Finding Interesting Subgraphs with Guarantees

Jose Cadena

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 Applications

Anil Vullikanti, Chair
Madhav Marathe
Naren Ramakrishnan
Chang-Tien Lu
Goran Konjevod

December 4, 2017
Blacksburg, Virginia

Keywords: Graph Mining, Graph Algorithms, Anomaly Detection, Finding Subgraphs,
Parameterized Complexity, Distributed Algorithms

Finding Interesting Subgraphs with Guarantees

Jose Cadena

(ABSTRACT)

Networks are a mathematical abstraction of the interactions between a set of entities, with extensive applications in social science, epidemiology, bioinformatics, and cybersecurity, among others. There are many fundamental problems when analyzing network data, such as anomaly detection, dense subgraph mining, motif finding, information diffusion, and epidemic spread. A common underlying task in all these problems is finding an "interesting subgraph"; that is, finding a part of the graph—usually small relative to the whole—that optimizes a score function and has some property of interest, such as connectivity or a minimum density.

Finding subgraphs that satisfy common constraints of interest, such as the ones above, is computationally hard in general, and state-of-the-art algorithms for many problems in network analysis are heuristic in nature. These methods are fast and usually easy to implement. However, they come with no theoretical guarantees on the quality of the solution, which makes it difficult to assess how the discovered subgraphs compare to an optimal solution, which in turn affects the data mining task at hand. For instance, in anomaly detection, solutions with low anomaly score lead to sub-optimal detection power. On the other end of the spectrum, there have been significant advances on approximation algorithms for these challenging graph problems in the theoretical computer science community. However, these algorithms tend to be slow, difficult to implement, and they do not scale to the large datasets that are common nowadays.

The goal of this dissertation is developing scalable algorithms with theoretical guarantees for various network analysis problems, where the underlying task is to find subgraphs with constraints. We find interesting subgraphs with guarantees by adapting techniques from parameterized complexity, convex optimization, and submodularity optimization. These techniques are well-known in the algorithm design literature, but they lead to slow and impractical algorithms. One unifying theme in the problems that we study is that our methods are scalable without sacrificing the theoretical guarantees of these algorithm design techniques. We accomplish this combination of scalability and rigorous bounds by exploiting properties of the problems we are trying to optimize, decomposing or compressing the input graph to a manageable size, and parallelization.

We consider problems on network analysis for both static and dynamic network models. And we illustrate the power of our methods in applications, such as public health, sensor data analysis, and event detection using social media data.

Finding Interesting Subgraphs with Guarantees

Jose Cadena

(GENERAL AUDIENCE ABSTRACT)

Networks are a mathematical abstraction of the interactions between a set of entities, with extensive applications in social science, epidemiology, bioinformatics, and cybersecurity, among others. There are many fundamental problems when analyzing network data, such as anomaly detection, dense subgraph mining, motif finding, information diffusion, and epidemic spread. A common underlying task in all these problems is finding an "interesting subgraph"; that is, finding a part of the graph—usually small relative to the whole—that optimizes a score function and has some property of interest, such as being connected.

Finding subgraphs that satisfy common constraints of interest is computationally hard, and existing techniques for many problems of this kind are heuristic in nature. Heuristics are fast and usually easy to implement. However, they come with no theoretical guarantees on the quality of the solution, which makes it difficult to assess how the discovered subgraphs compare to an optimal solution, which in turn affects the data mining task at hand. For instance, in anomaly detection, solutions with low anomaly score lead to sub-optimal detection power. On the other end of the spectrum, there have been significant progress on these challenging graph problems in the theoretical computer science community. However, these techniques tend to be slow, difficult to implement, and they do not scale to the large datasets that are common nowadays.

The goal of this dissertation is developing scalable algorithms with theoretical guarantees for various network analysis problems, where the underlying task is to find subgraphs with constraints. One unifying theme in the problems that we study is that our methods are scalable without sacrificing theoretical guarantees. We accomplish this combination of scalability and rigorous bounds by exploiting properties of the problems we are trying to optimize, decomposing or compressing the input graph to a manageable size, and parallelization.

We consider problems on network analysis for both static and dynamic network models. And we illustrate the power of our methods in applications, such as public health, sensor data analysis, and event detection using social media data.

Dedication

I dedicate this dissertation to my dearest parents and brother for their unconditional love, support, and encouragement.

Acknowledgments

I owe thanks to so many people who made the completion of this dissertation possible.

First and foremost I thank my PhD advisor, Anil Vullikanti, for being the best guide and mentor that a young researcher could ask for. I cannot thank Anil enough for patiently teaching me how to approach technical problems and productively reading scientific material, encouraging me, believing in the quality of my work, and always playing an active role as an advisor throughout my time in graduate school, despite all his numerous time commitments inside and outside the lab.

I thank Madhav Marathe, Naren Ramakrishnan, Chang-Tien Lu, and Goran Konjevod for serving in my PhD committee and providing their time and valuable feedback for my dissertation. I have had the opportunity to work with all of them and their students throughout my time at Virginia Tech, and I hope that we continue collaborating for many years to come.

I thank Madhav for having me in the Network Dynamics and Simulation Science Laboratory. I was privileged to work with a group of world-class researchers and experts on network science. I believe not many computer science students have the opportunity of being exposed to large-scale, multidisciplinary problems from day one in graduate school. I will wholeheartedly miss everyone at NDSSL.

I thank Naren for allowing me to be a part of the EMBERS project, which funded most of my PhD and got me started in data science and network analysis. This was a great experience, and I enjoyed working with Naren, Dr. Lu, and their students.

I thank Goran and the computational engineering division at Lawrence Livermore National Laboratory for hosting me for three summers at Livermore and for allowing me to experience how research is done outside an academic environment.

This dissertation would not have been possible without all my collaborators: Gizem Korkmaz, Chris Kuhlman, Saliya Ekanayake, Arinjoy Basak, Xinwei Deng, Achla Marathe, Charu Aggarwal, and Feng Chen. In particular, I thank Feng for his valuable insight and ideas for scalability in my scan statistics work, Charu for proposing that we work on the signed dense subgraph problem, and Saliya for his expertise on distributed algorithms.

I would also like to thank all the people who reviewed my papers before submission and

patiently sat through half-baked, poorly-delivered presentations while I was preparing for my preliminary exam and dissertation defense. I thank S. S. Ravi for all his feedback in numerous papers. I thank Elizabeth Tran for keeping me accountable and helping me set deadlines that I was not willing to miss in the crucial months preceding my preliminary exam, as well as for patiently reviewing my scan statistics work in all its many forms: papers, slides, and posters. I thank Aarathi Raghuraman for listening to my preliminary exam presentation multiple times, always listening intently and always asking interesting questions. I am forever grateful to Phan Nguyen for the Herculean task of listening to my 1-hour LLNL interview talk a total of eight times and giving me helpful feedback every single time. And I thank Vanessa Cedeno, Ryan Graham, and Arinjoy Basak for helping me practice my defense presentation.

When I was not in the lab or in class, I was fortunate enough to have friends with whom to share the graduate school experience. Living for over five years in a college town like Blacksburg, one sees many people come and go, and it would be difficult to exhaustively list everyone who made my time in Blacksburg a little better. I thank Sudarshan Aji and the R-Bar crowd for many fun weekend nights. I also thank the NDSSL coffee club folks for the nice mental break and caffeine recharge every afternoon. I thank Manpreet Hora and Aarathi Raghuraman for many fun dinners, potlucks, and random life conversations. I thank Iccha Sethi for her constant support, encouragement, and friendship. Especially, I thank my academic older sister, first scientific collaborator, and true friend Gizem Korkmaz for being there from the beginning of my PhD to the very end. The fact that she came to town just for my defense is something that will stay forever in my mind.

Last but not least, I thank my family without whom none of this would have ever been possible. I thank my father Jorge Washington for his love, encouragement, and always believing in me. I thank my brother Jorge Eduardo for always being there for me, lovingly and patiently. He has this wonderful ability to adopt some of my few virtues without absorbing my many flaws. I am forever thankful to Carlos and Carmen Izquierdo. Carlos was the first person to suggest that I should come to the United States to pursue higher education, believing in my academic success with a certainty that far surpassed my own. He and Carmen opened their home in Tampa to me in a way that, up to that point in my life, I thought was reserved to people related by blood. They have my eternal gratitude. Most of all, I thank my mother Narciza for her precious life lessons on the value of education, responsibility, gratitude, optimism in the face of adversity, and so many others I cannot fit here. I thank her for all her love and selfless sacrifice that far surpass the duties of a mother, and for her constant emotional and financial support, without which dreams and aspirations would have been just that. I recognize that I was born with many physical, social, and economic privileges that today allow me to write a PhD dissertation. But the single biggest privilege I was given—an unfair life advantage, one could say—is my mother, and everything I accomplish, big and small, will always be thanks to her.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

A network or graph is a mathematical representation of the interactions between a set of entities, providing a very powerful framework when studying a phenomenon in which interactions are of interest. In the past few decades, networks have been used extensively to study complex systems, such as (1) social networks [159,160,258] and contact networks [31,32,62], where entities are people and an edge represents friendship or contact, (2) computer networks [124,147,164], where entities are computers and an edge represents that two computers are directly connected to each other, (3) biological systems [83,197,204] , where, for instance, entities are proteins and edges represent bindings.

There are many fundamental problems in network analysis and graph mining, including frequent subgraph mining [120,263,267], motif finding [152,219], anomaly detection [9], dense subgraph mining [23, 51, 251], information diffusion [29, 98, 128], and epidemic spread [171, 253], among others. A common underlying task in these problems is finding an "interesting subgraph"; that is, finding a part of the graph—usually small relative to the whole—that has some property of interest. Consider, for instance, the anomaly detection problem. In Figure 1.1, we show four snapshots of a sensor network. This type of network has been used in a water distribution system [195] to detect pollution. Each node is a sensor, and an edge represents a water pipe between two sensors. We would like to detect pollution in the network, so that remedial action is taken. However, a sensor could become active due to noisy observations; similarly, a sensor that should be active may fail to detect pollution. For example, at times 2 and 3 in the figure, we observe some active sensors (colored blue), but these are not indicative of pollution. At time 4, on the other hand, we find a large subgraph of adjacent active sensors, which has a low likelihood of being just noise. Note, however, that we may have to include some inactive nodes if this allows us to discover a larger anomalous cluster. This event detection problem can be formalized as follows: given a sensor graph

$G(V, E)$ and a score function $f : 2^V \rightarrow \mathbb{R}$, we want to find a subset of connected nodes $S \subseteq V$, such that $f(S)$ is maximized. Here, f is a function that increases on the number of active sensors. We study this problem further in Chapter 3.

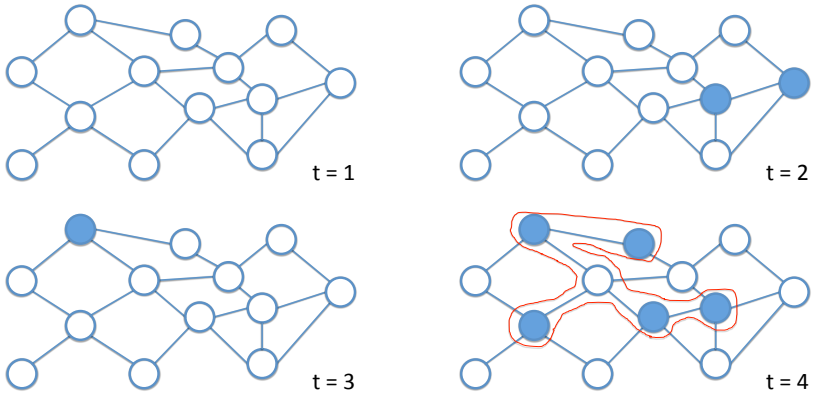


Figure 1.1: **Finding subgraphs for anomaly detection.** Four snapshots of a network of sensors. A blue node (sensor) indicates pollution at that part of the network. However, individual sensors may become active due to noise. This is the case at times 2 and 3. However, time 4 shows a large subgraph of active sensors. This event detection problem can be cast as finding a connected subgraph with a high proportion of blue nodes—possibly connected by some white nodes. In this case, we would like to detect the graph circled in red at time 4.

Finding subgraphs with a particular structure or obeying constraints, such as connectivity or minimum weight, is computationally hard. In particular, finding a specific subgraph in a network is an instance of the well-known subgraph isomorphism problem [252]. Similarly, the task of finding subgraphs with connectivity and weight constraints belongs to the family of Network Design problems [123], which is NP-Complete in general.

Over the last few decades, there have been significant advances on approximation algorithms for these challenging graph problems. One celebrated result, for instance, is the Primal-Dual schema of Goemans and Williamson [92] in 1995, which made it possible to develop algorithms for many network design problems with an approximation factor[1] of 2. However, *even though approximation algorithms offer rigorous guarantees on the quality of the solution, many of them tend to be impractical and do not scale to the large datasets that are common nowadays.*

As an alternative to approximation algorithms, there is a large body of work on heuristic methods for network analysis in the data mining community. State-of-the-art methods for various subgraph detection tasks are usually of this kind, and they have been shown to have very good empirical performance while, at the same time, being very efficient and scalable. Two areas of network analysis where heuristics have been very successful are

[1]We say that an approximation algorithm A for a minimization (maximization) problem has an approximation factor of α if, for any instance of the problem, A finds a solution S_A with value at most $\alpha \times \text{value}(S^*)$ (at least $\frac{1}{\alpha} S^*$ for maximization), where S^* is the optimal solution for the instance.

frequent subgraph mining and graph anomaly detection. In both areas, almost all of the existing methods are heuristics (see [120] and [9] for surveys on frequent subgraph mining and graph anomaly detection algorithms, respectively).

Despite the aforementioned successes, one notable drawback of heuristics is that they seldom offer rigorous theoretical guarantees on the quality of the solutions discovered. This raises some problems:

1. Because of the lack of guarantees, heuristics may perform poorly in some datasets. Many times, it is possible to show that an algorithm will have arbitrarily poor performance in graphs with a particular structure. Poor performance on a particular problem formulation affects the graph mining task at hand as well. For example, if anomaly detection is posed as an optimization problem, and a particular heuristic finds a solution far from optimal, then, we will fail to detect the desired anomalous events in the graph. Thus, whenever possible, it is desirable to have guarantees on the objective value of a solution.

2. As a complement to the previous point, we note that for most state-of-the-art heuristics, it is not known under what conditions the method will perform well and when it will be unacceptably suboptimal. When a heuristic is proposed, empirical evaluation only covers a handful of datasets, and, therefore, the empirical performance reported only holds for a limited class of networks. It is not clear that a method will have good performance in general.

3. Lastly, the lack of guarantees also makes it challenging to compare two competing heuristics. Again, evaluation is usually performed in a few datasets, and it is not clear whether one method is always superior to the other or whether both are better in different parts of the problem space. Furthermore, comparison studies evaluating different heuristics for the same problem are not performed frequently. For instance, despite the extensive work in frequent subgraph mining, we are only aware of one comparison study [263].

As an example of the points above, we discuss the `NPHGS` heuristic algorithm of [54], used to detect anomalies of the type illustrated in Figure 1.1 (i.e., with connectivity constraints). Briefly, `NPHGS` builds an anomalous connected subgraph S by starting at an active node v and trying to add neighbors of v to S if it increases the objective value $f(S)$. For the example in Figure 1.1, this algorithm would only detect two of the adjacent active sensors instead of the optimal set of 6 nodes that includes the inactive sensor.

1.2 Overview and Contributions

The focus of this dissertation is developing scalable algorithms with good theoretical guarantees for various network analysis problems. In particular, we make contributions to the areas of anomaly detection, dense subgraph mining, motif finding, and the study of diffusion processes in networks, namely information diffusion and epidemic spread. In all these problems, we are given a graph or graph time series with attributes—e.g., node weights, edge costs—and we want to find a subgraph that optimizes some function of the attributes, subject to constraints like connectivity, density, or subgraph isomorphism.

We find interesting subgraphs with guarantees by adapting techniques from parameterized complexity [12, 139], convex optimization [52, 79], and submodularity optimization [142]. While these techniques are now standard in the algorithm design literature, they lead to slow and impractical algorithms. One major technical contribution in each chapter is keeping the resulting algorithms scalable without losing the theoretical guarantees of these techniques. We accomplish this by exploiting properties of the functions that we are trying to optimize, decomposing or compressing the input graph to a manageable size, and parallelization.

We organize this dissertation into two main parts: 1) Using Static Properties and 2) Dynamics on Networks. In static graphs, functions capture a notion of anomaly score or density. In the context of dynamics, weights capture dynamical properties. The problems discussed in each part are summarized in Table 1.1.

Table 1.1: Dissertation organization

Part	Research Problem	Chapter
I: Using Static Properties	Anomalous Subgraph Detection	3–5
	Motif Finding	6
	Dense Subgraph Mining	7
II: Dynamics on Networks	Epidemic Spread	8
	Information Diffusion	9

1.2.1 Part I: Using Static Properties

In most applications, we analyze static network snapshots. This can be either a single snapshot or a time series of such snapshots—if what we are modeling is a dynamic process. We focus on three network analysis problems.

1.2.1.1 Anomalous Subgraph Detection

Anomaly detection on graphs is the task of finding a part of the graph (i.e., nodes, edges, subgraphs) where some "strange" or "unusual" behavior is taking place. Anomalies have

been defined in terms of the edges of the network (i.e. anomalous interactions between nodes) [40, 105, 174, 203, 256] and in terms of the nodes (i.e. members of the network who behave differently than other members) [8, 21, 232]. Anomalies have also been defined in terms of a subgraphs, as in [194, 237], where the authors model anomalies as structural patterns in a graph that have high compression cost according to the minimum description length principle (MDL) [33].

A number of methods have been proposed for anomaly detection in different applications (see Chapter 2). Among them, we have *window-based* methods, which are used to find anomalies in network time series. Usually, in this category, the anomaly detection task is posed as an optimization problem where the goal is to find the subgraph(s) that maximizes some distance function between a network at time t and the networks in the *time window* $[t - W, t - 1]$, for a window size W.

One widely popular type of window-based methods is *scan statistics*. This methodology typically involves formalizing a function of "anomalousness" and finding a subset that optimizes this function [233]. These were originally developed for spatial data and involve finding clusters that maximized a discrepancy score [144, 173, 183, 184, 185, 188]. Later, scan statistics were extended to network data by considering scores for *connected subgraphs* [53, 54, 233], and they have been applied to a number of domains, such as epidemiology [231], biological systems [101, 156], and social network analysis [54].

As a result of the diversity of applications, a large number of scan statistics have been independently developed, and, with them, many heuristic algorithms have been proposed. One challenge of anomaly detection based on scan statistics is that maximizing functions over connected subgraphs falls under the NP-Hard family of *Network Design* problems. In particular, many commonly used scan statistics have ties to the well-known Prize-Collecting Steiner Tree problem [122]. Heuristics for these problems have been used for network scan statistics [40, 212, 231, 233], but they do not give any rigorous guarantees on the solution quality.

Scan Statistics Optimization with Guarantees (Chapter 3): We develop a unified framework for designing algorithms for optimizing a large class of scan statistics for networks subject to connectivity constraints. Our algorithms provide rigorous guarantees on the objective value obtained; in contrast, most existing methods are heuristics with no bounds on quality. Extensive empirical evidence demonstrates the effectiveness and efficiency of our proposed algorithms in comparison with state-of-the-art methods. Further, our algorithms scale to networks with up to a million nodes, which is 1-2 orders of magnitude larger than prior applications. Our contributions are

1. *Rigorous algorithms.* We develop a *unified framework for optimizing a large class of parametric and non-parametric scan statistics for networks with connectivity constraints*, which scales linearly in the network size and as a function of a parameter defined as the "effective solution size". Our framework can be used to find provably

nearly-optimal solutions for over 20 scan statistics functions from the literature (Table 3.1); this contrasts with prior methods, which are developed for specific classes of functions.

2. *Flexibility.* The same core algorithmic techniques apply to all these network scan statistics. Furthermore, our approach also holds for the extensions of these functions with both node and edge weights, which generalizes Steiner connectivity problems.

3. *Scaling.* In practice, the effective solution size parameter is very small, making the time complexity of our algorithms better than prior methods, which are super-linear in the network size. Additionally, we develop a preprocessing and refinement technique that reduces the solution size without degrading the quality score beyond a provable constant factor. The resulting algorithms are able to scale to graphs with over a million nodes in minutes and are significantly faster than all the state-of-the-art baseline methods, which have only been run on graphs of up to 10^4 nodes.

4. *Interesting subgraphs in Twitter.* As an application of our algorithms, we explore event detection in Twitter follower graphs in Mexico and Venezuela. We model interactions—i.e. retweets, replies, direct messages—between Twitter users. We find that the subgraphs detected using our methods contain chatter about important national civil unrest events. Figure 1.2 shows an example of two events, one for Venezuela and one for Mexico. For Venezuela, the predominant terms of the tweets relate to a protest organized in the city of Tachira demanding the liberation of local students, who had been put in jail in the previous days for protesting. For Mexico, the two most common words in the extracted tweets form the phrase "energy reform" referring to a bill recently proposed by Mexican president Enrique Pena Nieto that would have a significant impact in the economy of the country.

Faster Graph Scan Statistics Optimization Using Algebraic Fingerprints (Chapter 4): We adapt an algebraic technique from the parameterized complexity literature to develop improved algorithms for scan statistics optimization. The main advantage over the algorithms in Chapter 3 is a provably better bound in running time and memory while maintaining the same theoretical guarantees. Our contributions are

1. *Improved algorithms with guarantees for graph scan statistics.* We develop MultilinearScan, which uses the multilinear detection technique [136] for optimizing a large class of both parametric and non-parametric graph scan statistics with connectivity constraints.

2. *Parallel algorithm.* We develop a parallel version of MultilinearScan using Giraph, which allows us to scale to very large instances with over 40 million edges; all earlier sequential and parallel methods have been shown to run on networks with less than 2 million edges.

(a) (b) (c) (d)

Figure 1.2: Examples of events found in Twitter follower networks for Venezuela (top) and Mexico (bottom)

3. *Scalability.* Approx-MultilinearScan improves over the methods in Chapter 3 by more than three orders of magnitude with respect to the running time and 1–2 orders of magnitude with respect to memory usage, with less than 1% loss in accuracy. Our approach also enables search for solutions with much larger effective solution size, leading to better quality solutions in many cases.

4. *Interesting subgraphs of under-vaccination.* We illustrate our methods by applying them to the task of finding clusters of low vaccination in a geographical region—we consider the state of Minnesota in particular. A preview of our results is presented in Figure 1.3.

Scan Statistics with Uncertainty (Chapter 5): Most existing work in anomaly detection in networks implicitly assumes that the reported data is either correct or has only a negligible amount of noise, which is often not a realistic assumption. The observed counts in data have uncertainty and do not exactly match the real world due to reporting errors, geocoding errors, missing entries, etc. All these sources of uncertainty would affect the problem formulations and algorithms for anomaly detection, but they are especially relevant when using scan statistics because the anomaly score is formalized in terms of the log likelihood of occurrence of observed data. The only scan-statistics-based method that incorporates uncertainty is the Bayesian scan statistic proposed in [187] and later extended to the multivariate case [186]. The authors add a conjugate prior to standard frequentist scan statistics, which allow them to quantify the uncertainty by controlling the variance in the prior. However, in general, more complex distributional assumptions for uncertainty would lead to expressions that do not have a closed form and can only be approximately optimized via sampling. We use methods from the theory of stochastic optimization to formally characterize scan statistic

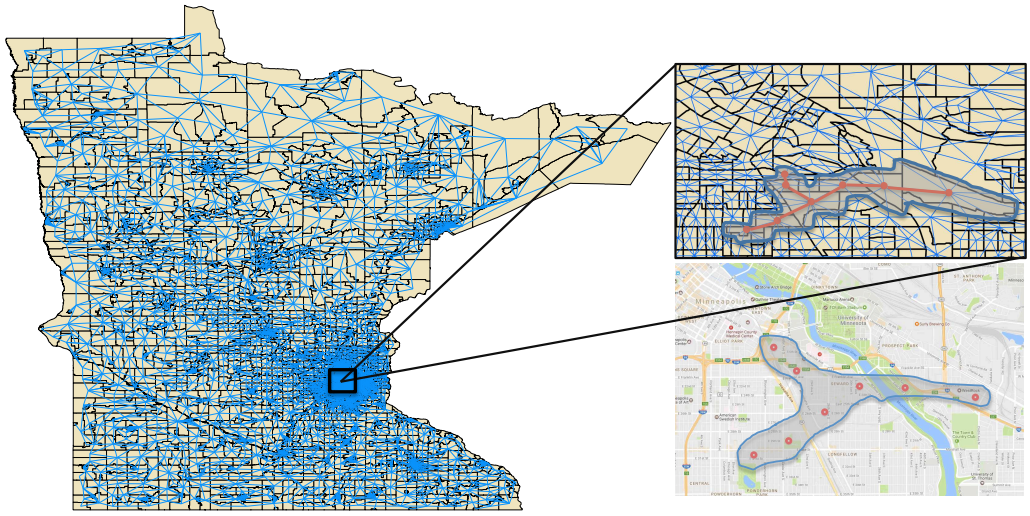


Figure 1.3: Under-vaccinated clusters in Minnesota

maximization with uncertainty and develop novel algorithms and heuristics for this problem. Our contributions are summarized below.

1. We propose two approaches for taking uncertainty into account: a sample average approximation (SAA) and a max-min formulation [37,202,222], and show that these are well motivated for scan statistics by analyzing instances with stochastic perturbations. When we account for uncertainty, the anomaly detection task becomes much more challenging. We show that, even without any connectivity constraints, finding clusters based on scan statistics that optimize the max-min objective function is NP-hard, whereas it can be done in linear time if there is no uncertainty.

2. We develop rigorous algorithms and heuristics for optimizing the scan statistics in both these formulations. For the SAA formulation, our algorithm AggregateSAA gives rigorous bounds on the approximation guarantee for finding solutions of a given "effective" size k, with or without connectivity constraints, and is a fixed parameter tractable algorithm. For the max-min formulation, we present two algorithms: (1) MaxMinLpRound for the case of no connectivity constraints, using linear programming rounding, which yields rigorous approximation bounds, and (2) a heuristic, BestMax, for the case with connectivity. Both AggregateSAA and BestMax use algorithms for scan statistics that we proposed in Chapter 3.

