl denotes the number of classes.

Dataset	$\|V\|$	$\|E\|$	# com	Domain
Work	92	757	5	contact
Cornell	195	304	5	web
School	182	2221	5	contact
Texas	187	328	5	web
Wash.	230	446	5	web
Wisc.	265	530	5	web
PolBlogs	1490	16783	2	web
Youtube	1.13M	2.97M	5000	social

Dataset	# graphs	# cl	# nodes	# labels
MUT	188	2	17.9	7
PTC	344	2	25.5	19
ENZ	600	6	32.6	3
PRT	1113	2	39.1	3
NC1	4110	2	29.8	37
NC109	4127	2	29.6	38

DeepWalk and Node2Vec are recent network embedding methods which learn feature representations of nodes in the network which we then cluster (in the same way as us) to obtain communities.

Table 9.2: Sub2Vec easily out-performs all baselines in all datasets. Average F-1 score is shown for each method. Winners have been bolded for each dataset. G stands for gain obtained by Sub2Vec in percentage.

Method	Work	School	PolBlogs	Texas	Cornell	Wash.	Wisc.	Youtube
Newman	0.32	0.34	0.58	0.17	0.33	0.21	0.16	0.04
Louvian	0.25	0.31	0.50	0.20	0.20	0.13	0.19	0.01
DeepWalk	0.40	0.48	0.80	0.25	0.32	0.29	0.29	0.15
Node2Vec	0.64	0.79	**0.86**	0.27	0.33	0.28	0.30	0.17
Sub2Vec-DM	**0.77**	**0.93**	0.85	**0.35**	**0.36**	**0.38**	**0.32**	**0.38**
Sub2Vec-DBON	0.65	0.57	0.82	**0.35**	0.34	0.37	**0.32**	0.36
G	**20.3**	**17.7**	**-1.2**	**29.6**	**9.1**	**31.0**	**6.6**	**123.5**

Results. We measure the performance of all the algorithms by computing the Average F1 score [259] against the ground-truth. See Table 9.2. Both versions of Sub2Vec significantly and consistently outperform all the baselines We achieve a significant gain of *123.5 %* over the closest competitor (Node2Vec) for Youtube. We do better than Node2Vec and DeepWalk because intuitively, we learn the feature vector of the neighborhood of each node for the community detection task; while they just do random probes of the neighborhood. Performance of Newman and Louvian is considerably poor in Youtube as these methods output non-overlapping communities. Performance of Node2Vec is satisfactory in sparse networks like Wash. and Texas. Node2Vec does slightly better ($\sim 1\%$) than Sub2Vec in PolBlogs—the network consists of homogeneous neighborhoods, which favors it. However, the performance of Node2Vec is significantly worse for dense networks like Work and School. On the other hand, performance of Sub2Vec is even more impressive in these dense networks (where the task is more challenging).

Table 9.3: Testing accuracy of graph classification. G is the % gain obtained by `Sub2Vec`.

Method	MUT	PTC	ENZ	PRT	NC1	NC109
WL-Kernel	0.80	0.56	0.27	0.72	0.80	0.80
DG-Kernel	0.82	0.60	0.53	0.71	0.80	0.80
Sub2Vec-N	0.74	0.59	**0.89**	0.92	**0.95**	0.90
Sub2Vec-S	**0.85**	**0.62**	0.85	**0.96**	**0.95**	**0.91**
G	**3.7**	**3.3**	**67.9**	**33.3**	**18.8**	**13.8**

9.3.2 Graph Classification

Setup. Here, we show an application of our method in the Graph Classification task [217, 258]. In the graph classification problem, the data consists of multiple (g_i, Y_i) tuples, where each g_i is a graph and Y_i is its class-label. Moreover, the nodes in each graph g_i are labeled. The goal is to predict the class Y_i for a given graph G_i. Since we can generate embeddings of each graph, we can train any off-the-shelf classifier to classify the graphs. In this experiment, we set the dimension of embedding as 300 and set the length of the random walk as 100000.

Approach. Our approach is to learn the embedding of each graph by treating them as a subgraph of a union of all the graphs. First we learn the feature representation of the graphs such that either the Neighborhood (`Sub2Vec`-N) or Structural (`Sub2Vec`-S) property is preserved. We then leverage four off-the-shelf classifiers: Decision Tree, Random Forest, SVM, and Multi-Layered Perceptron, to solve the classification task.

Datasets. We test on classic graph classification benchmark datasets. All the datasets are publicly available[2]. List of datasets is presented in Table 9.1.

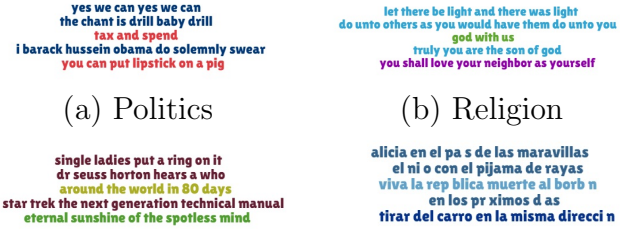
Baselines. We used two state-of-the-art methods as our competitors. `WL-Kernel` [217]: This is a graph kernel method based on the Weisfeiler-Lehman test of graph-isomorphism. `DG-Kernel` is a deep-learning version of WL-kernel [258], which relies on latent representation of sub-structures of the graphs. It uses the popular skip-gram model.

Results. We report the testing accuracy of a 5-fold cross validation. For both `Sub2Vec`-N and `Sub2Vec`-S, we run both of our models `Sub2Vec`-DM and `Sub2Vec`-DBON. We then train all four classifiers and show the best of them. See Table 9.3. The results show that both of our methods consistently outperform competitors. The gain of `Sub2Vec` over the state-of-the-art `DG-Kernel` is upto a significnt 67.9%. The better performance of `Sub2Vec`-S over `Sub2Vec`-N indicates that structural properties in these datasets are more discriminative. This is intuitive as different bonds between the elements results in different structure and also determine the chemical properties of a compound [54]. Interestingly, the Neighborhood property outperforms the Structural property in `ENZ` dataset. This suggests that, in `ENZ` dataset, which interestingly also has higher number of classes, the neighborhood property is more important than structural property in determining the graph class.

[2]http://mlcb.is.tuebingen.mpg.de/Mitarbeiter/Nino/Graphkernels/

9.3.3 Case Studies

We perform case-studies on `MemeTracker`[3] dataset to investigate if the embeddings returned by `Sub2Vec` are interpretable. Here, we run `Sub2Vec` to preserve the Neighborhood property. The `MemeTracker` consists of a series of cascades caused by memes spreading on the network of linked web pages. Each meme-cascade induces a subgraph in the underlying network. We first embed these subgraphs in a continuous vector space by leveraging `Sub2Vec`. We then cluster these vectors to explore what kind of meme cascade-graphs are grouped together and what characteristics of memes determine their similarity and distance to each other. For this case-study, we pick the top 1000 memes by volume, and cluster them into 10 clusters.



(a) Politics (b) Religion

(c) Entertainment (d) Spanish

We find coherent clusters which are meaningful groupings of memes based on topics. For example we find cluster of memes related to different topics such as entertainment, politics, religion, technology and so on. Visualization of these clusters is presented above. In the entertainment cluster, we find memes which are names of popular songs and movies such as "sweet home alabama", "Madagascar 2" and so on. Similarly, we also find a cluster of religious memes. These memes are quotes from the Bible. In the politics cluster, we find popular quotes from the 2008 presidential election season e.g. Barack Obama's popular slogan "yes we can" along with his controversial quotes like "you can put lipstick on a pig" in the cluster. Interestingly, we find that all the memes in Spanish language were clustered together. This indicates that memes in different language travel though separate websites, which matches with the reality as most webpages use one primary language.

9.4 Related Work

The network embedding problem, which seeks to generate low dimensional feature representation of nodes, has been well studied. Early work includes [36, 234, 25]. However, these methods are slow and do not scale to large networks. Recently, several deep learning based network embeddings algorithms were proposed. DeepWalk [182] and Node2Vec [103] learn feature representation based on contexts generated by random walks. SDNE [248] and LINE [231] learn feature representation of nodes while preserving first and second order proximity. Other recent works include [65, 193, 250]. However, all of them node embeddings, while our goal is to embed subgraphs.

[3] snap.stanford.edu

The most similar network embedding literature includes [194, 258, 166]. Risen and Bunke [194] propose to learn vector representations of graphs based on edit distance to a set of pre-defined prototype graphs. Yanardag et al. [258] and Narayanan et al. [166] learn vector representation of the subgraphs using the Word2Vec by generating "corpus" of subgraphs where each subgraph is treated as a word. The above works focuses on some specific subgraphs like graphlets and rooted subgraphs. None of them embed subgraphs with arbitrary structure.

9.5 Conclusions and Discussion

We focus on the embedding problem for a set of subgraphs by formulating two intuitive properties (Neighborhood and Structural). We developed `Sub2Vec`, a scalable embedding framework which gives interpretable embeddings such that these properties are preserved. We also demonstrate via detailed experiments that `Sub2Vec` outperforms traditional algorithms as well as node-level embedding algorithms in various applications, more so in challenging settings. Extending our framework to handle more properties of interest would be fruitful. For example, one may think of a 'Positional property' which relates to the *position or role* of nodes in the subgraph w.r.t. the overall graph.

Chapter 10

EpiDeep: Deep Learning for Influenza Forecasting

Seasonal influenza is a major health issue that affects many people across the world. The US national Centers for Disease Control and Prevention (CDC) reports that there were 30,453 laboratory-confirmed influenza related hospitalizations in the 2017/18 influenza season in the United States alone[1]. According to the same estimate, the 2017-18 season saw a larger number of deaths due to influenza than in any of the past five seasons. These statistics reveal that despite years of efforts, accurately predicting key indicators of the flu season and employing counter-measures remain major challenges.

FluSight task. To encourage research into making more accurate forecasts, the CDC has been hosting the 'FluSight' challenge for seasonal influenza forecasting at the national and regional levels [40, 41]. It involves predicting on a weekly basis, multiple aspects of the current influenza season, which is represented as a time series of the weighted Influenza-like Illness (wILI) data. The wILI data released by the CDC is collected by the Outpatient Influenza-like Illness Surveillance Network (ILINet) which consists of more than 3,500 outpatient healthcare providers all over the United States. Each week the healthcare providers voluntarily report the percentage of patients visiting for Influenza-like Illness (ILI). ILI is defined as "fever (temperature of 100°F [37.8°C] or greater) and a cough and/or a sore throat without a known cause other than influenza"[2]. The CDC compiles the ILI reports, weights the total percentage visits by the state population and computes the resulting wILI values for the national and local regions. It then releases this data typically with a delay of two weeks (weekly wILI incidence curves for each season since 1997/98 are publicly available[3]).

The FluSight challenge typically starts on week 40 of the calendar year and lasts till week 20 of the following year when the influenza activity is high [40, 41]. Given the wILI data, the

[1] https://www.cdc.gov/flu/about/season/flu-season-2017-2018.htm
[2] https://www.cdc.gov/flu/weekly/overview.htm
[3] https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

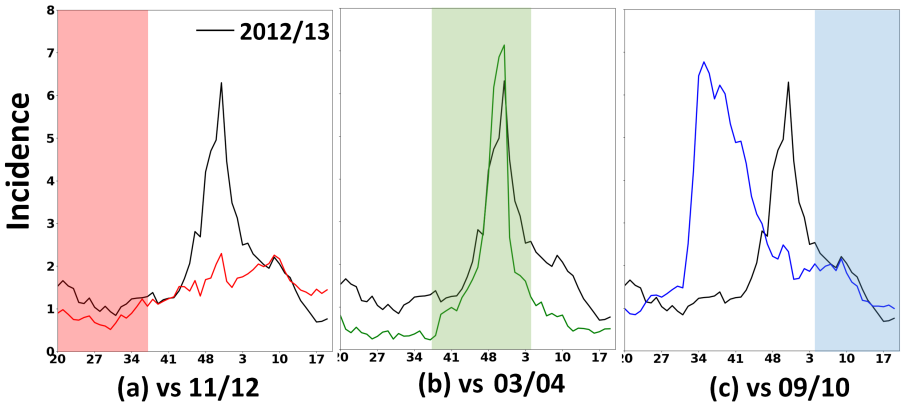


Figure 10.1: Changing similarities: wILI incidence curve for 2012/13 season in HHS Region 4 (black curve in all three figures) is most similar to that of 2011/12 season in the beginning of the season (red band), to 2003/04 in the middle of the season (green band), and finally to 2009/10 at the end of the season (blue band). Note that `EpiDeep` *automatically* learns to infer these closest seasons at various stages.

forecasting tasks include predicting the Future Incidences, Seasonal Peak Intensity, Seasonal Peak Time and Onset Week (we will discuss each forecasting target in more detail in the next section). In essence, this is a time-series forecasting problem with limited data. The same characteristics can be seen in other time-series data as well [115, 192].

Prior work. There has been a surge of research interest in developing methods for the flu forecasting tasks in recent times. Statistical methods for flu forecasting such as [51, 59] fit a predefined statistical model on historical data and use it for forecasting. These methods typically are either too simple, or fail to incorporate domain knowledge like the epidemiological dynamics, or require laborious feature engineering or are not interpretable. They often act as a "black-box" and fail in providing any interpretation or explanation of the forecasts. Note that interpretability is very important, as it provides an insight on how to fine tune the model for better predictions and also as it guides decision makers. For example, knowing that this season is similar to another one helps the policy experts to focus efforts [177, 176]: these similarities may be due to similarities in the environmental, geographical or biological (like similar strains of the virus) or other factors.

On the other hand, mechanistic models [266, 213, 176] are motivated by domain knowledge; they typically include various factors such as the epidemiological and associated human mobility models and make forecasts based on simulation and/or some simple aggregation. While insightful and usually interpretable, they require a lot of calibration, and are also usually too rigid to generalize well and accurately fit the data [177]. We can see our work in context of forecasting general time-series with limited data as well. Prior works typically rely on leveraging correlated time-series instances to overcome sparsity [115, 192]. While useful in

domains like E-commerce and real estate, where there are a large number of sparse instances (like sales for different items, housing price for each zip-code e.t.c), they do not directly apply to influenza forecasting as there is limitation in the number of instances as well. Hence, in this paper, inspired by the above significant gaps in literature, we develop an end-to-end neural learning model `EpiDeep`, which is flexible, does not require any feature engineering, and aids in interpretation while also maintaining an excellent forecasting performance.

Challenges. We focus on interpretability through finding similarities between seasons (as discussed before, knowing similar seasons is especially helpful for decision makers). This is challenging to capture, as influenza seasons are highly dynamic [176], which can be due to weather patterns, dominant virus types etc. For example, consider the wILI incidence curve for season 2012/13 for HHS Region 4 in Figure 10.1—the curve is similar to that of 2011/12 in the beginning of the season. However as the season progresses, the two curves start differing from each other. In the middle of the flu season, we find that the 2012/13 wILI curve matches closely with the 2003/04 season instead, and is closest to the much different 2009/10 season in the end. Hence traditional time-series forecasting methods like `ARIMA` are unlikely to perform well, as they do not have the flexibility to capture this dynamic nature without significant modelling input [177]. Neural networks, on the other hand, can overcome this issue as they can infer meaningful representations from the data (historical seasons) itself. However, data sparsity is also a major issue here (e.g. wILI data surveillance began only in the late 1990s). Hence sequence prediction neural models such as Long Short Term Memory (LSTM) [108], which typically require a large amount of data, are not an straightforward fit here. Therefore, we need an approach which can explicitly leverage the evolving similarities between historical seasons and yet be able to make accurate forecasts with the sparse data.

Approach and Contributions. Our main idea is that learning season similarities should help in both forecasting and interpretation. To this end, we design `EpiDeep` (Figure 10.2 summarizes the overall architecture). We feed the historical wILI incidence data to our model. It then learns to *embed* the historical seasons in a feature space and leverages the embeddings together with the current season's wILI incidence curve to forecast all the CDC targets. Given the embeddings of the seasons, we can easily answer questions like which season was the closest to the current season at the time of prediction, how does the similarity of the current season with others evolve over time, and so on. These patterns are non-trivial to find directly from the incidence curves as we may have to check all possible snippets and combinations of the historical data. Note in Figure 10.1, `EpiDeep` can automatically infer these closest seasons at various stages.

Our main contributions are as follows:

- **A Novel Deep Learning Approach:** We leverage recent development in deep clustering to design an end-to-end time-series embedding, clustering and forecasting approach. To the best of our knowledge, we are the first to do so.

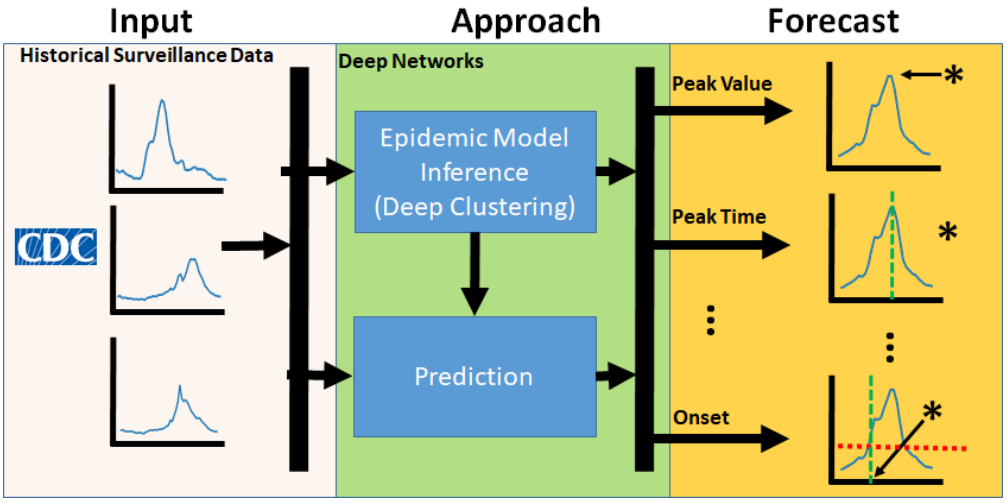


Figure 10.2: A visual representation of our approach. We leverage historical surveillance data to learn meaningful representations via evolving clustering and forecast multiple metrics of interest.

- **Interpretability with Performance:** Our learnt embeddings help in determining the closest historical seasons to the current season at the time of prediction, helping in both interpretability and performance. To the best of our knowledge, we are the first to propose a non-trivial deep learning model for epidemic forecasting which ensures both interpretability and excellent forecasting performance.
- **Rigorous Empirical studies:** We conduct extensive experiments in multiple settings using real ILI data. Our results show that `EpiDeep` outperforms non-trivial baselines by up to 40%, consistently across multiple metrics, and is also interpretable.

Next, we discuss the CDC challenge in detail and formulate the problems. We then describe `EpiDeep` in Section 10.2, discuss empirical studies in Section 10.3, related literature in Section 10.4 and finally conclude in Section 11.5. We defer some additional experiments and details to the supplementary for sake of space and readability.

10.1 Problem Statement

The CDC FluSight challenge provides a uniform prediction goal for influenza forecasting and hence we model our problem based on that task. As mentioned earlier, the historical wILI incidence reports are available till week $t-1$ while making predictions for week t. There are four forecasting targets: Future Incidence, Seasonal Peak Time, Seasonal Peak Intensity,

and Onset Week. All the target descriptions are based on the CDC FluSight 2018/2019 challenge[4].

Future Incidence: This refers to short term future incidence predictions. This includes the prediction of wILI data one to four weeks ahead of the latest wILI data release. Since the ILINet data is delayed by two weeks, at week t in the season, the short term forecasts correspond to predicting wILI values for week $t-1$, t, $t+1$, and $t+2$, given the data till week $t-2$.

Seasonal Peak Intensity: The peak intensity measures the maximum intensity of the influenza in the given season (i.e. the highest numerical value of wILI in the given season). Since the amount of resources needed to influenza prevention is directly affected by the peak intensity, it is an important task.