1.2.1.2 Distributed Algorithms for Motif Finding

Many problems in graph mining and social network analysis can be reduced to questions about different kinds of subgraphs; two important classes of such problems that we will focus on, are: (1) *Detecting subgraphs*, such as paths and trees, of a given size k—these are used for characterizing different kinds of networks, especially in biological models [11, 114]. (2) *Anomaly detection in network data using graph scan statistics*, which involves finding connected subgraphs of size k, optimizing different kinds of objectives [54, 101, 156, 182, 231].

Problems involving subgraph detection are NP-Hard in general, making them computationally very challenging. Various sequential heuristics have been proposed for both these problem classes [11, 54, 57, 101, 114, 156, 182, 231]. However, these do not scale to large instances, motivating the need for parallel implementations. There has been a fair amount of work on parallel algorithms for various kinds of subgraph problems [19, 22, 71, 218, 228, 229, 269, 270], but very little for anomaly detection using graph scan statistics, other than the work of [269] and our work in Chapter 4.

Most of these parallel algorithms involve heuristics based on partitioning, pruning, and enumeration for speeding up the computations to fairly large graphs—e.g., for maximal cliques [19,57,71,218,269]. One of the few rigorous approaches for subgraph counting that has been used for developing parallel algorithms is based on the "color coding" technique [11, 12], which gives a rigorous *fixed parameter tractable* algorithm for finding and counting subgraphs. Color coding is a dynamic programming algorithm, and multiple approaches have been developed for parallelizing it [228, 229, 270]. The FASCIA algorithm of [228, 229], which is implemented in MPI, is the state-of-the-art, and it can handle trees of size up to 12 in graphs with millions of nodes. We have used color coding to develop a sequential algorithm for graph scan statistics (Chapter 3), but no parallel adaptations exist so far. A big challenge of the color coding technique is the memory overhead, since it scales as 2^k, where k denotes the subgraph size, which limits further scaling to larger graphs or subgraphs.

Finding trees and anomalous subgraphs in parallel (Chapter 6). We develop a novel approach for designing highly scalable parallel algorithms for both the problem classes mentioned above, by adapting algebraic techniques for detecting multilinear terms in a multivariate polynomial developed by Koutis [136] and Williams [261]. Our contributions are

1. *Efficient algorithms for parallel multilinear detection and applications to subgraph analysis.* We develop a distributed MPI based algorithm for detecting multilinear terms with k variables of the form $x_{i_1} x_{i_2} \dots x_{i_k}$ (i.e., a term in which all variables have exponent 1) in a multivariate polynomial $P(x_1, \dots, x_n)$—this is referred to as the k-Multilinear Detection (k-MLD) problem [136, 261]. The problem of finding paths and trees can be reduced to the k-MLD problem [136]. We show that the network scan statistics problem can also be reduced to the k-MLD problem. Consequently, our parallel multilinear detection algorithm leads to parallel algorithms for both these problems. Our methods provably have better space and running time complexity

compared to the color coding technique. The space complexity is $O(kn)$ compared to $O(2^k n)$, and the running time complexity is proportional to $O(2^k)$, compared to $O(2^k e^k)$ [228, 229, 270].

2. *Comparison with prior methods.* Our algorithm for finding paths gives over two orders of magnitude improvement in time compared to FASCIA, the state of the art method based on color coding [228, 229]. There is only one parallel algorithm for network scan statistics [269], which is based on pruning heuristics, and our method improves over it by several orders of magnitude.

3. *Comparison with Giraph.* We compare our methods with recent parallel graph programming models, Giraph and Spark. Our results show that neither of these models scale past 40 million edges for these problems; further, Spark has worse performance than Giraph.

1.2.1.3 Dense Subgraph Mining

A common subproblem that arises in network analysis is that of dense subgraph mining, which has applications to bioinformatics [83], social network analysis [17, 258], community detection [230], and anomaly detection. For unsigned networks (i.e., networks where all edge weights are non-negative), there are many notions of density, such as the average degree, edge density, and triangle density [14, 23, 24, 51, 250, 251]. The two latter ones can be computed more efficiently; however, as discussed by Tsourakakis et al. [251], these notions might not give the densest subgraph in real networks, and the authors propose a different notion called the Optimal QuasiClique, which does much better at finding dense networks.

One limitation of existing algorithms and notions of density is that they are restricted to unsigned networks. Even though there is extensive work on dense subgraph mining for unsigned networks, the problem has been relatively under-studied in signed networks. Furthermore, the algorithms developed for the unsigned setting do not extend to the signed case.

Dense Subgraphs in Signed Networks (Chapter 7): We formalize the problem of finding dense subgraphs in signed network and present algorithms to find such subgraphs. Specifically, our contributions are as follows.

1. *Formalizing density problems and event detection in signed networks*: We introduce the Generalized Optimal Quasiclique (GOQC) problem for dense subgraph mining in signed networks and show that the problem is NP-complete. We also show that event detection in network time series can be naturally formalized using GOQC.

2. *Algorithms for GOQC*: We develop an algorithm called DenSDP for GOQC, using semidefinite programming (SDP) based rounding, which gives an $O(\log n)$ approximation to the optimal solution under certain conditions. In practice, the approximation

factor is much better. Furthermore, DenSDP can be easily modified to admit additional constraints of practical interest, such as finding dense subgraphs containing a specific set of query nodes. Although DenSDP runs in polynomial time, it does not scale very well to large networks. Motivated by the low diameter of dense subgraphs, we design another algorithm, EgoScan, with significantly faster running time, by modifying DenSDP using heuristics for pruning and scanning neighborhoods (ego-networks) of bounded size.

3. *Detection of dense subgraphs in signed and unsigned networks*: We evaluate DenSDP and EgoScan in more than 20 real networks and observe that they find solutions with very high density, significantly improving on adaptations of the best prior methods for both signed and unsigned networks. With a variant of EgoScan that optimizes the edge density instead of the GOQC score, we obtain EgoScan-δ, an algorithm that gives solutions with much higher edge and triangle density than all earlier methods. The improvement in edge density over previous methods is as much as 85% and usually over 50%. These results are consistent across signed and unsigned networks in different domains. The improvement in performance is even more significant for the constrained version involving finding subgraphs containing a subset of query nodes.

4. *Event detection using GOQC*: We use our approach for event detection in real datasets for which we have suitable ground truth events. We find that our method based on EgoScan significantly outperforms existing approaches and baseline methods in terms of the precision-recall tradeoff (by as much as 25-50% in some instances).

5. *Interesting subgraphs in political databases.* The dense subgraphs found by EgoScan are able to capture increase in protest activity in Venezuela in February 2014. Figure 1.4 shows a dense subgraph detected in the week of February 17, 2014. This week has many interactions between actors. In particular, civilians (CVL), government (GOV), and opposition (OPP) are connected to almost every other node in the subgraph. In contrast, the same set of nodes in the previous month is disconnected.

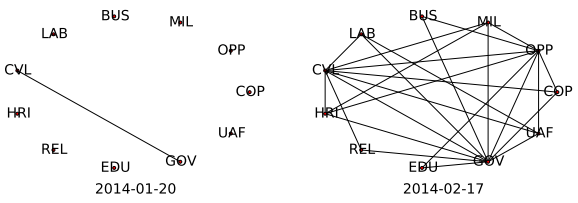


Figure 1.4: Dense Subgraph for February 17, 2014 in ICEWS Venezuela (right) compared with the corresponding graph from the prior month (left). The increased activity between the actors during February is consistent with nationwide protests in Venezuela.

1.2.2 Part II: Dynamics on Networks

It is only relatively recently that people have analyzed dynamical processes on networks instead of using the static snapshot approximation. Two main areas where graph dynamics are a natural methodology are epidemic spread—how a disease propagates in a social contact network—and information diffusion—how messages or rumors are disseminated. For the former area, we study the problem of finding *critical subgraphs* that can cause an epidemic outbreak even if the network is mostly immunized. For the latter, we study a particular model of information diffusion and provide theoretical bounds based on graph structure under which a message persists in a social network for a long time.

1.2.2.1 Finding Critical Subgraphs for Epidemic Spread

Infectious diseases constitute one of the largest causes of human mortality worldwide and account for more than 13 million deaths a year. Despite significant strides in medicine and public health practices, such as high vaccination coverage, large outbreaks continue to be an important source of concern. Further, there is limited understanding of the dynamics of infectious disease spread through human communities. For instance, despite the fact that the average vaccination rate for measles is over 95% in the United States, there were 667 cases of the disease in 2014 and 187 in 2015 according to the Centers for Disease Control and Prevention (CDC) [1]. This poses the question: are there pockets of the population that, if left unvaccinated, will spread a disease in spite of high vaccination rates elsewhere?

Critical Sets for Epidemic Control (Chapter 8): We focus on vaccination interventions for controlling epidemic spread under the SIR model. We formalize the problem of finding *critical epidemic subsets*; that is, subsets that cause a large disease outbreak in a population if left unvaccinated, even if the rest of the population has a high vaccination rate. We formalize this problem as the k-Critical Set problem, where we want to find a cluster of k nodes in a population that will maximize spread if left unvaccinated, even if the rest of the population follows a fairly strict vaccination policy. Our contributions are

1. *Critical sets for epidemic spread.* We formalize the notion of *criticality* of a subpopulation S—i.e., the expected number of infections if there is an outbreak in S and its inhabitants are not properly immunized. Then, we cast the problem of finding the most critical subpopulation as a constrained optimization problem. We propose the k-Critical Set (k-CSP) problem, which tasks us with finding a connected subpopulation of bounded size (at most k) that maximizes the spread of a disease. We show that k-CSP generalizes the Influence Maximization problem of Kempe et al. [128] and is an instance of submodularity maximization with size and connectivity constraints. We also propose the k-Critical Region (k-CRP) problem, which asks us to find a geographical area with high criticality. This problem formulation more readily translates to public policy than k-CSP.

2. *Algorithms for k-CSP and k-CRP.* We propose approximation algorithms and fast heuristics for k-CSP and k-CRP. Even though maximizing criticality is NP-Hard in general, we show conditions for which the function can be optimized in polynomial time by exploiting *local* structure. Loosely, in populations where most people are vaccinated, except for small clusters—which is the case with measles, for instance—the criticality function is *approximately modular*, and we can derive a polynomial-time algorithm with rigorous bounds. However, the resulting approximation algorithm does not scale, so we also propose a nearly linear time heuristic with good empirical performance.

3. *Experimental Results.* We evaluate our algorithm on a realistic, activity-based population and mobility model for the state of Minnesota, where we simulate an outbreak of a highly-infectious disease. We find that the sets we discover have very high criticality compared to reasonable baselines and heuristics commonly considered in the public health community.

1.2.2.2 Understanding Information Diffusion in Social Networks

Social unrest may also be studied as an infectious process. Recent news have drawn attention to happenings in Latin America, the Middle East, and Eastern Europe. Civilian populations mobilize, sometimes spontaneously and sometimes in an organized manner, to raise awareness of key issues or to demand changes in governing or other organizational structures. It is of key interest to social scientists and policy makers to forecast civil unrest using indicators observed on media like Twitter, news, and blogs.

Forecasting Social Unrest Using Activity Cascades (Chapter 9): We present an event forecasting model using a notion of activity cascades in Twitter (proposed by Gonzalez-Bailon et al., 2011) to predict the occurrence of protests in three countries of Latin America: Brazil, Mexico, and Venezuela. The basic assumption is that the emergence of a suitably detected activity cascade is a precursor or a surrogate to a real protest event that will happen "on the ground." Our model supports the theoretical characterization of large cascades using spectral properties and uses properties of detected cascades to forecast events. Our key contributions are focused on the following three questions:

1. *When do large activity cascades happen?* A common empirical observation is that cascades seldom become very large. We rigorously prove necessary and sufficient conditions for large cascades in terms of spectral properties of the underlying graph; these also imply a similar characterization for a class of Hawkes processes. We find that this characterization closely matches our empirical observations for synthetic traces. Specifically, our analyses show if the spectral radius of a cascade graph is below a particular constant, then large cascades are not possible. Our techniques build on approaches for analyzing the spread of epidemics [85, 200, 247] and are the first such results for cascades of this kind.

2. *Are there critical subsets of users that contribute to cascades?* We study the questions of identifying critical subsets of users responsible for formation and survival of cascades, and we formalize these as two complementary problems: CriticalSetFormation (CSFP) and CriticalSetShattering (CSSP). We show both to be NP-Complete, and we evaluate different greedy heuristics to approximate them empirically by studying large, monthly cascades for all (country, month) combinations for Brazil, Mexico, and Venezuela over a 15-month period. Our results for CSSP show that a very small set of users is critical for a cascade to exist—their non-participation causes the cascade to shatter. We also prove that a high-degree strategy gives a constant factor approximation for CSSP in random power law graphs. On the other hand, the results for CSFP suggest that unless a large fraction of users participate, a cascade cannot exist. Thus, one needs a reasonable fraction of users, plus some critical users for a cascade to exist.

3. *Can we forecast protests using activity cascades?* Since large cascades are not very common, their occurrence signals a big event. We analyze over 353 million tweets from three Latin American countries over a 1.5-year period, and we consider activity cascades formed by a filtered set of tweets. These tweets contain at least 3 keywords from a dictionary that has over 900 words in English, Spanish, and Portuguese, related to civil unrest activities. This ensures that the resulting activity cascades will be relevant to the topic of civil unrest. Next, we build a feature set based on the structural properties of the cascades, to be used as predictors of social unrest. Statistical models are then used to remove redundancies among features and make predictions of events from the reduced feature set. The model is tested on multiple countries for robustness and compared against a baseline model. We show that our approach can "beat the news," i.e., contribute a lead time of one to two days over the reporting of a protest in major news media, with an accuracy of over 0.75. It can even predict black swan events like the Brazilian Spring with an accuracy of 0.83.

In the following chapter, we discuss related work on graph mining and define various graph-theoretic concepts to be used in the rest of the dissertation.

Chapter 2

Preliminaries

2.1 Graph Theoretic Definitions

2.1.1 Graphs

A *network* or *graph* G is a tuple (V, E), where V is a set of *nodes* or *vertices* and $E \subseteq V \times V$ is a set of *edges* or *links*. We may denote the graph G as $G(V, E)$ when we want to be explicit about the node and edge sets. We may also write the node and edge sets as $V(G)$ and $E(G)$, respectively, when we want to emphasize that they are related to graph G. Intuitively, if u and v are nodes, an edge $(u, v) \in E$ represents a connection or interaction between the two nodes—for instance, an edge may represent a friendship relationship in a social network. If an edge (u, v) exists, we say that u and v are *connected*, or that u and v are *neighbors*. When it is clear from context, we will use n and m to denote the number of nodes and edges in a graph, respectively.

The most basic type of graph is the *simple graph*, which is undirected and unweighted, and doesn't have loops or parallel edges. Figure 2.1 shows an example of a simple graph. The edge (u, v) in a simple graph is the same as the edge (v, u) (i.e., the edge is an unordered pair).

Relationships are not always symmetric. In a network of scientific publications, for instance, a paper cites one other paper but not the other way around. We say that a graph is *directed* if each edge has a direction associated with it. In other words, in this case, the edge is an ordered pair. Let $(u, v) \in E$ be an edge of a directed graph. We say that u is the *tail* and v is the *head*. Figure 2.2 shows an example of a directed graph.

A graph is *weighted* if edge $e \in E$ has a weight $w(e)$ associated with it. Formally, let $w : E \to \mathbb{R}$ be a *weight function* that maps each edge to a real value. For an edge $e = (u, v)$, we may also write the weight as $w(u, v)$ when we want to be explicit about the endpoints.

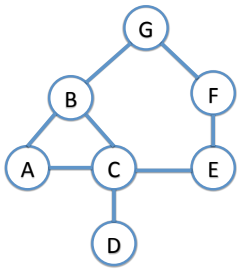


Figure 2.1: **Simple graph.** Graph $G(V, E)$ with node set $V = \{A, B, C, D, E, F, G\}$ and edge set $E = \{(A, B), (A, C), (B, C), (B, G), (C, D), (C, E), (E, F), (F, G)\}$.

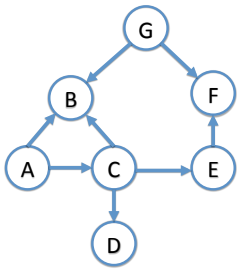


Figure 2.2: **Directed graph.**

Though in general, a weight can be any real number, in most applications, we assume that weights are only non-negative numbers. A graph whose weights are all non-negative is sometimes called *unsigned*; otherwise, we say the graph is *signed*.

A graph can also be *attributed*. Nodes and edges may have attributes associated with them. Attributes are formalized as functions that map from either the nodes or edges of the graph to some other set. Note that a weighted graph is an attributed graph whose edges have a weight attribute.

Let $G(V, E)$ be a graph. We say that a graph $H(V', E')$ is a *subgraph* of G if $V' \subseteq V$ and $E' \subseteq E$. We say that H is an *induced subgraph* of G if $V' \subset V$ and, for all pairs of nodes $u, v \in V'$, if $(u, v) \in E$, then $(u, v) \in E'$. In words, H has all the edges between nodes in V'

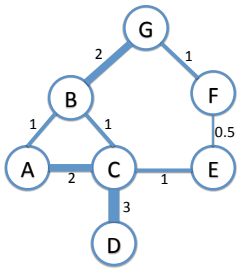


Figure 2.3: **Weighted graph.**

that appear in G. We may also say that the set V' *induces* the subgraph H. For example, the graph in Figure 2.4a is an induced subgraph of the graph in Figure 2.1; however, the graph in Figure 2.4b is not induced because the edge (A, C) is missing.

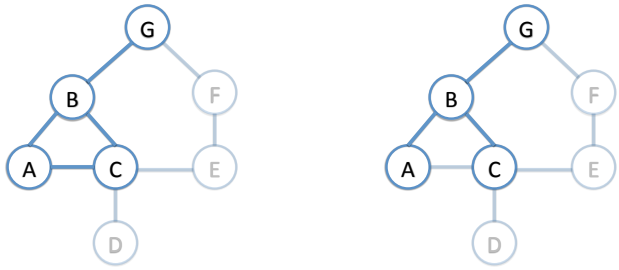


Figure 2.4: **Subgraph.** (a) An induced subgraph $H(V', E')$ of the graph in 2.1. $H' = \{A, B, C, D\}$ and $E' = \{(A, B), (A, C), (B, C), (B, G)\}$. (b) A subgraph that is not induced because the edge (A, C) is in G but not in H.

A *path* from node v_0 to node v_k is an ordered sequence of nodes and edges

$$v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, v_k,$$

such that the edge $e_i = (v_{i-1}, v_i)$ exists for all $i \in [1, k]$ and all the nodes in the path are distinct. The *length* of a path is k, the number of edges in the path. We say that node u is *reachable from* node v if there is at least one path from v to u. Note that a path in a graph G is a (not-induced) subgraph of G. Figure 2.5 shows an example of a path in a simple graph.

We say that a graph is *connected* if every node in the graph is reachable from any other node. If there is exactly one path between any pair of nodes, we say that the graph is a *tree*. We note that a path is a particular type of tree. Figure 2.1 is connected and the subgpraph in Figure 2.4b is a tree.

A graph $G(V, E)$ is *complete* if there are edges between every possible pair of nodes. In the case of simple graphs, there are $\binom{n}{2}$ edges. Figure 2.6 shows a complete graph of 5 nodes.

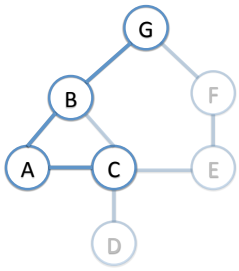


Figure 2.5: A path of length 3 for $v_0 = C, e_0 = (C, A), v_1 = A, e_1 = (A, B), v_2 = B, e_2 = (B, G), v_3 = G$

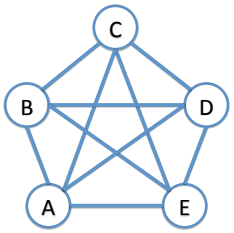


Figure 2.6: **Complete Graph.**

2.1.2 Graph Properties

The *degree* of a node in an undirected graph is the number of edges that are incident to the node. In a directed graph, the *in-degree* is the number of edges for which u is the head, and the *out-degree* is the number of edges for which v is the tail.

For a node u, let $Nbr(u)$ be the set of neighbors of u. The *ego-network* of u is the subgraph induced by u and all its neighbors. More generally, the *d-neighborhood* of a node u is the subgraph induced by u and all the nodes for which there is path of length d from u. Note that the ego-network is the 1-neighborhood.

For a weighted graph $G(V, E)$ with weight function $w(\cdot)$, we define the weight of the graph, $W(G)$ as the sum of the weight of its edges: $W(G) = \sum_{e \in E} w(e)$.

The *edge density* of a graph is the fraction of edges in the graph out of all the possible edges: $\frac{m}{\binom{n}{2}}$. The *average degree* of a graph is the number of edges divided by the number of nodes. $\frac{m}{n}$. These definitions can be extended to weighted graphs by replacing the number of edges (m) by the weight of the graph ($W(G)$).

2.2 Network Analysis and Mining

We provide a general overview of the research topics on graph mining relevant to this dissertation, namely graph anomaly detection, dense subgraph mining, and dynamical processes. These are by no means the only topics that have been considered. Other important problems under active research are frequent subgraph mining [43,65,109,112,116,120,150,193,263,264], clustering and community detection [30,61,77,82,104,108,163,170,191,192,198,210,211,248, 266], classification [48,126,168], link prediction [10,49,58,159,165,167,242], ranking and influence maximization [129,196].

2.2.1 Graph Anomaly Detection

Anomaly detection on graphs is the task of finding a part of the graph (i.e., nodes, edges, subgraphs) where some "strange" or "unusual" behavior is taking place. What constitutes strange behavior depends on the nature of the network. Anomalies have been defined in terms of the edges of the network (i.e. anomalous interactions between nodes) [40,105,174,203,256] and in terms of the nodes (i.e. members of the network who behave differently than other members) [8, 21, 232]. Anomalies have also been defined in terms of a subgraphs, as in [194, 237], where the authors model anomalies as structural patterns in a graph that have high compression cost according to the minimum description length principle (MDL) [33].

Akoglu et al. [9] give a good introduction to graph-based anomaly and event detection. They classify the different approaches depending on whether the graph at hand is *static*—a single snapshot—or *dynamic*—the nodes and/or edges change over time. Within static and dynamic graphs, the authors further classify existing methods depending on the graph characteristics that are used and the kind of events that are detected. We give some representatives examples of each class, but we will focus on the *window-based* methods.

Static Graphs. Informally, the problem of anomaly detection on a static graph calls for finding subgraphs that are significantly different than most of the "normal" patterns observed in that graph. One line of work uses **structural properties** of a network; these type of methods define anomalies as subgraphs whose structure is different than the rest of the graph. One notable example is OddBall [8] for identifying anomalous nodes. OddBall consists of two phases: first, we select a set of node-level features, such as the degree of the node, and the number of edges in its ego network. In the second phase, we plot the distribution of these features across all the nodes in the graph; then, a node is anomalous if its features differ significantly from the overall distribution. Another line of work leverages on the **community** of a graph to find anomalies. A *community* in a graph is loosely defined as a set of nodes that have many edges among each other (i.e., they are densely-connected) and few edges to other nodes of the graph. The community-based methods define anomalies as nodes or edges that do not clearly belong to any community; rather, these nodes act as "bridges" and lie in the boundary between two or more communities. An example of a community-based method is the work of Sun et al. [238] for community discovery and anomaly detection in bipartite graphs. The authors use an algorithm based on random walks on a graph to find the community of a query node v. If we start a random walk from v, and we visit node u many times, then, u and v have high likelihood of being in the same community. Anomalies are defined as links connecting nodes that have low likelihood of being in the same community.

The nodes or edges of a graph may also have **attributes**. For example, in a social network, attributes of a person (i.e., a node) would be their hometown, occupation, political and religious views, etc. This additional information may be used to define anomalous behavior. A well-known attributed-based method is the work of Noble and Cook [194]. The authors use an MDL approach for finding frequent subgraphs—subgraphs with low compression cost—

when each node has a label. The idea is that the opposite of "frequent" is anomalous, so graphs that are hard to compress are labeled as anomalous.

Dynamic Graphs. *Dynamic* or *time-evolving* graphs are time series of static graphs [9]. Each static graph in the series is called a snapshot; the nodes and/or edges of the graph may change from one snapshot to the next. The problem of anomaly detection in dynamic graphs may be summarized as follows: Given a dynamic graph, we want to find (1) a time stamp or time interval where an event or change point occurs, and (2) the subgraph where this change occurs. Akoglu et al. [9] split the different approaches to anomaly detection in dynamic graphs into four categories.

Proposed methods in the first category are **feature-based**. These methods first create a summary of each snapshot—for instance, converting the snapshot to a vector—and then compare consecutive snapshots using a distance function on the summaries. If the distance between two consecutive graphs is above some threshold, we say that there is a change point or anomaly between them. Two key challenges that feature-based methods have to solve are (1) how to create a summary of a graph, and (2) how to measure pairwise distance between summary representations of two graphs. For part (1), many ways of constructing a summary have been proposed. For instance, in [7], Akoglu and Faloutsos use singular value decomposition (SVD) on a matrix of node correlations to summarize a graph as a vector. For part (2) the authors use the cosine distance of these vectors as the distance function.

The second category focuses on **decomposition-based** events. These methods operate on the adjacency matrix representation of each snapshot or on the tensor representation of the full network time series. They are called decomposition-based because anomalies are discovered using a matrix [7, 239] or tensor [20, 140] decomposition.

A third class of methods discover **community-based** events. These methods define anomalies as snapshots in the time series whose community structure differs significantly from snapshots in the recent past. Peel and Clauset [198] propose a Bayesian model of community structure and a statistical test to detect change points in dynamic graphs. First, the authors use the snapshots in a time window $[t - W, t]$ to infer the community structure of the graph. Then, their method compares the community structure of snapshot $t+1$ with the one learned from the time window. Then, by computing the Bayes factor, we can asses the significance of the change. Algorithms based on the MDL principle have also been proposed.

In the last category, we have methods for **window-based** events. We first define a *time window* of past snapshots to model normal behavior. Subsequent snapshots are marked as anomalous if they differ significantly from the patterns observed in the time window. Usually, in this category, the anomaly detection task is posed as an optimization problem where the goal is to find the subgraph(s) that maximizes some distance function between the current snapshot and the time window. In this dissertation, we focus on window-based methods, so we discuss them further below.

2.2.1.1 Window-based methods

Scan Statistics. As observed in [233], scan statistics involve formalizing a notion of "anomalousness" for a subset of data, and then using a method to efficiently find a subset that optimizes a corresponding score. Originally scan statistics were developed for disease surveillance in spatial data and involved finding simple regions, such as disks, that maximized some form of discrepancy score based on a likelihood ratio test [144, 173, 183, 184, 188]). Later, scan statistics were extended to network data by considering scores for connected subgraphs. One of the earliest uses of scan statistics in networks is due to Priebe et al. [203], who use this methodology to find anomalies in the Enron dataset [133]. The authors define a score function that increases with the density of the k-neighborhood of a node. If a k-neighborhood is much denser than one would expect based on recent past observations, then it is marked as an anomaly. Subsequent work has focused on extending existing spatial scan statistics to the network setting, usually replacing the spatial proximity constraint by a connectivity constraint (i.e., the detected subgraph must be connected) [53, 54, 233, 256].

Because of connectivity constraints, optimizing scan statistics on graphs is challenging in general. Maximizing functions over connected subgraphs is an instance of the *Network Design* problem—this includes well-known graph optimization problems, such as the Minimum Spanning Tree (MST), Steiner Tree and its variants: Prize-Collecting Steiner Tree (PCST) and NetWorth, which are NP-hard. If we remove the connectivity requirement, then, the optimal value of a scan statistic can be computed in polynomial time because of a linear ordering property [53, 183, 185]. To attack the computational complexity of the problem, a number of heuristics have been developed for optimizing different scan statistics, and we summarize them in Table 2.1.

MEDEN and NetSpot. In [40], the authors define anomalies in a dynamic network as a connected subgraph with high total edge weight over some time interval. The authors formalize the anomaly detection task as the Heaviest Dynamic Subgraph (HDS) problem, where the goal is to find a subgraph and time interval where the subgraph has maximum total weight, over all connected subgraphs and time intervals. This problem is a variant of the Prize Collecting Steiner problem (PCST) [122] with NetWorth objective, for which there is no constant factor approximation, unless P=NP. The authors also propose a scalable heuristic called MEDEN to find an HDS and show applications to congestion detection in traffic networks. In posterior work, [178] propose NetSpot, a heuristic to find all non-overlapping subgraphs and time intervals whose weight is above some threshold. While this latter problem can be solved naively by repeated use of MEDEN, NetSpot is much faster.

EventAllPairs+ and EventTree+. Another window-based method was recently proposed by Rozenshtein et al. [212]. The authors define *activity networks*, where each node has a positive weight and each edge has a positive cost or distance. Node weights model the importance or intensity of the activity at that node—for instance, the number of posts made by a user in a social network. In order to find interesting events in these networks, the authors model activity using two different optimization problems based on network de-

sign. In the first formulation, the authors pose the problem of finding a connected subgraph with high weight and small total pairwise distance between the nodes (EventAllPairs+). The events found by solving this formulation correspond to circular and compact clusters of nodes in the network. The authors show that the objective function for this problem is submodular, so they can take advantage of existing results in submodularity optimization to design approximation algorithms with good theoretical bounds. The authors also show that this formulation is a variant of the MaxCut problem [93], which can be solved using standard semidefinite programming tools. One drawback of EventAllPairs+ is that the solutions for this problem correspond to clusters of circular shape, which limits the detection power; this approach would not discover, for instance, an anomaly with an elongated shape. This limitation is addressed by a second formulation called EventTree+, where the goal is to find a connected subgraph with high total node weight and minimum edge cost. This problem reduces to Prize Collecting Steiner Tree, and can be approximated using the primal-dual algorithm of Goemans and Williamson [94]. The authors also provide a simple heuristic, GreedyT, which is much faster and has similar empirical performance to the primal-dual approach. However, this heuristic does not have any theoretical guarantees.

Discussion. Many window-based algorithms have been proposed for graph anomaly detection; however, *no computationally tractable algorithms with rigorous guarantees are known for any of the objectives discussed in this dissertation*, other than the MaxCut and PCST objectives. Furthermore, the approximation algorithms for these two objectives are not scalable. Table 2.1 compares our work (Chapters 3 and 4) with several state-of-the-art algorithms on supported scoring functions, time and space complexity, and performance bound.

2.2.2 Dense Subgraph Mining

We say that an undirected, unweighted graph of n nodes is *dense* if it has close to $\binom{n}{2}$ edges, the maximum number of links possible between n nodes. Different notions of density have been proposed giving rise to many variations of the dense subgraph problem, which we briefly describe below.

In the **Densest Subgraph** problem, we are given a graph $G(V, E)$, and the goal is to find a subset of nodes $S \subseteq V$, such that the *average degree* of the graph induced by S, $\frac{E(S)}{|S|}$ is maximized. This problem can be solved optimally in polynomial time using Goldberg's flow-based algorithm [95]. There is also a linear-time greedy algorithm that yields a $\frac{1}{2}$-approximation [23, 51]; in practice, this algorithm finds subgraph with average degree close to optimal.

In real-world networks, subgraphs with maximal average degree have been found to be large—sometimes trivially spanning the entire node set V—and not very dense [251]. When we want to control the size of the subgraphs discovered, we can add a constraint k to the densest subgraph formulation. In the **Densest-k Subgraph** problem [80], the goal is to find a subset S of size k with maximum number of edges. This problem is NP-Hard. Asahiro et

Table 2.1: Summary of the related work on window-based methods for graph anomaly detection and how it compares to our contributions.

Method	Description	Time	Space
Algorithms for optimization of parametric scan statistics in networks			
Exact algorithms	Exhaustive search (optimal) over k-nearest-neighbors subgraphs [231, 241] Do not scale to graphs with more than 1000 nodes	$O(2^k n)$	$O(n+m)$
Simulated annealing	Based on a concept of "non-compactness" for penalizing. No guarantees. clusters [72]	$O(n \log n)$	$O(n+m)$
`AdditiveGraphScan`	Connects clusters based on a heuristic for Prize-Collecting Steiner Tree [233], used for nonlinear score functions. No guarantees.	$O(mn + n^2 \log n)$	$O(n+m)$
`EdgeLasso`	Sparse learning method based on edge-lasso regularization [224], handles quadratic score functions. No guarantees.	$O(l \cdot n^3)$ for l iterations	$O(n^2)$
`GraphLaplacian`	Spectral scan method based on graph Laplacian regularization [225], handles quadratic score functions. No guarantees.	$O(l \cdot n^3)$ for l iterations	$O(n^2)$
Algorithms for optimization of nonparametric scan statistics			
`NPHGS`	Local search heuristic for optimizing nonparametric scan statistics on general graphs [54], considers nonlinear functions. No guarantees.	$O(n \log n)$	$O(n+m)$
Reduction to variants of Network Design and using methods for Prize-Collecting Steiner Tree (PCST):			
`EventTree+`	Use PCST method [212], linear function. 2-approximation for PCST from [92]	$O(n^2 \log n)$	$O(n+m)$
`Meden`	Greedy algorithm for the Heaviest Dynamic Subgraph (HDS) problem equivalent to the NetWorth objective, which is a linear function. No guarantees. [40]	$O(m \log n)$	$O(n+m)$
This dissertation: Fixed-parameter tractable algorithms with theoretical guarantees for parametric and non-parametric scan statistics as well as Steiner-based formulations.			
ColCodeNP	Finds optimal solution with effective size at most k with probability $1-\epsilon$	$O((2e)^k m \log \frac{n}{\epsilon})$	$O(2^k n)$
MultilinearScan	Finds optimal solution with effective size at most k with probability $1/5$ or $f(\epsilon)$-approximate solution where $\epsilon > 0$ is a parameter	$O(2^k m k^2 n^2)$ exact, $O(2^k m k^2 \log_{1+\epsilon}(n))$ approximate	$O(kn^2)$ exact, $O(kn \log_{1+\epsilon}(n))$ approximate

al. propose a $O(k/n)$ greedy approximation algorithm for any value of k [24]; for particular values of k, Feige and Langberg are able to obtain better approximations using semidefinite programming [79]. When the constraint is to find a set S of size at least k, we obtain the **Densest-at-least-k Subgraph** problem; when the constraint is $|S| \leq k$, we have the **Densest-at-most-k Subgraph** problem. Both variants (also NP-Hard) were proposed by Andersen and Chellapilla [14]. We note that all these formulations have been studied on directed and weighted graphs as well [132].

Recently, Tsourakakis [250] proposed the **k-Clique densest subgraph** problem as a generalization of the densest subgraph problem. In this formulation, the goal is to find a set of nodes that maximizes $\frac{G_k(S)}{|S|}$, where G_k is the number of k-cliques induced by the nodes

in S. For $k = 3$, we obtain the *triangle-densest subgraph* problem. Tsourakakis shows this latter formulation discovers graphs that are denser than the ones found by maximizing the average degree.

Dense subgraphs are also related to cliques. A complete graph (i.e., a clique) is the densest subgraph one could have. In the **Maximum Clique** problem, we are given a network, and the goal is to find the largest subgraph that is a clique [41]. However, this formulation is not very practical for two reasons. First, the maximum clique problem is hard to approximate in polynomial time to a factor of $n^{1-\epsilon}$ unless P = NP [102]. Second, finding a clique is too restrictive because all of the edges have to be present. In practice, subgraphs of interest are not necessarily complete.

In [251], Tsourakakis et al. proposed the **Optimal Quasiclique** problem, where the goal is to find a set of nodes that maximizes $f_\alpha(S) = E(S) - \alpha\binom{|S|}{2}$, where α is a parameter. The authors propose a greedy method based on [51] and a local search algorithm to optimize the objective. They also show that subgraphs with a high f_α score have high edge and triangle density, and they have small diameter —all of these are desirable properties for dense subgraphs.

Discussion. Despite the extensive literature in dense subgraph mining, the problem of finding dense subgraphs in signed networks has been relatively unexplored. Signed networks bring a new set of challenges when it comes to density problems. The current methods and formulations assume the weights of the graph are non-negative, and there is no simple way to extend these methods to the signed case. There *is* work on community detection in signed networks [148], where communities are defined in terms of density; however, community detection and dense subgraph mining have fundamentally different objectives [155]. As part of this dissertation, we propose, to our knowledge, the first algorithms for mining dense subgraphs with positive and negative edge weights. Table 2.2 compares our contributions (Chapter 7) with existing work on the areas of dense subgraph mining and signed networks.

Table 2.2: Comparative summary of the related work on dense subgraph mining.

Category	Method	Dense Subgraph	Signed	Unsigned	Weighted	Theoretical guarantees	Scalable
Dense Subgraph Mining	Densest Subgraph [23, 51, 95]	✓	✗	✓	✓	✓	✓
	Densest-k Subgraph [24, 79]	✓	✗	✓	✓	✓	✗
	Densest At-Least (At-Most) k subgraph [14]	✓	✗	✓	✓	✓	✓
	Optimal Quasiclique [3, 250]	✓	✗	✓	✓	✗	✓
Signed Networks	Community Detection [39, 248]	✗	✓	✗	✗	✗	✓
	Spectral Clustering [148]	✓	✓	✓	✓	✗	✓
	Low-Rank Modeling [58]	✗	✓	✗	✗	✓	✓
This Dissertation	DenSDP	✓	✓	✓	✓	✓	✗
	EgoScan	✓	✓	✓	✓	✗	✓

2.2.3 Motif Finding

An important task in biological networks is that of finding *functional motifs*, or simply *motifs* [152,219]. A motif is a connected graph where each node has a color or label, which represents a distinct function of the node. Then, in the Graph Motif problem, we are given a graph G and a motif M (much smaller than G), and the goal is to find whether M is a subgraph of G. In general, this problem is NP-Hard. However, the problem is fixed-parameter tractable in k, the size of the motif, which makes it possible to design algorithms with running time in $O(c^k poly(n, m))$, where c is a small constant and $poly(n, m)$ is a polynomial function on the size of G. This running time is reasonable, since, in practice, motifs are usually small. The fastest known algorithm for Graph Motif runs in time $O(2.54^k k^2 m)$ [137], and it is based on the multilinear detection technique introduced by Koutis and refined by Williams [139].

2.2.4 Dynamics on Networks

2.2.4.1 Models of Epidemic Spread

Compartmental models have a long history in epidemiology [16,27,131]. These models involve splitting a population into groups (or compartments) according to their disease status. For example, in the *Susceptible-Infection-Recovered* (SIR) model, each individual is assumed to be in one of these three states depending on whether he/she has not contracted the disease yet (susceptible), is currently infected, or has already recovered. Individuals move from S to I at a rate $\beta|I|$, and from I to R at rate γ, where β and γ are parameters of the model. Figure 2.7a shows a graphical representation of an SIR process. This model has been used to study diseases that confer immunity after recovery, such as measles, chicken pox, or seasonal flu [171, 190]. Other variations with additional states, such as the Susceptible-Exposed-Infected-Recovered (SEIR) model [107] have been considered. For diseases that do not confer immunity—for instance, sexually transmitted diseases—the *Susceptible-Infected-Susceptible* (SIS) model is more appropriate. Here, an individual moves back and forth between the two states (Figure 2.7b).

Figure 2.7: **Compartmental models.** (a) SIR model. A susceptible individual becomes infected at rate $\beta|I|$, and he/she recovers at rate γ. (b) SIS model. An individual moves back and forth between the Susceptible and Infected states.

One weakness of compartmental model is that they assume that any pair of individuals in the population may come into contact, which is unrealistic in most cases. Later, network

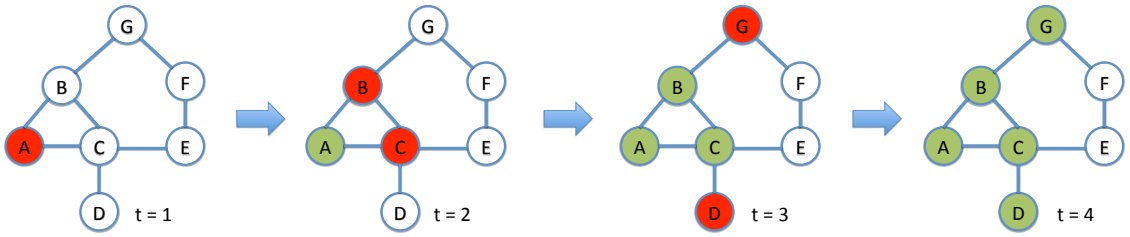


Figure 2.8: **SIR model on a network.** White nodes are susceptible, red nodes are infected, and green nodes are recovered. The infectious period is one time unit. Initially, A is the only node infected. It infects neighbors B and C. From time 2 to 3, C only infects one of its two neighbors (D). There are no new infections at time 4, so the process stops.

structure was added to the model to relax this assumption. Let $G(V, E)$ be a contact network where the nodes are individuals in some population, and there is an edge between two persons if they come into contact at some point in the day. An infected node v in this graph spreads the infection to each susceptible neighbor u with some probability $p(u, v)$, usually measured in terms of probability of infection per unit time. An infected node stays in state I for an *infectious duration* period and then moves to state R. Figure 2.8 shows an example of disease spread on a network under the SIR model for an infectious period of one time unit.

2.2.4.2 Information Diffusion in Online Social Networks

With the advent of online social networks in the past decade, modeling how information spreads has become a very active area of research. One way to reason about information diffusion is in terms of epidemic spread and peer influence. Under this view, we can model diffusion processes using epidemiological models, such as linear threshold and SI/SIS/SIR epidemic models [130, 157]. However, a more common modeling paradigm is that of *cascades* [4, 28, 151, 236]. Informally, cascades are subgraphs—often trees but forests have been considered recently [143]—that capture the spread of influence from a node to its newly activated/influenced neighbors and descendants. There are many notions of cascades, which afford varying levels of formal characterization and utility. For instance, in Bakshy et al. [28], an edge $A \rightarrow B$ is included in a cascade only if it can be argued that some action (e.g., a posting or use of a URL) by B can be directly attributed to A. Another common approach is to define cascades in terms of the (random) subgraph over which diffusion processes like linear threshold and independent cascade model spread. A simpler notion of cascades, referred to as *activity cascades*, was introduced by [42, 84, 97]. Informally, such a cascade consists of a tweet emitted by a user u, and the messages of the users who see/mention u's message given that those messages are sent within a small time interval. It is harder to attribute influence under this definition of cascade. However, as we discuss further in Chapter 9, advantages of this formulation is that it is amenable to rigorous analysis and that it has good predictive power for modeling civil unrest.

2.3 Algorithmic Foundation

2.3.1 Parameterized complexity

A problem is fixed parameter tractable (FPT) if it can be solved in time $f(k)|x|^{O(1)}$, where $f(\cdot)$ is an arbitrary function, k is a (usually) small parameter, and $|x|$ is the size of the problem instance. Intuitively, this means that the problem can be solved in polynomial time in the size of the input, but it may have exponential running time in the parameter k. This is one way to deal with complexity.

The k-Path problem is a well-known example of a problem in FPT. In k-Path, we are given a graph, and the goal is to decide whether there is a path of length k in the graph. In general, this problem is NP-Complete by reduction to Hamiltonian Path (i.e., when $k = n - 1$). However, the problem can be solved in time $O(2^k poly(n, m))$.

2.3.1.1 Color Coding

We use the k-Path problem as an example to explain the color coding technique. Consider the following algorithm for k-Path. For $0 \leq i \leq k$, let $\mathcal{P}(v, i)$ be the set of all paths of length i ending at node v. Note that G has a k-path if $\mathcal{P}(v, k) \neq \emptyset$ for some node v. We can compute $\mathcal{P}$ recursively using a dynamic program. For $i = 0$, $\mathcal{P}(v, 0) = \{\{v\}\}$. We construct a path in $\mathcal{P}(v, i)$ by appending node v to a path of length $i - 1$ ending at a neighbor of v if v is not already in that path. That is

$$\mathcal{P}(v, i) = \bigcup_{u \in N(v)} \{S \cup \{v\} | S \in \mathcal{P}(u, i - 1), v \notin S\}.$$

It can be verified that this algorithm correctly solves the k-Path problem. However, its running time is $O(n^k)$, which is prohibitively large, even for small values of k.

The $O(n^k)$ bound comes from the fact that we are potentially keeping track of $\binom{n}{i}$ paths for $0 \leq i \leq k$. The idea behind color coding is sampling a subset of all the possible paths instead of keeping track of all of them. In this way, the computation stays tractable. Note that, because of the random sampling, we may not keep track of any of the k-paths in the graph; however, we can show that, with high probability, we will indeed find a k-path if it exists.

With that intuition, we describe the algorithm. Let $K = \{1, \ldots, k\}$ be a *color set*. We define a *coloring* as a function $col : V \to K$ that maps nodes to colors; $col(u)$ is the color of node u. For a given set of nodes S and a color set $T \subseteq K$, we say that S is *colorful (with respect to T)* if every node in S has a distinct color from T; that is, for all $u, v \in S$, $col(u) \neq col(v)$ and $col(u), col(v) \in T$.

Going back to the k-Path problem, we now only keep track of colorful paths. Let $\mathcal{P}(v,i)$ be the set of colorful paths of length i ending at node v. Again, we will use a dynamic program to compute these sets for all $i \leq k$. When $i = 0$, $\mathcal{P}(v,0) = \{\{col(v)\}\}$. For $1 \leq i \leq k$, we construct a path in $\mathcal{P}(v,i)$ by appending node $col(v)$ to a path of length $i-1$ ending at a neighbor of v if $col(v)$ is not already in that path. That is

$$\mathcal{P}(v,i) = \bigcup_{u \in N(v)} \{S \cup \{col(v)\} | S \in \mathcal{P}(u, i-1), col(v) \notin S\}.$$

There is a k-path in G if $OPT(v,k) \neq \emptyset$ for some node v. If G does not have a k-path, the algorithm always returns a correct answer. However, if there is a k-path, we will only find it if it is colorful. By repeating the algorithm for many random colorings, a k-path will be colorful with high probability. For simplicity, suppose there is only one k-path in the graph. The probability that this path is colorful is

$$\frac{k!}{k^k} \geq e^{-k},$$

and the probability that the path is not colorful in any of t random colorings is

$$\left(1 - \frac{1}{e^k}\right)^t.$$

For $\epsilon > 0$, let $t = -e^k \ln \epsilon$, and we can bound the probability of incorrectly reporting that G does not have a k-path:

$$\left(1 - \frac{1}{e^k}\right)^{-e^k \ln \epsilon} \leq \epsilon.$$

Theorem 1 (Alon et al. [12]) *Color Coding yields an algorithm for the k-Path problem that finds a path with probability at least ϵ. The time complexity of the algorithm is $O((2e)^k m \log(1/\epsilon))$ and the memory requirement is $O(2^k n)$.*

2.3.1.2 Multilinear Detection

In [136], Koutis introduced the Multilinear k-Term problem and an algorithm that runs on $O(2^k poly(n))$ time, where n is the size of the input. It has been shown that many parameterized problems reduce to Multilinear k-Term. Examples include k-Path [136, 261], variations of the Graph Motif problem [99, 137], and parameterized set covering problems [136, 138].

Let $X = x_1, \ldots, x_n$ be a set of variables, and let $P(X)$ be a polynomial, which is a sum of monomials on X. An example of a polynomial on 4 variables is $P(x_1, x_2, x_3, x_4) = x_1^2 x_2 + x_1 x_2 x_3 + x_2 x_4^2$. A monomial is called *multilinear* or *square-free* if all its variables have

exponent 1. Furthermore, the *degree* of a monomial is the sum of the exponents of all its variables. For instance, in the example above, $x_1x_2x_3$ is the only multilinear monomial, and all the monomials have degree 3. We can now state the Multilinear k-Term problem: Given variables $X = x_1, \ldots, x_n$ and a polynomial $P(X)$, the goal is to decide whether or not $P(X)$ has a multilinear monomial of degree exactly k.

For the case when a multilinear monomial appears an odd number of times in $P(X)$, Koutis proposed a randomized algorithm that runs in $O(2^k poly(n))$ time and returns an affirmative answer with probability at least 1/4 [136]. This algorithm was later extended by Williams [261] to allow even repetitions.

We describe the main ideas in [136]. Let $\mathbb{Z}_2^k$ be the group formed by all the k-dimensional binary vectors, and let the entry-wise XOR operator be the group multiplication. For example, $\mathbb{Z}_2^2$ consists of the vectors $v_0 = (0,0), v_1 = (0,1), v_2 = (1,0), v_3 = (1,1)$. We note that v_0 is the multiplicative identity of the group, and each element is its own multiplicative inverse: $v_i \cdot v_i = v_0$. Now, we define a group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$. Each element in the group algebra is a sum of elements from $\mathbb{Z}_2^k$ with coefficients from $\mathbb{Z}_2$ (i.e., either 1 or 0):

$$\sum_{v \in \mathbb{Z}_2^k} a_v v,$$

where $a_v \in \{0,1\}$. Such element may also be interpreted as a subset of $\mathbb{Z}_2^k$—because of the binary coefficients. The addition operator of the group algebra is

$$\sum_{v \in \mathbb{Z}_2^k} a_v v + \sum_{v \in \mathbb{Z}_2^k} b_v v = \sum_{v \in \mathbb{Z}_2^k} (a_v + b_v) v,$$

where the addition of the coefficients is modulo 2. The multiplication is defined as

$$\left(\sum_{v \in \mathbb{Z}_2^k} a_v v \right) \left(\sum_{u \in \mathbb{Z}_2^k} b_u u \right) = \sum_{v \in \mathbb{Z}_2^k} (a_v \cdot b_u)(v \cdot u).$$

The first key insight in [136] is that, for any $v_i \in \mathbb{Z}_2^k$,

$$(v_0 + v_i)^2 = v_0^2 + 2(v_0 \cdot v_i) + v_i^2 = v_0 + (0 \mod 2) v_i + v_0 = 2v_0 = 0 \mod 2.$$

If we are given a polynomial $P(X)$, and we assign uniformly at random an element $(v_0 + v_i)$ from $\mathbb{Z}_2[\mathbb{Z}_2^k]$ to each x_i variable, then, any monomial with a square term will evaluate to 0. This is called the *annihilation* property. The second key idea is that, under this random assignment, a multilinear monomial does not evaluate to 0 with high probability. This is called the *survival* property. Finally, Koutis shows that a polynomial $P(x_1, \ldots, x_n)$, where the variables are elements from $\mathbb{Z}_2[\mathbb{Z}_2^k]$, can be evaluated in time $O(2^k poly(n))$ and space $O(poly(n))$.

Because of the choice of binary coefficients in the group algebra and the modulo 2 addition operation, any monomial that appears an even number of times in $P(X)$ will evaluate to 0. In [261], Williams proposes working with the group algebra $GF(2^{3+\log_2 k})[\mathbb{Z}_2^k]$, where $GF(p)$ is the finite field of order p [180]. By using this group algebra, it is unlikely that multilinear monomials evaluate to 0 merely due to repetition, and, at the same time, the annihilation property of [136] is preserved. We have the following theorem.

Theorem 2 (Koutis [136] and Williams [261]) *There exists an algorithm that, given an instance $P(x_1, \ldots, x_n)$ of the Multilinear k-Term problem, returns "no" if $P(X)$ does not contain a multilinear term. Otherwise, it returns "yes" with probability at least 1/5. Furthermore, the algorithm has time complexity $O(2^k poly(n))$ and space complexity $O(kpoly(n))$.*

2.3.2 Semidefinite Programming

Most of the notation and definitions here are from [262].

2.3.2.1 Positive Semidefinite Matrix

A matrix $X \in \mathbb{R}^{n \times n}$ is *positive-semidefinite* if and only if for every vector $y \in \mathbb{R}^n$, we have that $y^T X y \geq 0$. Furthermore, there exists some matrix $V \in \mathbb{R}^{m \times n}$, where $m \leq n$, such that $X = V^T V$. We may also use the notation $X \succeq 0$ to denote that matrix X is positive semidefinite. For the rest of the discussion, we assume that X is symmetric.