Seasonal Peak Week: The peak week is the time when the peak intensity is observed. The CDC defines the seasonal peak week as the surveillance week when the wILI value rounded to one decimal point is the highest. The peak week prediction is an important problem as it allows for planning ahead and strategic resource allocation.

Onset Week: The onset week represents the week when the flu season 'takes-off'. The CDC defines it as the surveillance week when percentage of visits for influenza-like illness (ILI) reaches or exceeds a pre-defined baseline value for three consecutive weeks. The onset week is the first week of such three weeks. The baseline value may vary from year to year and from region to region. The onset week prediction is an important as the start of the flu season determines when many precautionary and preventive measures are deployed. It also gives healthcare providers an early notification that a rise in ILI cases is impending.

At week $t+2$, we are given the time-series $\mathcal{Y}_c = \{y_c^1, y_c^2, \ldots, y_c^t\}$ representing initial stage of the current season c till week t. The values y_c^i represent the wILI values for the week i. Our goal is, given $\mathcal{Y}_c$, predict all four targets for the season c: short term forecasts, peak intensity, peak week, and onset. Formally:

Problem 13 EpidemicPrediction
Given*: a time-series $\mathcal{Y}_c = \{y_c^1, y_c^2, \ldots, y_c^t\}$ representing the current season c till week t and a CDC baseline b_c*
Predict:

- ***Task 1:*** *Future Incidence Prediction:* $\forall_{i=t+1}^{t+4} y_c^i$
- ***Task 2:*** *Peak Intensity Prediction:* $\max y_c^i \forall_{i=1}^{T}$*, where T is the last week of the season.*
- ***Task 3:*** *Peak Time Prediction:* $\arg\max_i y_c^i \forall_{i=1}^{T}$*, where T is the last week of the season.*
- ***Task 4:*** *Onset Prediction: Week j such that* $\forall_{i=j}^{j+3}, y_c^i \geq b_c$

[4]https://predict.cdc.gov/post/5ba1504e5619f003acb7e18f

10.2 Our Approach

Overview. We first give an overview of `EpiDeep`. Here, we are given historical wILI time-series (the 'training set'), using which we propose a learning based approach to solve Problem 13. Specifically, we design a deep neural network to encode various aspects of our data. At a high level, our model tries to leverage similarities between the observed stage of the current season with the past seasons to make accurate future predictions (using the future observed trends of the past seasons). We train the deep model by leveraging a set of historical wILI incidence $\mathrm{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, \mathcal{Y}_3, \ldots, \mathcal{Y}_{c-1}\}$. Once the model is trained, we use it to predict various metrics of interest for the current season $\mathcal{Y}_c$. Following literature [51], we define each season $\mathcal{Y}_i$ to begin at week 20 of the calendar year.

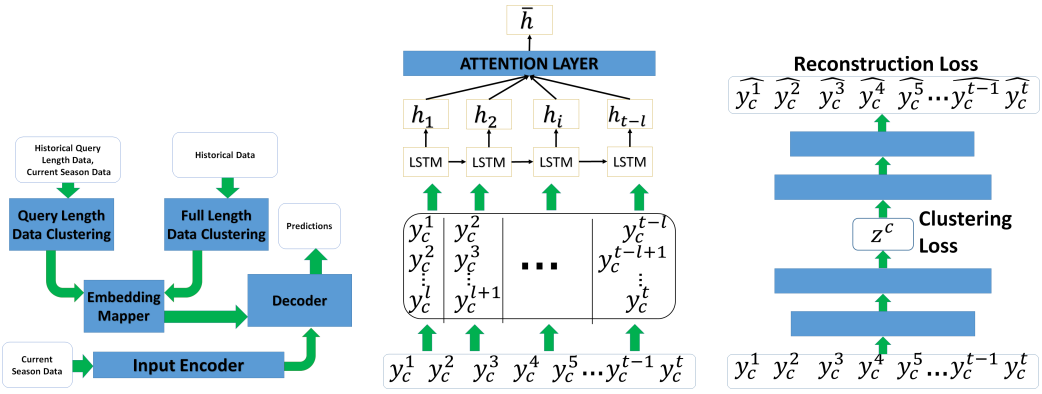
The main challenge in implementing our idea of 'evolving similarities for prediction' is that the current season $\mathcal{Y}_c$ is observed only till week t, whereas the historical seasons in Y are fully observed. Hence, naïve ideas like computing distance between the curves are not suitable here. We resolve this issue in two steps. First, we learn the embeddings to capture similarities between the current season and historical seasons restricted till time t. However, this is not enough as the entire season-length historical data will have useful information to aid in forecasting of the current season. Hence after that, we also find embeddings to capture similarities between the full length historical seasons *without* the current season. We finally learn a mapping function to map between these representations and then further leverage the embeddings for the forecasting tasks. Figure 10.3 (a) shows the overall architecture of `EpiDeep`. Next, we discuss these steps in detail.

Our first step is accomplished by the 'Query Length Data Clustering' module, which leverages deep clustering techniques to learn feature representations. Here, we treat the observed part of the current season $\mathcal{Y}_c$ (for which we need to forecast) as a 'query' to the historical fully observed season-length incidence curves. We refer to the snippets of the historical season till the observed week t as query length historical wILI data. This module learns the feature representation of query length historical wILI data with the current season $\mathcal{Y}_c$. To capture the similarity between the current season and historical seasons, this module learns embeddings for each season in Y in conjunction with $\mathcal{Y}_c$ and clusters them by leveraging a deep clustering method.

Similarly for the second step, the 'Full length Data clustering' component embeds the full length historical data in a continuous space, such that the clustering of the embeddings are meaningful. Since, we do not have access to the complete incidence curve for the current season $\mathcal{Y}_c$, we learn a mapping function to convert the embeddings from the query length space to the space representing complete seasons. This allows us to learn embeddings of the current season in the space of fully observed historical wILI data.

We also have the 'Input Encoder' module, which provides a succinct representation of the current season $\mathcal{Y}_c$. The input encoder is designed in a way such that it can extract important information from any snippets in $\mathcal{Y}_c$. Finally, the 'Decoder' module combines outputs from

the clustering layers and the encoder to make prediction for the final forecasting targets. In the following, we describe the various components of our model.



(a) (b) (c)

Figure 10.3: (a) The overall architecture of `EpiDeep`. It consists of clustering/embedding, encoder, and decoder modules. (b) The architecture of the encoder module. (c) The architecture of the deep clustering module.

10.2.1 Encoding the Input

Here we adopt a recurrent neural network (RNN) to capture temporal dynamics of of the time-series $\mathcal{Y}_c = \{y_c^1, y_c^2, \ldots, y_c^t\}$. RNN processes variable length input sequence by maintaining a hidden state $h_j \in \mathrm{R}^K$ and continuously updating it as $h_j = f(y_c^j, h_{j-1})$, where f is a non-linear activation function. Here, we leverage Long short-term memory networks (LSTM network) [98], a special type of RNN designed to work better with long sequences of data. Given a sequence, $\mathcal{Y}_c = \{y_c^1, y_c^2, \ldots, y_c^t\}$, we first convert it to a matrix $\mathbf{Y} \in \mathrm{R}^{l \times (t-l+1)}$, such that j^{th} column of Y consists of $[y_c^j, y_c^{j+1}, \ldots, y_c^{j+l-1}]$. Now, the j^{th} input to our LSTM network is the j^{th} column of the matrix $\mathbf{Y}$, i.e. $\mathbf{Y}[:j]$. Now, the LSTM equations for j^{th} input are as follows:

$$i_j = \sigma(\mathbf{W}_i \mathbf{Y}[:j] + \mathbf{U}_i \mathbf{h}_{i-1} + \mathbf{b}i) \tag{10.1}$$

$$f_j = \sigma(\mathbf{W}_f \mathbf{Y}[:j] + \mathbf{U}_f \mathbf{h}_{i-1} + \mathbf{b}f) \tag{10.2}$$

$$C_j = i_j \odot \tanh(\mathbf{W}_c \mathbf{Y}[:j] + \mathbf{U}_c \mathbf{h}_{i-1} + \mathbf{b}c) + f_j \odot C_{j-1} \tag{10.3}$$

$$\mathbf{o}_j = \sigma(\mathbf{W}_o \mathbf{Y}[:j] + \mathbf{U}_o \mathbf{h}_{i-1} + \mathbf{b}o) \tag{10.4}$$

$$\mathbf{h}_j = o_j \odot \tanh(C_j) \tag{10.5}$$

Here, the matrices $\mathbf{W} \in \mathrm{R}^{\mathrm{h} \times \mathrm{l}}$ and $\mathbf{U} \in \mathrm{R}^{\mathrm{h} \times \mathrm{h}}$ are the weight matrices and h represents the size of hidden units.

Typically, only the last output of the LSTM is leveraged for the prediction. For our query series $\mathcal{Y}_c = \{y_c^1, y_c^2, \ldots, y_c^t\}$, this would mean only $\mathbf{o}_t$ would be leveraged for prediction.

However, such an approach has a clear disadvantage. It is known that the official estimates for ILI surveillance data are often delayed and revised multiple times before they stabilize [59]. In such scenario, the data pertaining to the final time-stamp y_c^t is most vulnerable to revision. Therefore, over reliance on the last time-stamp could harm the predictive power of the model. To overcome this issue, we require a mechanism to assign varying degree of importance to the earlier states of the model.

A mechanism typically used in NLP to give partial importance to each state of the RNN is known as the *attention mechanism* [152]. The main idea here is to produce the output of the RNN as a weighted sum of all previous hidden states. The context vector $\bar{\mathbf{h}}_j$, for j^{th} input then is computed as $\bar{\mathbf{h}}_j = \sum_z \alpha_{jz}^a \mathbf{h}_{jz}$, where

$$\alpha_{js}^a = \frac{\exp(u_{jx}^T u_\alpha)}{\sum_z \exp(u_{zx}^T u_a)} \tag{10.6}$$

$$u_{js} = \tanh(\mathbf{W}_a^T \mathbf{h}_{js} + \mathbf{b}_a) \tag{10.7}$$

Since, the context vector $\bar{\mathbf{h}}_j$ is the weighted sum of previous hidden states, the model can infer variable weights for each hidden state and has the flexibility to put more/less attention to each input. The context vector $\bar{\mathbf{h}}_j$ can now be fed into the decoder network to make predictions.

The complete encoder module is shown in Figure 10.3 (b). As mentioned earlier, the input time series $\mathcal{Y}_c$ is converted to the matrix Y. Each column of the matrix is fed into the LSTM network. The output of the LSTM network is combined using the attention mechanism to learn a single context vector $\bar{h}_j$.

10.2.2 Closest Season based on Deep Clustering

A simple way to train our model is to feed the context vector $\bar{\mathbf{h}}_j$ from the attention model to the decoder. The assumption here is that the deep model would be able to infer relevant information from the historical incidence time-series Y and leverage it for correct prediction. However, due to the sparsity of the data available, a more explicit model to extract similarities between season is warranted. Our main idea is to learn the embedding $\mathbf{z}_c$ of the partially observed current season $\mathcal{Y}_c$ in the latent space such that the distance between z_c and other 'similar' historical season is minimized. To this end, we develop an embedding layer with multiple deep components.

Clustering Query Length Data: Here we are concerned with embedding the similar seasons such that seasons which are closer to each other are embedded together. Recall that in the current season $\mathcal{Y}_c = \{y_c^1, y_c^2, \ldots, y_c^t\}$, we have only observed incidence till time t. Hence, we leverage deep clustering [256] of the set $\mathrm{Y}_t = \{\mathcal{Y}_i[0:t] | \forall_{i=1}^{c-1}\} \cup \{\mathcal{Y}_c\}$ to learn meaningful embedding of $\mathcal{Y}_c$. Here we adopt the Improved Deep Embedded clustering (IDEC) [106]

method to cluster Y_t and to learn embeddings. IDEC clusters given input by augmenting clustering loss to the reconstruction loss.

Let, the vector $\mathbf{z}_i^t$ be the encoding of the the season $\mathcal{Y}_i$ given by the encoder. Let, μ_j be the cluster center for cluster j. Then the clustering objective in IDEC is given by the following:

$$L_c^t = KL(P||Q) = \sum_i \sum_j p_{ij} \log \left(\frac{p_{ij}}{q_{ij}} \right) \tag{10.8}$$

where q_{ij} is the similarity between the embedding $\mathbf{z}_i$ and cluster center μ_j as given the the Student's t-distribution, i.e.

$$q_{ij} = \frac{(1 + ||\mathbf{z}_i^t - \mu_j||^2)^{-}1}{\sum_j (1 + ||\mathbf{z}_i^t - \mu_j||^2)^{-}1} \tag{10.9}$$

and p_ij is the target distribution given by

$$p_{ij} = \frac{q_{ij}^2 / \sum_i q_{ij}}{\sum_j (q_{ij}^2 / \sum_i q_{ij})} \tag{10.10}$$

In addition to the clustering loss L_c parameterized by the target distribution, IDEC also minimizes the reconstruction loss L_r^t, as a regular auto-encoder, to preserve the local structure of the data.

The embedding z_c^t generated by the deep clustering network only tries to capture the similarities between the historical season Y and the current season $\mathcal{Y}_c$ only till time t, as the current season is observed only till time t.

Clustering Full Length Data: Once trained, the vectors $\mathbf{z}^t$ give us meaningful embedding of the set Y_t. However, our main goal is to infer how similar the seasons would look like at the end of the season rather than at the prediction time t. In other words, given the current season $\mathcal{Y}_c$ at time t, can we learn embedding of the $\mathcal{Y}_c$ in the space of Y rather than Y_t. Since, we have not observed $\mathcal{Y}_C$ entirely, we are unable to leverage clustering to embed it together with Y.

To avert this issue, our idea is to leverage IDEC, as earlier, to cluster and embed the historical seasons in Y which are all observed entirely till time T. To this end, we optimize for the loss L_c^T (in the same manner as in Equation 10.8) and the reconstruction loss L_r^T for the historical seasons Y. Hence, for each season $y_i \in$ Y, we obtain a full-length embedding $\mathbf{z}_i^T$.

Mapping the Embeddings: Now, our problem reduces to translating the embedding learned from query length data to the space of full length data, i.e, mapping $\mathbf{z}_c^t$ to $\mathbf{z}_c^T$. To this end, we learn the mapping function f_{emd} to map z_i^t to z_i^T. Our idea is to leverage,$\mathbf{z}_i^t$ and $\mathbf{z}_i^T$ for each historical season in Y to learn the function f_{emd}. To this end, we optimize the objective $L_{emd} = \sum_i ||z_i^T - f_{emd}(z_i^t)||_2^2$

Here, $f_{emd}(z_i^t)$ is the translation of z_i^t to the space of z_i^T. We represent the function f_{emd} as a feed-forward neural network. Once the complete network is trained, we obtain $z_c^T = f_{emd}(z_c^t)$ as our embedding for the current season $\mathcal{Y}_c$.

10.2.3 Prediction

The next component of our Deep Model `EpiDeep`, leverages the encoding $\bar{\mathbf{h}}_j$ of the input $\mathcal{Y}_c$ and the embedding z_c^T to predict the metrics of interest. As explained earlier, we focus on four types of predictions, namely *future incidence*, *peak time*, *peak intensity*, and *onset*. We train our model for both point and binning-based probabilistic predictions. Let us first focus on the point predictions.

Task 1: Future Incidence Prediction: Here, given the current season $\mathcal{Y}_c = \{y_c^1, y_c^2, \ldots, y_c^t\}$, the goal is to predict y_c^i for $i \in \{t+1, t+2, t+3, t+4\}$. For simplicity we explain the training process for y_c^{t+1}. Our goal here is to learn function f_{next} that maps the encoding learned to the output $\hat{y} \in \mathrm{R}$, i.e. $\hat{y} = f_{next}(\bar{\mathbf{h}}_j, z_c^T)$.

We represent f_{next} as a feed-forward neural network. We train the network by leveraging the historical data Y. Hence, our objective function becomes $L_{pred} = \sum_{k \in \mathrm{Y}} ||y_k^{t+1} - \hat{y}||_2^2$.

Task 2: Peak Intensity Prediction: Here the approach is similar to future incidence intensity prediction. Instead of training to predict the next incidence, we train the network to directly predict the peak intensity.

Task 3: Peak Time Prediction: As in previous metrics, the goal here is to leverage the encoding to predict peak time (in weeks). Here, the prediction process is slightly different. We have $x_t = \mathbf{W} f_{next}(\bar{\mathbf{h}}_j, z_c^T)$ and

$$P(t|x_t) = \frac{\exp(x_t)}{\sum_i \exp(x_i)}, \tag{10.11}$$

where $\mathbf{W}$ is a weight matrix and $f_{next}(\cdot)$ is represented as a feed-forward neural network. The term $P(t|x_t)$ represents the probability that the peak occurs at time t. As shown above, we use the softmax function to compute the probability. Finally the peak time is given by $\hat{t} = \arg\max P(t|x_t)$. As above, historical data Y is leveraged for the training. Here we adopt the cross-entropy as the objective function L_{pred}.

Task 4: Onset Prediction: We adopt similar approach as Peak Time prediction for the onset prediction.

The overall objective is as follows:

$$\theta^* = \arg\min_{\theta} L_{emd} + L_c^t + L_r^t + L_c^T + L_r^T + L_{pred} \tag{10.12}$$

Note that the prediction loss L_{pred} is different for each task as mentioned above. We train separate networks for each task. We start by pre-training the clustering and mapping layers first and then jointly training the entire model. The adaptive moment estimation (Adam) optimization algorithm [131] was used to infer the model parameters. The model was coded using the automated differentiation package in PyTorch[5].

Note: We train for the CDC binning-based probabilistic predictions in a similar fashion. Instead of predicting point estimates, we assign probabilities to each bin pre-defined by the CDC.

10.3 Empirical Study

Table 10.1: `EpiDeep` consistently performs well across all the tasks, outperforming all the methods in majority of the scenarios. Comparison of performance of all the methods for all the four tasks for seasons starting from 2010/11 till 2016/17. **R** is RMSE, **M** is MAPE and **LS** is the average Log-Score. A "–" means that the method can not be used for that prediction. For the 2011/12 season, the national wILI incidence curve did not cross the baseline, so there was no onset & we mark the cells with "×" signs.