2.3.2.2 Semidefinite Program

A semidefinite program is an optimization problem of the form

$$\begin{aligned} \text{maximize or minimize} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} & \\ \text{subject to} \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ijk} x_{ij} = b_k & \text{for all } k \\ & x_{i,j} = x_{ji} & \text{for all } 1 \leq ij \leq n \\ & X = (x_{ij}) \succeq 0 & \end{aligned}$$

Here, we are looking for a matrix of variables X that optimizes $\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$ under the given constraints. We note that the objective and the constraints are linear in terms of the x_{ij} variables, and X is required to be positive semidefinite.

Equivalently, we can formulate a *vector program* in which the variables are vectors in $\mathbb{R}^n$. It is often more convenient to use this form to reason about approximation algorithms.

$$\begin{aligned} \text{maximize or minimize} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}(v_i \cdot v_j) & \\ \text{subject to} \quad & \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ijk}(v_i \cdot v_j) = b_k & \text{for all } k \\ & v_i \in \mathbb{R}^n & \text{for all } 1 \leq i \leq n \end{aligned}$$

Here, $v_i \cdot v_j$ is the inner product of the vectors—which is a scalar. The quadratic and vector programs shown above are equivalent. To see why, we can take a solution X to the quadratic program and compute (approximately) the decomposition $X = V^T V$ in polynomial time. Let v_i be the i^{th} column of V; then, $x_{ij} = v_i \cdot v_j$, and the v_i vectors are a feasible solution to the vector program. Conversely, we can take a solution to the vector program and construct a matrix V whose i^{th} column is v_i. Let $X = V^T V$; then, X is symmetric and positive semidefinite, and $x_{ij} = v_i \cdot v_j$, so X is a feasible solution to the semidefinite program.

2.3.3 Submodularity

Let V be a finite set, and let $f : 2^V \to \mathbb{R}$ be a set function that maps a subset $S \subseteq V$ to a real number. We say that f is *submodular* if for every $A, B \subseteq V$,

$$f(A) + f(B) \geq f(A \cup B) + f(A \cap B).$$

An equivalent definition states that f is submodular if for every $A \subseteq B \subseteq V$ and $x \in V$, we have

$$f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B).$$

This is the *diminishing returns* property. Intuitively, adding an element to the smaller set A yields more benefit than adding it to the larger set B. This property has proven useful to develop algorithms for optimizing submodularity functions. Most of the time, we will focus on *monotone* submodular functions. We say that a submodular function f is monotone if for every $A \subseteq B \subseteq V$, $f(A) \leq f(B)$.

2.3.3.1 Submodular Function Maximization

We focus on the problem of maximizing a submodular function given a size constraint. Formally, given a submodular function f, a set V, and a parameter k, we want to find a set $S \subseteq V$, such that $|S| \leq k$ and $f(S)$ is maximized.

The problem above is NP-Hard. However, there is a simple greedy algorithm with a good approximation bound when f is nonnegative and monotone. We start with the empty set $S_0 = \emptyset$. We construct a set S_i by adding an element to S_{i-1} that yields the highest increase in objective value:

$$S_i = S_{i-1} \cup \underset{x \in V \setminus S_{i-1}}{\operatorname{argmax}} \{f(S_{i-1} \cup \{x\}) - f(S_{i-1})\}$$

The algorithm returns a set S_k with objective value at least $1 - \frac{1}{e}$ of the optimal.

Theorem 3 (Nemhauser et al. [189]) *Let $f : 2^V \to \mathbb{R}_+$ be a non-negative monotone submodular function, and let S_k be the set found by the greedy algorithm above. Then,*

$$f(S_k) \geq \left(1 - \frac{1}{e}\right) \max_{|S| \leq k} f(S).$$

Submodular functions over graphs have also been explored. In this setting, we are given a graph $G(V, E)$, and a submodular function f on the node set V. One relevant problem to our work is maximizing a submodular function with connectivity constraints. Formally, we are given a graph $G(V, E)$, a non-negative monotone submodular function $f : 2^V \to \mathbb{R}_+$, and a parameter k. We want to find a set of nodes $S \subseteq V$, such that (1) $|S| \leq k$, (2) the nodes in S form a connected subgraph, and (3) $f(S)$ is maximized.

Recently, Kuo et al. [149] proposed a $O(\sqrt{k})$ approximation algorithm to this problem. Their approach consists of first relaxing the connectivity constraint and using the greedy algorithm from above to find a set S' of size at most $\lfloor\sqrt{k}\rfloor$, over the $\lfloor\sqrt{k}\rfloor$-neighborhood of every node. The authors show that it is possible to make the set S' connected by adding at most $\lfloor\sqrt{k}\rfloor$ nodes to S' to obtain a final solution S with objective value $\frac{1}{O(\sqrt{k})}$ of the optimal.

Part I

Using Static Properties

Chapter 3

Near-Optimal Algorithms for Graph Scan Statistics

3.1 Introduction

Detecting "hotspots" and "anomalies" is a recurring problem in a wide range of applications, such as social network analysis, epidemiology, finance, bio-surveillance. A number of methods have been proposed, but the paradigm of "scan statistics" is among the most powerful and widely used. As observed in [233], these methods typically first involve formalizing a notion of "anomalousness" for a subset of data, and then a method to efficiently find a subset that optimizes a corresponding score. These were originally developed for spatial data and involved finding simple regions (e.g., disks) that maximized some form of discrepancy score based on a likelihood ratio test—see, e.g., [144, 173, 183, 184, 188]).

These methods have been extended to network data by considering scores for connected subgraphs [53,54,233] and have important applications in a variety of domains, such as detection of disease outbreaks in a geographic network [231], significantly mutated subnetworks in a genome-scale interaction network for cancer research [101, 156], intrusion attacks in a computer network [182], pollution in a water distribution network [231], and civil unrest events in a social media network [54].

Consider, for instance, the snapshots of a toy sensor network in Figure 3.1. This type of network has been used to detect pollution on a water distribution system [195]. Each node is a sensor, and an edge represents a water pipe between two sensors. We would like to detect pollution in the network, so that remedial action is taken. However, a sensor could become active due to noisy observations; similarly, a sensor that should be active may fail to detect pollution. For example, at times 2 and 3 in the figure, we observe some active sensors (colored blue), but these are not indicative of pollution. At time 4, on the other hand, we find a large subgraph of adjacent active sensors, which has a low likelihood of being just

noise. Note, however, that we may have to include some inactive nodes if this allows us to discover a larger anomalous cluster.

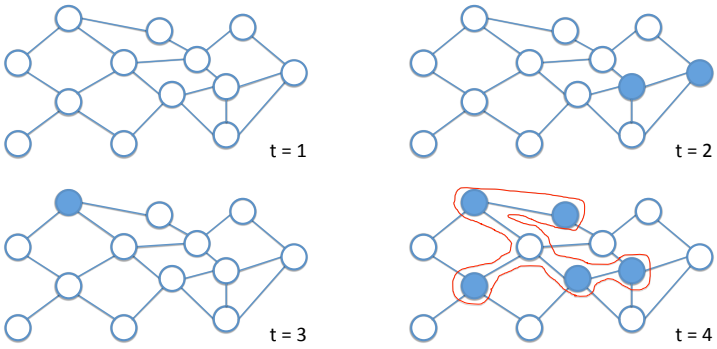


Figure 3.1: Anomalous subgraph detection. Four snapshots of a network of sensors. A blue node (sensor) indicates pollution at that part of the network. However, individual sensors may become active due to noise. This is the case at times 2 and 3. However, time 4 shows a large subgraph of active sensors. This event detection problem can be cast as finding a connected subgraph with a high proportion of blue nodes—possibly connected by some white nodes. In this case, we would like to detect the subgraph circled in red at time 4.

The connectivity constraint ensures that subgraphs reflect changes due to *localized* in-network processes. This is important in domains like public health and monitoring physical systems for two reasons. First, when an anomaly is detected, remedial action involves a site visit to the anomalous location. In the previous figure of the sensor network, a cleaning crew would come to the subgraph at time 4 to clear the pollution. As another example, in a public health setting, an anomalous subgraph might model a geographical region with poor vaccination practices, and a public health agency could focus resources on improving vaccination in this region. Second, the connectivity constraint helps to find subgraphs that are easier to interpret. In the public health example, a practitioner could correlate low vaccination with demographic properties or infrastructure in the target region—e,g., income or access to health services—which would be more difficult to do if the discovered nodes were disconnected and scattered over the entire graph.

Depending on whether the notion of anomalousness is with respect to an underlying model for the data or historical values, scan statistics can be *parametric* or *non-parametric* (Section 3.2). As a result of the diversity of applications, a large number of scan statistics have been developed. We briefly describe some applications below.

- *Social Science:* Detection of human rights events [55] and civil unrest [54]
- *Disease surveillance:* Early detection of respiratory disease outbreaks [185] and clusters with high incidence of breast cancer [44].
- *Security:* Network intrusion detection and illicit activities in shipment data [173].

3.1.1 Contributions

Maximizing functions over connected subgraphs generalizes *Network Design* problems—this includes well-known graph optimization problems, such as the Steiner Tree problem and its variants, Prize-Collecting Steiner Tree (PCST) and NetWorth, all NP-hard. Heuristics for these problems have been used for network scan statistics [40, 212, 231, 233], but they do not give any rigorous guarantees on the solution quality. Here, we present a *unified algorithmic framework for graph scan statistics with connectivity constraints*. Our contributions are

1. A *unified framework for optimizing a large class of parametric and non-parametric scan statistics for networks with connectivity constraints*, which scales linearly in the network size and is a function of a parameter defined as the "effective solution size". We also give rigorous bounds on the solution quality (summarized in Theorem 5). In other words, our framework encompasses many different network scan statistics—this contrasts with all prior methods, which are developed for specific statistics; further, our approach also holds for the extensions of these functions with both node and edge weights, which generalize Steiner connectivity problems. *In practice, the effective solution size parameter is very small* (see Section 3.6.6), making the time complexity of our algorithms better than prior methods, which are super-linear in the network size.

2. Preprocessing and refinement techniques that reduce the solution size without degrading the quality score beyond a provable constant factor (Section 3.4.4). The resulting algorithms are able to scale to graphs with over a million nodes in minutes and are significantly faster than state-of-the-art methods, which have only been run on graphs of up to 10^4 nodes.

3. Significant improvement over the objective scores computed by different baselines, with over 25% improvement in some instances, compared to the best baseline method (Section 3.6.3). Better objective scores also translate to higher anomaly detection power with 3% improvement on accuracy and F1 score over state-of-the-art methods. Our algorithmic framework has the added advantage that different score functions can be optimized by just modifying the specific objective function *within the same implementation.*

3.2 Preliminaries

We are given a graph $G = (V, E)$, where V is a set of n vertices or nodes, and $E \subseteq V \times V$ is a set of m edges. Each vertex $v \in V$ has two values associated with it: (1) a *population* count, $b(v)$, which indicates the count that we expect to see at the node v—for instance, the number of people in a county, corresponding to node v—and (2) an *event count*, $c(v)$, which

indicates how many occurrences of an event of interest are seen at the node—for instance, the number of cases of a disease in a county. These values vary over time, but we will not indicate the time in the notation in order to keep it simple. Our notation is summarized in Table 3.2.

3.2.1 Parametric Scan Statistics.

Parametric scan statistics assume that counts observed at each node are generated from some parameterized distribution and formalize anomaly detection as a hypothesis testing problem [144, 185]. Common choices are distributions from the exponential family, such as Poisson or Normal. Under the alternative hypothesis $H_1(S)$, an underlying anomalous phenomenon is characterized in the following manner: features of a majority of the vertices are generated from the same background distribution, and features of a small connected subset $S \subseteq V$ of vertices are generated from a different distribution. The goal is to maximize an appropriate scan statistic function $F(S)$, typically a likelihood ratio. These score functions can be expressed as

$$F(S) = g(C(S), B(S)), \tag{3.1}$$

where $C(S) = \sum_{v \in S} c(v)$, $B(S) = \sum_{v \in S} b(v)$, and the function g is defined depending on the score function considered. A well-known example of this class of functions is the Kulldorff scan statistic, commonly used in disease surveillance [72, 144, 145, 185] and defined as

$$C(S) \log \left(\frac{C(S)}{B(S)} \right) + (C(V) - C(S)) \log \left(\frac{C(V) - C(S)}{B(V) - B(S)} \right) - B(V) \log \left(\frac{C(V)}{B(V)} \right),$$

3.2.2 Non-Parametric Scan Statistics.

Non-parametric scan statistics do not assume an underlying distribution or process on the graph. Instead, they first estimate a p-value for each vertex based on empirical calibration by comparing the current feature ($c(v)$ and $b(v)$) of this vertex with its features in the historical data. The problem of anomaly detection has been formalized as a hypothesis testing problem for testing whether the empirical p-values are uniformly distributed on $[0, 1]$ [26, 172, 185, 205, 223, 254, 268]). Let $\alpha \in [0, 1]$ be *significance level* and let $w(v, \alpha)$ denote the *weight* of a node v as a function of α. For a set of nodes S, let $W(S, \alpha) = \sum_{v \in S} w(v, \alpha)$ and $N(S)$ be a function of the cardinality of the set. Then, the score functions can be expressed in the following general form:

$$F(S) = \max_{\alpha \leq \alpha_{max}} \phi(W(S, \alpha), N(S), \alpha), \tag{3.2}$$

The significance level α can be optimized between 0 and some constant α_{max}. We use $w(v)$ and $W(S)$ to denote $w(v, \alpha)$ and $W(S, \alpha)$, respectively, whenever α is clear from the context.

For clarity, from now on, we consider the specific case when $N(S) = |S|$ is the cardinality of S and $w(v, \alpha)$ is 1 if the p-value of v is less than α, 0 otherwise—we say these nodes are *significant at level* α. Then, $W(S, \alpha)$ is the number of significant nodes in S. An example of a scan statistic with this structure is the Berk-Jones scan statistic (BJ) [36] defined as

$$\max_{\alpha \leq \alpha_{max}} |S| \left[\frac{W(S,\alpha)}{|S|} \log \left(\frac{W(S,\alpha)/|S|}{\alpha} \right) + \left(1 - \frac{W(S,\alpha)}{|S|} \right) \log \left(\frac{1 - W(S,\alpha)/|S|}{1-\alpha} \right) \right].$$

Table 3.1 shows non-parametric and parametric scan statistics that can be optimized using our proposed methods.

Limitations of scan statistics. The suitability of scan statistics depends on the application and the assumptions underlying the dataset. We refer to [144,172,185,188] for a more detailed discussion of the advantages and limitations of these approaches.

3.2.3 Problem Formulation.

From the discussion above, the graph anomaly detection task can be posed as the following constrained optimization problem.

Problem 1 *Given a graph $G = (V, E)$, a scan statistic $F(\cdot)$, and the associated counts for vertices—represented by vectors* $\mathbf{c}$ *and* $\mathbf{b}$*—find a connected subset $S \subseteq V$ that maximizes $F(S)$.*

3.3 Hardness

Because of connectivity constraints, optimizing scan statistics on graphs is challenging in general. Note that this contrasts with the case without any connectivity requirement, where the optimal value of the scan statistic can be computed in polynomial time because of a linear ordering property [185].

We present the proof of hardness for maximizing the Berk-Jones (BJ) statistic over connected subgraphs. A similar style of proof can be used to establish hardness results for other functions in Table 3.1. Through the results below, we also formalize connections between scan statistic optimization and Steiner connectivity, which is of both theoretical and practical interest.

The BJ statistic for a subset of nodes S is defined as

$$F(S) = \max_{\alpha \leq \alpha_{max}} |S| \cdot KL(W(S,\alpha)/|S|, \alpha).$$

Table 3.1: Scan statistics functions that can be optimized with our framework.

Non-Parametric Scan Statistics (The following definitions are by default, unless otherwise indicated) $F(S) = \max_{\alpha \leq \alpha_{max}} \phi(W(S,\alpha), N(S), \alpha)$, $p(v)$ refers to the p-value of node v, $N(S) = \lvert S \rvert, W(S,\alpha) = \sum_{v \in S} I(p(v) \leq \alpha)$, where $I(\text{True}) = 1$ and $I(\text{False}) = 1$.		
Name	**Original Form**	**General Form**
Berk-Jones [36]	$F(S) = \max_{\alpha \leq \alpha_{max}} N(S) KL(\frac{W(S,\alpha)}{N(S)}, \alpha)$	$\phi(a,b,\alpha) = b \cdot KL(a/b, \alpha)$, where $KL(x,\alpha) = x \log\left(\frac{x}{\alpha}\right) + (1-x)\log\left(\frac{1-x}{1-\alpha}\right)$
Higher Criticism [67]	$F(S) = \max_{\alpha \leq \alpha_{max}} \frac{W(S,\alpha) - N(S)\alpha}{\sqrt{N(S)\alpha(1-\alpha)}}$	$\phi(a,b,\alpha) = (a - b\cdot\alpha)/\sqrt{b \cdot \alpha(1-\alpha)}$
Kolmogorov-Smirnov [260]	$F(S) = \max_{\alpha \leq \alpha_{max}} \sqrt{N(S)} \cdot \left(\frac{W(S,\alpha)}{N(S)} - \alpha\right)$	$\phi(a,b,\alpha) = \sqrt{b}\left(\frac{a}{b} - \alpha\right)$
Anderson-Darling [75]	$F(S) = \max_{\alpha \leq \alpha_{max}} \sqrt{N(S)} \cdot \left(\frac{W(S,\alpha)}{N(S)} - \alpha\right) / \sqrt{\frac{W(S,\alpha)}{N(S)} \cdot \left(1 - \frac{W(S,\alpha)}{N(S)}\right)}$	$\phi(a,b,\alpha) = \sqrt{b}\left(\frac{a}{b} - \alpha\right) / \sqrt{\frac{a}{b} \cdot \left(1 - \frac{a}{b}\right)}$
Jager-Wellner [118]	$F(S) = \max_{\alpha \leq \alpha_{max}} \sqrt{N(S)} \cdot \left(1 - \sqrt{\frac{N^{-}_{\alpha}(S)}{N(S)} \cdot \alpha} - \sqrt{(1 - \frac{N^{-}_{\alpha}(S)}{N(S)})(1-\alpha)}\right)$	$\phi(a,b,\alpha) = \sqrt{b}\left(1 - \sqrt{\frac{a}{b} \cdot \alpha} - \sqrt{(1-\frac{a}{b})(1-\alpha)}\right)$
Stochastic Ordering of p-Values [13]	$F(S) = N(S) \int_0^{\alpha_{max}} \frac{(W(S,\alpha)/N(S) - \alpha)^2}{\alpha(1-\alpha)} \mathrm{d}\alpha$	$\phi(a,b,\alpha) = b \int_0^{\alpha_{max}} \frac{(a/b - \alpha)^2}{\alpha(1-\alpha)} \mathrm{d}\alpha$
Fisher's Test [81]	$F(S) = -\sum_{v \in S} \log p(v)/N(S)$	$W(S,\alpha) = \sum_{v \in S} \log p(v)$, $\phi(a,b,\alpha) = -a/b$
Truncated Fisher's Test	$F(S) = \max_{\alpha \leq \alpha_{max}} -\frac{\sum_{v \in S} I(p(v) \leq \alpha) \log p(v)}{N(S)}$	$W(S,\alpha) = \sum_{v \in S} I(p(v) \leq \alpha) \log p(v)$, $\phi(a,b,\alpha) = -a/b$
Weighted Fisher's Test	$F(S) = -\sum_{v \in S} \log(w(v)p(v)) / \sum_{v \in S} w(v)$, where $w(v)$ is the predefined weight of vertex v.	$W(S,\alpha) = \sum_{v \in S} \log(w(v)p(v))$, $N(S) = \sum_{v \in S} w(v)$, $\phi(a,b,\alpha) = -a/b$
Stouffer's Test [235]	$F(S) = -\frac{\sum_{v \in S} \Phi^{-1}(1-p(v))}{\sqrt{N(S)}}$	$W(S,\alpha) = \sum_{v \in S} \Phi^{-1}(1-p(v))$, $\phi(a,b,\alpha) = -a/b$, where $\Phi^{-1}(\cdot)$ refers to the inverse cumulative density function of standard Gaussian distribution
Edgington's Test [74]	$F(S) = -\sum_{v \in S} \log p(v)/N(S)$	$W(S,\alpha) = \sum_{v \in S} \log p(v)$, $\phi(a,b,\alpha) = -a/b$
Parametric Scan Statistics (The following defintions are by default, unless otherwise indicated) $F(S) = g(C(S), B(S))$, $C(S) = \sum_{v \in S} c(v)$, $B(S) = \sum_{v \in S} b(v)$		
Positive Elevated Mean Scan Statistic [205]	$F(S) = \sum_{i \in S} x_i / \sqrt{N(S)}$	$g(a,b) = a/\sqrt{b}$
Elevated Mean Scan Statistic [205]	$F(S) = (\sum_{i \in S} x_i)^2 / N(S)$	$g(a,b) = a^2/b$
Expectation-based Poisson Scan Statistic [185]	$F(S) = C(S)\log(C(S)/B(S)) + B(S) - C(S)$	$g(a,b) = a\log(a/b) + b - a$
Kulldorff Scan Statistic [144]	$F(S) = C(S)\log\left(\frac{C(S)}{B(S)}\right) + (C - C(S))\log\left(\frac{C-C(S)}{B-B(S)}\right) - C\log\left(\frac{C}{B}\right)$, where $C = \sum_{v \in \mathbb{V}} c(v)$ and $B = \sum_{v \in \mathbb{V}} b(v)$.	$g(a,b) = a\log(\frac{a}{b}) + (C-a)\log(\frac{C-a}{B-b}) - C\log\left(\frac{C}{B}\right)$
Expectation-based Gaussian Scan Statistic [185]	$F(S) = (C(S) - B(S))^2/(2B(S))$, where $\sigma(v)$ refers to the standard deviation of $c(v)$ that is calibrated based on its historical observations, $C(S) = \sum_{v \in S}(c(v)b(v))/\sigma(v)^2$, and $B(S) = \sum_{v \in S} b(v)/\sigma(v)^2$	$g(a,b) = (a-b)^2/(2b)$
Expectation-based Exponential Scan Statistic [185]	$F(S) = B(S)\log(B(S)/C(S)) + C(S) - B(S)$, where $C(S) = \sum_{v \in S} c(v)/b(v)$, $B(S) = \lvert S \rvert$	$g(a,b) = a\log(a/c) + b - a$
Spatial Scan Statistic for Multinomial Data [125]	$F(S) = \sum_k \{C_k(S)\log(\frac{C_k(S)}{C(S)}) + (C_k - C_k(S))\log\frac{C_k - C_k(S)}{C - C(S)}\} - \sum_k C_k \log(C_k/C)$, where $C_k(S)$ refers to the count of vertices of category k, $C(S) = \lvert S \rvert$, and $C = \lvert \mathbb{V} \rvert$.	$C(S) = \sum_k \{C_k(S)\log(\frac{C_k(S)}{C(S)}) + (C_k - C_k(S))\log\frac{C_k - C_k(S)}{C - C(S)}\}$, $B(S) = \sum_k C_k \log(C_k/C)$, $g(a,b) = a - b$

Here, each node has a p-value, $p(v) \in [0,1]$, $W(S,\alpha)$ is the number of nodes in S with p-value at most α, and α_{max} is a parameter. $KL(\cdot,\cdot)$ is the truncated KL-divergence:

$$KL(\delta,\gamma) = \begin{cases} 0 & 0 \leq \delta < \gamma \\ \delta \log\left(\frac{\delta}{\gamma}\right) + (1-\delta)\log\left(\frac{1-\delta}{1-\gamma}\right) & \gamma \leq \delta < 1 \\ \log\left(\frac{1}{\gamma}\right) & \delta = 1 \end{cases}$$

Sometimes, we will only be interested in the number of nodes with p-value at most α and not in the particular set being evaluated. In those cases, we will consider the following version of the BJ statistic:

$$F'(i,j,\alpha) = (i+j)\cdot KL(i/(i+j),\alpha) = \begin{cases} 0 & 0 \leq i/(i+j) < \alpha \\ i\log\left(\frac{i/(i+j)}{\alpha}\right) + j\log\left(\frac{j/(i+j)}{1-\alpha}\right) & \alpha \leq i/(i+j) < 1 \\ i\log\left(\frac{1}{\alpha}\right) & i/(i+j) = 1 \end{cases}$$

Notice that $F(\cdot)$ can be computed from $F'(\cdot)$ by letting $F(S) = \max_{\alpha \leq \alpha_{max}} F'(W(S,\alpha), |S| - W(S,\alpha), \alpha)$.

Lemma 1 *The function $F'(i,j,\alpha)$ is increasing on i and decreasing on j when $\frac{i}{i+j}$ is in the interval $(\alpha,1)$.*

Proof: The derivative of $F'(\cdot)$ with respect to i is

$$\frac{\partial F'}{\partial i} = \log\left(\frac{i}{i+j}\right) - \log(\alpha),$$

which is greater than 0 in the desired interval. Similarly, the derivative of $F'(\cdot)$ with respect to j is

$$\frac{\partial F'}{\partial j} = \log\left(\frac{j}{i+j}\right) - \log(1-\alpha),$$

which is less than 0 in the interval.

■

In the decision version of anomalous subgraph detection, we want to find whether or not there is a connected subgraph with objective value $F(\cdot)$ at least some lower bound.

Problem 2 (BJ-Decide (BJ-D)) ***Given:*** *an undirected graph $G = (V,E)$ with p-values, $p(v) \in [0,1]$, a parameter α_{max}, and a parameter $\tau \geq 0$.* ***Decide:*** *Is there a connected set of nodes $S \subseteq V$, such that $F(S) \geq \tau$?*

For the proof, we will consider the node version of the Steiner Tree problem.