	10/11			11/12			12/13			13/14			14/15			15/16			16/17		
Method	**R**	**M**	**LS**	**R**	**M**	**LS**	**R**	**M**	**LS**	**R**	**M**	**LS**	**R**	**M**	**LS**	**R**	**M**	**LS**	**R**	**M**	**LS**
Task 1: Future Incidence Prediction																					
Hist	1.29	0.4	39.82	0.44	0.21	49.57	1.4	0.36	46.92	0.77	0.23	54.88	1.12	0.26	61.96	0.65	0.23	39.0	0.9	0.22	53.12
ARIMA	0.65	**0.15**	–	**0.28**	**0.12**	–	0.89	**0.17**	–	0.65	**0.12**	–	0.88	0.17	–	0.42	0.13	–	0.67	0.15	–
KNN	0.76	0.26	57.32	1.57	0.71	75.11	0.81	0.24	75.97	1.06	0.37	76.86	**0.61**	0.24	75.1	0.98	0.36	80.41	0.65	0.2	77.75
LSTM	0.92	0.36	40.12	0.72	0.32	58.41	0.93	0.32	54.46	1.25	0.40	79.76	0.82	0.19	64.45	0.78	0.33	56.97	0.98	0.31	72.06
EB	0.81	0.31	43.29	0.97	0.5	60.11	1.04	0.24	65.42	0.67	0.24	58.37	0.87	0.21	61.8	0.93	0.39	47.79	1.06	0.32	56.61
EpiDeep	**0.59**	0.17	**26.61**	0.36	0.16	**32.2**	**0.68**	**0.17**	**29.89**	**0.45**	**0.12**	**36.03**	0.73	**0.15**	**41.01**	**0.41**	**0.13**	**29.75**	**0.58**	**0.15**	**35.15**
Task 2: Peak Intensity Prediction																					
Hist	1.39	0.31	∞	1.0	0.42	∞	2.55	0.42	∞	0.78	0.17	1.18	1.94	0.32	1.2	0.78	0.22	0.93	**0.54**	**0.11**	∞
ARIMA	2.47	0.52	–	0.64	0.25	–	4.01	0.65	–	2.76	0.6	–	3.93	0.65	–	1.55	0.41	–	2.82	0.54	–
KNN	1.31	0.28	∞	3.28	1.35	∞	0.61	0.1	∞	0.9	0.19	32.47	**0.18**	**0.03**	∞	1.49	0.4	53.73	0.58	0.09	∞
LSTM	1.43	0.32	89.91	1.35	0.56	77.84	1.94	0.51	56.92	0.84	0.17	22.41	2.83	0.42	0.94	1.42	0.37	43.77	1.98	0.38	95.41
EB	**0.96**	0.21	**60.73**	**1.21**	**0.48**	82.16	2.48	0.41	**42.86**	1.1	0.24	0.46	2.3	0.38	0.45	**0.24**	**0.06**	**11.24**	1.41	0.28	64.29
EpiDeep	0.99	**0.2**	71.4	1.59	0.6	**71.03**	**1.36**	**0.21**	71.43	**0.56**	**0.1**	**0.37**	2.26	0.37	**0.34**	0.46	0.11	18.64	0.87	0.15	**43.9**
Task 3: Peak Time Prediction																					
Hist	17.0	0.3	∞	21.0	0.33	∞	8.0	0.15	0.67	6.0	0.12	0.57	5.0	0.1	0.51	13.0	0.21	∞	7.0	0.12	0.95
ARIMA	33.53	0.58	–	37.1	0.58	–	27.3	0.51	–	27.08	0.51	–	26.93	0.5	–	38.69	0.62	–	34.8	0.59	–
KNN	12.0	0.21	∞	5.9	0.09	∞	16.82	0.32	0.47	16.82	0.32	0.33	11.0	0.21	0.14	**6.6**	**0.1**	∞	10.17	0.17	∞
LSTM	7.09	0.14	64.13	5.6	0.08	81.32	9.35	0.24	1.48	9.73	0.22	0.29	19.24	0.41	21.25	11.45	0.23	50.44	8.85	0.32	88.4
EB	1.09	**0.02**	60.7	**4.5**	**0.07**	78.6	**5.4**	**0.1**	**0.3**	**1.0**	**0.02**	**0.2**	**1.4**	**0.02**	**0.2**	8.04	0.11	75.0	**6.3**	**0.1**	64.2
EpiDeep	**1.0**	**0.02**	**33.2**	5.1	0.08	**29.2**	6.0	0.12	0.33	6.0	0.12	0.26	3.71	0.05	0.28	10.3	0.16	**29.6**	6.65	0.11	**21.6**
Task 4: Onset Prediction																					
Hist	23.0	0.43	∞	×	×	×	15.0	0.29	∞	12.0	0.24	0.57	8.0	0.16	∞	14.0	0.25	∞	6.0	0.12	∞
ARIMA	51.21	0.96	–	×	×	×	48.66	0.94	–	49.18	0.98	–	47.96	0.97	–	55.89	0.97	–	51.5	0.98	–
KNN	17.19	0.32	60.9	×	×	×	21.63	0.42	∞	23.0	0.46	0.16	19.0	0.39	∞	15.13	0.26	∞	19.96	0.38	60.9
LSTM	6.71	0.28	92.4	×	×	×	5.41	0.64	58.43	17.40	0.33	21.48	11.22	0.24	56.58	9.56	0.15	56.6	8.28	0.29	63.79
EB	**2.38**	**0.04**	∞	×	×	×	2.35	0.05	64.63	3.59	0.07	68.03	3.0	0.06	∞	8.04	**0.13**	47.01	3.67	0.07	61.15
EpiDeep	4.0	0.08	**31.99**	×	×	×	**2.0**	**0.04**	**17.47**	**0.88**	**0.02**	0.0	**0.35**	**0.0**	**24.33**	**7.76**	**0.13**	**40.28**	**2.92**	**0.06**	24.66

[5]https://pytorch.org/docs/stable/autograd.html

10.3.1 Setup

We describe our experimental setup next. All experiments are conducted using a 4 Xeon E7-4850 CPU with 512GB of 1066 Mhz main memory. The code is available for academic purposes[6].

Data: We used wILI surveillance data collected and released by the CDC. The data is publicly available[7]. Following the guidelines provided by the CDC, we measure success of the methods on four major tasks as described earlier. For each prediction reported, only the historical data observed prior to the time of prediction is leveraged.

Baselines: While several methods exist for influenza epidemic forecasting, most of them require additional data such as twitter feeds, weather data, and so on. In contrast, `EpiDeep` forecasts given only the historical wILI data. Hence, we compare our performance against many non-trivial baselines which can forecast given only the wILI data.

- `Hist`: It is inspired by the traditional approach for flu forecasting. Here, we compute historical average of all previous seasons and make predictions using the average.
- `ARIMA` is a popular auto-regressive method typically used for prediction on time-series data. Here we leveraged `ARIMA` (7,0,1) as it performed the best.
- `KNN`: Here, we select the top k closest historical seasons to the current season and make predictions based on the average. Many model based approaches for flu forecasting [176] leverage a similar strategy of utilizing the closest historical season.
- `LSTM`: We leverage Long Short Term Memory network for forecasting. Note that it is a version of [245] without climate and geographical data and can be considered a simpler version of `EpiDeep` without the embeddings and attention.
- `EB` is an empirical bayes framework for epidemic prediction [51]. In this approach, translation of a historical season is fitted to the observed part of the current season to make a prediction. It is the publicly available version of the approach which has won several of the past iterations of the FluSight challenge.

Evaluation Metrics: There has been much discussion on the correct metric for evaluating models for epidemic predictions [228]. Hence we utilize the following multiple metrics to evaluate the predictive power of all the methods:

- **RMSE:** The root mean squared error is the square root of the average squared error i.e. RMSE $= \sqrt{\frac{\sum_{i=1}^{N} e_i^2}{N}}$.
- **MAPE:** The mean absolute percentage error measures the average of absolute percentage error, i.e. MAPE $= \frac{1}{N}\sum_{i=1}^{N}\left|\frac{e_i}{y_i}\right|$.

[6]http://people.cs.vt.edu/ bijaya/codes/EpiDeep.zip
[7]https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

- **Log Score:** For the probabilistic predictions, we leverage the logarithmic scoring rule leveraged by the CDC [200]. The performance is measured by the negative log score, defined as $\log(p, i) = -\log(p_i)$ for the probability assigned in the bin i (containing the ground-truth).

10.3.2 National Predictions

Here, we compare the performance of all the methods for all four tasks at the United States national level starting from 2010/11 till 2016/17. For the Future Incidence Prediction task, we ran all the methods to predict future wILI values starting from epidemiological week 40, when the flu season typically starts, till epidemiological week 20, when the season ends. For Peak (Intensity and Time) and Onset Prediction tasks, we predicted the metric starting from week 20 until it was observed for each season. The results for each method is summarized in Table 10.1. Since `ARIMA` does not produce probabilistic predictions, we were unable to compute the log-score.

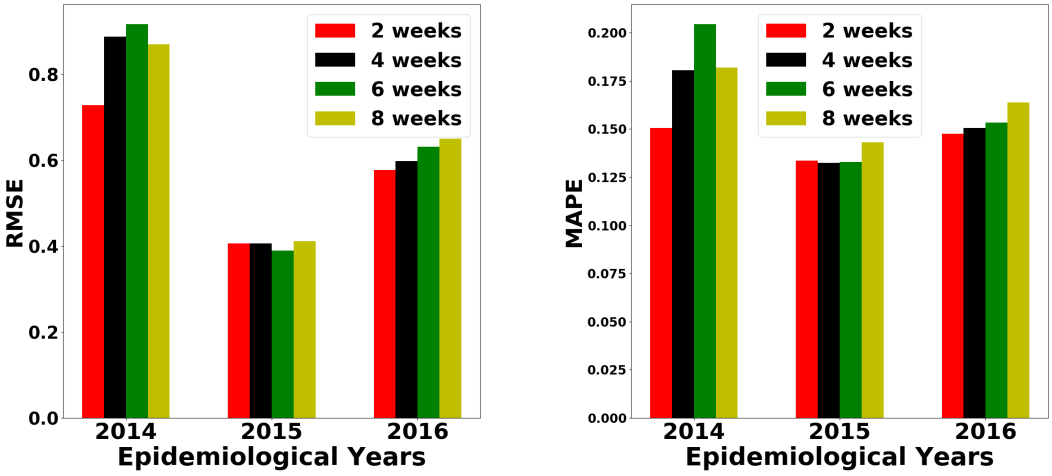
As shown in the table, `EpiDeep` outperforms all the baselines in the majority of the settings. It actually outperforms non-trivial baseline `EB` in three of the four tasks, namely Peak Intensity, Onset, and Future wILI prediction by an impressive margin of 16%, 14%, and 40% on average in terms of RMSE. This is partly due to the fact that `EB` is constrained by a rigid base function, whereas `EpiDeep` is not. Overall, `EpiDeep` outperforms all the baselines in 17 out of 21 measures for Future wILI prediction, in 10 out of 21 measures for Peak Intensity Prediction, in 7 of 21 measures for Peak Time Prediction, and in 16 of the 21 measurements for Onset Prediction. It is a close second/third in the rest. Simpler baselines such as `Hist`, `ARIMA`, and `KNN` have reasonably satisfactory and stable performance for the Future Incidence Prediction task. However, the performance of these methods are at two different extremes for all other tasks. On the other hand, `LSTM`, `EB`, and `EpiDeep` have a stable performance across all tasks. Note that, `EpiDeep` outperforms `LSTM` in almost all measurements. This is due to the fact that `EpiDeep` has the flexibility of `LSTM` in addition to the meaningful embeddings which it can exploit for accurate forecasting.

10.3.3 Delayed Data Arrival

As mentioned in Section 11, the ILINet data has a delay of about 2 weeks. An interesting question is how the performance of `EpiDeep` varies with the delay. Will the performance of `EpiDeep` vastly vary if the delay is increased significantly? Will it remain stable?

To answer these questions, we performed experiments with simulated larger delays. Specifically, we leveraged `EpiDeep` to forecast future wILI incidence with delay of 2, 4, 6, and 8 weeks. We repeated the experiments for three seasons, namely 2014/15, 2015/16, and 2016/17. Since peak (time/intensity) predictions and onset prediction already have a large

gap between the time when prediction is made and the time when the data is observed, we conducted this study only for future incidence prediction for the national data.



(a) RMSE (b) MAPE

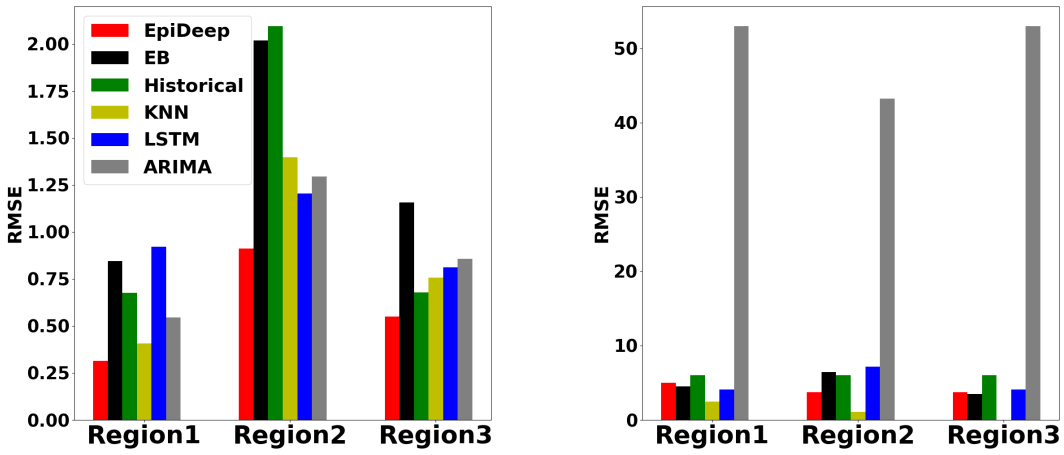
Figure 10.4: RMSE and MAPE for future wILI incidence predictions for delayed data arrival. `EpiDeep`'s performance remains stable even when data is delayed by up to 8 weeks.

The result is summarized in Figure 10.4. As shown in figure, there is minimal change in performance of `EpiDeep` even when the data is delayed by up to 8 weeks. It highlights the fact that `EpiDeep` makes stable predictions even in scenarios where there is a bigger delay in data arrival.

10.3.4 Regional Forecasting

The US Department of Health and Human Services has divided the country into 10 regions, commonly refereed to as the HHS regions. The CDC reports ILINet wILI values for each of these regions individually as well. Here we leverage all methods for influenza forecasting for different regions. For different regions the influenza pattern can be different. We want to see if `EpiDeep` and other methods can detect these differences and perform well in each regions. Hence, here we leveraged all the methods for all four tasks in all the regions.

In summary, we observe that `EpiDeep` consistently performs well in all predictions. For space, we report RMSE results for the 2016/17 season only for HHS regions 1, 2 and 3 for Future Incidence and Onset prediction tasks. We observe similar results in all other tasks across other metrics (in supplementary). See Figure 10.5: `EpiDeep` outperforms all the baselines in future incidence prediction in all three HHS regions. Similarly, we observe that the all the methods except `ARIMA` perform well in onset prediction for all three regions. `EpiDeep`'s



(a) Future Incidence Prediction (d) Onset Prediction

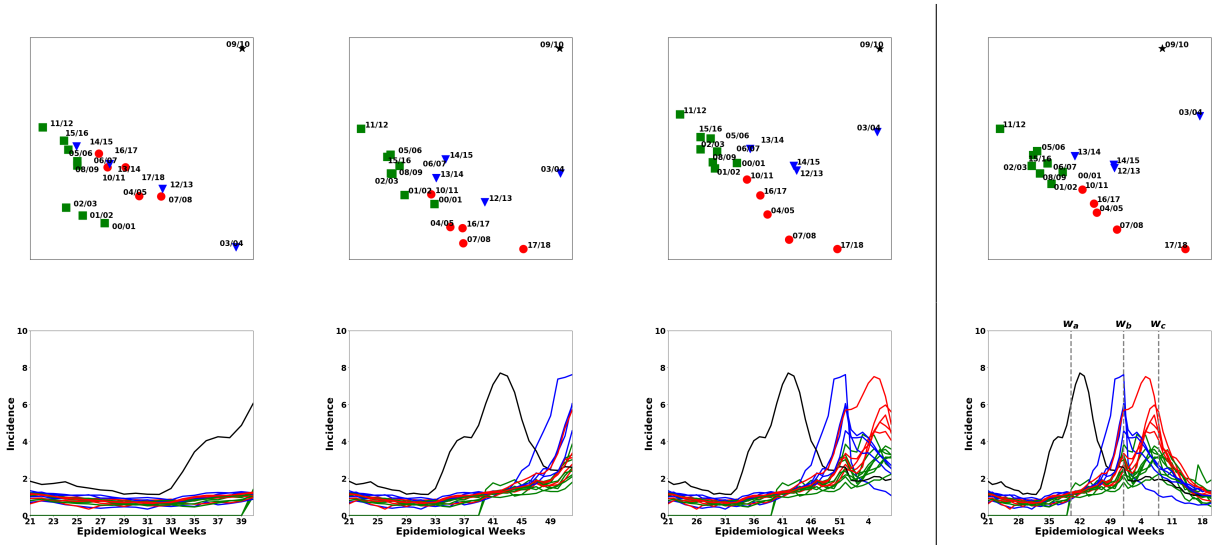
Figure 10.5: RMSE for regional predictions of two of the tasks for 2016/17 season. `EpiDeep` consistently performs well. The figure is best viewed in colour.

performance is competitive in all three regions.

10.3.5 Interpretability

An important advantage of `EpiDeep` is that its embedding/clustering components help in interpretability of the forecasts. Here we demonstrate how to leverage the embeddings learnt by `EpiDeep` for interpreting the influenza forecasting. Our experiments are designed to answer questions that epidemiologists and authorities like the CDC have. We focus on the following questions:

- *Question 1:* Can we infer 'clusters' of historical seasons based on their incidence curve even when partially observed?
- *Question 2:* What relationship can we infer between different HHS regions across multiple seasons?
- *Question 3:* Which historical season is closest to the current 'query' season at the time of forecasting? Does the closest season evolve over time as more data is observed?



(a) $w_a = 40$ (b) $w_b = 52$ (c) $w_c = 8$ (d) Whole Season

Figure 10.6: (d)-bottom row shows full season length wILI curve for all historical seasons. Bottom row (a), (b), and (c) shows snippets of the historical seasons till week w_a, w_b, and w_c respectively. The top row shows 2-d projection of learnt embeddings of corresponding snippets.

Question 1: Qualitative Evaluation of the Embeddings and Inferred Clustering

We can leverage `EpiDeep` to learn the embeddings of the historical seasons. Figure 10.6 shows the 2-d projection of the embeddings generated by `EpiDeep` at various weeks w (when partial data is observed) in the top row and corresponding incidence curves in the bottom row. The colors of the markers in the top plot represent the cluster memberships. For each cluster, the same color is used to draw the incident curves in the bottom row. The top row in Fig 10.6 (d) shows the embeddings generated when the complete data is observed. From the figure, we can make the following key observations.

Observation 3 ***Season Clusters:*** `EpiDeep` *embeddings showcase different meaningful clusters of wILI trends.*

The first cluster (in black) has only one member, the 2009/10 H1N1 pandemic season. Clearly from the incidence curves (bottom row Fig 10.6 (d)), it is obvious that the 2009/10 season was very different than the rest [56]. The cluster in green, representing seasons such as seasons 2011/22, 2005/06, 2015-14, and so on, have a distinct characteristics: they all have a low peak (green curves in (bottom row Fig 10.6 (d)). As reported by the CDC these were the only seasons to peak in March [57]. The defining characteristic of the blue cluster is that the seasons peak relatively early (late December/early January) and have high intensity.

Finally, we observe that the seasons in the red clusters have a late peak and a high intensity. These intuitive clustering of the seasons shows that `EpiDeep` learns meaningful embeddings of the historical seasons.

Question 2: Regional and Seasonal Embeddings.

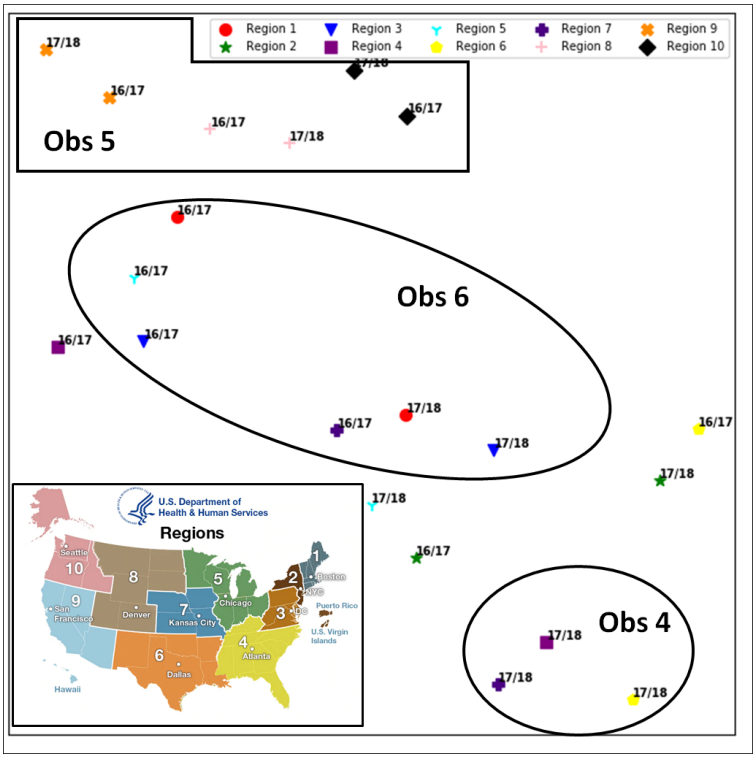


Figure 10.7: 2-d projections of embeddings of 2016/17 and 2017/18 seasons' wILI curves for all 10 HHS regions (inset).