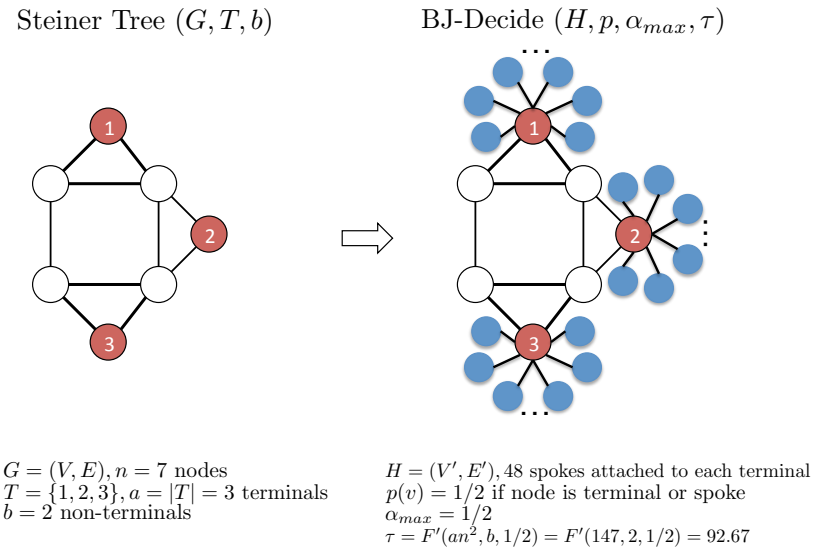


Figure 3.2: An example of the reduction in Theorem 4. For each terminal in the instance of Steiner Tree, we add $n^2 - 1$ spokes to the corresponding node in the instance of BJ-Decide. There is a Steiner Tree containing at most b non-terminal nodes in G if and only if there is a connected subgraph with BJ objective at least $\tau = F'(an^2, b, 1/2)$ in H.

Problem 3 (Steiner Tree (ST)) ***Given:*** *an undirected graph $G = (V, E)$, a set of terminals $T \subset V$, such that $|T| = a$, and a parameter $b \geq 0$.* ***Decide:*** *Is there a connected set of nodes $S \subseteq V$, such that $|S| \leq a + b$ and $T \subseteq S$? i.e., Is it possible to connect all the terminals using at most b non-terminal nodes?*

Theorem 4 BJ-D *is NP-complete.*

Proof: First, we note that **BJ-D is in NP**. Given any set of nodes S, we can verify that the nodes are connected in time polynomial in the size of the input graph, and we can evaluate $F(S)$ in time $O(|S|^2)$ to check whether or not $F(S) \geq \tau$. The $O(|S|^2)$ bound comes from the facts that 1) for a fixed α, we have to compute $W(S, \alpha)$, which involves checking whether each node has p-value at most α or not—this takes $O(|S|)$ time—and 2) we have to evaluate the function for at most $|S|$ different values of α.

Now, we show that **ST is polynomial-time reducible to BJ-D**. Given an instance $(G = (V, E), T, b)$ of ST, we construct an instance $(H = (V', E'), p, \alpha_{max}, \tau)$ of BJ-D as follows. Let n be the number of nodes in G; H is going to be a graph with the same nodes and edges as G, but, in addition, we are going to attach $n^2 - 1$ new vertices to each terminal node. We will refer to these new vertices as *spokes*. Formally, for a terminal $t \in T$, let $S^t = \{s_1^t, \ldots, s_{n^2-1}^t\}$; then, $V' = V \cup \{S^t | \forall t \in T\}$ and $E' = E \cup \{(t, s_1^t), \ldots, (t, s_{n^2-1}^t) | \forall t \in T\}$. For the p-values, the terminals and the spokes have $p(v) = 1/2$; all other nodes have p-value 1. Finally, we

let $\alpha_{max} = 1/2$ and $\tau = F'(an^2, b, 1/2)$, where a is the number of terminals and b is the parameter in the instance of ST. This reduction is illustrated in Figure 3.2.

Claim 2 *There is a Steiner tree S_G with b non-terminal nodes in G if and only if there is a connected set of nodes S_H with BJ score τ in H.*

Proof: The **first direction**—i.e., S_G implies S_H—is straightforward. By construction of H, S_G is a connected set of nodes in H. Let S_H be the set formed by S_G and all the spokes in the graph. That is, $S_H = S_G \cup \{S^t | \forall t \in T\}$. S_H has $b + a + a(n^2 - 1)$ vertices—i.e., the non-terminals, the terminals, and the spokes. Out of those, an^2 nodes have p-value $1/2$ and the remaining b have p-value 1. Therefore, this subgraph has BJ score of

$$F(S_H) = \max_{\alpha \leq (1/2)} F'(an^2, b, \alpha) = F'(an^2, b, 1/2) = \tau.$$

To prove the **converse**, notice that a connected subgraph S_H with $F(S_H) \geq \tau$ has at most b nodes with p-value greater than $1/2$. By construction, these b nodes correspond to non-terminal nodes in G. All that is left to show is all the terminals are included in S_H, as this implies that G contains a connected graph with all the terminals and at most b non-terminals.

To see that S_H must in fact have all the terminal nodes, consider the highest scoring subgraph that **does not** include some terminal. In the best possible case, we would be able to connect the remaining $(a-1)$ terminals and their respective spokes without using any Steiner nodes. Such subgraph would have a score of

$$F'((a-1)n^2, 0, 1/2) = (a-1)n^2 \log(2).$$

We want to compare this value to τ, which is given by

$$\begin{aligned}
\tau &= F'(an^2, b, 1/2) \\
&\geq F'(an^2, n-a, 1/2) \\
&= an^2 \log\left(2 \times \frac{an^2}{an^2 + n - a}\right) + (n-a) \log\left(2 \times \frac{n-a}{an^2 + n - a}\right),
\end{aligned}$$

where the inequality holds because 1) $F'(\cdot)$ is decreasing on b (Lemma 1) and 2) in the worst case, we need to use all the vertices in G to connect the terminals—i.e., $(n-a)$ non-terminal nodes. Now, we have a lower bound on τ and an upper bound on the best score that does not have all the terminals. We want to show that

$$an^2 \log\left(2 \times \frac{an^2}{an^2 + n - a}\right) + (n-a) \log\left(2 \times \frac{n-a}{an^2 + n - a}\right) > (a-1)n^2 \log(2) \tag{3.3}$$

$$an^2 \log\left(\frac{an^2}{an^2 + n - a}\right) + (n-a) \log\left(\frac{n-a}{an^2 + n - a}\right) > (n^2 + n - a) \log(1/2), \tag{3.4}$$

where we get inequality (4) by moving the $an^2\log(2)$ and $(n-a)\log(2)$ terms to the right-hand side and rearranging. We show that the following two inequalities, which together imply (4), hold for sufficiently large n.

$$an^2\log\left(\frac{an^2}{an^2+n-a}\right) > \frac{1}{2}(n^2+n-a)\log(1/2) \tag{3.5}$$

$$(n-a)\log\left(\frac{n-a}{an^2+n-a}\right) > \frac{1}{2}(n^2+n-a)\log(1/2). \tag{3.6}$$

Proving (5):

$$\begin{aligned}
an^2\log\left(\frac{an^2}{an^2+n-a}\right) &> \frac{1}{2}(n^2+n-a)\log(1/2)\\
an^2\log\left(\frac{an^2+n-a}{an^2}\right) &< \frac{1}{2}(n^2+n-a)\log(2)\\
\log\left(1+\frac{n-a}{an^2}\right) &< \frac{1}{2a}(1+\frac{n-a}{n^2})\log(2)\\
1+\frac{n-a}{an^2} &< 2^{\frac{1}{2a}(1+\frac{n-a}{n^2})}
\end{aligned}$$

Because $1 < a < n$, we can express a as βn, such that $1 > \beta > 1/n$. Replacing, we obtain

$$1+\frac{n-\beta n}{(\beta n)n^2} < 2^{\frac{1}{2}(\frac{1}{\beta n}+\frac{n-\beta n}{(\beta n)n^2})} \tag{3.7}$$

$$1+\frac{1-\beta}{\beta n^2} < 2^{\frac{1}{2}(\frac{1}{\beta n}+\frac{1-\beta}{\beta n^2})} \tag{3.8}$$

$$1+\frac{\delta}{n^2} < 2^{\frac{1}{2}(\frac{1}{\beta n}+\frac{\delta}{n^2})} \qquad (\text{let } \delta = \frac{1-\beta}{\beta}) \tag{3.9}$$

$$2^{\frac{1}{2}(1-\frac{1}{\beta n})}\left(1+\frac{\delta}{n^2}\right) < 2^{\frac{1}{2}(1+\frac{\delta}{n^2})} \qquad (\text{multiply both sides by } 2^{\frac{1}{2}(1-\frac{1}{\beta n})}) \tag{3.10}$$

Let $\epsilon = \log_2(1+\delta/n^2)$, so that $1+\delta/n^2 = 2^\epsilon$; then, inequality (10) holds for

$$2^{\frac{1}{2}(1-\frac{1}{\beta n})}2^\epsilon < 2^{\frac{1}{2}2^\epsilon} \tag{3.11}$$

$$\frac{1}{2}(1-\frac{1}{\beta n})+\epsilon < 2^{\epsilon-1} \tag{3.12}$$

$$1-\frac{1}{\beta n}+2\epsilon < 2^\epsilon. \tag{3.13}$$

Substituting $1 + \delta/n^2 = 2^\epsilon$ into (13), we obtain

$$1 - \frac{1}{\beta n} + 2\epsilon < 1 + \frac{\delta}{n^2}$$
$$2\log(1 + \frac{\delta}{n^2}) < \frac{\delta}{n^2} + \frac{1}{\beta n}$$

Since $(1 - \beta) < 1$, we prove the stronger inequality $2\log(1 + \delta/n^2) < \frac{\delta}{n^2} + \frac{\delta}{n}$ by analyzing the growth rate of both functions:

$$\begin{aligned}
\lim_{n\to\infty} \frac{\frac{\delta}{n^2} + \frac{\delta}{n}}{2\log(1 + \frac{\delta}{n^2})} &= \lim_{n\to\infty} \frac{-\frac{2\delta}{n^3} - \frac{\delta}{n^2}}{2\frac{1}{1+\frac{\delta}{n^2}}\frac{-2\delta}{n^3}} && \text{(taking the derivative of both functions)} \\
&= \lim_{n\to\infty} \frac{2 + n}{\frac{4}{1+\frac{\delta}{n^2}}} && \text{(taking out } n^3 \text{ and } \delta\text{)} \\
&= \lim_{n\to\infty} \frac{2 + n}{\frac{4n^2}{n^2+\delta}} \\
&= \lim_{n\to\infty} n/4 = \infty.
\end{aligned}$$

Proving (6):

$$(n - a)\log\left(\frac{n - a}{an^2 + n - a}\right) > \frac{1}{2}(n^2 + n - a)\log(1/2)$$
$$(n - a)\log\left(\frac{an^2 + n - a}{n - a}\right) < \frac{1}{2}(n^2 + n - a)\log(2)$$
$$\log\left(1 + \frac{an^2}{n - a}\right) < \frac{1}{2}(1 + \frac{n^2}{n - a})\log(2)$$
$$1 + \frac{an^2}{n - a} < 2^{\frac{1}{2}(1+\frac{n^2}{n-a})}$$

By letting $a = \beta n$ for $1 > \beta > 1/n$:

$$1 + \frac{(\beta n)n^2}{n - \beta n} < 2^{\frac{1}{2}(1+\frac{n^2}{n-\beta n})} \tag{3.14}$$
$$1 + \frac{\beta n^2}{1 - \beta} < 2^{\frac{1}{2}(1+\frac{n}{1-\beta})} \tag{3.15}$$
$$1 + \delta n^2 < 2^{\frac{1}{2}(1+\frac{n}{1-\beta})} \qquad (\text{let } \delta = \frac{\beta}{1 - \beta}) \tag{3.16}$$
$$2^{\frac{1}{2}(\delta n^2 - \frac{n}{1-\beta})}\left(1 + \delta n^2\right) < 2^{\frac{1}{2}(1+\delta n^2)} \qquad (\text{multiply both sides by } 2^{\frac{1}{2}(\delta n^2 - \frac{n}{1-\beta})}) \tag{3.17}$$

Let $1 + \delta n^2 = 2^\epsilon$. Then, inequality (17) holds for

$$2^{\frac{1}{2}(\delta n^2 - \frac{n}{1-\beta})} 2^\epsilon < 2^{\frac{1}{2}2^\epsilon} \tag{3.18}$$

$$\frac{1}{2}(\delta n^2 - \frac{n}{1-\beta}) + \epsilon < 2^{\epsilon-1} \tag{3.19}$$

$$\delta n^2 - \frac{n}{1-\beta} + 2\epsilon < 2^\epsilon \tag{3.20}$$

Substituting $1 + \delta n^2 = 2^\epsilon$ into (20), we obtain

$$\begin{aligned} \delta n^2 - \frac{n}{1-\beta} + 2\epsilon &< (1 + \delta n^2) \\ n &> (2\epsilon - 1)(1 - \beta) \\ n &> (2\log_2(1 + \delta n^2) - 1)(1 - \beta). \end{aligned}$$

The stricter inequality $n > (2\log_2(1 + \delta n^2) - 1)$ is true for sufficiently large n, since the function on the left of the ">" sign grows faster.

■

We have shown that BJ-D is in NP and that ST is polynomial-time reducible to BJ-D. This completes the proof. ■

In light of the NP-completeness in Theorem 4, it is unlikely that we would be able to compute the optimal solutions to the score functions efficiently. One approach that has been successfully used to combat computational hardness is *fixed parameter tractable algorithms*: the idea is to find an algorithm whose running time is $O(c^k f(n, m))$, where $f(n, m)$ is a polynomial function of the number of nodes, n, and the number of edges, m, c is a constant, and k is a parameter. In other words, the algorithm is exponential in a parameter other than the input size, but polynomial in the input size. In Section 3.4, we develop algorithms to maximize $F(S)$, restricted to sets with $|S| \leq k$, where k is a parameter that represents the *solution size*. In Section 3.4.4, we show that we can compress specific subsets of nodes into "supernodes", using a process we refer to as *refinement*. The size of a set S computed in terms of these supernodes will be referred to as the *effective solution size*, and it becomes significantly smaller than the original size of S. Our final algorithms will find solutions with effective solution size at most k.

3.4 Algorithms for Non-Parametric Scan Statistics

In this section, we present an algorithm for non-parametric functions that are characterized by equation (3.2) and then discuss techniques to scale it without losing the quality guarantees. Our algorithm relies on two main ideas, namely *monotonicity* and *constraining the solutions*.

[Definitions and notation]

Table 3.2: Definitions and notation used in this chapter

Term	**Description**
$b(v)$, $c(v)$	population and event counts of node v
α, α_{max}	significance level, maximum significance level
Significant node (at level α)	a node with p value below α
$Nbr(v)$	set of neighbors of v
$w(v)$, $w(v, \alpha)$	weight of node v, based on its p-value and the significance level α
$W(S)$, $W(S, \alpha)$	denotes $\sum_{v \in S} w(v, \alpha)$
$F(S)$	any of the functions in Table 3.1
K	The set $\{1, \ldots, k\}$
$col(u)$	color of node u from set K
T	Subset of K (denotes colors)
$M(v, T)$	$\max_S W(S)$, where the maximization is over connected colorful sets $S \subseteq V$, such that $v \in S$ and $\{col(u) : u \in S\} = T$
$\psi_i, \psi_i(\alpha)$	$\max_{T:\|T\|=i} M(v, T)$. Maximum weight over connected colorful sets of size i
S_i^*, $S_i^*(\alpha)$	Set with weight ψ_i
$OPT(F, k)$	$\max_{S:\|S\|\leq k} F(S)$, where the maximum is over connected subsets S of size $\leq k$

3.4.1 First idea: Monotonicity

A key observation is that the functions $\phi(W(S, \alpha), N(S), \alpha)$ we consider in Table 3.1 are monotonically increasing functions of $W(S, \alpha)$ if $N(S)$ is fixed, and if $W(S, \alpha) \geq \alpha$. For example, in the Berk-Jones (BJ) statistic, $W(S, \alpha)$ is the number of nodes in S signicant at level α, and the function increases with the number of significant nodes. Further, given two node sets of the same size, the set with more significant nodes scores higher according to the BJ statistic. We have the following lemma.

Lemma 3 *The non-parametric scan statistics functions characterized by equation (2) are increasing functions of $W(S)$ if $W(S) \geq \alpha$ and $N(S)$ is fixed.*

3.4.2 Second idea: constraining the solutions

We introduce the idea of a *coloring* of the nodes, and only consider connected subgraphs S, in which all nodes have distinct colors—such a solution is defined to be "colorful". Our approach builds on the color-coding technique of [12], but it involves several new techniques to scale the algorithm up to much larger graphs than the standard color-coding can accomodate.

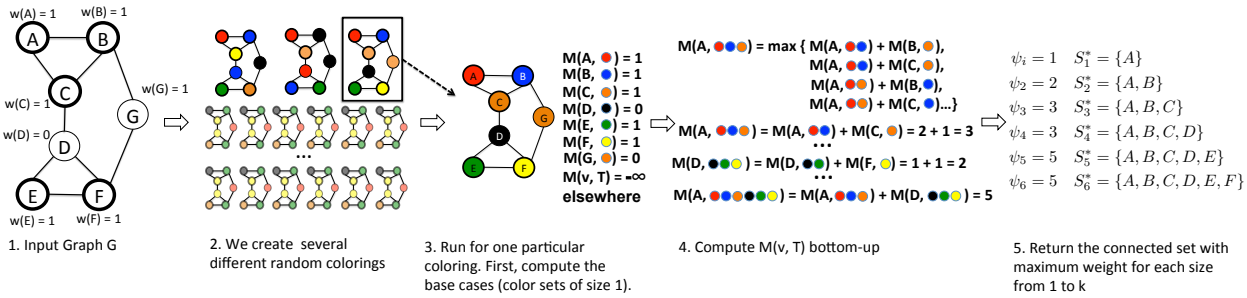


Figure 3.3: Example illustrating the MaxWeight procedure for $k = 6$ colors. 1) Nodes A, B, C, E, and F in the input graph have weight 1; D and G have weight 0. 2) We generate many random colorings of the node, as per the error parameter ϵ'. 3) For each coloring, we solve the dynamic program given in Lemma 4. M(A, {red}) is 1 because node A has weight 1 and its color is red; in other words, there exists a tree that is colorful with respect to {red} and contains node A. For all other colors c, M(A, {c}). 4) We compute M(v,T) bottom up. M(A, {red, blue, orange}) is maximized by adding M(A, {red, blue}) and M(C, {orange}). The corresponding colorful subtrees have nodes {A, B, C}, {A, B}, and {C}, respectively. In other words, the weight of tree {A, B, C} is the sum of weights of its subtrees {A, B}, and {C}. 5) We return the maximum weight and corresponding subtree of sizes 1 to k. The optimal subtree may not be colorful in one particular coloring, but, over all the random colorings, we will find the optimal subtree for each size up to k with high probability.

Let $K = \{1, 2, \ldots k\}$ be a set of colors —where k is a parameter— and let $col(v) \in K$ denote the color for node v. We say that a subgraph induced by set $S \subseteq V$ is *colorful* if $col(u) \neq col(v)$, for all $u, v \in S$. We let $M(v, T) = \max_S W(S)$, where the maximization is over all connected and colorful sets $S \subseteq V$, such that $v \in S$, $|S| = |T|$, and $\{col(u) : u \in S\} = T$. In other words, we only consider a set S if each node in the set has a different color from T. These definitions are illustrated in Figure 3.3. A key observation is that $M(v, T)$ can be computed by a dynamic program with a recurrence given in the lemma below.

Lemma 4 *Let $M(v, T)$ be defined as above. For any node v and color s, $M(v, \{s\}) = w(v)$ if $col(v) = s$, else $M(v, \{s\}) = -\infty$. If $|T| \geq 2$:*

$$M(v, T) = \max_{\substack{u \in Nbr(v) \\ T_1, T_2 \subseteq T}} \{M(v, T_1) + M(u, T_2)\},$$

where the maximum is over all partitions $T_1 \cup T_2 = T$ of the set T and all neighbors u of v.

Proof: Suppose $M(v, T)$ is achieved for a connected set S, such that $|S| = |T|$ and $\{col(u) : u \in S\} = T$, with $M(v, T) = \sum_{i \in S} w(i, \alpha)$. We claim that there exists $u \in Nbr(v)$, and partitions $T = T_1 \cup T_2$, and $S = S_1 \cup S_2$, such that: (1) $M(v, T) = M(v, T_1) + M(u, T_2)$, (2) $\{col(i) : i \in S_1\} = T_1$ and $\{col(i) : i \in S_2\} = T_2$, and (3) the subsets of nodes S_1 and S_2 are connected. Since S is connected, there exists a tree H that spans S and contains node

v. Further, there must exist a node $u \in Nbr(v)$ such that $(u, v) \in T$, since $|T| = |H| \geq 2$. Let H_1 and H_2 be the trees rooted at nodes v and u, respectively, that result when edge (u, v) is deleted in H. Let S_1 and S_2 denote the sets of nodes in H_1 and H_2, respectively. Let T_1 and T_2 be the colors used by S_1 and S_2, respectively. By construction, we have $M(v, T) = M(v, T_1) + M(u, T_2)$, so that the partitions $S = S_1 \cup S_2$ and $T = T_1 \cup T_2$ satisfy the requirements mentioned earlier. Therefore, the recurrence follows. ■

3.4.3 ColCodeNP

In Algorithm 1, we present ColCodeNP for optimizing non-parametric statistics. Recall the notation in Table 3.2. Let $F(S)$ denote any of the non-parametric functions in Table 3.1, and let $OPT(F, k) = \max_{S:|S|\leq k} F(S)$, where the maximum is over all connected subsets S of size $\leq k$, for a given α_{max}. Algorithm ColCodeNP takes the size bound k as input, and an *error parameter* ϵ, which indicates the probability of not finding the optimum solution.

Algorithm 1 ColCodeNP$((G(V, E), \alpha_{max}), k, \epsilon)$.

1: **Input**: Instance $(G(V, E), \alpha_{max})$, parameters k, ϵ
2: **Output**: Set S^* with score $OPT(F, k)$
3: Let A be the set of p-values of nodes in V below α_{max}
4: **for** $\alpha \in A$
5: Let $\mathbf{w}$ be a weight vector with $w(v) = w(v, \alpha)$
6: $\{S_i^*(\alpha) : i \in K\} =$ MaxWeight$(G(V, E), \mathbf{w}, k, \epsilon/n^2)$
7: $S^* = argmax_{i \in K, \alpha \in A} F(S_i^*(\alpha))$
8: **return** S^*
9:
10: **procedure** MaxWeight$(G(V, E), \mathbf{w}, k, \epsilon')$
11: **Input**: Instance $(G(V, E), \mathbf{w})$ and parameters k, ϵ'
12: **Output**: $\{S_i^* : i \in K\}$, such that S_i^* has weight ψ_i
13: Let $\psi_i = -\infty$ for all $i \in K$
14: **for** $j = 1$ to $e^k \log(1/\epsilon')$
15: For each node v, pick random color $col(v) \in K$
16: **for** $v \in V, s \in K$
17: $M(v, \{s\}) = w(v)$ if $col(v) = s$; $-\infty$ otherwise
18: **for** $v \in V$ and $T \subseteq K$, with $|T| \geq 2$
19: Use Lemma 4 to compute $M(v, T)$
20: **If** $M(v, T) > \psi_{|T|}$ **update** $\psi_{|T|} = M(v, T)$
21: **return** $\{S_i^* : \sum_{v \in S_i^*} w(v) = \psi_i, \text{for } i \in K\}$

Main steps. We describe the main steps of ColCodeNP connecting with the two ideas from above.

- The set A in line 3 of ColCodeNP denotes the set of distinct p-values of the nodes

less than α_{max}; it suffices to find the maximum of $\phi(W(S,\alpha), N(S), \alpha)$ for $\alpha \in A$. The **for** loop in lines 4—6 finds the best solution for each $i \in K$ and any given α (by calling MaxWeight in line 6), and the maximum is computed in line 7.

- MaxWeight finds the best solution S_i^* of size i, for each $i \in K$ using the idea described in Section 3.4.2. We show an example in Figure 3.3.
- Each iteration of the outer **for** loop in lines 14—20 starts with a random coloring (line 15). This is Step 2 in Figure 3.3.
- The inner **for** loop in lines 16—17 computes the base case of the dynamic program from Lemma 4; then, we solve the program bottom-up in lines 18–19. These are Steps 3 and 4 in Figure 3.3.
- ψ_i keeps track of the maximum weight solution restricted to size i, and it is updated if $M(v,T)$ denotes a better solution for size $|T|$.

Theorem 5 *For any non-parametric function $F(\cdot)$ in Table 3.1, algorithm* ColCodeNP *returns solution S^* satisfying $\Pr[F(S^*) = OPT(F,k)] \geq 1-\epsilon$, in time $O(2^k e^k |A| m \log(n/\epsilon))$, and using space $O(2^k n)$, where A is the set defined in line 3 of Algorithm 1.*

Before proving this theorem, the following lemma establishes that we can find the set with maximum weight for a particular size i by solving the recurrence from Lemma 4 for many different random colorings.

Lemma 5 *Let $\epsilon \in (0,1)$ be any constant and define. For any fixed i, α, consider ℓ random colorings of the nodes of graph G using a random color from the set $\{1,\ldots,i\}$ for each node. Let $X_j = \max_{v,T:|T|=i} M(v,T)$ for the j^{th} coloring. Then, $\Pr[\max_j X_j = \psi_i(\alpha)] \geq 1-\epsilon$, if $\ell \geq e^i \log 1/\epsilon$.*

Proof: Let $T = \{1,2,\ldots,i\}$ be a color set and let S_i^* be the node set that achieves ψ_i. For a random coloring of G, the probability that the set S_i^* is colorful is

$$p = \frac{i!}{i^i}.$$

For ℓ random colorings of G, the probability that S_i^* is not colorful in *any* of the colorings is $(1-p)^\ell$. We want this probability to be bounded by some small constant ϵ. Then, the number of random colorings that we should explore can be estimated as follows:

$$(1-p)^\ell = \left(1 - \frac{i!}{i^i}\right)^\ell < \left(1 - \frac{1}{e^i}\right)^\ell.$$

If we let ℓ be at least $-e^i \log(\epsilon)$, we have

$$\left(1 - \frac{1}{e^i}\right)^{\ell} \leq \left(1 - \frac{1}{e^i}\right)^{-e^i \log(\epsilon)} \leq e^{log(\epsilon)} = \epsilon.$$

■

Now, we present the proof of Theorem 5.