In the previous question, we explored the quality of embeddings and their evolution. Here, we study the embeddings generated by `EpiDeep` for each HHS region over two seasons (2016/17 and 2017/18). We are interested in questions like does `EpiDeep` capture meaningful geographical relationship between HHS regions? Which regions are similar to each other and so on. The 2-d projections of the embeddings learnt by `EpiDeep` is shown in Figure 10.7. The first observation we make from the figure is as follows.

Observation 4 ***Neighbor Similarities:*** *Learnt embeddings reveal neighboring HHS regions have very similar incidence curves.*

The neighboring regions 4, 6, and 7 witnessed a very similar influenza season in 2017/18. It turns out all three of these regions peaked at week 4 and had a very high peak intensity.

Similar observations were made by prior works [157] for multiple diseases. Geographical correlation between HHS regions are leveraged for forecasting as well [157, 245].

Another nontrivial observation from the embeddings is that the incidence curve for some of the non-neighboring seasons are similar to each other.

Observation 5 ***Long Distance Similarities:*** `EpiDeep` *embeddings discover geographically distant regions having similar influenza incidence curves for multiple seasons.*

We observe that regions 1, 3, and 5 are embedded close to each other for both seasons. Their similarity is explained by the fact that all three seasons saw the influenza intensity peak at week 6 and 7 for 2016/17 season and peak at week 6 for 2017/18 season. Similarly, all three regions saw significant rise in the peak intensity in 2017/18 season as compared to that of 2016/17 season.

Additional Observations:

We make the following interesting observations additional to the ones discussed. Observation 4 is with respect to Question 1, Observation 5 is with respect to Question 2, and Observation 6 is with respect to question 3. Due to the lack of space, we defer detailed discussion on the significance of these observations to the Supplementary.

Observation 6 ***Intensity Separation:*** `EpiDeep` *embeddings distinguish seasons with higher intensities from the ones with lower intensities.*

Observation 7 ***Temporal Similarities:*** *The learnt embeddings reveal temporal similarities between different seasons in the same region.*

Observation 8 ***Evolution of Season Similarity:*** *The similarity/distances between the seasons evolve as more data is observed and* `EpiDeep` *is able to capture this phenomenon.*

Note that all the observations, from 1 to 6, are made directly from the embeddings learned by `EpiDeep`. It is quite challenging to extract all these patterns from the raw incidence curves as it is hard to compare all possible snippets of all the historical curves in each region.

10.4 Related Work

Epidemic Forecasting: In addition to the closely related works discussed earlier, there are other statistical [236, 67] and modelling based approaches [214, 266] for flu forecasting,

which suffer from the challenges discussed before. Additionally, orthogonal to this paper, there has also been much interest in leveraging signals from external data sources such as search engine [99, 261], social media [62, 139], environmental and weather reports [213, 230], and a combination of heterogeneous data [59]. Deep learning for flu forecasting has barely been explored except for [245] which basically uses a simple LSTM with geographical and climate constraints and [246] which uses LSTM to predict influenza activities specifically in the military population by incorporating twitter data. As we saw in Section 10.3, LSTM does not perform well as it requires a large amount of data. In contrast, we give a novel architecture which ensures excellent performance even with sparse data.

Time Series Analysis: Time-series prediction is a well-studied area with several methods from different perspectives including auto-regression, kalman-filters and groups/panels [48, 206, 115]. Recently recurrent neural architectures [108, 70] have also become popular. However these prediction methods are ill suited for flu forecasting as they are too specialized or usually not flexible enough to capture the seasonal inconsistency in wILI activity [177]. In contrast, we design an end-to-end approach which automatically embeds, clusters, and forecasts giving it the flexibility to capture the seasonal inconsistency in the data.

10.5 Discussions and Conclusions

Here we proposed a novel deep learning model `EpiDeep` to learn feature representations of historical epidemic seasons in conjunction with the observed current season and leveraged it for four epidemic forecasting tasks. We compared the performance of `EpiDeep` against multiple baselines on extensive historical data and showed that it outperforms non-trivial baselines by up to 40%. It also promises gains in *interpretability*. The embeddings learnt by `EpiDeep` are meaningful and non-trivial. We also observed that these embeddings evolve as the season progresses to capture the most meaningful relationship between the historical seasons.

Our method was designed to overcome specific challenges in influenza forecasting like the data sparsity issue and leveraging some domain knowledge for interpretability, but it also flexible and extensible. Due to its modular neural structure, as future work, we believe it has the potential to be useful in overcoming other challenges in a systematic manner as well. For example, since `EpiDeep` uses an end-to-end representation learning based framework, we can try to learn to jointly embed multiple heterogeneous data sources in addition to ILINet (say social media, weather data etc) and leverage these embeddings for prediction. We can also try to directly take data from epidemiological models as inputs into our model. Further, usually ILI data has geographical structure (e.g. flu incidence in nearby states would be expected to be similar [151]). These types of constraints can also be explicitly codified in the loss functions of the predictor module of `EpiDeep` (though notably, as we saw, it discovers many of these relationships automatically). We also believe that `EpiDeep` can be leveraged for forecasting general time-series with limited data. We plan on exploring it in future.

Chapter 11

Incorporating Guidance for Deep Epidemic Forecasting

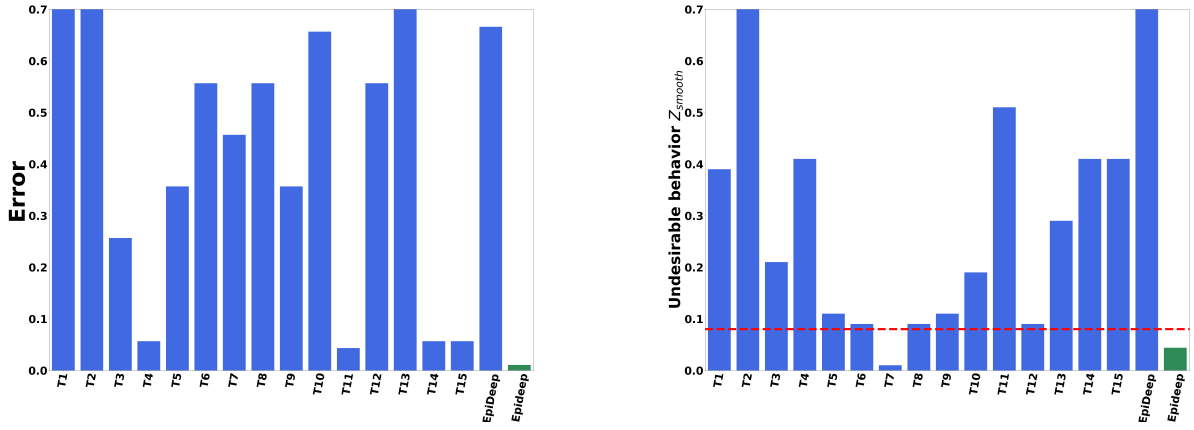
Epidemic outbreaks incur heavy burden in terms of both health and economic costs (like the ongoing 2019-Covid corona virus epidemic). According to the world health organization (WHO), more than 15 thousand lives were lost due to the Ebola outbreak in West Africa between 2013 and 2016[1]. The economic cost of Ebola is estimated to be more than 53 billion dollars[2]. Timely forecasting of epidemic outbreaks is critical. Accurately forecasting various metrics of an epidemic outbreak informs practitioners and policymakers about impending scenarios and helps them devise strategic countermeasures, such as quarantining subpopulations, increasing vaccination availability, and school closures.

In this paper, we focus on influenza forecasting, motivated by the CDC FluSight prediction challenge [40] which seeks to predict the inclidence of Influenza-like-Illnesses (ILI) in the US. Influenza is a major disease in the United States and beyond, causing thousands of fatalities every year. ILI is a symptomatic definition of illnesses that serves as a bellwether for real influenza incidence in a population. There has been a surge in recent research interest in influenza forecasting giving rise to a variety of mechanistic [266, 213] and statistical approaches [8, 51]. Mechanistic approaches predict influenza burden using simulation and aggregation of large epidemiological models. These models require a lot of calibration and hence are limited by their parameters to generalize well and fit the data [177]. Hence many researchers have begun exploring statistical approaches for this task, which train on historical ILI data and use the trained model to make forecasts for the current season.

Influenza seasons tend to be highly dynamic and have high variability due to numerous factors (e.g., weather, human mobility, virus strains circulating amongst the population) affecting the overall characteristics of the season. Moreover, different seasons and regions have different

[1] http://apps.who.int/gho/data/view.ebola-sitrep.ebola-summary-latest

[2] https://www.reuters.com/



(a) Error (b) Guidance

Figure 11.1: Comparison of approaches in terms of (a) error in forecasting and (b) a guidance metric. In both plots lower is better. The red line in (b) is the threshold determined by guidance. T1 to T14 is the performance of teams participating in the 2015 FluSight challenge. Our method Guided-Epideep (GEpideep in the plot) is the only method which satisfies the guidance and gives the lowest prediction performance error.

dominating influenza virus types. Further, the surveillance data collected (using ILINet) is a composite of multiple sources, is non-uniform, and is biased in many domain-specific ways. Hence while statistical approaches can frequently perform more accurate predictions than mechanistic models, they often show undesirable, unexplainable, or otherwise unexpected behavior.

For example, consider influenza incidence during the annual holiday season in the US. During this period, patients typically self-select and refrain from going to health providers, unless the situation is serious. This causes a temporary drop in recorded ILI incidence. However, as human mobility is high, flu activity rapidly increases in the following weeks. This can not be modeled using standard mechanistic epidemiological models [178]. At the same time, statistical approaches 'over-correct' and exaggerate the temporary 'dip'. Hence if we can ensure that the forecasting model's predictions are reasonably 'smooth', such a behavior can be avoided. This 'smoothness' of the forecasts is well-motivated from other epidemiological considerations as well. As Figure 11.1 (b) shows, almost all the methods used in the 2015 CDC FluSight challenge show this 'lack of smoothness' behavior (lower is better).

To tackle such issues, in this paper, we propose incorporating *expert guidance* into statistical models for epidemic forecasting. In the case above, may be the expert can give the guidance that week-to-week forecast should be smooth, which can alleviate the over-correction problem. Indeed, incorporating this guidance helps our approach outperform the baselines while maintaining accuracy. Our approach 'Guided EpiDeep' is the only method to show desirable behavior (having guidance metric Z_{smooth} below the predefined threshold (red line)) while

also getting the lowest errors (Figure 11.1).

The Christmas effect described above is just one example of how an epidemiological expert's domain knowledge can be brought to bear upon influenza forecasting. If such expert insights can be incorporated in the forecasting framework, the resulting forecasts will be better, actionable, and insightful. As another example, an expert will also able to guide the forecasting model to demonstrate desirable behaviors, as undesirable prediction could lead to undesirable health policies. For example, if rural regions have systematically lower prediction accuracy in forecasting models than other, more urban regions, that will lead to systematic under-allocation of resources for certain regions. Incorporating expert guidance to prevent this will allow the framework to have behavior that is more desirable from a policy viewpoint as well.

To design a forecasting framework as envisioned here, there are several challenges. The first challenge is (a) how to design a general framework for any influenza statistical forecasting model to ingest and leverage expert guidance. Designing a general framework to incorporate guidance allows existing approaches to include expert guidance. The second challenge is (b) how to ensure that the framework is easy to use and generates useful feedback to the user. Moreover, the framework should communicate the extent to which the guidance was successfully incorporated and whether the guidance is helpful or not. Such a framework will aid in selecting guidance and make the forecast interpretable with respect to the guidance provided. None of the existing approaches is able to tackle these challenges.

In this paper, we leverage the Seldonian Optimization framework proposed in AI safety to enforce expert guidance (desired behavior) and prevent undesirable behavior. Our framework provides feedback to the user regarding the success or failure in the incorporating the expert insights. In case the framework fails to incorporate the insight, it communicates the failure to the user, who in turn can take steps to alter/improve the insight or change data or modify model hyper-parameters. Our contributions are as follows.

- **Novel method for incorporating expert guidance:** We explore a novel problem & adapt a successful framework to obtain domain-based consistency (and guidance), and perform extensive experiments to show properties of the framework.
- **Flexible user interaction framework:** The framework adapts to the user's requirements.
- **Real data case-study:** We present concrete case studies showing examples of expert guidance motivated by epidemiologist observation, and how our method helps to achieve experts requirements.

The rest of the paper is organized in the following way: we first motivate our problem, and formulate it. Then we present our method, and then empirical studies on real CDC data. We finally end with related work and conclusions.

11.1 Problem Formulation

In this section, we introduce the novel problem of aiding statistical epidemic forecasting models with an expert's guidance. Before, we formalize our problem, we present the problem setting.

11.1.1 Epidemic Forecasting

Motivated by the setup of the CDC FluSight challenge, we study the epidemic forecasting problem from a temporal seasonality standpoint, such as in influenza. For this problem, we are given data $\mathcal{D}$ in the form of time series (e.g. the wILI burden per week for every season) and a predictive task $\mathcal{T}_w$, which sets what the target is and the time w (usually a week) when this prediction is to be made. Examples of targets are immediate-future incidences, peak intensity for season i, and the time when the peak value occurs.

The annual FluSight Challenge hosted by CDC asks to forecast metrics related to the current influenza season for the national and regional levels [40, 41]. The CDC releases influenza surveillance data, referred to as weighted Influenza-like Illness wILI, each week for every region. Given the latest partially observed influenza season, often represented as a time-series, the challenge asks to perform four different types of prediction tasks $\mathcal{T}_w$. They involve forecasting the incidence (wILI) value for the next four weeks, the onset of the season, the peak incidence value, and the time when the peak occurs. wILI incidence curves for each season since 1997/98 are publicly available[3].

11.1.2 Expert guidance

Expert guidance for epidemic forecasting is about leveraging multiple forms of domain knowledge and other preferences. An expert may want to guide a statistical model based on many considerations. Such considerations may include the epidemiology of the disease, characteristics of active virus strains (e.g. transmissability, reproduction rate), activity intensity in other other latitudes that dealt with the same virus strain, or efficacy of the vaccines to active strains of virus. It can also include some auxiliary knowledge. For example, it is well known that the Christmas holiday season in the US has specific impact on the flu spread which can not be captured by regular mechanistic epidemiological models [178]. It can include other public health policy considerations too, to ensure desirable behaviors like fairness in resource allocations. We give two specific examples we will use in our paper.

Example 1: Holiday correctness. As mentioned earlier, during the holiday season recorded epidemic activity temporarily drops due to patients' tendency to not seek healthcare. However, an expert notices that predictions of current statistical models are not

[3]https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

'smooth' i.e. they change a lot week-week and 'over-correct' during this time (a fact we demonstrate later in our experiments using the predictions of all the teams which participated in the CDC FluSight 2015 challenge). Hence s/he may want to more accurately forecast influenza incidence during the Holiday season incorporating the 'smoothness' property.

Example 2: Regional equity. In this case, a CDC expert wants to make sure that forecasting errors across rural/urban regions are not lopsided and are balanced, so that intervention resources can be distributed more fairly. We again demonstrate that existing models (from the CDC FluSight challenge) fail to have this important property. Hence the expert wants to ensure that a statistical forecasting model has this property, but also maintain accuracy.

11.1.3 Desired Properties of Guidance

In this paper, we focus on incorporating such types of guidance into statistical epidemic forecasting models. Designing a framework which can incorporate guidance must be able to exhibit some ideal properties, especially as it is meant for experts who may not know the internals or any technical details of the statistical models.

- **P1.** Promote one or more desirable behaviors during training.
- **P2**. Have a mechanism to guarantee tolerance on deployment.
- **P3.** Be flexible to any generic ad-hoc guidance and be compatible with state of the art epidemic forecasting models.
- **P4.** Be easy to use for the user/expert.
- **P5.** Provide feedback to user if guidance could not be incorporated.

Let us discuss the underlying reasons behind the importance of each of the properties described above. P1 is an important facet of guidance: when training, the preference should be given to candidate models aligned with guidance goals. In addition, an ideal framework should be able to enforce more that one desirable behavior. Once the training completed, one can not expect that guidance will be met in unseen data at all times. Hence, it is natural to think about a probability of the trained model in meeting the guidance in unseen data. P2 sets guarantees on the expected probability of a model to exhibit the desirable behavior on unseen (test) data. P3 takes into consideration that experts' requirements may be related to any characteristic of the epidemic season. Furthermore, to leverage existing statistical forecasting models, the framework should provide a path to easily incorporate them. P4 aims to provide the user an easy interface to leverage the framework, treating the statistical model essentially as a black box. Finally, P5 importantly aspires to clearly communicate the result of attempting to incorporate the proposed guidance. Note that in our context, expert

guidance can be motivated by practical considerations, and the data is really a composite signal (ILI cases rather than exactly flu cases). Hence sometimes expert guidance can indeed not be borne out by data, or be 'completely wrong' (unlike theory-guided data science [126], where scientific knowledge is considered ground truth) – so our framework needs a principled mechanism to signal this fact and provide feedback. The feedback provided opens possibilities to fruitful interactions as the expert may explore with different behaviors and tolerances to find the most suitable, and even test 'what-if' scenarios.

11.1.4 Definitions

Taking these properties in consideration, we make the following definitions to then state our problem.

Definition 12 *Expert guidance: We represent expert guidance as a tuple* $\langle g, \delta \rangle$*, where* $g : \Theta \rightarrow \mathbb{R}$ *is a function that maps a candidate forecasting model* $\theta \in \Theta$ *to a measure of desirable or undesirable behavior of* θ*, and* $\delta \in [0, 1]$ *is a tolerance which constraints the probability of the model to exhibit this behavior.*

Definition 13 *Successful incorporation of guidance: We successfully incorporate guidance when we obtain a forecasting model* θ *for which our desired tolerance is met.*

Note that our definition of expert guidance allows any framework which adopts it to exhibit all five desirable properties. Since the function g encodes one or more desirable behavior quantitatively, it can be used to enforce the behavior, satisfying P1. The parameter δ is the tolerance of undesirable behavior as mentioned in P2. Our definition of guidance is general enough to incorporate wide range of user insights to meet property P3. The only requirement is that the deviation from the desired behavior needs to be captured by the function g. Similarly, the user/expert do not need to be aware of underlying optimization framework and statistical model to incorporate the guidance as the function g is independent of both, satisfying P4. Similarly, if the value of the function g is greater than the threshold δ, the framework can communicate with the user regarding its inability to meet the guidance. We show how we can adapt our examples given before using our framework later (in Section 11.2.4).

11.1.5 Problem Statement

Having defined the notion of guidance that meets all the desired properties, we can state our problem as follows:

Guided Epidemic Forecasting: *Given a forecasting model which defines hypothesis space* Θ*, data* $\mathcal{D}$*, a predictive task* $\mathcal{T}_w$*, and expert guidance* $\langle g(\theta), \delta \rangle$*, we are required to*

return an optimal model θ, if found, that successfully incorporates expert guidance or return feedback that such θ could not be found.

In this paper, the predictive task we consider is the future incidence forecasting. Our task $\mathcal{T}_w$ asks for influenza incidence at week $w+1$ given that the incidence till week w is observed. And as the problem states, our goal is to enforce expert guidance, while solving for the predictive task. However our framework can easily handle other predictive tasks as well (like peak prediction etc).

11.2 Our Method

As stated above, the Guided Epidemic Forecasting problem requires a base forecasting model upon which the guidance is enforced. To enforce the guidance, we need a framework which optimizes for performance with respect to the predictive task $\mathcal{T}_w$ as well as ensures that the constraint imposed by the guidance $\langle g(\theta), \delta \rangle$ is met. Here we leverage Seldonian Optimization which does this.

11.2.1 Seldonian Optimization

The Seldonian optimization framework [235] was recently proposed for Artificial Intelligence (AI) safety. It is designed to prevent AI models from showing undesirable behaviour such as gender or racial bias. Traditional AI algorithms optimize an objective function to select a model θ as a solution from the space of all possible models Θ. This framework precludes undesirable behaviour of AI model by enforcing behavioral constraints on the optimization objective. Hence, a probabilistic constraint is added to the optimization such that the probability that the value of a predefined undesirable behavior metric $g(\theta)$ is greater than 0. After training, to ensure the behavioral constraint will be met when the solution is deployed, this framework has a safety test mechanism, which is performed in unseen data. If the model meets the requirements of the safety test, the trained model is returned, else the framework returns no solution found (NSF).

A natural question that arises is what kind of base forecasting model (which is required by our problem) best works with the Seldonian optimization framework. Intuitively, models which learn/train by back-propagating errors are most suited for the Seldonian Framework as it learns through back-propagating as well. Hence, here we chose a recently proposed deep learning based influenza forecasting model EpiDeep [8] as the base model upon which the guidance is to be enforced. We desribe Epideep briefly next. However we wish to emphasize that our framework is general.

11.2.2 EpiDeep

Epideep [8] is a recent deep neural architecture designed specifically for influenza forecasting. It exploits seasonal similarity between the current season and historical seasons via deep clustering [256]. The clustering module learns a latent low dimensional embedding of the seasons, such that the similarity between the seasons in the embedding space is meaningful for the task at hand. The clustering module in EpiDeep is designed such that it is possible to learn the embedding of the partially observed current season in the space of fully observed historical seasons. Epideep also uses long short-term memory (LSTM) [98] to encode in-season patterns of the current season. It then combines the embeddings from the clustering module and the LSTM and feeds the aggregated embedding to the decoder module, which make predictions for task $\mathcal{T}_w$. For the set of seasons $\mathcal{S}$ where each season $S \in \mathcal{S}$ is represented as a time series $S = s_1, s_2, \ldots, s_T$ in the training season, to predict the incidence observed in week w EpiDeep is trained with a loss function $\mathcal{L}(\theta) = \sum_{S \in \mathcal{S}} ||\hat{y} - s_w|| + \beta$, where $\theta \in \Theta$ is the trained model,$\hat{y}$ is the prediction made by θ and s_w is the observed incidence and β is the internal loss for Epideep not directly related to the task T_w. Note that while training only the weeks prior to week w is leveraged.

11.2.3 Expert-guided EpiDeep

The next natural question is how to adapt the Seldonian optimization framework to train EpiDeep with expert guidance. Before we answer that, let us define some notations. Let us have several different expert guidance to incorporate $\{\langle g_i, \delta_i \rangle\}_{i=1}^{n}$. We adopt the convention that if $g_i(\theta) \leq \epsilon$ for some small $0 \leq \epsilon$, the forecasting model θ does not exhibit undesirable behavior. Hence we impose probabilistic constraint on $g_i(\theta)$, on the model optimization. Hence, our updated optimization objective is as follows.

$$\begin{aligned} &\arg\min_{\theta} \sum_{S \in \mathcal{S}} ||\hat{y} - s_w|| + \beta \\ s.t. \Pr\left(g_i(\theta) \leq \epsilon\right) &\geq 1 - \delta_i, \forall i \in \{1, \ldots, n\} \end{aligned} \tag{11.1}$$

Here, $\hat{y}$ is the prediction made by model θ for the prediction task $\mathcal{T}_w$. The objective above indicates that we want to ensure the probability that the desirable behavior (i.e, $g_i(\theta) \leq \epsilon$) occurs is greater than $1 - \delta_i$ for some small $0 \leq \delta \leq 1$, while the difference between predicted incidence value and the eventually observed value is minimized. Note that, we also want to ensure that the probability that the desirable behavior holds even in test/deployment stage.

Following [235], we ensure that our approach optimizes the objective while not violating the constraints and it generalizes to other unseen data with high confidence. It does so by dividing the given training data $\mathcal{D}$ into two partitions D_c and D_s. D_c partition is used for the model selection/optimization, while D_s partition is only used to verify that the guidance behavior is met in unobserved data. If the guidance behaviour is not met in D_s, then the

framework ensures that no model is returned. In Algorithm 13, we leverage the Seldonian framework to design our algorithm Guided EpiDeep as follows.

Algorithm 13 Guided EpiDeep

1: **Input:** $\mathcal{D}$, $\langle g, \delta \rangle$, $U_{\mathcal{L}}$.
2: Partition $\mathcal{D}$ into D_c (for candidate selection) and D_s (for safety test)
3: $\theta_c \in \arg\min_{\theta \in \Theta}$ CandidateLossFunction$(D_c, \delta, \epsilon, U_{\mathcal{L}}, |D_s|)$
4: { Safety test using D_s }
5: **if** UpperBound$(\theta_c, D_s, \delta, U_{\mathcal{L}}) \leq \epsilon$ **then**
6: return θ_c
7: **else**
8: return No Solution Found (NSF)

Here, the function UpperBound in line 5 measures if the behavior of candidate model θ_c is desirable as per the guidance provided. Based on predictions made by θ_c, variable Z is defined to quantify the deviation from the desirable behaviour for each prediction made. We discuss how variable Z is constructed for the guidance we consider in Section 11.2.4. Once Z is defined, we use compute UpperBound as suggested in [235]. We employ an empirical upper bound on the magnitude of $\mathcal{L}$, which is denoted as $U_{\mathcal{L}}$. This bound is necessary to prevent gradient explosion when switching losses in our CandidateLossFunction. Next we present the CandidateLossFunction subroutine in line 3 of Algorithm 14.

In the CandidateLossFunction subroutine, we use the D_c partition of the training data to train on both the objective with respect to the task $\mathcal{T}_w$ and to ensure that the returned model, θ is consistent with the guidance. To do so, variable $Z = \{Z_i | \forall i \in D_c\}$ is created using the predictions made by the model θ. Then the upper bound on variable Z is computed. If the upper bound computed is less than ϵ, indicating that the model is showing desirable behavior with respect to the guidance, then loss on $\mathcal{L}(\theta)$ is returned else, the loss on the bound is returned. Note that internally, $\lambda \in R_{>=0}$ balances the trade-off between loss and guidance.

Algorithm 14 CandidateLossFunction

1: **Input:** Candidate θ_c, D_c, $\langle g, \delta \rangle$, $U_{\mathcal{L}}$, $|D_s|$
2: Create an array of Z_i, where $i \in D_c$
3: $\hat{U} =$ PredictedBound$(Z_i, \delta, |D_s|)$
4: **if** $\hat{U} \leq \epsilon_i$ **then**
5: return $\sum_{S \in \mathcal{S}} ||\hat{y} - s_w|| + \beta + \lambda \frac{1}{|Z|} \sum_{i=1}^{|Z|} |Z_i|$
6: return $U_{\mathcal{L}} + \hat{U} + (\lambda - 1)\epsilon$

Now, the question is how to define the variables Z for a given guidance. We discuss it next.

11.2.4 Constructing Behavioral Constraints

In this paper, we select two distinct expert guidance for seasonal influenza forecasting, namely smoothness and regional equity. In this section, we show the construction of constraint objectives for these expert guidance in the form of the Z variables. The guidance g and the variable Z are related to each other as follows:

$$g_i(\theta) = Z_i(\theta) - \epsilon \tag{11.2}$$

Smoothness

Mechanistic epidemiological models reveal that the epidemiological curves tend to be smooth with a single peak [214]. Hence, we expect epidemic influenza seasons to be generally smooth. In fact, we observe influenza incidence curve to be smooth with the consecutive values not changing drastically. Usually, incidence are low in the beginning of the season, they gradually rise till the peak is observed and then decline near monotonically. Hence, forecasting that the influenza incidence while ensuring that the predictions are smooth is a desirable property.

The smoothness also helps in correcting the drop observed in the influenza activity during the holiday season (discusses earlier), which arises due to the artifact of data collection. The existing approaches tend to overcompensate for the drop. Enforcing smoothness in forecasts prevents such undesirable behaviour. Here we describe smoothness as follows:

Definition 14 *Smoothness is the max allowed difference ϵ between the predicted value and its predecessor.*

We have a smoothness parameter ϵ, which is the maximum change allowed between current influenza incidence and the next incidence. In simple words, we want to ensure that the probability of smoothness function being greater than ϵ.

$$g(\theta) = E(|\hat{y}_{t+1} - Y_t|) - \epsilon \leq 0 \tag{11.3}$$

Here, $E(|\hat{y}_{t+1} - Y_t|)$ is the expected absolute difference between the predicted incidence $\hat{y}_{t+1}$ by the model θ and the last observed incidence Y_t. The guidance function $g(\theta)$ quantifies the smoothness by computing the difference between predicted and the previously observed value. Now, we have

$$E(|\hat{y}_{t+1} - Y_t|) \leq \epsilon \tag{11.4}$$

The equation above, highlights that the expected difference between the forecasted value and the previously observed value should be less than some constant ϵ ensuring that the

forecast maintains week-to-week smoothness. Following this, we define the varaible Z for smoothness as follows:

$$Z_{\text{smooth}} = |\hat{y}_{t+1} - Y_t| \tag{11.5}$$

Regional Equity

The performance of forecasting model for different regions is governed by the quality of surveillance mechanisms in place in a given region. Hence forecasting models tend to perform poorer in the regions where the surveillance data is poor. Moreover, the bias in the quality of forecast due to quality of data could be magnified, leading to less accurate forecasts in regions which already have limited surveillance resources. This could result in under-preparation, which could be catastrophic or over-preparation, which is wasteful in such regions. Hence, to ensure that the regions with poor surveillance system do not suffer more financial/health burden regional equity in the quality of forecast is required. Here we define Regional Equity as follows:

Definition 15 *Regional Equity is the maximum allowed difference ϵ between the quality of forecast μ between any two regions.*

Here the function μ could be any measure of forecast quality such as root mean squared error (RMSE), mean average prediction error (MAPE), and so on. In this paper, we use RMSE. Now, the guidance function for smoothness with respect to quality metric μ and two arbitrary regions R_1 and R_2 can be written as follows:

$$g(\theta) = E(|\mu(\theta, t+1, R_1) - \mu(\theta, t+1, R_2)|) - \epsilon \leq 0 \tag{11.6}$$

Here,$E(|\mu(\theta, t+1, R_1) - \mu(\theta, t+1, R_2)$ is the expected difference in quality of forecast in regions R_1 and R_2. Now, we have:

$$E(|\mu(\theta, t+1, R_1) - \mu(\theta, t+1, R_2)|) \leq \epsilon \tag{11.7}$$

The equation above represents our desire that under the regional equity guidance, we wish to ensure that the error in forecasting for two regions R_1 and R_2 should be less than some small constant ϵ. Hence, we define the variable Z for regional equity as follows:

$$Z_{\text{regional}} = |\mu(\theta, t+1, R_1) - \mu(\theta, t+1, R_2)| \tag{11.8}$$

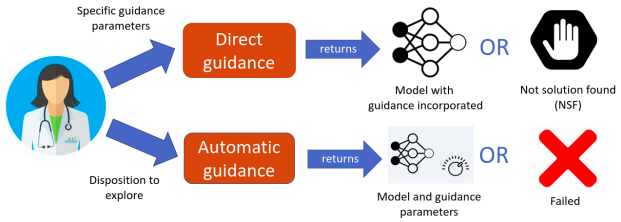


Figure 11.2: Flow diagram of expert interaction with Guided EpiDeep. Expert is given two modes: direct guidance and automatic guidance. The choice depends on the underlying motivation of the expert. Depending on the mode selected, the feedback is adapted to report success or failure.

11.2.5 Expert Interaction

As mentioned in Section 11.1 two of the desirable properties of a framework to incorporate guidance is that it should be easy to use (P4) and should be able to provide feedback to user (P5). In this section, we present how our framework can be leveraged for exploration as well as demonstrate how an user might be able to interact with the framework.

From a user perspective, our framework provides three knobs: data, model, tolerance. An user is able to decide on how to partition the data into D_c for candidate model selection and D_s for safety test. Similarly, the user can decide on the base model suitable for the task at hand. The final knob corresponds to the tolerance with which the failure to incorporate the provided guidance is allowed. An expert/user can interact with the system by varying the values corresponding to the knobs.

Since our model has a mechanism for the safety test, it may return 'No Solution Found' (NSF) indicating that the guidance provided could not be met given the values of the knobs. If the model returns NSF, it is an indication for the user to either consider the guidance provided or to vary the knobs. For example, if guidance related to smoothness is decided to be changed, this can be changed from $\epsilon = 0.5$ to $\epsilon = 1$. If tolerance is changed, confidence in guidance is changed. Hence, the model might be able to incorporate the guidance with a lower confidence on its generalizability. On the other hand, an expert can also change data by deciding to exclude some historical seasons that are preventing the guidance provided from being.

For ease of usage and interaction, our framework provides two modes of usages, namely Direct guidance and Automatic guidance, and depicted in Figure 11.2. We discuss each of the usages next.

Direct Guidance

In this mode, the user specifies guidance along with all the *all* parameters. Then our framework tries to incorporate the guidance within the constraints imposed by the parameters. If the framework fails to find a forecasting model which guarantees guidance incorporation, the the framework returns NSF. The direct guidance framework is presented in Algorithm 13.

Automatic Guidance

The user or epidemiological expert may not have data science/mathematical background to estimate the parameters with which the guidance can be incorporated and is willing to explore. Hence, in such cases, the framework tries to find the parameters which ensures that the guidance is met and the performance is maintained.

Our framework is able to provide such a exploration mode for users. Here the user may specify a *subset* of the parameters, and requirements in terms of performance. Our framework then explores the parameter space to find such a model. If none of the parameters explored is able to induce a model which satisfies the user requirements, the the framework returns NSF, indicating tha the guidance could not be incorporated. In this paper, as an example of automatic guidance, we ask our framework to explore the parameter ϵ_i such that no compromise is made in terms of RMSE.

11.3 Experiments

11.3.1 Setup

We describe the experimental setup next. All experiments are conducted using a 4 Xeon E7-4850 CPU with 512GB of 1066 Mhz main memory. Our method is very fast, training for one prediction task (on 1 week) in about 3 mins. We will release the code for academic purposes.

Data Here we use the weighted Influenza-like Illness (wILI) data released and updated by the CDC. CDC collects the wILI data through the Outpatient Influenza-like Illness Surveillance Network (ILINet) which consists of more than 3,500 outpatient healthcare providers all over the United States.CDC defines Influenza-like Illness (ILI) as fever (temperature of 100F [37.8C] or greater) and a cough and/or a sore throat without a known cause other than influenza. Weekly wILI incidence curves for each season since 1997/98 are publicly available[4].

Research questions to address. In our experiments we want to compare the performance of our approach Guided-EpiDeep with the baselines for both the smoothness and

[4]https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html

regional equity guidance. We also want to evaluate the automatic guidance mode of GUIDED-EPIDEEP. Specifically, we are interested in answering the following questions.

Q1. Is GUIDED-EPIDEEP successful in incorporating guidance?

Q2. Does GUIDED-EPIDEEP give feedback?

Q3. Is GUIDED-EPIDEEP successful in realistic scenarios on real WILI data?

Evaluation. Here, we use the test data T, which is separated out during training to evaluate the performance of GUIDED-EPIDEEP. Note that the test data is not used in either partition of the training data, namely D_c used for candidate model selection and D_s used for safety test.

To evaluate GUIDED-EPIDEEP with respect to Q1, we will test if the guidance is incorporated in the test data. For Q1, we train the model on D_c and D_s to incorporate the guidance. Once the model is trained we evaluate whether the behavior of the model in forecasting influenza season in T is desirable with respect to the given guidance. To evaluate the degree to which the behaviour mandated by guidance is met in the test, we compute the probability that the behavior defined by the guidance $g(\theta)$, as defined in Section 11.2, falls outside the bounds. We name this metric as the failure rate of the model θ. Formally, we define the failure rate as $\Pr(g_i(\theta))$. To evaluate GUIDED-EPIDEEP with respect to Q2 and Q3, we perform several case-studies.

Baselines. We use EpiDeep for performance and state of art baselines from the FluSight challenge for our case studies to show how they perform in a real-world scenario as posed by the CDC FluSight challenge. The complete list of teams we use is presented in the supplementary.

11.3.2 Direct Guidance

As mentioned earlier, in the direct guidance mode, the user/expert provides guidance as well as other parameters. Here, GUIDED-EPIDEEP searches for the model which in able to incorporate the guidance within the constraints imposed by the parameters. For direct guidance, we evaluate GUIDED-EPIDEEP in terms of Q1 and Q3.

Performance

Here, we want to quantify the rate at which GUIDED-EPIDEEP is able to ensure that the behavior imposed by the guidance is met in the test set. To do so, here we split the historical seasonal influenza data into the training set D which consists 80% of the seasons and test set T, which has the remaining seasons. GUIDED-EPIDEEP is trained on D with the smoothness constraint with a $\epsilon = 0.25$ and $\delta = 0.2$ to return a model θ. We repeat the experiment to make forecasts starting at week 40 of the epidemiological season till week 17. We then

measure the failure rate, as defined earlier, of θ on the test set T for each week. Then we repeat the experiment with $\epsilon = 0.5$ and $\delta = 0.1$. The result is presented in Figure 11.3.

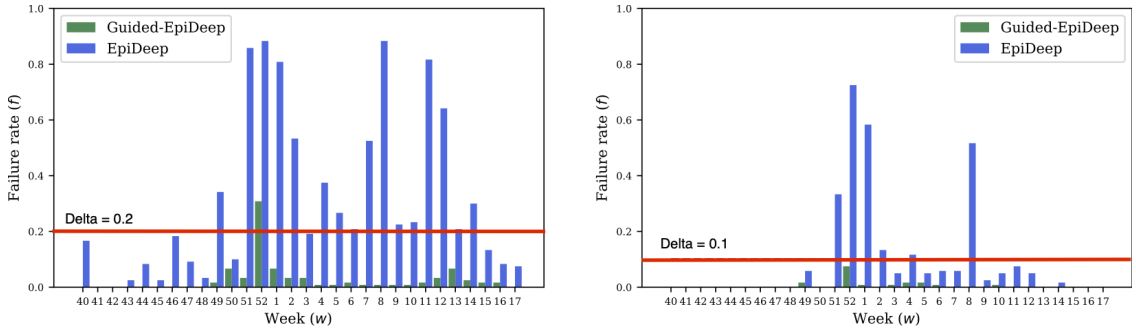


Figure 11.3: Performance of GUIDED-EPIDEEP in specific guidance. Figures show failure rate (f) for different combinations of ϵ and δ: (left) $\epsilon = 0.25$ and $\delta = 0.2$; (right) $\epsilon = 0.5$ and $\delta = 0.1$. Guided EpiDeep is successful incorporating expert guidance in epidemic task $\mathcal{T}_w$ for every week w in standard flu season as it is mostly within the bounds given by δ. Note that f in EpiDeep is higher than the required tolerance δ, but GUIDED-EPIDEEP is able to exhibit the desired behavior within the required tolerance.

As seen in both Figure 11.3, for both settings, GUIDED-EPIDEEP almost always ensures that the behavior imposed by the guidance is carried to the test data T. We observe that only 1 out of 80 observations, only one GUIDED-EPIDEEP has a failure rate higher than the threshold δ. On the other hand, the baseline EpiDeep has significantly higher failure rate consistently, with the failure rate for many observations greater than δ. This experiment demonstrates that GUIDED-EPIDEEP ensures that the desirable behavior is observed while forecasting on test data, while the baselines fail to do so.