Proof: We start with the proof of correctness of our algorithm, which involves three parts. The first observation is that within the outer for loop in the procedure MaxWeight, for each random coloring $col(\cdot)$, $\max_v M(v, \{1, \ldots, k\})$ is correctly computed. This follows because the algorithm is a dynamic program that computes all $M(v, T)$ for $T \subseteq K$ using the recurrence in Lemma 4.

Next, we observe that the algorithm correctly finds $\psi_i(\alpha)$ —the maximum weight among sets of size i for a given α—for each i, α, with probability at least $1 - \epsilon/n^2$. The procedure MaxWeight is called with parameter $\epsilon' = \epsilon/n^2$ and $e^k \log(1/\epsilon')$ colorings. Let X_{ij} be the maximum weight found over subsets of size i in the j^{th} random coloring. By Lemma 5, $\Pr[\max_j X_{ij} \neq \psi_i(\alpha)] \leq \epsilon' = \epsilon/n^2$. The number of possible choices for α is $|A|$, which satisfies $|A| \leq n$. Therefore, by a union bound, it follows that for all $i, \alpha \in A$, we have $\Pr[\max_j X_{ij} \neq \psi_i(\alpha)] \leq n^2\epsilon' \leq \epsilon$, and the algorithm correctly computes $\psi_i(\alpha)$ for all i, α with probability $1 - \epsilon$.

Finally, for any fixed i, α, by Lemma 3, $\phi(W(S), N(S), \alpha)$ is an increasing function of $W(S)$ when $N(S)$ is fixed. This implies that $\max_{S:N(S)=i} \phi(W(S), N(S), \alpha) = \psi_i(\alpha) = F(S_i^*(\alpha))$. Therefore, $\max_{i \in K, \alpha \in A} F(S_i^*(\alpha)) = OPT(F, k)$, and it follows that Algorithm ColCodeNP correctly computes $OPT(F, k)$ with probability at least $1 - \epsilon$.

Next, we consider the space and time complexity. The algorithm maintains the array M, indexed by nodes and all possible color sets, which leads to the space complexity of $O(2^k n)$, since there are at most $2^k - 1$ possible non-empty color sets. The running time is the result of solving the recurrence for each node v, and for each color set T; this requires examining each possible partition $T_1 \cup T_2 = T$, and each neighbor $u \in Nbr(v)$, which requires time $O(|Nbr(v)|2^{|T|})$. Therefore, the total running time for each coloring is $O(\sum_v \sum_{i=0}^{k} |Nbr(v)|2^i) = O(\sum_v |Nbr(v)|2^k) = O(2^k m)$. The algorithm considers $O(e^k \log(n^2/\epsilon)) = O(e^k \log(n/\epsilon))$ colorings, so the running time follows. ■

3.4.4 Additional techniques for further scaling

We now discuss two techniques for scaling ColCodeNP to networks with over a million nodes without losing on the approximation guarantees significantly. For this section, we focus on non-parametric scan statistics functions of the form $\phi(W(S, \alpha), N(S), \alpha)$, where $W(S, \alpha) = N_\alpha(S) = \sum_{v \in S} I(p(v) \leq \alpha)$, and we refer to a node with p-value less (greater)

than α as an *anomalous* (*non-anomalous*) node. We notice that most of the functions in Table 3.1 are of this form. We take advantage of a *locality* property of these functions. Namely, if we are given a set S with score $F(S)$, we can increase the score of the set by adding any anomalous node that is a neighbor of a node already in S. This idea is formalized in the following lemma.

Lemma 6 *Let $G(V, E)$ be a network, and let F be a non-parametric scan statistic function. Given a set $S \subseteq V$, suppose there exists an anomalous node $u \notin S$ and an edge $(u, v) \in E$, for some $v \in S$. Then, $F(S \cup \{u\}) \geq F(S)$.*

Proof: Non-parametric scan statistics are increasing on the ratio $\frac{N_\alpha(S)}{N(S)}$. Since u is anomalous, we have that

$$\frac{N_\alpha(S \cup \{u\})}{N(S \cup \{u\})} = \frac{N_\alpha(S) + 1}{N(S) + 1} \geq \frac{N_\alpha(S)}{N(S)},$$

and the proof follows. ■

Graph refinement. An implication of Lemma 6 that neighboring anomalous nodes can be merged into a single *component* without loss in quality. We illustrate with an example in Figure 3.4a. In the figure, orange nodes are anomalous. The key idea is that any solution containing node A should also include node B and C, since $\phi(W(S, \alpha), N(S), \alpha)$ is increasing in the number of anomalous nodes. We call this compression operation *graph refinement.* In the figure, the number of anomalous nodes is reduced from 5 to 2 making it possible to discover the subgraph A through F using $k = 3$ instead of $k = 6$. In the experiments (Section 3.6.6), we show that this refinement is very effective in terms of size reduction in real networks.

More formally, we propose a preprocessing algorithm, RefineNP. Given a network $G(V, E)$, a vector $\mathbf{p}$ of p-values, and a significance level α, we define $V_1, \ldots, V_r$ as the connected components (or *supernodes*) of G induced by the set of anomalous nodes $V_\alpha = \{v : p(v) \leq \alpha\}$. We create a new graph $H(V', E')$, whose node set consists of $V_1, \ldots, V_r$ and the non-anomalous nodes in G, $V \setminus V_\alpha$. Edges between non-anomalous nodes are preserved in H, and we put an edge from a non-anomalous node u to V_i if the edge (u, v) exists, for some $v \in V_i$. Finally, we create a new vector of p-values, $\mathbf{p}'$.[1] The p-value of a node in H is $|V_i|$ for each supernode V_i, and 0 otherwise. This procedure is equivalent to removing all the nodes in V_α and replacing them with the respective supernode V_i.

Lemma 7 *Let $((G(V, E), \mathbf{p}, \alpha), \epsilon, k)$ be an input to* ColCodeNP*, and let S^*_G be the solution returned by the algorithm. Similarly, let $((H(V', E'), \mathbf{p}', \alpha), \epsilon, k)$ be the corresponding instance generated by* RefineNP*, and let S^*_H be the solution returned by* ColCodeNP *in this instance. Then, $F(S^*_H) = F(S^*_G)$. Furthermore, suppose $|S^*_G| = k$; then, $|S^*_H| = k'$, for*

[1] Note that the "p-values" in this new graph do not have a statistical meaning. It is more appropriate to think of these values as *weights* for the nodes.

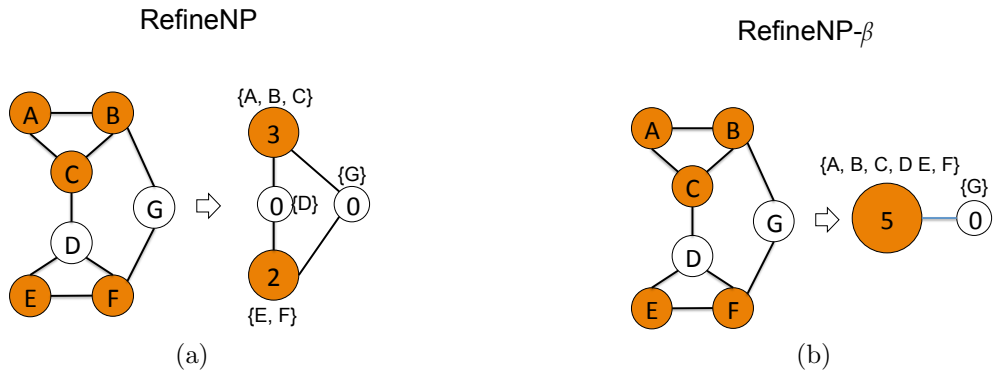


Figure 3.4: (a) **Example of exact graph refinement.** Colored nodes are anomalous (i.e., $p(v) \leq \alpha$). Nodes A, B, and C are merged into one component of weight 3, and E and F form a component of weight 2. The numbers of anomalous decreases from 5 to 2 allowing us to use a smaller parameter k without any loss in the quality of the solution. (b) **Example of graph refinement with** $\beta = 5/6$. By allowing some non-anomalous nodes to be merged, we obtain a single anomalous component with an approximation guarantee bounded by β.

some $k' \leq k$*, so it is possible to execute* ColCodeNP *with parameter* k' *and still obtain* $F(S^*_H) = F(S^*_G)$.

Proof: The lemma follows from Lemma 6. ■

We extend this idea to an *approximate refinement*. For a parameter $\beta \in [0, 1]$, we keep adding nodes to a component as long as the number of anomalous nodes remains more than β times the size of the component. Figure 3.4b shows an example for $\beta = 5/6$. By allowing non-anomalous node D to be merged, we are able to combine nodes A through F in a single component. Note that when $\beta < 1$, the solution may not be optimal, but the following lemma describes the effect on the approximation bound.

Lemma 8 *Let* S^* *be the set that maximizes* F *and let* $r(S^*) = \frac{N_\alpha(S^*)}{N(S^*)}$. *There is a solution on the instance* H *with ratio* $r(S)$ *at least* $\beta r(S^*)$.

Proof: We split the set S^* into anomalous nodes, $N_\alpha(S^*)$, and non-anomalous nodes, $N^-_\alpha(S^*)$, such that $N(S) = N_\alpha(S^*) + N^-_\alpha(S^*)$. We now show that, in the instance H, there exists a set S with ratio

$$r(S) = \frac{N_\alpha(S)}{N(S)} \geq \frac{N_\alpha(S^*)}{N_\alpha(S^*) + N^-_\alpha(S^*) + \frac{1-\beta}{\beta} N_\alpha(S^*)}.$$

We define S' as the set formed by the supernodes V_i corresponding to the anomalous nodes in S^*:

$$S' = \{V_i : (p(v) \leq \alpha) \wedge (v \in S^*) \wedge (v \in V_i)\},$$

and we note that $N_\alpha(S') \geq N_\alpha(S^*)$; for simplicity, we assume equality. By construction, the cardinality of S' is $N(S') = N\alpha(S') + \frac{1-\beta}{\beta}N_\alpha(S') = N_\alpha(S^*) + \frac{1-\beta}{\beta}N_\alpha(S^*)$. Note that the nodes in S' may be disconnected; however, we can connect them using a set of anomalous nodes S'' of size at most $N^-_\alpha(S^*)$. Finally, we form a set $S = S' \cup S''$ that has the desired ratio.

To conclude the proof, we compute $r(S)/r(S^*)$:

$$\frac{r(S)}{r(S^*)} = \frac{\frac{N_\alpha(S)}{N(S)}}{\frac{N_\alpha(S^*)}{N(S^*)}} \geq \frac{\frac{N_\alpha(S^*)}{N(S^*)+\frac{1-\beta}{\beta}N_\alpha(S^*)}}{\frac{N_\alpha(S^*)}{N(S^*)}} = \frac{1}{1+\frac{1-\beta}{\beta}\times\frac{N_\alpha(S^*)}{N(S^*)}}.$$

Noticing that $\frac{N_\alpha(S^*)}{N(S^*)} \leq 1$, we obtain $r(S) \geq \beta r(S^*)$. ■

Low radius subgraphs. Let $B_G(v, r)$ (referred to the ball of radius r at v) denote the set of nodes at distance at most r from v in the graph G. It suffices to run the algorithm restricted to the balls centered around anomalous nodes; this gives significant speedup if the number of anomalous nodes is small.

Algorithm FastColCodeNP shows the modified algorithm using the above observations.

Algorithm 2 FastColCodeNP$(G(V,E), \alpha_{max}, k, \epsilon, \beta)$.

Input: Instance $(G(V,E), \alpha_{max})$, parameters k, ϵ and β
Output: Set S^* with score $OPT(F,k)$
Let A be the set of p-values of nodes in V below α_{max}
for $\alpha \in A$
  Perform approximate refinement with parameter β.
  Let $H = (V', E')$ be the refined graph with weights $\mathbf{w}'$
  $\{S^*_i(\alpha) : i \in K\} =$ MaxWeight$(H(V', E'), \mathbf{w}', k, \epsilon/n^2)$
$S^* = argmax_{i\in[1,k],\alpha\in A} F(S^*_i(\alpha))$
return S^*

Theorem 6 *Consider an instance* $(G(V,E), c(\cdot), b(\cdot), k, \epsilon, \beta, r)$. *Let*

$$OPT(F,k,r) = \max_{S\subset V': |S|\leq k, S\subset B(v,r) \text{ for some } v} F_H(S),$$

where $F_H(S)$ *is the objective value of any non-parametric function* $F(\cdot)$ *from Table 3.1 on set* S *in graph* H. *Then, the solution* S^* *returned by* FastColCodeNP *satisfies* $\Pr[F(S^*) \geq h(\beta)OPT(F,k)] \geq 1-\epsilon$, *where the function* $h(\cdot)$ *depends on the specific function* $F(\cdot)$.

3.5 Algorithms for Parametric Scan Statistics and Extensions

In parametric scan statistics, both $c(v)$ and $b(v)$ are used as arguments to the function, which makes the problem more challenging than for non-parametric functions. Furthermore, there exist other score functions for graph anomaly detection where both nodes and edges have weights [40, 212]. Optimizing such functions reduces to the Prize Collecting Steiner Tree (PCST) problem [122], which is NP-Hard. We can extend the methods described above to these settings by keeping additional information in the dynamic program. We propose algorithm FastColCodeP for parametric scan statistics and algorithm ColCodeNW for PCST and its variants.

3.5.1 Extensions to Parametric Scan Statistics.

In Algorithm 3, we describe ColCodeP for parametric scan statistics maximization. For these functions, each node v of the input graph has two weights associated with it: $c(v)$ and $b(v)$. Therefore, we need a more general algorithm than ColCodeNP. Analogous to Lemma 3, we make use of the following property:

Lemma 9 *The parametric scan statistics functions characterized by equation (3.1) are increasing functions of $C(S)$ if $C(S) > B(S)$ and $B(S)$ is constant.*

Given a graph $G(V, E)$ and vectors $\mathbf{c}$ and $\mathbf{b}$, let $M(v, T, j)$ be the maximum value $C(S)$ over all connected subsets S, such that (1) $v \in S$, (2) S is colorful with respect to T, and (3) $B(S) = j$. Here j ranges from 1 to $B(V)$. $M(v, T, j)$ can be computed by a dynamic program with the following recurrence.

Lemma 10 *Let $M(v, T, j)$ be defined as above. For any node v and color s, $M(v, \{s\}, j) = c(v)$ if $col(v) = s$ and $b(v) = j$, else $M(v, \{s\}, j) = -\infty$. If $|T| \geq 2$:*

$$M(v, T, j) = \max_{\substack{u \in Nbr(v) \\ T_1, T_2 \subseteq T \\ j_1 + j_2 = j}} \{M(v, T_1, j_1) + M(u, T_2, j_2)\}.$$

where the maximum is over all partitions $T_1 \cup T_2$ of the set T, all integers j_1, j_2 with $j_1 + j_2 = j$. and all neighbors u of v.

Proof: Suppose $M(v, T, j)$ is achieved for a connected set S, such that $|S| = |T|$, $\{col(u) : u \in S\} = T$, and $B(S) = j$ with $M(v, T, j) = \sum_{i \in S} c(s)$. We claim that there exists $u \in Nbr(v)$, integers j_1, j_2: $j = j_1 + j_2$, and partitions $T = T_1 \cup T_2$, and $S = S_1 \cup S_2$,

such that: (1) $M(v,T,B) = M(v,T_1,j_1) + M(u,T_2,j_2)$, (2) $\{col(i) : i \in S_1\} = T_1$ and $\{col(i) : i \in S_2\} = T_2$, and (3) the node sets S_1 and S_2 are connected. Since S is connected, there exists a tree H that spans S rooted at node v. Further, there must exist a node $u \in Nbr(v)$ such that $(u,v) \in T$, since $|T| = |H| \geq 2$. Let H_1 and H_2 be the trees rooted at nodes v and u, respectively, that results when edge (u,v) is deleted in H. Let S_1 and S_2 denote the sets of nodes in H_1 and H_2, respectively. Let T_1 and T_2 be the colors used by S_1 and S_2, respectively. By construction, we have $\psi(v,T,j) = \psi(v,T_1,j_1) + \psi(u,T_2,j_2)$, so that the partitions $S = S_1 \cup S_2$ and $T = T_1 \cup T_2$ satisfy the requirements mentioned earlier. Therefore, the recurrence follows. ■

3.5.1.1 ColCodeP

Our algorithm for maximizing parametric scan statistics, ColCodeP, is presented in Algorithm 3. The procedure MaxWeightP uses Lemma 10 to compute $\psi_i = \max_{T:|T|=i} M(v,T,j)$ for all $i \leq k$.

Algorithm 3 ColCodeP$((G(V,E),\mathbf{C},\mathbf{B}),k,\epsilon)$.

Input: Instance $(G(V,E),\mathbf{C},\mathbf{B})$, parameters k and ϵ
Output: Set S^* with score $OPT(F,k)$
$\{S_i^* : i \in K\} =$ MaxWeightP$(G(V,E),\mathbf{C},\mathbf{B},k,\epsilon/n)$
$S^* = argmax_{i\in[1,k]} F(S_i^*)$
return S^*

procedure MaxWeightP$(G(V,E),\mathbf{C},\mathbf{B},k,\epsilon')$
Input: Instance $(G(V,E),\mathbf{C},\mathbf{B})$ and parameter k
Output: Set S_i^* with maximal weight ψ_i for all $i \in [1,k]$
Let $\psi_i = -\infty$ for all $i \in [1,k]$
for $t = 1$ to $e^k \log(1/\epsilon')$
  For each node v, pick random color $col(v) \in K$
  for $v \in V, s \in K, j \leq B(V)$
    $M(v,\{s\},j) = c(v)$ if $col(v) = s$ and $j = b(v)$; $-\infty$ otherwise
  for $v \in V$, $T \subseteq K$, with $|T| \geq 2$, $j \leq B(V)$
    Use Lemma 10 to compute $M(v,T,j)$
    If $M(v,T,j) > \psi_{|T|}$ **update** $\psi_{|T|} = M(v,T,j)$
return $\{S_i^* : \sum_{v\in S_i^*} c(v) = \psi_i, \text{for } i \in K\}$

Theorem 7 *Let $F(\cdot)$ be any of the parametric scan statistics in Table 3.1, and let $OPT(F,k) = \max_{S:|S|\leq k} F(S)$, where the maximum is over all connected subsets S of size $\leq k$.* ColCodeP *returns solution S^* satisfying $\Pr[F(S^*) = OPT(F,k)] \geq 1-\epsilon$, in time $O(2^k e^k m B_{max}^2 \log(n/\epsilon))$, and using space $O(2^k n B_{max})$, where $B_{max} = B(V)$.*

Proof: We prove below that the procedure MaxWeightP correctly returns ψ_i for $i \leq k$, within the required time and space bounds. From Lemma 9, it follows that $\max_{i\in[1,k]} \psi_i = \max_{|S|\leq k} g(C(S), B(S))$. The correctness of the algorithm follows from the proof of the recurrence in Lemma 10 and the bound on the success probability from Lemma 5. The algorithm maintains the array M, indexed by nodes, all possible color sets, and all integers in $[1, B_{max}]$, which leads to the space complexity of $O(2^k n B_{max})$, since there are at most 2^{k-1} possible non-empty color sets. Lemma 5 considers the probability of error, i.e., $\Pr[\max_j X_j \neq \psi_i]$ for one value of i. We need to consider this for k possible values. Therefore, taking $\epsilon' = \epsilon/n$ in Lemma 5 ensures that for each i, the probability $\Pr[\max_j X_j \neq \psi_i] \leq \epsilon/n^2$, so that for all i, α, the algorithm correctly finds ψ_i. The running time is the result of solving the recurrence for each node v, for each color set T, and value B; this requires examining each possible partition $T_1 \cup T_2 = T$, $B_1 + B_2 = B$, and each neighbor $u \in Nbr(v)$, which requires time $O(|Nbr(v)|B2^{|T|})$. Therefore, the total running time for each coloring is $O(\sum_v \sum_{i=0}^{k} \sum_{j=1}^{B_{max}} |Nbr(v)|2^i j) = O(\sum_v |Nbr(v)|2^k B_{max}^2) = O(2^k m B_{max}^2)$. Since the algorithm considers $O(e^k \log n/\epsilon)$ colorings, the running time bound follows. ■

We note that by approximating $B(S)$ within a factor of $(1+\delta)$, the running time in Theorem 7 can be improved to $O(2^k e^k m \frac{n^2}{\delta} \log(n/\epsilon))$, while losing a constant factor in terms of the approximation. We define $\mu = \delta B_{max}/n$, for some $\delta > 0$. Then, we define a vector $\mathbf{b}'$, where $b'(v) = \lfloor b(v)/\mu \rfloor$, for each node v. By invoking ColCodeP on the instance $(G = (V, E), \mathbf{c}', \mathbf{b}')$, we obtain a $(1 + \delta)$ approximation on the weight of S^*.

3.5.1.2 Scaling

There is a notion of graph refinement for parametric scan statistics analogous to the one presented for non-parametric functions. We note that the parametric functions in Table 3.1 are increasing on the ratio $r(S) = C(S)/B(S)$. Therefore, if we are given a set S with score $F(S)$, we can increase the score of the set by adding any node that is a neighbor of a node already in S as long as $r(S)$ does not decrease. This idea is formalized in the following lemma.

Lemma 11 *Let $G(V, E)$ be a network, and let $F(\cdot)$ be a parametric scan statistic function. Given a set $S \subseteq V$, suppose there exists an node $u \notin S$ and an edge $(u, v) \in E$, for some $v \in S$, such that $C(S \cup \{u\})/B(S \cup \{u\}) \geq C(S)/B(S)$; then, $F(S \cup \{u\}) \geq F(S)$.*

Exact refinement. For parametric scan statistics, the exact refinement consists of merging nodes into components as long as the ratio $r(S)$ of the component does not decrease. Given a network $G(V, E)$ and event $(c(v))$ and population $(b(v))$ counts for every $v \in V$, we maintain a list of components or supernodes $\mathcal{V} = V_1, \ldots, V_r$ and the graph $H(V', E')$ induced by those. There is an edge between V_i and V_j if there exist nodes $v_i \in V_i$ and $v_j \in V_j$, such that v_i and v_j are neighbors in G. Initially, every node is its own component. The exact refinement iterates through the list of current components trying to merge them until no more merges are

possible. In each iteration, we first sort the current components in descending order of ratio $r(S)$. Then, in that order, we merge a component with its neighbors if the ratio does not decrease. After this exact refinement, we run Algorithm 3 for the instance $(H(V', E'), \mathbf{c}', \mathbf{b}')$, where $\mathbf{c}'$ and $\mathbf{b}'$ are the event and population counts of the supernodes, respectively.

Approximate refinement. We can obtain larger components by allowing merges that decrease the ratio of the component. Our approximate refinement procedure takes two parameters: $\beta \in [0, 1]$ and $\delta > 1$. We allow a component V_i to grow as long as two constraints are not violated:

1. **Population size constraint.** The population size of V_i is at most δ times the smallest population size in the component: $B(V_i) \leq \delta \min_{v \in V_i} b(v)$.
2. **Event size constraint.** The event size of V_i is at least $\beta\delta$ times the largest event size in the component: $\frac{C(V_i)}{\delta} \geq \beta \max_{v \in V_i} c(v)$.

Lemma 12 *Let S^* be the set that maximizes a parametric scan statistic G and let $r(S^*) = \frac{C(S^*)}{B(S^*)}$. There is a solution S on the instance H constructed as above with ratio $r(S) \geq \beta r(S^*)$.*

Proof: Let S be the set formed by the supernodes V_i containing the nodes in S^*:

$$S = \{V_i : (v \in S^*) \wedge (v \in V_i)\}.$$

For simplicity, and without loss of generality, let us assume that each node of S^* is in a separate component in H. Then, we can analyze the ratio $r(S)/r(S^*)$:

$$\frac{r(S)}{r(S^*)} = \frac{\frac{C(V_1)+\ldots+C(V_{|S|})}{B(V_1)+\ldots+B(V_{|S|})}}{\frac{c(v_1)+\ldots+c(v_{|S|})}{b(v_1)+\ldots+b(v_{|S|})}}.$$

By the population size constraint δ,

$$\frac{\frac{C(V_1)+\ldots+C(V_{|S|})}{B(V_1)+\ldots+B(V_{|S|})}}{\frac{c(v_1)+\ldots+c(v_{|S|})}{b(v_1)+\ldots+b(v_{|S|})}} \geq \frac{\frac{C(V_1)+\ldots+C(V_{|S|})}{\delta b(v_1)+\ldots+\delta b(v_{|S|})}}{\frac{c(v_1)+\ldots+c(v_{|S|})}{b(v_1)+\ldots+b(v_{|S|})}} = \frac{\frac{C(V_1)+\ldots+C(V_{|S|})}{\delta}}{c(v_1)+\ldots+c(v_{|S|})}.$$

And, by the event size constraint β, we have

$$\begin{aligned}\frac{\frac{C(V_1)+\ldots+C(V_{|S|})}{\delta}}{c(v_1)+\ldots+c(v_{|S|})} &= \frac{\frac{C(V_1)}{\delta}+\ldots+\frac{C(V_{|S|})}{\delta}}{c(v_1)+\ldots+c(v_{|S|})} \\ &\geq \frac{\beta c(v_1)+\ldots+\beta c(v_{|S|})}{c(v_1)+\ldots+c(v_{|S|})} = \beta,\end{aligned}$$

so we conclude that $r(S) \geq \beta r(S^*)$. ■

3.5.2 Functions with Node and Edge Weights.

Both the Heaviest Subgraph [40] and EventTree+ problems [212] reduce to Prize Collecting Steiner Tree (PCST) with NetWorth objective [122]. In PCST, we are given a graph $G(V, E)$ with non-negative node prizes, $\boldsymbol{\pi}$, and non-negative edge costs, $\mathbf{w}$, and the goal is to find a tree $S(V(S), E(S))$ that maximizes the NetWorth objective:

$$W(S) = \sum_{v \in V(S)} \pi(v) - \sum_{e \in E(S)} w(e).$$

Using our framework, we can design an algorithm to find a tree with maximal NetWorth and size up to k, where k is a parameter. This implies an algorithm for HS and EventTree+.