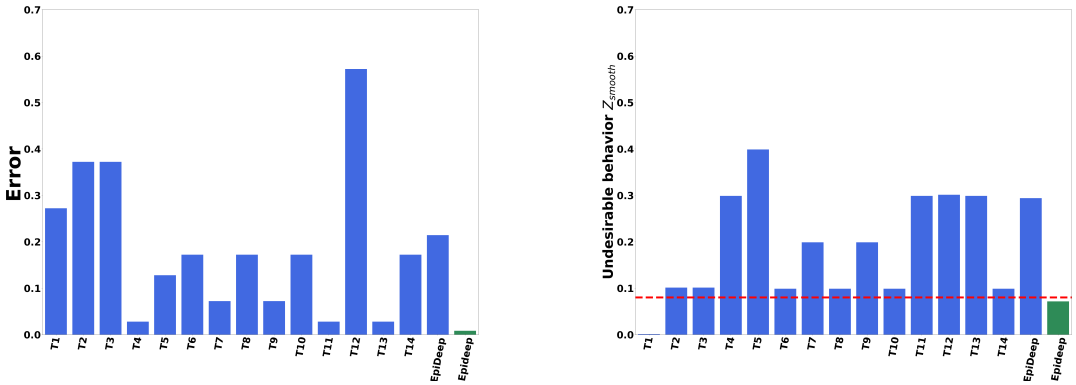
Failure in Incorporation of Guidance. In the rare case when the GUIDED-EPIDEEP fails to return a model (NSF) or the returned model does not ensure that the desirable behavior is observed in test data, as in week 52 in Figure 11.3 (left), the user is free to adjust one or more of the three knows our framework, data, model, and tolerance to allow the framework to search for a better forecasting model. For example, in the same example, setting a higher δ may ensure that the selected model satisfies the constraint in the test data as well.

Case study: Holiday Correctness

As mentioned earlier, during the holiday season in late December/early January the influenza incidence temporarily drops. Typically, the observed drop in the influenza incidence around the holiday season is not drastic. Hence, here we expect enforcing smoothness leads to better forecasting performance.

Here we gathered the submission made by different models forecasting the influenza incidence in the first week of January 2016 from the 2015-2016 influenza season in the US national

region as well as HHS Region 1. For each of the model, we measure the smoothness of prediction as the difference between the observed incidence in the last week of December and the forecast for the first week of January. Similarly, we also measure the performance of each model as the RMSE between the forecast and eventually observed ground truth incidence in the first week of January. We simulate the forecast for EpiDeep and our approach Guided-Epideep and compute both the metric. The result for the national region is presented in Figure 11.1 and the result for the HHS Region 1 is presented in Figure 11.4.



(a) Performance (b) Guidance

Figure 11.4: Comparison of approaches in terms of (a) RMSE in forecasting and (b) Z_{smooth} for the smoothness constraint. n both plots lower is better. The red line in (b) is the threshold determined by guidance. T1 to T14 is the performance of teams participating in the 2015 FluSight challenge in HHS region 1. Guided-Epideep (GEpideep in the plot) is the only method to satisfy the smoothness constraint and maintain a low performance error.

As observed in both Figures 11.1 and 11.4, only Guided-Epideep and a baseline model $T4$ satisfy the smoothness constraint. All other methods fail to satisfy the constraint. Note that while $T4$ satisfies the constraint, it has a very high performance error due to the fact that it forecasts the incidence to be nearly the same as the last observed incidence. However, Guided-EpiDeep avoids this issue as ensures that the smoothness constraint is met, while the performance is also maintained.

11.3.3 Automatic Guidance

As mentioned earlier, in the automatic guidance mode, the user/expert provides the guidance. However, the other parameters are not known. Here, Guided-EpiDeep searches for the ideal parameter set which can incorporate the guidance. Here, for automatic guidance, we evaluate Guided-EpiDeep in terms of all the questions.

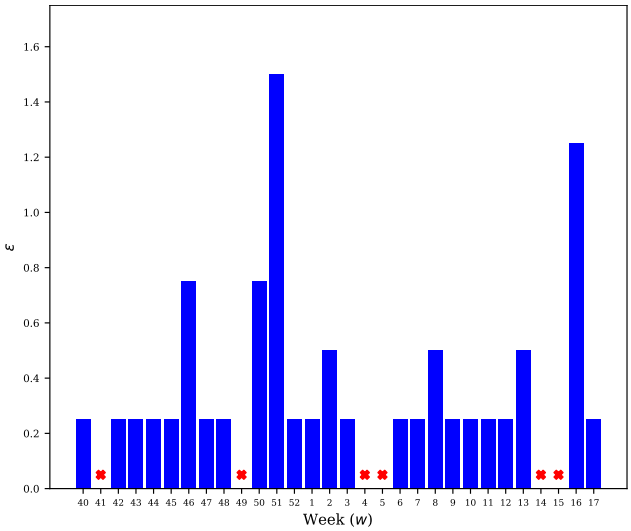


Figure 11.5: Automatic guidance over weeks. The y axis shows the value of ϵ found by GUIDED-EPIDEEP in automatic guidance mode. The red crosses represent the weeks where no suitable ϵ was found.

Performance

Here, we want to measure if GUIDED-EPIDEEP can find a parameter which can satisfy the constraints imposed by the guidance provided by the user. Here we use the same setup as in Direct guidance. We split the data into training D and test T sets with 4:1 ratio. The expert's requirement here is to ensure that the performance of GUIDED-EPIDEEP is better than the baseline model EpiDeep. We do so by enforcing that the ratio of RMSE of GUIDED-EPIDEEP to the RMSE of Epideep is less than 1. And the parameter to explore/detect is ϵ.We repeated the experiment for each week in the influenza season. Figure 11.5 shows the result.

As seen in the figure, for most of the week GUIDED-EPIDEEP is able to find an ϵ such that the constraint defined by the user is met. Among, 40 weeks GUIDED-EPIDEEP fails to find ϵ in only 6 weeks, demonstrating that our framework is able to incorporate expert's guidance in the automatic guidance mode. For the weeks where ϵ could not be found, GUIDED-EPIDEEP communicates its inability to find a solution to the user.

Case study: Regional Equity

It has been well documented that forecasting models have better performance for some regions than others [59]. Typically, regional error inconsistency is exhibited when comparing rural and urban regions. Therefore, an expert may want to make forecasting error balanced and achieve a more fair distribution of resources.

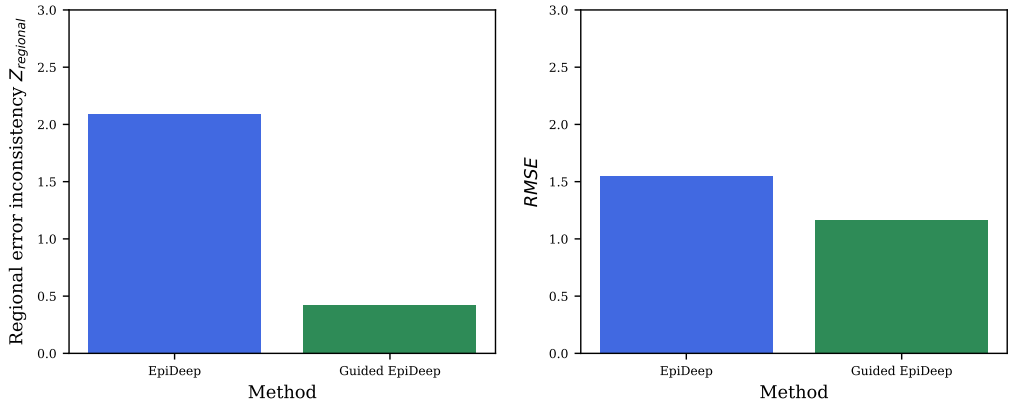


Figure 11.6: Automatic guidance for our case study on regional equity. Expert wants to make EpiDeep's predictions in Region 1 and Region 6 more equate/fair. Guided EpiDeep in automatic guidance finds out that for $\epsilon = 1.0$, we are able to reduce regional inconsistency in addition to reduce average RMSE of the two models.

In this case study, from the pool of models participating in the CDC Flusight challenge in 2018/19 season, the expert selects EpiDeep as the model for forecasting the current season. In the middle of the forecasting season, she realizes that EpiDeep performs well for HHS Region 1 but poorly for HHS Region 6. As she does not know the internals of EpiDeep, she prefers to use the automatic guidance mode, which allows her to figure out if regional equity can be incorporated without compromising shared accuracy.

To do so, here we run Guided-EpiDeep with the regional equity consistency as discussed earlier for week $w = 5$ in the automatic guidance mode. Guided-EpiDeep returned $\epsilon = 1.0$ and the model which performs best on train data in terms of accuracy. We then compare the performance of this model with Epideep both in terms of RMSE and regional equity consistency Z_{regional}. The result is summarized in Figure 11.6.

As shown in the figure, Guided-EpiDeep is able to find an ϵ which is able to enforce the regional equity constraint. Moreover, it is also able to minimize RMSE. On the other hand, the baseline EpiDeep has high variance in the error in prediction between the regions and also has higher overall error. Hence, the automatic guidance mode of Guided-EpiDeep is well suited for guidance incorporation when the parameters are unknown.

11.4 Related Work

Epidemic Forecasting: Epidemic forecasting models and generally categorized into statistical [236, 8, 178, 52] and mechanistic based approaches [214, 266]. Ensemble of mechanistic and statistical approaches too have been proposed [190]. There also has been interest in leveraging external data source in epidemic forecasting such as social media [62, 139],

search engine [99, 261], environmental and weather reports [213, 230], and a combination of heterogeneous data [59].

Recently, there has been surging interest in leveraging deep learning for influenza forecasting.Adhikari et al. [8] proposed EpiDeep which leverages deep architecture to exploit seasonal similarity for epidemic forecasting. Similarly, Wang et al. proposed DEFSI [249] which exploits intra and inter seasonal data for forecasting. Other approaches like [245, 246] have limited use case (example, for military population) and/or require external data sources (example, twitter, weather). However, none of these approaches are able to incorporate expert guidance.

Time Series Analysis: A field related to o epidemic forecasting in data mining is Time Series Analysis. Several approaches have been proposed such as auto-regression, kalman-filters and groups/panels [48, 206, 115]. Several deep learning approaches have also been used for time series analysis [108, 70].

Guided prediction framework: The Seldonian optimization framework [235] discussed earlier presents a general framework for expert guided prediction framework. Based on the Seldonian framework, Metevier et al. proposed Robinhood, an algorithm for fairness in a bandit setting. Several other approaches have been proposed for specific fairness objectives as well [124, 123]. However, to the best of our knowledge, we are the first to introduce a guidance-based machine learning approach for epidemic forecasting.

11.5 Conclusions

In this paper, we study the novel general problem of incorporating expert guidance to statistical epidemic forecasting methods, using influenza prediction as an example. Leveraging the Seldonian optimization framework, we showcase a flexible, adaptable framework which can ensure that the given guidance can be incorporated subject to some probabilistic tolerance, whilst also maintaining performance accuracy. Additionally, our method also gives valuable feedback to the expert, if the guidance can not be successfully incorporated, to promote interactions. Via two natural guidance scenarios (smoothness and regional consistency) we show on real CDC surveillance data, that our method bounds the probability of undesirable behavior while also reducing RMSE by 17%. As future work, one can focus on extending this framework to more types of guidance, and also handling probabilistic predictions (as opposed to point predictions we considered here).

Chapter 12

Discussions and Conclusions

12.1 Summary of contributions

In this thesis, we presented optimization frameworks, where we formulate tractable problems, and representation learning, where we design novel neural architectures inspired by domain knowledge for various selection, projection, prediction and inference tasks on dynamics over networks. Our frameworks enable better analysis of the actual underlying processes which leads to better selection algorithms. For example, to detect *C. Difficile* outbreaks in a hospital setting, we propose a discrete lattice-based optimization framework to complement a more accurate two mode disease model which captures multiple pathways of outbreaks. Similarly, we incorporate geographically correlated failure model in an information theoretic optimization framework for failure inference in critical infrastructure networks. Additionally, as we saw in various chapters, realistic frameworks allow one to analyze the mechanistic processes themselves at a finer resolution paving way for better generalization from sparse data. At the same time, we also presented low-dimensional domain-aware embeddings, which allow us to capture domain specific properties (like incidence-based similarity between historical influenza seasons) more efficiently from sparse data. These embeddings eventually lead to better performance in downstream tasks such as projection and prediction. Since our embeddings capture the information regarding dynamical processes directly from the data, without any modeling assumptions, they generalize better to new or unknown processes. In this thesis, we demonstrated that incorporating domain-knowledge leads to better optimization frameworks and neural architectures which outperform state-of-the-art baselines in several important tasks in multiple domains.

12.2 Discussion and Takeaways

In this section, we discuss the key takeaways and lessons learnt in this thesis. The main points are as follows:

Domain knowledge can be powerful: The thesis is centered on leveraging domain knowledge for problems in dynamics over networks. A key lesson learnt in the early stages of building this thesis was that domain-specific characteristics are powerful and if incorporated properly could lead to useful tools. This can be done in multiple ways. Domain knowledge can be incorporated to formulate problems from the first principles. For example, formulation based on geographically correlated failure model specific to the critical infrastructure networks led to two chapters (Chapter 6 and 7) on missing failure inference in this thesis. Similarly, domain-specific properties can be exploited for better algorithms. We were able to cluster queries observed in an e-commerce platform based on user intents by exploiting the characteristics of e-commerce networks we built in Chapter 4. Similarly, key domain-specific characteristics may inspire a better neural architecture. For example, the dynamic seasonal similarities between the historical influenza incidence curves motivated the architecture of `EpiDeep` in Chapter 10, which is able to generalize better from sparse data. Similarly, guidance from domain experts can be incorporated as constraints and loss function as demonstrated for epidemic forecasting in Chapter 11. These observations highlight a key point: domain knowledge, if incorporated correctly, leads to powerful approaches.

Heuristics without rigorous bounds can still be effective: In this thesis, we presented several guaranteed near optimal algorithms (Chapter 3, 4, and 6). However, for several problems discussed in this thesis, bounding the performance of the proposed algorithms is not straightforward. For example, in Chapter 5, we proposed a top-k network summarization approach. Similarly, in Chapter 3 and 7, we proposed greedy heuristic for minimizing outbreak detection time in hospitals and inferring failed components in critical infrastructure networks respectively. Despite lacking a rigorous bound, these approaches outperformed several state-of-the-art baselines as they were well motivated from a domain perspective. It will be interesting to better analyze these algorithms to obtain performance bounds based on data characteristics in future.

Experience in real world is valuable: Due to several issues like noise, confounding variables, and uncertainty in the data, the performance of machine learning/data mining model may suffer in real world challenges/deployment. As the problems explored in this thesis have high socio-economic value, there is no substitute of using the built models in practice. The experience of participating in real world challenges highlights the shortcomings of the models. For example, participating in the 2018-19 CDC Flusight challenge exposed some weaknesses of our influenza forecasting approach which were not evident in the development/research stage. Once these shortcomings are identified, they ultimately motivate better research.

12.3 Future work

Our work paves way for research in several new promising directions. We present some of the future directions below.

Heterogeneous graph mining for hospital acquired infections: In this thesis, we demonstrated that outbreak detection for HAIs can be formulated as a sensor selection problem in a heterogeneous network. However, several important problems for HAIs remain unexplored. Examples include how do we change mobility patterns to curtail infection outbreaks (intervention), and how do we find missing infections (inference)? These problems can be posed as optimization problems over heterogeneous networks. We propose to explore these in future.

Machine learning for general graph mining: Recently deep neural approaches for unstructured graph data have gained significant research interest. Most of the approaches are specifically designed for node/graph classification tasks and solved in a supervised or semi-supervised setting [132, 244]. The success of deep approaches in these tasks opens new avenues for designing graph convolutional networks for inherently unsupervised general graph mining tasks like community detection, network summarization, and sensor set selection. Designing deep graph approaches for these problems will bridge the gap between the current graph convolutional architectures and general graph mining problems such as the ones discussed in the first half of the thesis.

'What-if' influenza forecasting framework: Influenza forecasting has received much research interest lately. However, none of the approaches can answer what-if scenarios, which can be useful for policy makers like the CDC. Deep learning frameworks provide a unique opportunity to combine multiple data sources from sequential, structured, and unstructured data and learn efficient representations which are useful for downstream tasks. We propose to develop deep learning tools to forecast influenza under a given scenario. Such a framework is specially useful as it can be used to answer various questions regarding different scenarios such as interventions to aid decision making.

Bibliography

[1] http://www.sociopatterns.org/.

[2] T. Adachi and B. R. Ellingwood. Comparative Assessment of Civil Infrastructure Network Performance under Probabilistic and Scenario Earthquakes. *Journal of Infrastructure Systems*, 16(1):1–10, 2010.

[3] L. A. Adamic and B. A. Huberman. Power-law distribution of the world wide web. *science*, 287(5461):2115–2115, 2000.

[4] B. Adhikari, B. Lewis, A. Vullikanti, J. M. Jimenez, and B. A. Prakash. Fast and Near-Optimal Monitoring for Healthcare Acquired Infection Outbreaks.

[5] B. Adhikari, L. Li, N. Rao, and K. Subbian. Finding needles in heterogeneous haystacks. 2020.

[6] B. Adhikari, P. Rangudu, B. A. Prakash, and A. Vullikanti. Near-Optimal Mapping of Network States using Probes. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 108–116. SIAM, 2018.

[7] B. Adhikari, P. Sondhi, W. Zhang, M. Sharma, and B. A. Prakash. Mining E-commerce Query Relations using Customer Interaction Networks. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1805–1814. International World Wide Web Conferences Steering Committee, 2018.

[8] B. Adhikari, X. Xu, N. Ramakrishnan, and B. A. Prakash. EpiDeep: Exploiting Embeddings for Epidemic Forecasting.

[9] B. Adhikari, Y. Zhang, S. E. Amiri, A. Bharadwaj, and B. A. Prakash. Propagation-based temporal network summarization. *IEEE Transactions on Knowledge and Data Engineering*, 30(4):729–742, 2018.

[10] B. Adhikari, Y. Zhang, A. Bharadwaj, and B. A. Prakash. Condensing temporal networks using propagation. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 417–425. SIAM, 2017.

[11] B. Adhikari, Y. Zhang, N. Ramakrishnan, and B. A. Prakash. Distributed representations of subgraphs. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 111–117. IEEE, 2017.

[12] B. Adhikari, Y. Zhang, N. Ramakrishnan, and B. A. Prakash. Sub2vec: Feature learning for subgraphs. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 170–182. Springer, 2018.

[13] P. K. Agarwal, A. Efrat, S. K. Ganjugunte, D. Hay, S. Sankararaman, and G. Zussman. The Resilience of Wdm Networks to Probabilistic Geographical Failures. *IEEE/ACM Trans. Netw.*, 21(5):1525–1538, oct 2013.

[14] C. Aggarwal and K. Subbian. Evolutionary network analysis: A survey. *ACM Computing Survey*, 2014.

[15] C. C. Aggarwal, S. Lin, and S. Y. Philip. On Influential Node Discovery in Dynamic Social Networks. In *SDM*, 2012.

[16] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 2002.

[17] S. E. Amiri, B. Adhikari, A. Bharadwaj, and B. A. Prakash. NetGist: Learning to Generate Task-Based Network Summaries. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 857–862. IEEE, 2018.

[18] S. E. Amiri, L. Chen, and B. A. Prakash. SnapNetS: Automatic Segmentation of Network Sequences with Node Labels. 2017.

[19] G. Amitai, A. Shemesh, E. Sitbon, M. Shklar, D. Netanely, I. Venger, and S. Pietrokovski. Network analysis of protein structures identifies functional residues. *Journal of molecular biology*, 344(4):1135–1146, 2004.

[20] A. Anandkumar, A. Hassidim, and J. Kelner. Topology Discovery of Sparse Random Graphs with Few Participants. *SIGMETRICS Perform. Eval. Rev.*, 39(1):253–264, jun 2011.

[21] R. M. Anderson, B. Anderson, and R. M. May. *Infectious diseases of humans: dynamics and control*. Oxford university press, 1992.