Let $M(v, T) = \max_S W(S)$, where the maximization is over all connected and colorful sets $S \subseteq V$, such that $v \in S$, $|S| = |T|$, and $\{col(u) : u \in S\} = T$. $M(v, T)$ can be computed by a dynamic program:

Lemma 13 *Let $M(v, T)$ be defined as above. For any node v and color s, $M(v, \{s\}) = \pi(v)$ if $col(v) = s$, else $M(v, \{s\}) = -\infty$. If $|T| \geq 2$:*

$$M(v, T) = \max_{\substack{u \in Nbr(v) \\ T_1, T_2 \subseteq T}} \{M(v, T_1) + \max\{M(u, T_2) - w(v, u), 0\}\},$$

where the maximum is over all partitions $T_1 \cup T_2$ of the set T and all neighbors u of v.

Proof: The proof is analogous to that of Lemma 4, but this time we have to account for the weight of the edge connecting two subsets in the recursive step. Suppose $M(v, T)$ is achieved for a connected set S, such that $|S| = |T|$ and $\{col(u) : u \in S\} = T$, with $M(v, T) = \sum_{i \in S} \pi(i)$. We claim that there exists $u \in Nbr(v)$, and partitions $T = T_1 \cup T_2$, and $S = S_1 \cup S_2$, such that: (1) $M(v, T) = M(v, T_1) + M(u, T_2) - w(v, u)$, (2) $\{col(i) : i \in S_1\} = T_1$ and $\{col(i) : i \in S_2\} = T_2$, and (3) $G[S_1]$ and $G[S_2]$ are connected. Since $G[S]$ is connected, there exists a tree H that spans $G[S]$, rooted at node v. Further, there must exist a node $u \in Nbr(v)$ such that $(u, v) \in T$, since $|T| = |H| \geq 2$. Let H_1 and H_2 be the trees rooted at nodes v and u, respectively, that result when edge (u, v) is deleted in H. Let S_1 and S_2 denote the sets of nodes in H_1 and H_2, respectively. Let T_1 and T_2 be the colors used by S_1 and S_2, respectively. By construction, we have $M(v, T) = M(v, T_1) + \max\{M(u, T_2) - w(v, u), 0\}$, where the second term is negative if the weight $w(v, u)$ is greater than the NetWorth $M(u, T_2)$, in which case $M(v, T) = M(v, T_1)$. The partitions $S = S_1 \cup S_2$ and $T = T_1 \cup T_2$ satisfy the requirements mentioned earlier. Therefore, the recurrence follows. ■

3.6 Experiments

Our experiments address the following questions.
1. Optimization power. Do our algorithms find high-scoring subgraphs in real networks

and synthetic benchmarks? How do they compare with existing methods? (Section 3.6.3)
2. Event detection power. Do our algorithms correctly identify anomalous subgraphs? How do the precision and recall compare with other baselines?(Section 3.6.4)
3. Scalability. How do our algorithms scale to networks with more than 10^5 nodes? (Section 3.6.5)
4. Understanding the performance guarantees in real data. How does the performance in real datasets compare with the worst case bounds? (Section 3.6.6).

We focus on one scan statistic from each class as illustrative examples: (1) Berk Jones's (BJ) statistic, (2) positively-elevated mean scan (EMS), and (3) the Heaviest Subgraph (HS) and EventTree+ functions, as examples of non-parametric, parametric, and generalized functions with edge weights, respectively.

3.6.1 Datasets

We use datasets from different domains, including social networks, infrastructure networks, and standard synthetic benchmarks. Most of these datasets contain multiple instances, corresponding to snapshots of the networks at different times. A brief summary of the datasets is provided in Table 3.3.

CitHepPh[2]. This is a network of scientific collaborations between authors of papers submitted to the High Energy Physics - Phenomenology category of arXiv. The p-value of each node v for a specific snapshot was calculated as the ratio of nodes in the current graph snapshot whose citations are greater than or equal to the citations of this node.

NEast [145]. The Northeastern USA Benchmark is a well-known dataset in the spatial scan statistics community. The benchmark contains census information of 245 counties as well as synthetically generated cases of a disease.

Traffic[3]. The highway network of Los Angeles County, California and its activity on May, 2014. Nodes in the graph are sensors that record traffic statistics, such as average speed and the number of vehicles passing through. We assume a normal distribution for the average speed recorded by each sensor. In each snapshot t, the p-value of a node v is the cumulative distribution function of a normal distribution with mean $x_v^{[1,t-1]}$ and standard deviation $\sigma_v^{[1,t-1]}$, where $x_v^{[1,t-1]}$ and $\sigma_v^{[1,t-1]}$ are, respectively, the sample mean and standard deviation for node v from snapshots 1 to $t-1$.

Twitter. A sample of the follower graph of Venezuela collected between July 1, 2013 and December 31, 2013. We assign p-values based on the tweeting behavior of the users. Formally, let x_u^t be the number of tweets generated by node u at time t; we model x_u^t as a draw from a Poisson distribution with parameter λ_u. We take a Bayesian approach and

[2] `https://snap.stanford.edu/data/cit-HepPh.html`
[3] `http://pems.dot.ca.gov/`

consider λ_u to be drawn from a Gamma distribution with parameters α_u and β_u. These parameters are updated as we see new data every snapshot. The p-value of a node at time t is its posterior probability $p(n_e^t|n_e^{[1,t-1]})$, which follows a negative binomial distribution by our choice of prior.

Battle of the Water Sensor Networks (BWSN) [195]. This dataset is a benchmark originally used to evaluate different sensor network designs in terms of early detection of contaminants in a water system. The dataset includes "ground truth" subgraphs representing parts of the network that are contaminated, which we use for evaluation in Section 3.6.4.

PCST [122]. A standard benchmark to evaluate algorithms for the Prize Collecting Steiner Tree Problem. We use the "K" instances of the benchmark to evaluate methods that reduce to PCST (Section 3.6.3.3).

In addition, we also consider three large networks (i.e., over 10^5 nodes) from the SNAP repository [162] to evaluate the scalability and performance-runtime tradeoff of our proposed methods (Section 3.6.5). Since we do not have data of events in these networks, we plant events according to the statistical assumptions of the BJ scan statistic. We select a set of seed nodes and their neighbors in the graph. With probability α, each node has p-value less than 0.05. The remaning nodes have p-value uniformly distributed in $[0, 1]$.

Table 3.3: Datasets used in our experiments

Dataset	**Description**	**Nodes**	**Edges**	**Instances**
CitHepPh	Citation network	11,895	76,284	4
NEast	Network of counties in Northeastern USA	245	683	10,000
Traffic	Traffic Network of Los Angeles Country, CA	1,870	1,993	1,488
Twitter	Follower network collected through Twitter API	2,645	17,108	182
BWSN	Battle of the Water Sensors	12,527	14,831	22
PCST	Benchmark for the Prize Collecting Steiner Tree Problem	100-400	284-1,576	34
Email-EuAll	Email Network	224,832	340,795	1
Higgs-Retweet	Retweet Network	223,833	307,884	1
RoadNet-PA	Traffic Network of Pennsylvania	1,088,092	1,541,898	1

3.6.2 Baseline Methods

We compared our proposed algorithm with six state-of-the-art methods that are organized in three categories:
(1) `NPHGS` [54]: A local search heuristic for optimizing the BJ scan statistic. (2) `AdditiveGraphScan` [233] and `DepthFirstScan` [231]: The state-of-the-art algorithms for optimizing parametric scan statistics that satisfy the Linear-Time-Subset-Scanning (LTSS) property. The EMS statistic also belongs to this category.
(3) `GraphLaplacian` [225] and `EdgeLasso` [224]: The representative methods for anomalous subgraph detection that optimize their own specific score functions, but are often considered as baseline methods in connected subgraph detection papers (e.g., [205]).
(4) `EventTree+` [212] and `MEDEN` [40]: Algorithms for optimizing anomaly score functions with node and edge weights. The basic ideas of these methods are reviewed in Section 3.8.

Parameter Tuning. We note that some of the methods depend on user-specified parameters. When possible, we use the values prescribed by the authors of the method; this is the case with `NPHGS`, `EventTree+`, and `MEDEN`. In the case of `GraphLaplacian` and `EdgeLasso`, we tune the parameters separately for each dataset. In particular, we take a sample of 20 instances for each dataset and choose a parameter from $\{0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1\}$ that maximizes the average score. Notice that the parameter for each dataset may be different.

3.6.3 Optimization Power

3.6.3.1 Non-Parametric Scan Statistics

We compare our method FastColCodeNP to other algorithms in terms of the Berk-Jones (BJ) scan statistic. In Table 3.4, we report the average BJ score obtained by each method, where the average is taken over all the instances in each dataset. We observe that FastColCodeNP achieves higher scores than all other methods in every dataset. The difference in score is more pronounced in the NEast dataset, where FastColCodeNP more than doubles the score of `EdgeLasso` (EL) and `GraphLaplacian` (GL). We also note that `AdditiveGraphScan` has performance comparable to our algorithm, which is reasonable, since this method uses a sophisticated heuristic that avoids the usual challenges of Steiner connectivity problems.

3.6.3.2 Parametric Scan Statistics

Next, we compare our algorithm FastColCodeP to other baselines with respect to the Positively-Elevated Mean Scan Statistic (EMS). Table 3.5 shows the average score for different datasets. We find that FastColCodeP has the best performance in all datasets, except for NEast, where `AdditiveGraphScan` (GS) scores higher, but only by 0.6%.

Table 3.4: **Non-parametric scan statistics optimization.** The BJ score obtained by state-of-the-art NPSS optimization methods. We report the average over multiple instances for each dataset (See Table 3.3 for sizes). Our method FastColCodeNP obtains higher scores than all other methods in every dataset.

	Berk-Jones Scan Statistic					
	FastColCodeNP	GS	EL	GL	DFS	NPHGS
CitHepPh	**1138.011**	1135.029	559.176	1118.352	1130.874	1118.353
NEast	**68.525**	64.214	23.366	23.211	55.541	17.696
Traffic	**128.740**	128.722	116.732	125.824	14.412	121.632
Twitter	**1722.790**	1722.388	1722.388	1722.388	1720.243	1457.410
BWSN	**602.164**	599.972	530.850	530.457	536.200	531.280

Table 3.5: **Parametric scan statistics optimization.** We evaluate different methods with respect to the Postively-Elevated Mean scan statistic. FastColCodeP has better performace than existing methods in most datasets, except for NEast, where GS is 0.6% better.

	Elevated Mean Scan Statistic			
	FastColCodeP	GS	EL	GL
CitHepPh	**43.611**	8.578	14.959	41.830
NEast	41.903	**42.164**	5.570	7.607
Traffic	**11.763**	9.920	4.526	8.752
Twitter	**23.019**	11.337	22.660	19.110
BWSN	**109.097**	21.64	108.933	107.459

3.6.3.3 Functions with Node and Edge Weights

We also test our algorithm on two objective functions for event detection that consider edge weights in addition to node weights: the Heaviest Subgraph (HS) [40] and `EventTree+` [212] problems. The methods proposed in these two works are `MEDEN` [40] and `GreedyT` [212], respectively. Both problem formulations reduce to the Prize Collecting Steiner Tree (PCST) problem [122]. Thus, for the evaluation, we use our framework to design an algorithm for the PCST objective; we call this algorithm ColCodeNW, and we compare it in terms of objective score to `MEDEN` and `GreedyT` on the PCST benchmark of [122]. For the comparison with `MEDEN`, we convert the PCST instances to HS instances and run the TopDown heuristic described in [40]. For `GreedyT`, we convert the instances to a complete graph representation where an edge between two nodes has weight equal to the shortest path between the nodes, as described in [212]. Figure 3.5 shows the scores of the two heuristics relative to the score of ColCodeNW. Our algorithm finds subgraphs of higher quality than the heuristic methods. The score is as much as 4 times higher compared to `GreedyT` and 1.5 times higher compared to `MEDEN`.

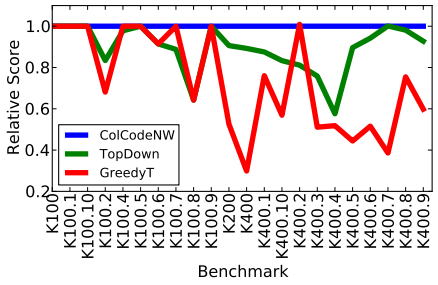


Figure 3.5: PCST objective score in a set of hard PCST benchmarks.

3.6.4 Event Detection Power

Now, we evaluate FastColCodeNP in terms of event detection power. We test whether our algorithm is able to identify anomalous subgraphs with and without noise. For this experiment, we use the ground truth provided with the BWSN dataset, and we evaluate in terms of accuracy, precision, recall, and the $F1$ score. Let R be the set of nodes in the anomalous subgraphs and let S be the detected subgraph; then, we define

(1) $\text{Accuracy}(R, S) = \frac{|R \cap S|}{|R \cup S|}$.

(2) $\text{Precision}(R, S) = \frac{|R \cap S|}{|S|}$

(3) $\text{Recall}(R, S) = \frac{|R \cap S|}{|R|}$

(4) $\text{F1 score} = 2\left(\frac{\text{Precision}_{(R,S)} \cdot \text{Recall}_{(R,S)}}{\text{Precision}_{(R,S)} + \text{Recall}_{(R,S)}}\right)$.

In order to assess the performance of our method under noise, we introduce a random percentage of uniform noise in each instance. For a given parameter l, each non-anomalous node becomes anomalous with probability l, and, vice versa, each anomalous node becomes non-anomalous with probability l. We refer to l as the *noise level*.

In Table 3.6, we compare the performance of FastColCodeNP (FCCNP) with previous algorithms at different noise levels. As in the previous section, we observe that our algorithm achieves higher objective scores compared to other methods, even when noise is present. As for the event detection power, FastColCodeNP has higher accuracy and recall for all the noise levels and higher F1 score at almost every level. `EdgeLasso` (EL) and `GraphLaplacian` (GL) have higher precision; however, this comes at the cost of recall because the solutions discovered by these methods are only a small subset of the real anomalous subgraph.

3.6.5 Scalability

Experiments on large networks. With the techniques presented in Section 3.4.4, we are able to run FastColCodeNP for non-parametric scan statistic evaluation in networks with over one million nodes. We note that in previous work only networks of up to $80,000$ nodes

Table 3.6: Average precision, recall, F1 score, accuracy, and objective value at different levels of noise.

	Precision				Recall				F1 Score				Accuracy				$F(S)$			
	GS	EL	GL	FCCNP	GS	EL	GL	FCCNP	GS	EL	GL	FCCNP	GS	EL	GL	FCCNP	GS	EL	GL	FCCNP
0%	0.980	**0.999**	0.901	0.977	0.943	0.856	0.856	**0.955**	0.948	0.895	0.820	**0.952**	0.966	0.855	0.820	**0.973**	599.972	530.850	530.457	**602.164**
2%	0.974	0.991	**0.995**	0.973	0.967	0.796	0.772	**0.975**	**0.970**	0.854	0.842	0.957	0.946	0.789	0.769	**0.950**	579.197	427.984	437.783	**580.977**
4%	0.945	**0.985**	0.984	0.966	0.955	0.687	0.663	**0.971**	0.952	0.775	0.757	**0.963**	0.912	0.678	0.652	**0.929**	565.363	393.930	387.914	**571.231**
6%	0.959	0.964	**0.973**	0.954	0.937	0.567	0.542	**0.953**	0.946	0.683	0.664	**0.953**	0.901	0.558	0.536	**0.912**	522.694	318.593	300.118	**531.497**
8%	0.928	0.960	**0.966**	0.931	0.888	0.561	0.502	**0.919**	0.905	0.670	0.626	**0.923**	0.830	0.544	0.490	**0.860**	483.127	315.720	291.227	**491.657**

have been considered, and much smaller than that in most cases (Table 3.1). In Table 3.7, we compare our algorithm to existing methods in terms of objective score and running time. If an algorithm did not finish executing after 10 hours or failed because of memory constraints, we show a "-" sign. First, we compare FastColCodeNP to the best-performing heuristic: `AdditiveGraphScan` (GS). We note that both methods achieve the same score in all datasets, but the running time of the latter is one order of magnitude larger, and it didn't complete after 10 hours for the road network. Second, we observe that `GraphLaplacian` (GL) does not scale to large networks due to the fact that this is a this is a constrained quadratic programming method, and has space complexity $O(n^2)$ and time complexity $O(n^3)$. We also note that our algorithm is much faster on the road network than on the other two; this is because nodes in this planar network have low degree, thus making our low-radius technique more effective. Finally, we note that `EdgeLasso`, `DFS`, and `NPHGS` are relatively fast compared to FastColCodeNP —especially `NPHGS`— however, for these large networks, the performance in terms of optimization is significantly worse than our method.

Table 3.7: Performace-runtime tradeoff in large datasets for non-parametric scan statistic evaluation.

	Berk-Jones Scan Statistic (Time in seconds)					
	FastColCodeNP	GS	EL	GL	DFS	NPHGS
Email-EuAll	**420.46** (2,376)	420.46 (20,254)	275.08 (679)	-	392.53 (3,671)	275.08 (10)
Higgs-Retweet	**839.18** (1,015)	839.18 (32,340)	421.16 (585)	-	721.70 (3,213)	421.16 (5)
RoadNet-PA	**24.66** (22)	-	24.66 (7,919)	-	21.22 (1,584)	13.28 (15)

3.6.6 Understanding the performance guarantees in real data

Effect of graph refinement. In Table 3.8, we report the average number of effective anomalous components (at a significance level of 0.15) before and after the graph refinement operation (Section 3.4.4) for $\beta \in \{1, 0.95, 0.90\}$. Each dataset initially contains hundreds of anomalous nodes, with Twitter being close to $1,000$. However, after we merge the anomalous nodes into components, we are left with a much smaller search space without any loss in quality. If we allow a few anomalous nodes into each component (as few as 5%), we are able to further reduce the effective number of anomalous nodes, down to a single digit in most cases.

Table 3.8: Number of anomalous nodes after graph refinement. Our preprocessing step compresses the anomalous nodes into a small number of supernodes in each networks. This allows us to discover high quality solutions using a small value of k.

Dataset	N_α ($\alpha = 0.15$)	**Graph Refinement**		
		$\beta = 1$	$\beta = 0.95$	$\beta = 0.90$
CitHepPh	624	20	3	1
Neast	65	24	24	21
Traffic	157	61	61	60
Twitter	991	80	2	1
BWSN	333	4	2	2

Convergence in few iterations. Algorithm FastColCodeNP uses $\ell = e^k \log n^2/\epsilon$ random colorings, in order to guarantee a solution with probability $1 - \epsilon$. In practice, we find the number of colorings needed is much smaller—this is shown in Figure 3.6 for the PCST dataset. Each line in the plot represents the solution obtained for one instance of size 100; the y-axis shows the objective value obtained normalized by the objective obtained in the last iteration of the algorithm. The theoretical bound requires $3,285$ random colorings to guarantee 95% probability. However, the best solution is found in less than 20 iterations for all instances, and sooner for most of them. We observed similar results in the other networks in Table 3.3.

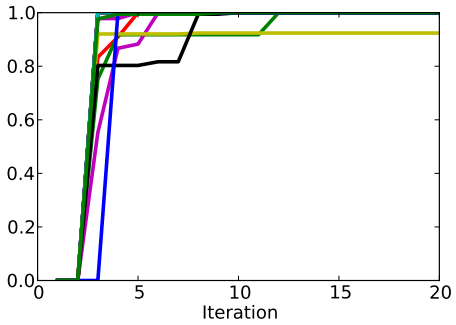


Figure 3.6: Objective value of the solution found (normalized by that of the optimum) as a function of the number of iterations

3.7 Application

As an application of our algorithms, we explore event detection in Twitter follower graphs using the Heaviest Dynamic Subgraph formulation of [40].

We model interactions –i.e. retweets, replies, direct messages– between Twitter users using a Poisson distribution specific to the corresponding edge between the users in the follower

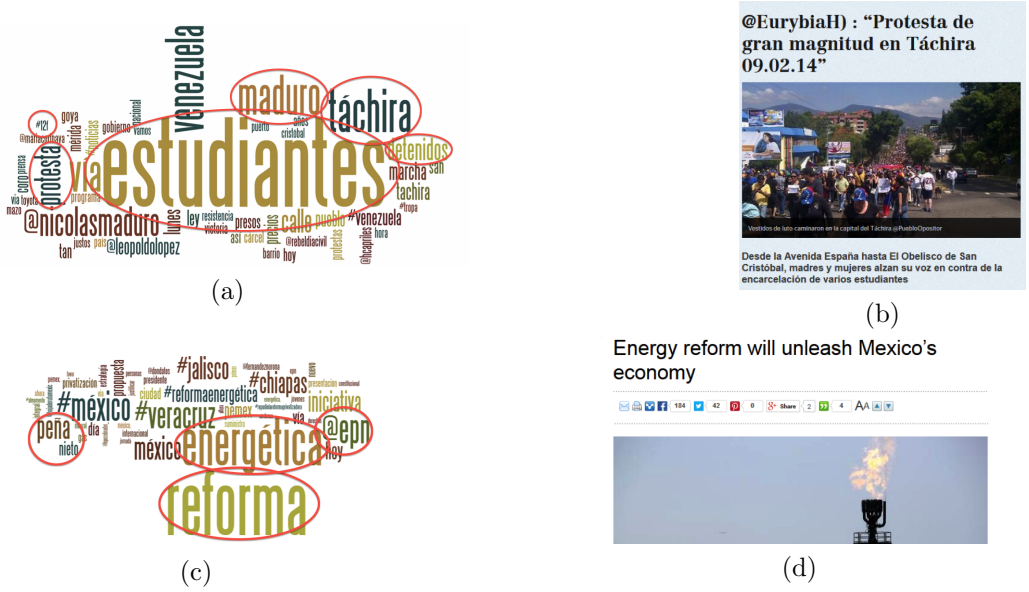


Figure 3.7: Examples of events found in the heavy subgraphs

graph. Formally, let n_e^t be the number of tweets that generated at edge e at time t; we model n_e^t as a draw from a Poisson distribution with parameter λ_e. We take a Bayesian approach and consider λ_e to be drawn from a Gamma distribution with parameters α_e and β_e. These parameters are updated as we see new data every day. Finally, the weight of an edge is given by $-\log(P(n_e^t|n_e^{[1,t-1]}))/\mu$, where $P(n_e^t|n_e^{[1,t-1]})$ is the posterior probability of n_e^t given the counts in the previous days, and $\mu = 0.05$ is a significance threshold. This weighing function was proposed by [178]; it has the desired properties of being positive-increasing if the posterior probability is less than μ and negative-decreasing otherwise.

After assigning weights to edges as described above, we solve the HDS problem in the Twitter dataset using the ColCodeNW for Heaviest Subgraph (see Section 3.5). For Venezuela, the temporal subgraph obtained spans the period of time between January 4, 2014 and March 31, 2014, which was a period of intense nationwide protests against the central government of president Nicolas Maduro.

We are also interested on interpreting the heavy subgraphs. To this end, we examined the subgraphs found by our algorithm and extracted the corresponding tweets of this subgraph. We find that these tweets contain chatter about important national events. Figure 3.7 shows an example of two events extracted from heavy subgraphs, one for Venezuela and one for Mexico. For Venezuela, the predominant terms of the tweets relate to a protest organized in the city of Tachira demanding the liberation of local students, who had been put in jail in the previous days for protesting. For Mexico, the two most common words in the extracted tweets form the phrase "energy reform" referring to a bill recently proposed by Mexican president Enrique Pena Nieto that would have a significant impact in the economy of the country.

3.8 Related Work

Although a large number of detection algorithms have been proposed for different kinds of scan statistics in networks, *no computationally tractable algorithms with rigorous guarantees are known for any of the objectives discussed below*, other than the PCST objective. Table 3.9 compares our method with several state-of-the-art algorithms on supported scoring functions, time complexity, and performance bound. Many of these heuristics have reasonable performance on some datasets, but are not consistent, as we find in our experiments. Since better approximation bounds often imply better detection power, this can be a problem in practice.

Table 3.9: Comparison of subgraph detection methods. n and m are the total numbers of nodes and edges in the input graph, respectively; d is a maximum depth parameter; l is the number of iterations; t is the number of snapshots; k is the solution size of our algorithm and is ≤ 10 in most cases.

Method	**Score function**	**Time Complexity**	**Performance Bound**
`Meden` [40]	Linear	$O(nt \log^2 t)$	No
`EventTree+` [212]	PCST objective [122]	$O(n^2 \log n)$	2-approximation
`AdditiveGraphScan` [233]	Nonlinear	$O(mn + n^2 \log n)$	No
`DepthFirstScan` [231]	Nonlinear	$O(n \cdot 2^d)$	No
`EdgeLasso` [224]	Quadratic	$O(l \cdot n^3)$	No
`GraphLaplacian` [225]	Quadratic	$O(l \cdot n^3)$	No
`NPHGS` [54]	Nonlinear	$O(n \log n)$	No
Our algorithms	Linear, Nonlinear	$O(2^k \cdot e^k m \log \frac{n}{\epsilon})$	$(1-\epsilon)$-approximation

3.8.1 Algorithms for optimization of parametric scan statistics.

These fall into three categories:

1. **Exact algorithms**, such as exhaustive search over all connected subgraphs [241], a branch-and-bound method for Kulldorff's spatial scan statistic, and upper level set scan statistic [231]. However, *these methods do not scale to graphs with more than 1000 nodes.*

2. **Heuristic algorithms**, which include (i) a simulated annealing approach that is based on a concept of "non-compactness" for penalizing clusters [72], (ii) the Additive GraphScan algorithm, which connects clusters based on shortest path distances [233], (iii) sparse learning method based on edge-lasso regularization [224], (iv) spectral scan method based on graph Laplacian regularization [225], and (v) submodular optimization algorithm based on Lovasz extensions [223]. *No quality guarantees are known for these methods*, when used for optimizing parametric scan statistics, in general.

3. **Algorithms based on density or Steiner connectivity**, a semi-definite programming based method [205], and the use of standard solutions of MAXCUT and PCST [212]. These methods work well in practice and give guarantees for the PCST objective (based on [94]), but they do not directly optimize a specific scan statistic function.

A number of other methods in this category consider relatively simple graphs, such as lines, lattices or trees, and planar graphs, and optimize over subgraphs of special forms, such as rectangles, balls, or some other low-dimensional parametric shapes [5, 64, 144, 184]. These methods are inapplicable to general graphs and are not reviewed here.