[22] T. Anwar, C. Liu, H. L. Vu, and M. S. Islam. Roadrank: Traffic diffusion and influence estimation in dynamic urban road networks. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1671–1674. ACM, 2015.

[23] S. Arianos, E. Bompard, A. Carbone, and F. Xue. Power grid vulnerability: A complex network approach. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 19(1):013119, 2009.

[24] J. Aspnes, K. Chang, and A. Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *J. Comput. Syst. Sci.*, 2006.

[25] F. R. Bach and M. I. Jordan. Learning spectral clustering. In *NIPS*, volume 16, 2003.

[26] E. Bacry, A. Iuga, M. Lasnier, and C. Lehalle. Market impacts and the life cycle of investors orders. *Market Microstructure and Liquidity*, 1(02), 2015.

[27] R. Baeza-Yates. Graphs from search engine queries. *SOFSEM 2007: Theory and Practice of Computer Science*, pages 1–8, 2007.

[28] R. A. Baeza-Yates, C. A. Hurtado, M. Mendoza, et al. Query Recommendation Using Query Logs in Search Engines. In *EDBT workshops*, volume 3268, pages 588–596. Springer, 2004.

[29] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[30] N. T. Bailey et al. The mathematical theory of epidemics. 1957.

[31] A.-L. Barabási et al. *Network science*. Cambridge university press, 2016.

[32] M. J. Barber. Modularity and community detection in bipartite networks. *Physical Review E*, 76(6):066102, 2007.

[33] C. L. Barrett, K. R. Bisset, S. G. Eubank, X. Feng, and M. V. Marathe. EpiSimdemics: an efficient algorithm for simulating the spread of infectious disease over large realistic social networks. In *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pages 1–12. IEEE, 2008.

[34] A. Bay and B. Sengupta. Approximating meta-heuristics with homotopic recurrent neural networks. *arXiv preprint arXiv:1709.02194*, 2017.

[35] D. Beeferman and A. Berger. Agglomerative clustering of a search engine query log. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 407–416. ACM, 2000.

[36] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *NIPS*, volume 14, pages 585–591, 2001.

[37] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

[38] Y. Berezin, A. Bashan, M. M. Danziger, D. Li, and S. Havlin. Localized attacks on spatially embedded networks with dependencies. *Scientific Reports*, 2015.

[39] S. Bhagat, G. Cormode, and S. Muthukrishnan. Node classification in social networks. In *Social network data analytics*, pages 115–148. Springer, 2011.

[40] M. Biggerstaff, D. Alper, M. Dredze, S. Fox, I. C.-H. Fung, K. S. Hickmann, B. Lewis, R. Rosenfeld, J. Shaman, M.-H. Tsou, et al. Results from the centers for disease control and preventions predict the 2013–2014 Influenza Season Challenge. *BMC infectious diseases*, 16(1):357, 2016.

[41] M. Biggerstaff, M. Johansson, D. Alper, L. C. Brooks, P. Chakraborty, D. C. Farrow, S. Hyun, S. Kandula, C. McGowan, N. Ramakrishnan, et al. Results from the second year of a collaborative effort to forecast influenza seasons in the United States. *Epidemics*, 2018.

[42] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[43] P. Boldi, F. Bonchi, C. Castillo, D. Donato, A. Gionis, and S. Vigna. The query-flow graph: model and applications. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 609–618. ACM, 2008.

[44] P. Boldi, F. Bonchi, C. Castillo, D. Donato, and S. Vigna. Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data*, pages 56–63. ACM, 2009.

[45] P. Bonacich. Some unique properties of eigenvector centrality. *Social networks*, 29(4):555–564, 2007.

[46] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 946–957. SIAM, 2014.

[47] O. Bousquet and L. Bottou. The tradeoffs of large scale learning. In *NIPS*, pages 161–168, 2008.

[48] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. L"jun"g. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[49] L. Briesemeister, P. Lincoln, and P. Porras. Epidemic profiles and defense of scale-free networks. In *Proceedings of the 2003 ACM workshop on Rapid malcode*, pages 67–75. ACM, 2003.

[50] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer networks*, 33(1):309–320, 2000.

[51] L. C. Brooks, D. C. Farrow, S. Hyun, R. J. Tibshirani, and R. Rosenfeld. Flexible modeling of epidemics with an empirical Bayes framework. *PLoS computational biology*, 11(8):e1004382, 2015.

[52] L. C. Brooks, D. C. Farrow, S. Hyun, R. J. Tibshirani, and R. Rosenfeld. Nonmechanistic forecasts of seasonal influenza with iterative one-week-ahead distributions. *PLoS computational biology*, 14(6):e1006134, 2018.

[53] T. Brown. Multiple modeling approaches and insights for critical infrastructure protection. *NATO Security through Science Series D-Information and Communication Security*, 13:23, 2007.

[54] F. A. Carey and R. J. Sundberg. *Advanced Organic Chemistry: Part A: Structure and Mechanisms*. Springer Science & Business Media, 2007.

[55] A. Cassini, D. Plachouras, T. Eckmanns, M. A. Sin, H.-P. Blank, T. Ducomble, S. Haller, T. Harder, A. Klingeberg, M. Sixtensson, et al. Burden of six healthcare-associated infections on European population health: estimating incidence-based disability-adjusted life years through a population prevalence-based modelling study. *PLoS medicine*, 13(10):e1002150, 2016.

[56] CDC. Summary of the 2009-2010 Influenza Season. https://www.cdc.gov/flu/pastseasons/0910season.html, 2010. Accessed: 2018-11-05.

[57] CDC. Summary of the 2015-2016 Influenza Season. https://www.cdc.gov/flu/about/season/flu-season-2015-2016.html, 2016. Accessed: 2018-11-05.

[58] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos. Epidemic thresholds in real networks. *TISSEC*, 10(4):1, 2008.

[59] P. Chakraborty, P. Khadivi, B. Lewis, A. Mahendiran, J. Chen, P. Butler, E. O. Nsoesie, S. R. Mekaru, J. S. Brownstein, M. V. Marathe, et al. Forecasting a moving target: Ensemble models for Ili case count predictions. In *Proceedings of the 2014 SIAM international conference on data mining*, pages 262–270. SIAM, 2014.

[60] S. Chawla, R. Krauthgamer, R. Kumar, Y. Rabani, and D. Sivakumar. On the hardness of approximating multicut and sparsest-cut. *computational complexity*, 15(2):94–114, 2006.

[61] C. Chen, H. Tong, L. Xie, L. Ying, and Q. He. FAscInaTe: Fast cross-layer dependency inference on multi-layered networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 765–774. ACM, 2016.

[62] L. Chen, K. T. Hossain, P. Butler, N. Ramakrishnan, and B. A. Prakash. Syndromic surveillance of Flu on Twitter using weakly supervised temporal topic models. *Data mining and knowledge discovery*, 30(3):681–710, 2016.

[63] L. Chen, X. Xu, S. Lee, S. Duan, A. G. Tarditi, S. Chinthavali, and B. A. Prakash. Hotspots: Failure cascades on heterogeneous critical infrastructure networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1599–1607. ACM, 2017.

[64] W. Chen, Y. Wang, and S. Yang. Efficient influence maximization in social networks. In *KDD*, 2009.

[65] K. Cheng, J. Li, and H. Liu. Unsupervised Feature Selection in Signed Social Networks. In *KDD 2017*, pages 777–786. ACM, 2017.

[66] N. A. Christakis and J. H. Fowler. Social network sensors for early detection of contagious outbreaks. *PloS one*, 5(9):e12948, 2010.

[67] R. Chunara, E. Goldstein, O. Patterson-Lomba, and J. S. Brownstein. Estimating influenza attack rates in the United States using a participatory cohort. *Scientific reports*, 5:9540, 2015.

[68] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson. An estimate for the condition number of a matrix. *SIAM Journal on Numerical Analysis*, 16(2):368–375, 1979.

[69] T. Coffman, S. Greenblatt, and S. Marcus. Graph-based technologies for intelligence analysis. *Communications of the ACM*, 47(3):45–47, 2004.

[70] J. T. Connor, R. D. Martin, and L. E. Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.

[71] Cosgrove, S.E., Y. Qi, K. Kaye, S. Harbarth, A. Karchmer, and Y. Carmeli. The Impact of Methicillin Resistance in Staphylococcus aureus Bacteremia on Patient Outcomes: Mortality, Length of Stay and Hospital Charges. *Infection Control and Hospital Epidemiology*, pages 166–174, 2005.

[72] A. Costa, Y. Yamaguchi, A. Traina, C. T. Jr., and C. Faloutsos. RSc: Mining and Modeling Temporal Activity in Social Media. In *KDD*, pages 269–278, 2015.

[73] N. Craswell and M. Szummer. Random Walks on the Click Graph. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 239–246, New York, NY, USA, 2007. ACM.

[74] P. Cui, S. Jin, L. Yu, F. Wang, W. Zhu, and S. Yang. Cascading outbreak prediction in networks: a data-driven approach. In *KDD*, pages 901–909, 2013.

[75] H. Dai, B. Dai, and L. Song. Discriminative embeddings of latent variable models for structured data. In *ICML*, 2016.

[76] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song. Learning Combinatorial Optimization Algorithms over Graphs. *NIPS*, 2017.

[77] D. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes*. Springer-Verlag, 2003.

[78] A. Das Sarma, N. Parikh, and N. Sundaresan. E-commerce product search: personalization, diversification, and beyond. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 189–190. ACM, 2014.

[79] M. Denil, L. Bazzani, H. Larochelle, and N. Freitas. Learning where to Attend with Deep Architectures for Image Tracking. *Neural Computation*, pages 2151–2184, 2012.

[80] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11), 2007.

[81] M. Diligenti, M. Gori, and M. Maggini. A unified representation of web logs for mining applications. *Information Retrieval*, 14(3):215–236, 2011.

[82] N. Du, L. Song, M. G. Rodriguez, and H. Zha. Scalable influence estimation in continuous-time diffusion networks. In *Advances in neural information processing systems*, pages 3147–3155, 2013.

[83] N. Du, L. Song, M. Yuan, and A. Smola. Learning networks of heterogeneous influence. In *NIPS*, pages 2780–2788, 2012.

[84] S. Duan, S. Lee, S. Chinthavali, and M. Shankar. Best effort broadcast under cascading failures in interdependent critical infrastructure networks. *Pervasive and Mobile Computing*, 43:114–130, 2018.

[85] N. G. Duffield, J. Horowitz, F. L. Presti, and D. Towsley. Multicast topology inference from measured end-to-end loss. *IEEE Trans. Inf. Theory*, 2002.

[86] S. E Amiri, B. Adhikari, M. Dowling, J. Wenskovitch, C. North, and B. A. Prakash. Learning to generate effective network summaries. 2020.

[87] S. E Amiri, B. Adhikari, M. Dowling, J. Wenskovitch, C. North, and B. A. Prakash. Netreact: Learning network visualizations for text analytics using interactions. 2020.

[88] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *ACM SIGCOMM computer communication review*, volume 29, pages 251–262. ACM, 1999.

[89] FEMA. Multi-hazard Loss Estimation Methodology: Earthquake Model, Hazus-Mh 2.1: Technical Manual. Technical report, FEMA, Washington, D.C., USA, 2013.

[90] B. M. Fonseca, P. B. Golgher, E. S. de Moura, and N. Ziviani. Using association rules to discover search engines related queries. In *Web Congress, 2003. Proceedings. First Latin American*, pages 66–71. IEEE, 2003.

[91] J. Fournet and A. Barrat. Contact patterns among high school students. *PloS one*, 9(9):e107878, 2014.

[92] A. P. Francisco, R. Baeza-Yates, and A. L. Oliveira. Mining query log graphs towards a query folksonomy. *Concurrency and Computation: Practice and Experience*, 24(17):2179–2192, 2012.

[93] A. P. Francisco, R. A. Baeza-Yates, and A. L. Oliveira. Clique Analysis of Query Log Graphs. In *SPIRE*, volume 5280, pages 188–199. Springer, 2008.

[94] A. Ganesh, L. Massoulié, and D. Towsley. The effect of network topology on the spread of epidemics. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 2, pages 1455–1466. IEEE, 2005.

[95] F. R. Gantmacher and J. L. Brenner. *Applications of the Theory of Matrices*. Courier Corporation, 2005.

[96] N. T. Gayraud, E. Pitoura, and P. Tsaparas. Diffusion Maximization in Evolving Social Networks. In *COSN*, 2015.

[97] M. Génois, C. L. Vestergaard, J. Fournet, A. Panisson, I. Bonmarin, and A. Barrat. Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers. *Network Science*, 3(03), 2015.

[98] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with LstM. 1999.

[99] J. Ginsberg, M. H. Mohebbi, R. S. Patel, L. Brammer, M. S. Smolinski, and L. Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012, 2009.

[100] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[101] C. Gomez, A. D. González, H. Baroud, and C. D. Bedoya-Motta. Integrating Operational and Organizational Aspects in Interdependent Infrastructure Network Recovery. *Risk Analysis*, 39(9):1913–1929, 2019.

[102] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. MIT press Cambridge, 2016.

[103] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[104] grunwald. A tutorial introduction to the minimum description length principle. In *Advances in Minimum Description Length: Theory and Applications*. MIT Press, 2004.

[105] C. Gulcehre, F. Dutil, A. Trischler, and Y. Bengio. Plan, attend, generate: Planning for sequence-to-sequence models. In *NIPS*, pages 5480–5489, 2017.

[106] X. Guo, L. Gao, X. Liu, and J. Yin. Improved deep embedded clustering with local structure preservation. In *International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 1753–1759, 2017.

[107] H. W. Hethcote. The mathematics of infectious diseases. *SIAM review*, 42(4):599–653, 2000.

[108] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[109] S. A. Hosseini, K. Alizadeh, A. Khodadadi, A. Arabzadeh, M. Farajtabar, H. Zha, and H. R. Rabiee. Recurrent Poisson Factorization for Temporal Recommendation. In *KDD*, pages 847–855, 2017.

[110] M. R. Islam, S. Muthiah, B. Adhikari, B. A. Prakash, and N. Ramakrishnan. DeepDiffuse: Predicting the' Who' and' When' in Cascades. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1055–1060. IEEE, 2018.

[111] M. R. Islam, S. Muthiah, B. Adhikari, B. A. Prakash, and N. Ramakrishnan. Prediction on a pittance: Neural models for cascade analysis. 2020.

[112] K. Järvelin and J. Kekäläinen. Cumulated Gain-based Evaluation of Ir Techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, oct 2002.

[113] M. Jenders, G. Kasneci, and F. Naumann. Analyzing and Predicting Viral Tweets. In *WWW*, pages 657–664, 2013.

[114] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, 2000.

[115] A. Jha, S. Ray, B. Seaman, and I. S. Dhillon. Clustering to forecast sparse time-series data. In *Data Engineering (ICDE), 2015 IEEE 31st International Conference on*, pages 1388–1399. IEEE, 2015.

[116] C. Ji, Y. Wei, and H. V. Poor. Resilience of Energy Infrastructure and Services: Modeling, Data Analytics, and Metrics. *Proceedings of the IEEE*, 105(7):1354–1366, 2017.

[117] D. Jiang, J. Pei, and H. Li. Mining search and browse logs for web search: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(4):57, 2013.

[118] S. Jiang, Y. Hu, C. Kang, T. Daly Jr, D. Yin, Y. Chang, and C. Zhai. Learning Query and Document Relevance from a Web-scale Click Graph. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 185–194. ACM, 2016.

[119] J. M. Jiménez, B. Lewis, and S. Eubank. Hospitals as Complex Social Systems: Agent-Based Simulations of Hospital-Acquired Infections. In *International Conference on Complex Sciences*, pages 165–178. Springer, 2012.

[120] J. M. Jimenez, B. L. Lewis, and S. Eubank. The application of macroergonomics and simulation to improve control of healthcare acquired infections. In *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*, pages 3938–3939. IEEE Press, 2013.

[121] X. Jin, W. Yiu, G. S. H. Chan, and Y. Wang. Network topology inference based on end-to-end measurements. *IEEE Journal on Selected Areas in Communications*, 24(12):2182–2195, 2006.

[122] D. S. Johnson. Approximation algorithms for combinatorial problems. *Journal of computer and system sciences*, 1974.

[123] M. Joseph, M. Kearns, J. Morgenstern, S. Neel, and A. Roth. Meritocratic fairness for infinite and contextual bandits. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 158–163, 2018.

[124] M. Joseph, M. Kearns, J. H. Morgenstern, and A. Roth. Fairness in learning: Classic and contextual bandits. In *Advances in Neural Information Processing Systems*, pages 325–333, 2016.

[125] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[126] A. Karpatne, G. Atluri, J. H. Faghmous, M. Steinbach, A. Banerjee, A. Ganguly, S. Shekhar, N. Samatova, and V. Kumar. Theory-Guided Data Science: A new Paradigm for Scientific Discovery from Data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, 2017.

[127] M. Karsai, N. Perra, and A. Vespignani. Time varying networks and the weakness of strong ties. *Scientific Reports*, 4:4001, 2014.

[128] G. Karypis and V. Kumar. Multilevel algorithms for multi-constraint graph partitioning. In *Supercomputing, 1998. SC98. IEEE/ACM Conference on*, pages 28–28. IEEE, 1998.

[129] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the Spread of Influence Through a Social Network. In *KDD*, pages 137–146, 2003.

[130] J. O. Kephart and S. R. White. Measuring and modeling computer virus prevalence. In *IEEE Research in Security and Privacy*, 1993.

[131] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[132] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[133] R. M. Klevens, M. Morrison, J. Nadle, K. Gershman, S. Ray, L. Harrison, R. Lynfield, G. Dumyati, J. Townes, A. Craig, E. Zell, G. Fosheim, L. McDougal, R. Carey, and S. Fridkin. Invasive Methicillin-Resistance Staphylococcus aureus Infections in the United States. *JAMA*, pages 1763–71, 2007.

[134] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[135] H. Larochelle and G. Hinton. Learning to combine foveal glimpses with a third-order Boltzmann machine. In *NIPS*, pages 1243–1251, 2010.

[136] M. Latapy, C. Magnien, and N. Del Vecchio. Basic notions for the analysis of large two-mode networks. *Social networks*, 30(1):31–48, 2008.

[137] Q. V. Le and T. Mikolov. Distributed Representations of Sentences and Documents. In *ICML*, volume 14, pages 1188–1196, 2014.

[138] B. Leclère, D. L. Buckeridge, P.-Y. Boëlle, P. Astagneau, and D. Lepelletier. Automated detection of hospital outbreaks: A systematic review of methods. *PloS one*, 12(4):e0176438, apr 2017.

[139] K. Lee, A. Agrawal, and A. Choudhary. Real-time disease surveillance using twitter data: demonstration on flu and cancer. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1474–1477. ACM, 2013.

[140] J. Leskovec. Stanford Network Analysis Project. http://snap.stanford.edu/index.html, 2011.

[141] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD09*, 2009.

[142] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ' 07*, page 420, San Jose, California, USA, 2007. ACM Press.

[143] B. Lewis, S. Eubank, A. M. Abrams, and K. Kleinman. In silico surveillance: evaluating outbreak detection with simulation models. *BMC medical informatics and decision making*, 13(1):12, jan 2013.

[144] W. Liu, A. Kan, J. Chan, J. Bailey, C. Leckie, J. Pei, and R. Kotagiri. On compressing weighted time-evolving graphs. In *CIKM12*. ACM, 2012.

[145] X. Liu, W. Liu, T. Murata, and K. Wakita. A framework for community detection in heterogeneous multi-relational networks. *Advances in Complex Systems*, 17(06):1450018, 2014.

[146] Y. Liu, T. Safavi, A. Dighe, and D. Koutra. Graph Summarization Methods and Applications: A survey. *ACM Computing Surveys (CSUR)*, 51(3):62, 2018.

[147] Z. Liu, C. Chen, L. Li, J. Zhou, X. Li, L. Song, and Y. Qi. Geniepath: Graph neural networks with adaptive receptive paths. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4424–4431, 2019.

[148] Z. Liu, G. Singh, N. Parikh, and N. Sundaresan. A large scale query logs analysis for assessing personalization opportunities in e-commerce sites. *WSCD ?2014 New York, New York USA, ACM-2014*, 2014.

[149] E. Lofgren, S. Cole, D. Weber, D. Anderson, and R. Moehring. Hospital-acquired Clostridium difficile Infections: Estimating All-Cause Mortality and Length of Stay. *Epidemiology*, pages 570–75, 2014.

[150] E. T. Lofgren, R. W. Moehring, D. J. Anderson, D. J. Weber, , and N. H. Fefferman. A mathematical Model to Evaluate the Routine Use of Fecal Microbiota Transplantation to Prevent Incident and Recurrent Clostridium difficile Infection. *Infection Control and Hospital Epidemiology*, 2013.

[151] F. S. Lu, M. W. Hattab, C. L. Clemente, M. Biggerstaff, and M. Santillana. Improved state-level influenza nowcasting in the United States leveraging Internet-based data and network approaches. *Nature communications*, 10(1):147, 2019.

[152] M.-T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[153] A. M' Kendrick. Applications of mathematics to medical problems. *Proceedings of the Edinburgh Mathematical Society*, 44:98–130, 1925.

[154] M. Marathe and A. Vullikanti. Computational Epidemiology. *Communications of the ACM*, 56(7):88–96, 2013.

[155] M. Mathioudakis, F. Bonchi, C. Castillo, A. Gionis, and A. Ukkonen. Sparsification of influence networks. In *KDD*, 2011.

[156] L. Matikainen, M. Lehtomäki, E. Ahokas, J. Hyyppä, M. Karjalainen, A. Jaakkola, A. Kukko, and T. Heinonen. Remote sensing methods for power line corridor surveys. *ISPRS Journal of Photogrammetry and Remote Sensing*, 119:10–31, 2016.

[157] Y. Matsubara, Y. Sakurai, W. G. Van Panhuis, and C. Faloutsos. FUnnEl: automatic mining of spatially coevolving epidemics. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 105–114. ACM, 2014.

[158] S. McClendon and A. C. Robinson. Leveraging Geospatially-Oriented Social Media Communications in Disaster Response. *International Journal of Information Systems for Crisis Response and Management (IJISCRAM)*, 5(1):22–40, 2013.

[159] J. Medlock and A. P. Galvani. Optimizing influenza vaccine distribution. *Science*, 325(5948):1705–1708, 2009.

[160] R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. *Machine learning: An artificial intelligence approach*. Springer Science & Business Media, 2013.

[161] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[162] S. Milojević. Power law distributions in information science: Making the case for logarithmic binning. *Journal of the American Society for Information Science and Technology*, 2010.

[163] M. Mitzenmacher. A brief history of generative models for power law and lognormal distributions. *Internet mathematics*, 1(2):226–251, 2004.

[164] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[165] C. Moore and M. E. Newman. Epidemics and percolation in small-world networks. *Physical Review E*, 61(5):5678, 2000.

[166] A. Narayanan, M. Chandramohan, L. Chen, Y. Liu, and S. Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *arXiv preprint arXiv:1606.08928*, 2016.

[167] S. Navlakha, R. Rastogi, and N. Shrivastava. Graph summarization with bounded error. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 2008.

[168] R. R. Nelson. Economic development as an evolutionary process. In *Handbook of Alternative theories of economic development*. Edward Elgar Publishing, 2016.

[169] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions?I. *Mathematical Programming*, 14(1):265–294, 1978.

[170] M. E. Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.

[171] M. E. Newman. Mixing patterns in networks. *Physical Review E*, 67(2):026126, 2003.

[172] M. E. Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[173] J. Ni, H. Xie, S. Tatikonda, and Y. Yang. Efficient and Dynamic Routing Topology Inference From End-to-End Measurements. *IEEE/ACM Transactions on Networking*, 2010.

[174] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pages 2014–2023, 2016.

[175] A. Nowak, S. Villar, A. S. Bandeira, and J. Bruna. A note on learning algorithms for quadratic assignment with graph neural networks. *arXiv preprint arXiv:1706.07450*, 2017.

[176] E. O. Nsoesie, R. Beckman, M. Marathe, and B. Lewis. Prediction of an epidemic curve: A supervised classification approach. *Statistical communications in infectious diseases*, 3(1), 2011.

[177] E. O. Nsoesie, J. S. Brownstein, N. Ramakrishnan, and M. V. Marathe. A systematic review of studies on forecasting the dynamics of influenza outbreaks. *Influenza and other respiratory viruses*, 8(3):309–316, 2014.

[178] D. Osthus, J. Gattiker, R. Priedhorsky, S. Y. Del Valle, et al. Dynamic Bayesian influenza forecasting in the United States with hierarchical discrepancy (with discussion). *Bayesian Analysis*, 14(1):261–312, 2019.

[179] M. Ouyang. Review on modeling and simulation of interdependent critical infrastructure systems. *Reliability engineering & System safety*, 121:43–60, 2014.

[180] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[181] R. Pastor-Satorras and A. Vespignani. Epidemic spreading in scale-free networks. *Physical review letters*, 86(14):3200, 2001.

[182] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710. ACM, 2014.

[183] B. A. Prakash, D. Chakrabarti, M. Faloutsos, N. Valler, and C. Faloutsos. Threshold Conditions for Arbitrary Cascade Models on Arbitrary Networks. In *ICDM*, 2011.

[184] B. A. Prakash, H. Tong, N. Valler, M. Faloutsos, and C. Faloutsos. Virus propagation on time-varying networks: Theory and immunization algorithms. In *ECML/PKDD10*, 2010.

[185] B. A. Prakash, J. Vreeken, and C. Faloutsos. Spotting Culprits in Epidemics: How many and Which ones? In *ICDM*, 2012.

[186] M. Purohit, B. A. Prakash, C. Kang, Y. Zhang, and V. Subrahmanian. Fast influence-based coarsening for large networks. In *KDD14*.

[187] Q. Qu, S. Liu, C. S. Jensen, F. Zhu, and C. Faloutsos. Interestingness-driven diffusion process summarization in dynamic networks. In *ECML/PKDD*, 2014.

[188] S. Rayana and L. Akoglu. Less is more: Building selective anomaly ensembles with application to event detection in temporal graphs. In *SDM*, 2015.

[189] R. Rehurek and P. Sojka. Software framework for topic modelling with large corpora. In *LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010.

[190] N. G. Reich, C. J. McGowan, T. K. Yamana, A. Tushar, E. L. Ray, D. Osthus, S. Kandula, L. C. Brooks, W. Crawford-Crudell, G. C. Gibson, et al. Accuracy of real-time multi-model ensemble forecasts for seasonal influenza in the Us. *PLoS computational biology*, 15(11), 2019.

[191] B. Y. Reis, I. S. Kohane, and K. D. Mandl. An epidemiological network model for disease outbreak detection. *PLoS medicine*, 4(6):e210, 2007.

[192] Y. Ren, E. B. Fox, A. Bruce, et al. Clustering correlated, sparse data streams to estimate a localized housing price index. *The Annals of Applied Statistics*, 11(2):808–839, 2017.

[193] L. F. Ribeiro, P. H. Saverese, and D. R. Figueiredo. struc2vec: Learning Node Representations from Structural Identity. In *KDD 2017*, pages 385–394. ACM, 2017.

[194] K. Riesen and H. Bunke. *Graph classification and clustering based on vector space embedding*. World Scientific Publishing Co., Inc., 2010.

[195] J. Rissanen. Minimum Description Length Principle. In S. Kotz and N. L. Johnson, editors, *Encyclopedia of Statistical Sciences*, volume V, pages 523–527. John Wiley and Sons, New York, 1985.

[196] A. Rodriguez, B. Adhikari, C. N. Andres D. Gonzalez, A. Vullikanti, and B. A. Prakash. Mapping network states using connectivity quesries. 2020.

[197] A. Rodriguez, B. Adhikari, A. Gonzlez, C. Nicholson, A. Vullikanti, and B. A. Prakash. Mapping network states using connectivity queries. In *IISE Annual Conference and Expo 2019 (Data Analytics and Information Systems Track)*.

[198] A. Rodriguez, B. Adhikari, N. Ramakrishnan, and B. A. Prakash. Incorporating expert's guidance on epidemic forecasting. 2020.

[199] E. M. Rogers. *Diffusion of innovations*. 2010.

[200] R. Rosenfeld, J. Grefenstette, and D. Burke. A proposal for Standardized Evaluation of Epidemiological Models, 2012.

[201] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *science*, 290(5500):2323–2326, 2000.

[202] P. Rozenshtein, A. Gionis, B. A. Prakash, and J. Vreeken. Reconstructing an Epidemic Over Time. In *KDD*, pages 1835–1844, 2016.

[203] S. Saha, A. Adiga, B. A. Prakash, and A. Vullikanti. Approximation Algorithms for Reducing the Spectral Radius to Control Epidemic Spread. In *SIAM SDM*, 2015.

[204] H. Saito. Analysis of Geometric Disaster Evaluation Model for Physical Networks. *IEEE/ACM Transactions on Networking*, 2015.

[205] P. Sambaturu, B. Adhikari, B. A. Prakash, S. Venkatramanan, and A. Vullikanti. Designing near-optimal temporal interventions to contain epidemics. 2020.

[206] N. I. Sapankevych and R. Sankar. Time series prediction using support vector machines: a survey. *IEEE Computational Intelligence Magazine*, 4(2), 2009.

[207] P. Sarkar, D. Chakrabarti, and M. Jordan. Nonparametric link prediction in dynamic networks. In *ICML*, 2012.

[208] Y. Satsangi, S. Whiteson, and F. A. Oliehoek. Exploiting submodular value functions for faster dynamic sensor selection. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[209] D. J. Schwab, R. F. Bruinsma, J. L. Feldman, and A. J. Levine. Rhythmogenic neuronal networks, emergent leaders, and k-cores. *Physical Review E*, 82(5):051911, 2010.

[210] E. Sefer and C. Kingsford. Diffusion archeology for diffusion progression history reconstruction. *Knowledge and information systems*, 49:403–427, 2016.

[211] D. Shah and T. Zaman. Detecting sources of computer viruses in networks: theory and experiment. In *sigmetric10*, pages 203–214, 2010.

[212] N. Shah, D. Koutra, T. Zou, B. Gallagher, and C. Faloutsos. TimeCrunch: Interpretable Dynamic Graph Summarization. In *KDD15*.

[213] J. Shaman, E. Goldstein, and M. Lipsitch. Absolute humidity and pandemic versus epidemic influenza. *American journal of epidemiology*, 173(2):127–135, 2010.

[214] J. Shaman and A. Karspeck. Forecasting seasonal outbreaks of influenza. *Proceedings of the National Academy of Sciences*, 109(50):20425–20430, 2012.

[215] H. Shao, K. Hossain, H. Wu, M. Khan, A. Vullikanti, B. A. Prakash, M. Marathe, and N. Ramakrishnan. Forecasting the Flu: designing social network sensors for epidemics. *SIGKDD epiDAMIK Workshop*, 2018.

[216] Z. Shen and N. Sundaresan. eBay: an E-commerce marketplace as a complex network. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 655–664. ACM, 2011.

[217] N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(Sep):2539–2561, 2011.

[218] L. Shi, H. Tong, J. Tang, and C. Lin. Vegas: Visual influence graph summarization on citation networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(12):3417–3431, 2015.

[219] E. Shim. Optimal strategies of social distancing and vaccination against seasonal influenza. *Mathematical Biosciences & Engineering*, 10(5&6):1615–1634, 2013.

[220] T. Soma and Y. Yoshida. Maximizing monotone submodular functions over the integer lattice. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 325–336. Springer, 2016.

[221] Y. Song, D. Zhou, and L.-w. He. Query suggestion by constructing term-transition graphs. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 353–362. ACM, 2012.

[222] G. W. Stewart. *Matrix perturbation theory*. 1990.

[223] P. W. Stone. Economic burden of healthcare-associated infections: an American perspective. *Expert review of pharmacoeconomics & outcomes research*, 9(5):417–422, 2009.

[224] K. Subbian, B. Prakash, and L. Adamic. Detecting Large Reshare Cascades in Social Networks. In *WWW*, pages 597–605, 2017.

[225] S. Sundareisan, J. Vreeken, and B. A. Prakash. Hidden Hazards: Finding Missing Nodes in Large Graph Epidemics. In *Proc. of SIAM International Conference on Data Mining (SDM)*, 2015.

[226] R. S. Sutton and A. G. Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[227] A. Tabassum, S. Chinthavali, L. Chen, and B. A. Prakash. Data Mining Critical Infrastructure Systems: Models and Tools. *IEEE Intell. Info. Bulletin*, pages 31–38, 2018.

[228] F. S. Tabataba, P. Chakraborty, N. Ramakrishnan, S. Venkatramanan, J. Chen, B. Lewis, and M. Marathe. A framework for evaluating epidemic forecasts. *BMC infectious diseases*, 17(1):345, 2017.

[229] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[230] J. D. Tamerius, J. Shaman, W. J. Alonso, K. Bloom-Feshbach, C. K. Uejio, A. Comrie, and C. Viboud. Environmental predictors of seasonal influenza epidemics across temperate and tropical climates. *PLoS pathogens*, 9(3):e1003194, 2013.

[231] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei. Line: Large-scale information network embedding. In *WWW*, pages 1067–1077. ACM, 2015.

[232] Y. Tang, Y. Shi, and X. Xiao. Influence Maximization in Near-Linear Time: A martingale Approach. In *SIGMOD*, pages 1539–1554, 2015.

[233] C. Tantipathananandh and T. Y. Berger-Wolf. Finding communities in dynamic social networks. In *ICDM11*.

[234] J. B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[235] P. S. Thomas, B. C. da Silva, A. G. Barto, S. Giguere, Y. Brun, and E. Brunskill. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468):999–1004, 2019.

[236] M. Tizzoni, P. Bajardi, C. Poletto, J. J. Ramasco, D. Balcan, B. Gonçalves, N. Perra, V. Colizza, and A. Vespignani. Real-time numerical forecast of global epidemic spreading: case study of 2009 A/h1N1pdm. *BMC medicine*, 10(1):165, 2012.

[237] H. Toivonen, F. Zhou, A. Hartikainen, and A. Hinkka. Compression of weighted graphs. In *KDD*, 2011.

[238] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *CIKM*, 2012.

[239] H. Tong, B. A. Prakash, C. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau. On the vulnerability of large graphs. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 1091–1096. IEEE, 2010.

[240] J. Travers and S. Milgram. The small world problem. *Phychology Today*, 1:61–67, 1967.

[241] O. Tsur and A. Rappoport. What' s in a Hashtag?: Content Based Prediction of the Spread of Ideas in Microblogging Communities. In *WSDM*, pages 643–652, 2012.

[242] N. C. Valler, B. A. Prakash, H. Tong, M. Faloutsos, and C. Faloutsos. Epidemic spread in mobile ad hoc networks: Determining the tipping point. In *International Conference on Research in Networking*, pages 266–280. Springer, 2011.

[243] E. van Kleef, J. V. Robotham, M. Jit, S. R. Deeny, and W. J. Edmunds. Modelling the transmission of healthcare associated infections: a systematic review. *BMC infectious diseases*, 13(1):e1001172, jun 2013.

[244] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[245] S. R. Venna, A. Tavanaei, R. N. Gottumukkala, V. V. Raghavan, A. Maida, and S. Nichols. A novel data-driven model for real-time influenza forecasting. *bioRxiv*, page 185512, 2017.

[246] S. Volkova, E. Ayton, K. Porterfield, and C. D. Corley. Forecasting influenza-like illness dynamics for military populations using neural networks and social media. *PloS one*, 12(12):e0188941, 2017.

[247] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[248] D. Wang, P. Cui, and W. Zhu. Structural deep network embedding. In *KDD*, 2016.

[249] L. Wang, J. Chen, and M. Marathe. DEfsI: Deep learning based epidemic forecasting with synthetic information. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9607–9612, 2019.

[250] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang. Community Preserving Network Embedding. pages 203–209, 2017.

[251] J.-R. Wen, J.-Y. Nie, and H.-J. Zhang. Clustering user queries of a search engine. In *Proceedings of the 10th international conference on World Wide Web*, pages 162–168. acm, 2001.

[252] J. J. Whang, I. S. Dhillon, and D. F. Gleich. Non-exhaustive, overlapping k-means. In *SDM*, pages 936–944. SIAM, 2015.

[253] B. Wiegmans. Gridkit extract of entso-e interactive map. *Zenodo: Oldenburg, Germany*, 2016.

[254] Y. Xia and D. Tse. Inference of Link Delay in Communication Networks. *IEEE Journal on Selected areas in Communications*, 2006.

[255] H. Xiao, P. Rozenshtein, N. Tatti, and A. Gionis. Reconstructing a cascade from temporal observations. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 666–674. SIAM, 2018.

[256] J. Xie, R. Girshick, and A. Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pages 478–487, 2016.

[257] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015.

[258] P. Yanardag and S. Vishwanathan. Deep graph kernels. In *KDD*, 2015.

[259] J. Yang and J. Leskovec. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 587–596. ACM, 2013.

[260] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical Attention Networks for Document Classification. In *HLT-NAACL*, pages 1480–1489, 2016.

[261] Q. Yuan, E. O. Nsoesie, B. Lv, G. Peng, R. Chunara, and J. S. Brownstein. Monitoring influenza epidemics in china with search query from baidu. *PloS one*, 8(5):e64323, 2013.

[262] F. Zhang. *Matrix theory: basic results and techniques*. Springer Science and Business Media, 2011.

[263] H. Zhang, D. D. Yao, and N. Ramakrishnan. Detection of stealthy malware activities with traffic causality and scalable triggering relation discovery. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 39–50. ACM, 2014.

[264] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th international conference on World Wide Web*, pages 221–230. ACM, 2007.

[265] N. Zhang, Y. Tian, and J. M. Patel. Discovery-driven graph summarization. In *ICDE*, 2010.

[266] Q. Zhang, N. Perra, D. Perrotta, M. Tizzoni, D. Paolotti, and A. Vespignani. Forecasting seasonal influenza fusing digital indicators and a mechanistic disease model. In *Proceedings of the 26th International Conference on World Wide Web*, pages 311–319. International World Wide Web Conferences Steering Committee, 2017.

[267] W. Zhang, P. Lin, N. Wang, C. Nicholson, and X. Xue. Probabilistic prediction of postdisaster functionality loss of community building portfolios considering utility disruptions. *Journal of Structural Engineering*, 144(4):04018015, 2018.

[268] Y. Zhang, B. Adhikari, S. T. Jan, and B. A. Prakash. Meike: Influence-based communities in networks. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, pages 318–326. SIAM, 2017.

[269] Y. Zhang, A. Adiga, S. Saha, A. Vullikanti, and B. A. Prakash. Near-optimal algorithms for controlling propagation at group scale on networks. *IEEE Transactions on Knowledge and Data Engineering*, 28(12):3339–3352, 2016.

[270] Y. Zhang and B. A. Prakash. DAva: Distributing Vaccines over Networks under Prior Information. In *Proceedings of the SIAM Data Mining Conference*, SDM ' 14, 2014.

[271] Y. Zhang and B. A. Prakash. Scalable Vaccine Distribution in Large Graphs given Uncertain Data. In *Proceedings of the 23st ACM international conference on Information and knowledge management*. ACM, 2014.

[272] Y. Zhang, A. Ramanathan, A. Vullikanti, L. Pullum, and B. A. Prakash. Data-driven immunization. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 615–624. IEEE, 2017.

[273] Z. Zhang and O. Nasraoui. Mining search engine query logs for query recommendation. In *Proceedings of the 15th international conference on World Wide Web*, pages 1039–1040. ACM, 2006.

[274] V. Zlatić, M. Božičević, H. Štefančić, and M. Domazet. Wikipedias: Collaborative web-based encyclopedias as complex networks. *Physical Review E*, 74(1):016115, 2006.