We also note that there has been a lot of work on parametric scan statistic optimization in non-network data [144, 185]. An important result due to Neill [185] is that unconstrained maximization of scan statistics can be performed efficiently.

3.8.2 Algorithms for optimization of non-parametric scan statistics.

Although non-parametric scan statistics have been widely used in a variety of pattern detection applications [68, 121], their applications to anomalous cluster detection have only been explored recently. Several papers apply non-parametric scan statistics to detect anomalous clusters in non-graph data [173, 183, 188]. Chen and Neill presented a fast heuristic algorithm to optimize non-parametric scan statistics on general graphs [54], with applications to detection of civil unrest, disease outbreaks [53], and human rights events [55], but this algorithm does not provide worst-case theoretical guarantees.

3.8.3 Reduction to variants of Network Design.

A common approach for dealing with connectivity requirements is to use algorithms for Prize Collecting Steiner Tree [122] (PCST) and other kinds of network design problems. For instance, the EventTree and EventTree+ problems in [212] are exactly the PCST problem. The authors propose a simple greedy heuristic. In [40] and [178], the authors propose the Heaviest Dynamic Subgraph (HDS) problem and a more general version called Significant Anomalous Regions (SAR). Both formulations reduce to the NetWorth objective, which is a complement of PCST. In, [54] and [205], two different scan statistic methods are proposed. These formulations have connections to the Quota Steiner tree problem and Budgeted Steiner tree, respectively. Rigorous guarantees are known for some of these objectives, such as for PCST and the budgeted steiner tree objective. However, no guarantees are known for the NetWorth and Quota objectives.

3.9 Conclusions

Anomaly detection is a fundamental task in network analysis with a large number of applications to different domains, and scan statistics is one particular methodology that has been widely applied for this task.

The detection power of methods based on scan statistics is reliant on the degree to which we can optimize a given "anomalousness" function over connected subgraphs. This connectivity constraint makes the problem very challenging, compared to the looser constraint of spatial adjacency that has been studied extensively. In fact, here, we show strong connections to the classical Steiner Tree problem in graph theory.

In order to tackle the computational complexity, recent papers have proposed various heuristics for scan statistics on graphs, which have been shown to have good empirical performance while, at the same time, being very efficient and scalable. However, one notable drawback of heuristics is that they seldom offer rigorous theoretical guarantees on the quality of the solutions discovered. Because of the lack of guarantees, heuristics may perform poorly in some datasets, and this will affect the ability to detect anomalous events in the graph. Furthermore, the lack of guarantees also makes it challenging to compare two competing heuristics. It is not clear whether one method is always superior to the other or whether both are better in different parts of the problem space.

Instead, we propose algorithms based on parameterized complexity. We find that fixed parameter tractability is a powerful, but underexplored approach for designing efficient algorithms, and it is likely to be useful in other graph mining applications. We present a unified framework for optimizing a broad class of graph scan statistics with connectivity constraints, with the following novel characteristics: (1) it gives rigorous guarantees for a large class of parametric and non-parametric score functions, and (2) it can be scaled to large graphs with over a million nodes.

Even though we mostly discuss our methods on the context of single graph snapshots, the methods extend trivially when the node observations vary over time, but the graph structure remains static. In this setting, an anomaly is roughly defined as a subgraph where the observations differ significantly from the empirical distribution; that is the distribution given by the observations up to this point in history. The scan statistics methodology handles this temporal setting, and our maximization algorithms extend without any modification.

An interesting direction of future work is handling streaming data, where instead of obtaining updated values for all nodes of the graph at once, these values come one by one in a streaming fashion, and we want to maintain the most anomalous subgraph at each point in time. Given the connections with Steiner connectivity, we conjecture that some ideas from the online Steiner tree problem [100] could be used for this streaming setting.

Chapter 4

Faster Graph Scan Statistics Optimization Using Algebraic Fingerprints

4.1 Introduction

Following on the track of fixed-parameter tractable algorithms for anomalous subgraph detection, we adapt a recently developed algebraic technique from the parameterized complexity literature to design improved sequential and parallel algorithms for the problems we studied in Chapter 3. Our contributions in this chapter are the following.

1. **Improved algorithms with guarantees for graph scan statistics.** We develop MultilinearScan, which uses the multilinear detection technique [136] for optimizing a large class of both parametric and non-parametric graph scan statistics with connectivity constraints (Section 4.3.4).

2. **Parallel algorithm.** We develop a parallel version of MultilinearScan using Giraph (Section 4.3.5), which allows us to scale to very large instances with over 40 million edges; all earlier sequential and parallel methods have been shown to run on networks with less than 2 million edges.

3. **Experimental results**. We evaluate MultilinearScan on a number of real and synthetic networks, and we observe significant improvement over all prior sequential and parallel algorithms with respect to multiple criteria (Section 3.6):
(1) ***Scalability***: MultilinearScan improves over ColCodeNP (Chapter 3) by more than three orders of magnitude with respect to the running time and 1–2 orders of magnitude with respect to memory usage, with less than 1% loss in accuracy (see

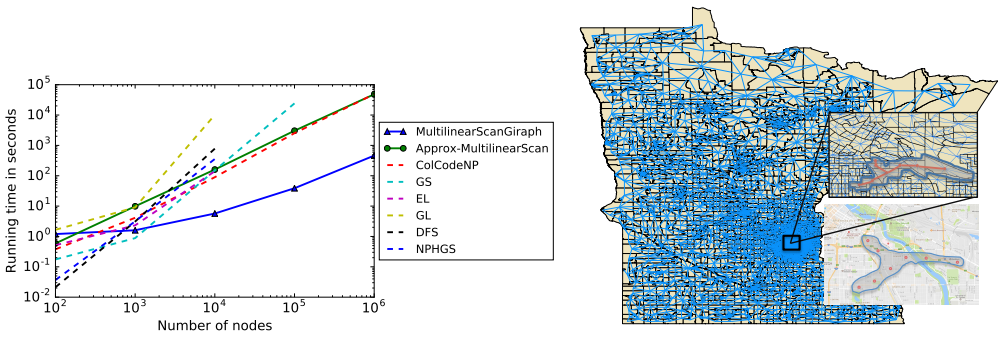


Figure 4.1: Summary of results. (Top) Comparison of the running times (seconds) for different graph sizes (see Section 3.6.5 for details). Our algorithm, MULTILINEARSCANGIRAPH, is a parallel Pregel algorithm implemented in Giraph, has rigorous theoretical guarantees (Theorem 10 and Lemma 14), and scales to graphs with millions of nodes, in contrast with most previous methods. (Bottom) Using MULTILINEARSCAN, we discover low-vaccination clusters in the census block group graph of Minnesota (see Section 4.4.5).

Figure 4.1 (top)). This implies similar improvements in the running time over other sequential heuristic methods discussed in the previous chapter. MULTILINEARSCAN also improves over the distributed method of [269], both in terms of running time and performance.
(2) ***Finding solutions with larger effective size and improved performance***: Our approach enables search for solutions with much larger effective solution size than possible with COLCODENP [47], leading to better quality solutions in many cases.

4. **Case study: finding unvaccinated clusters**. We illustrate our methods by applying them to the task of finding clusters of low-vaccination in a geographical region—we consider the state of Minnesota in particular. A preview of our results is presented in Figure 4.1.

4.2 Preliminaries

4.2.1 Multilinear Detection

Let $X = x_1, \ldots, x_n$ be a set of variables, and let $P(X)$ be a polynomial, which is a sum of monomials on X. An example of a polynomial on 4 variables is $P(x_1, x_2, x_3, x_4) = x_1^2 x_2 + x_1 x_2 x_3 + x_2 x_4^2$. A monomial is called *multilinear* or *square-free* if all its variables have exponent 1, and its *degree* is the sum of the exponents of all its variables. For instance, in the example above, $x_1 x_2 x_3$ is the only multilinear monomial, and all the monomials have degree 3. Given variables $X = x_1, \ldots, x_n$ and a polynomial $P(X)$, the goal in the k-Multilinear

Detection (k-MLD) problem is to decide whether or not $P(X)$ has a multilinear monomial of degree exactly k. We note that the polynomial $P(X)$ may have an arbitrary number of terms—i.e., exponential on the size of n—therefore, the problem is not as simple as writing the polynomial explicitly and checking each term. Rather, we assume that $P(X)$ is given succintly in a recursive form, and the "yes"/"no" decision has to be made without unrolling this recursion.

4.2.2 Algorithm for Multilinear Detection

The main idea in the algorithm of [136] is that, if we evaluate a polynomial over the "right" algebra, monomials that have square terms evaluate to $\bar{0}$ (which is the additive identity in the algebra), and the remaining terms, which are multilinear, do not cancel out, with high probability. Then, a polynomial $P(X)$ has a k multilinear term if $P(X) \neq \bar{0}$. Let $\mathbb{Z}_2^k$ be the group formed by all the k-dimensional binary vectors, and define the group multiplication operation as entry-wise XOR. For example, $\mathbb{Z}_2^2$ consists of the vectors $v_0 = (0,0), v_1 = (0,1), v_2 = (1,0), v_3 = (1,1)$. We note that v_0 is the multiplicative identity of the group, and each element is its own multiplicative inverse: $v_i \cdot v_i = v_0$. Now, we define a group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$. Each element in the group algebra is a sum of elements from $\mathbb{Z}_2^k$ with coefficients from $\mathbb{Z}_2$ (i.e., either 1 or 0): $\sum_{v \in \mathbb{Z}_2^k} a_v v$, where $a_v \in \{0,1\}$. The addition operator of the group algebra is

$$\sum_{v \in \mathbb{Z}_2^k} a_v v + \sum_{v \in \mathbb{Z}_2^k} b_v v = \sum_{v \in \mathbb{Z}_2^k} (a_v + b_v) v,$$

where the addition of the coefficients is modulo 2, and the multiplication is defined as

$$\left(\sum_{v \in \mathbb{Z}_2^k} a_v v \right) \left(\sum_{u \in \mathbb{Z}_2^k} b_u u \right) = \sum_{v \in \mathbb{Z}_2^k} (a_v \cdot b_u)(v \cdot u).$$

The key insight in [136] is that, for any $v_i \in \mathbb{Z}_2^k$, the square of the term $(v_0 + v_i) \in \mathbb{Z}_2[\mathbb{Z}_2^k]$ evaluates to $\bar{0}$:

$$(v_0 + v_i)^2 = v_0^2 + 2(v_0 \cdot v_i) + v_i^2 = v_0 + (0 \mod 2) v_i + v_0 = 2v_0 = \bar{0}.$$

Then, the algorithm of [136] is roughly as follows:

1. For each variable x_i, sample a vector v_i uniformly at random from $\mathbb{Z}_2^k$ and assign $x_i = (v_0 + v_i) \in \mathbb{Z}_2[\mathbb{Z}_2^k]$.
2. Evaluate the polynomial $P(x_1, \ldots, x_n)$ on this random assignment.
3. If $P(x_1, \ldots, x_n) \neq \bar{0}$ return "yes"; otherwise, return "no", where $\bar{0}$ is the additive identity of the group algebra.

Example. For $k = 2$, the group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$ has $2^{2^2} = 16$ elements, such as

$$x_1 = 0 \cdot \begin{bmatrix}0\\0\end{bmatrix} + 1 \cdot \begin{bmatrix}0\\1\end{bmatrix} + 1 \cdot \begin{bmatrix}1\\0\end{bmatrix} + 0 \cdot \begin{bmatrix}1\\1\end{bmatrix}, \text{ which we also write as } \begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}$$

$$x_2 = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}$$

We have

$$x_1 + x_2 = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} + 2\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}$$

$$x_1 x_2 = \left(\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}\right) \cdot \left(\begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}\right) = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix}$$

It is easy to check that

$$x_1^2 x_2 = 0 \cdot \begin{bmatrix}0\\0\end{bmatrix} + 0 \cdot \begin{bmatrix}0\\1\end{bmatrix} + 0 \cdot \begin{bmatrix}1\\0\end{bmatrix} + 0 \cdot \begin{bmatrix}1\\1\end{bmatrix} = \bar{\mathbf{0}} \text{ (additive identify)}$$

We can show that, under this random assignment, a multilinear monomial does **not** evaluate to $\bar{0}$ with high probability, whereas a monomial with squares is always $\bar{0}$ (as in the box above). However, monomials that appear an even number of times will also evaluate to $\bar{0}$. The method was later refined in [261] to avoid this problem, by evaluating the polynomial over the group algebra $GF(2^{3+\log_2 k})[\mathbb{Z}_2^k]$, where $GF(p)$ is the finite field of order p [180].

4.2.3 Reducing Space Using Matrix Representations

The algorithm for k-MLD as stated above has a space complexity proportional to the size of the group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$ because we need to represent and store this many elements. The space complexity is then $O(2^k poly(n))$.

We can improve this bound by evaluating the polynomial over a matrix representation, as discussed in [136].

4.2.3.1 Matrix Representation

Let $M^{d\times d}$ be the group of $d \times d$ matrices of integers, with the group operation being matrix multiplication. Then, there is a one-to-one map $\phi : \mathbb{Z}_2^k \to M^{2^k \times 2^k}$, such that $\phi(v_i v_j) =$

$\phi(v_i)\phi(v_j)$ [136, 245]. For example, the four binary vectors of $\mathbb{Z}_2^2$ have the following map:

$$\phi(v_0) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \phi(v_1) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix},$$

$$\phi(v_2) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \phi(v_3) = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

We can verify that, for instance, $\phi(v_2v_3) = \phi(v_1) = \phi(v_2)\phi(v_3)$. We can also verify that $\phi(v_0)$ is the identity of the matrix group and each element is its own inverse. One useful property of the matrices given by ϕ is that they are simultaneously diagonalizable; that is, there is a matrix U, such that, for all $v \in \mathbb{Z}_2^k$, $\phi(v) = U^{-1}\Lambda_v U$, where U^{-1} is the inverse of U and Λ_v is the (diagonal) matrix of eigenvalues of $\phi(v)$.

Now, we define an algebra $\mathcal{M}^{d\times d}$ of $d \times d$ matrices of integers, with matrix addition, matrix multiplication, and multiplication by a scalar. The map ϕ for the group of binary vectors can be used to create a one-to-one map Φ from the group algebra $\mathbb{Z}[\mathbb{Z}_2^k]$ of binary vectors with *integer* coefficients—not only 0 or 1—to $\mathcal{M}^{2^k\times 2^k}$:

$$\Phi\left(\sum_{v\in\mathbb{Z}_2^k} a_v v\right) = \sum_{v\in\mathbb{Z}_2^k} a_v \phi(v).$$

4.2.3.2 From $O(2^k)$ to $O(k)$ Space Complexity

Instead of evaluating the polynomial on the algebra with binary coefficients, $\mathbb{Z}_2[\mathbb{Z}_2^k]$, we can evaluate on the algebra of integer coefficients, $\mathbb{Z}[\mathbb{Z}_2^k]$, and just check the parity of said coefficients.

$$\text{If } \; P_{\mathbb{Z}_2[\mathbb{Z}_2^k]}(X) = \sum_{v\in\mathbb{Z}_2^k} a_v v, \text{ then, } \; P_{\mathbb{Z}[\mathbb{Z}_2^k]}(X) = \sum_{v\in\mathbb{Z}_2^k} (a_v \mod 2)v,$$

where $P_{\mathcal{A}}(X)$ denotes that when we evaluate the polynomial P, the variables X are assigned elements of the algebra $\mathcal{A}$.

Furthermore, instead of evaluating the polynomial on the algebra of integer coefficients, $\mathbb{Z}[\mathbb{Z}_2^k]$, we can evaluate on the algebra of matrices, $\mathcal{M}^{2^k\times 2^k}$, and just check the trace of the polynomial:

$$\text{If } \; P_{\mathbb{Z}_2[\mathbb{Z}_2^k]}(X) = \bar{0}, \text{ then, } \; trace(\Phi(P_{\mathbb{Z}[\mathbb{Z}_2^k]}(X))) = 0 \mod 2^{k+1}.$$

So, instead of assigning $x_i = (v_0 + v_i)$ (Section 3.2), we assign $x_i = \Phi(v_0 + v_i)$, evaluate P over the matrix algebra, and compute the trace (modulo 2^{k+1}).

The trace of the polynomial $P_{\mathcal{M}^{2^k \times 2^k}}(X)$ can be expressed in terms of eigenvalues because the matrices in $\mathcal{M}^{2^k \times 2^k}$ are simultaneously diagonalizable. Let Λ_i denote the eigenvalue matrix of $\Phi(v_0 + v_i)$ and let $P_{\mathcal{A}}(\bar{\Lambda})$ denote the polynomial where we assign $x_i = \Lambda_i$. Then, we have

$$trace(P_{\mathcal{M}^{2^k \times 2^k}}(X)) = trace(U^{-1} P_{\mathcal{M}^{2^k \times 2^k}}(\bar{\Lambda}) U) = trace(P_{\mathcal{M}^{2^k \times 2^k}}(\bar{\Lambda})).$$

The last idea is that we can compute the trace as a sum over rows in the eigenvalue matrices. Let Λ_{it} be the t^{th} eigenvalue of $\Phi(v_0 + v_i)$—i.e., the t^{th} diagonal entry in Λ_i. Let $P_{\mathbb{Z}_2^{k+1}}(\bar{\Lambda}_t)$ denote the polynomial where we assign $x_i = \Lambda_{it}$ and evaluate taking modulo 2^{k+1}. Then,

$$trace(P_{\mathcal{M}^{2^k \times 2^k}}(\bar{\Lambda})) = \sum_{t=1}^{2^k} P_{\mathbb{Z}_2^{k+1}}(\bar{\Lambda}_j).$$

We can compute the t^{th} eigenvalue in Λ_i as $\Lambda_{it} = 1 + (-1)^{v_i^T \cdot (t-1)_{bin}}$, where $(t-1)_{bin}$ is the binary representation of $(t-1)$ in k bits.

The algorithm of [136] can now be written as follows:

1. For each variable x_i, sample a vector v_i uniformly at random from $\mathbb{Z}_2^k$.
2. For $t = 0$ to $2^k - 1$:
   (a) Assign $x_i = 1 + (-1)^{{v_i}^T \cdot t_{bin}}$
   (b) Evaluate the polynomial $P^t(x_1, \ldots x_n)$ on this assignment
3. If $\sum_{t=0}^{2^k-1} P^t \neq 0 \mod 2^{k+1}$, return "yes"; otherwise, return "no".

With this algorithm, we evaluate the polynomial 2^k times, and each evaluation takes space $O(kpoly(n))$, since all the operations are now modulo 2^{k+1}. This is the version that we implement as part of MultilinearScan. We have the following theorem.

Theorem 8 (Koutis [136] and Williams [261]) *There exists an algorithm that, given an instance $P(x_1, \ldots, x_n)$ of the k-MLD problem, correctly returns "no" if $P(X)$ does not contain a k multilinear term. Otherwise, if $P(X)$ has a k multilinear term, it returns "yes" with probability at least 1/5. The algorithm has time complexity $O(2^k poly(n))$ and space complexity $O(kpoly(n))$.*

4.3 Scan Statistics Using Multilinear Detection

Recall from Chapter 3 that our goal is to solve the following optimization problem.

Problem 4 (Most Anomalous Connected Subgraph) *Given a graph $G = (V, E)$, a scan statistic $F(\cdot)$, the associated counts for vertices—$\mathbf{w}$ and $\mathbf{b}$—and a parameter $k \ll |V|$, find a connected subset $S \subseteq V$ that maximizes $F(S)$ with $B(S) \leq k$.*

Our approach involves reducing Problem 4 to k-MLD for suitably defined polynomials. The overall idea of our algorithm is illustrated in Figure 4.2.

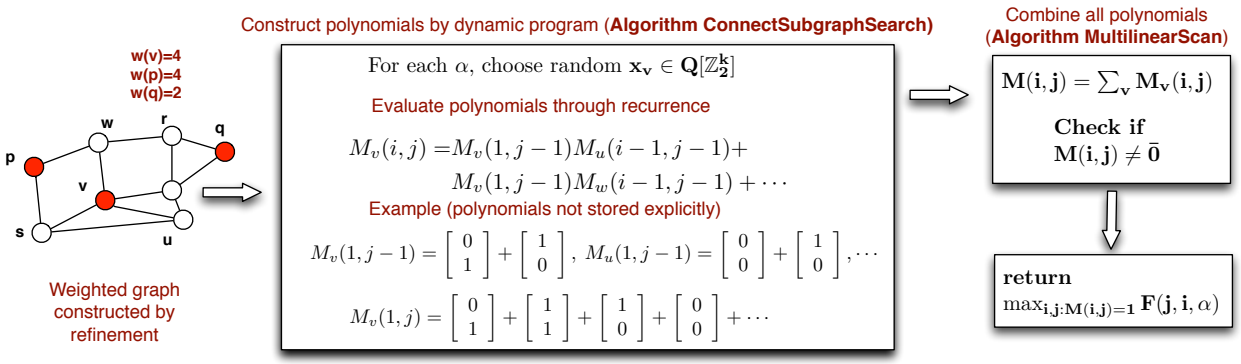


Figure 4.2: Overview of graph scan statistics using Algorithm MultilinearScan. As described in Section 3.2, our input is a weighted graph G, which is constructed as in Figure 3.4. The subroutine MaxWeight constructs and evaluates the polynomials $M_v(i, j)$ for each node v, size $i \leq k$ and weight in the interval $[(1+\epsilon)^{j-1}, (1+\epsilon)^j]$. More terms of the polynomial $M_v(3, 4)$ are shown in Figure 4.3, where node 1 corresponds to node v here. Q denotes the field $GF(2^{3+\log_2 k})$, and $\bar{0}$ denotes the additive identity of $Q[\mathbb{Z}_2^k]$.

4.3.1 Subgraphs as Monomials

We encode each subgraph of interest as a monomial. We define the sets $K = \{1, 2 \ldots, k\}$ where k is a *size parameter*, and $R = \{0, 1, 2, \ldots, W(V)\}$, where $W(V)$ is the weight of the entire node set—this is an upper bound on the weight of any subgraph. After the graph refinement procedure described in Chapter 3 and illustrated in Figure 3.4, the input graph will have a weight $w(v)$ for each node $v \in V$.

We now define a set of variables $\{x_v : v \in V\}$, and we construct a polynomial over these variables. Every term in the polynomial will represent a connected subgraph of size at most k and weight at most $W(V)$. For $i \in K$ and $j \in R$, let $M_v(i, j)$ be the polynomial corresponding to a subgraph (1) containing node v, (2) of size i, and (3) weight j.

The polynomials $M_v(i, j)$ will be constructed and evaluated recursively. As a result, there can be monomials in the polynomial representation for $M_v(i, j)$ with squared terms, e.g., the monomial $x_1^2 x_2$ in Figure 4.3—these terms correspond to the same node being considered multiple times during the recursive construction. More importantly, *we do not store all the terms of the polynomials explicitly, since there are too many of them.* Instead, we evaluate all the polynomials over random elements from the group algebra $Q[\mathbb{Z}_2^k]$, where $Q = GF(2^{3+\log_2 k})$, and we only store the result of the evaluation. $M_v(i, j)$ is non-zero if and only if there exists a subgraph with properties (1), (2), and (3) above.

Some polynomials constructed for the example in Figure 4.3.

$$
\begin{aligned}
M_1(1,0) &= M_1(1,1) = M_1(1,2) = M_1(1,3) = 0 \\
M_1(1,4) &= x_1 \\
M_1(2,4) &= M_1(1,4)M_2(1,0) + M_1(1,4)M_4(1,0) + \dots \\
&\quad + M_1(1,2)M_2(1,2) + M_1(1,2)M_4(1,2) + \dots \\
&= x_1x_2 + x_1x_4 + x_1x_5 + x_1x_6 \\
M_1(3,8) &= M_1(1,8)M_2(1,0) + M_1(1,8)M_4(1,0) + \dots \\
&\quad + M_1(1,4)M_2(1,4) + M_1(1,4)M_4(1,4) + \dots \\
&= x_1x_2x_3 + x_1x_3x_4 + x_1^2x_2 + x_1^2x_4 + \dots
\end{aligned}
$$

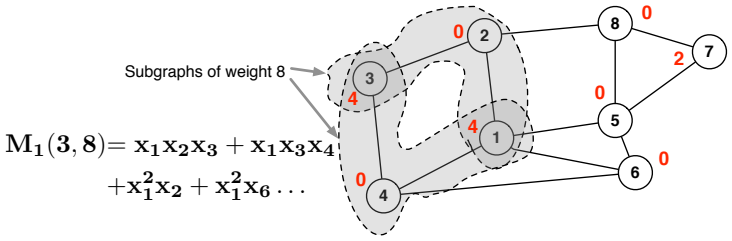


Figure 4.3: Example showing the graph $G(V, E)$ constructed in Figure 4.2, with $|V| = 8$, and the node weights shown in red next to the node ID. $M_1(3, 8)$ is the polynomial consisting of node 1 and two other nodes, with weight 8. Two of the terms in the polynomial, $x_1x_2x_3$ and $x_1x_3x_4$, are multilinear, but there are other terms, such as $x_1^2x_2$ and $x_1^2x_6$ that arise because they also satisfy the weight constraint. These will be canceled out when the polynomial is evaluated.

4.3.2 Overview of Algorithm MultilinearScan

We give the intuition behind the algorithm in Figure 4.2:

- The preprocessing step converts a sequence of graphs with varying attributes to an input graph with event counts, as defined in Section 3.2. We start by performing the

graph refinement from Chapter 3, which transforms graph G into a weighted graph with a weight $w(v)$ for each node v.

- The subroutine MaxWeight (Algorithm 4) constructs and evaluates the polynomial $M_v(i,j)$ for each node v, size $i \in K$ and weight $j \in R$. These polynomials are constructed recursively through a dynamic program. Only the results of their evaluation are stored, and not the complete representation of each $M_v(i,j)$. Finally, $M(i,j) = \sum_v M_v(i,j)$ is returned to Algorithm MultilinearScan, for all i, j.
- The algorithm MultilinearScan evaluates $F(j,i,\alpha)$ for each i, j such that $M(i,j) \neq \bar{0}$, and for each α, and returns the best solution.
- *Main idea underlying* MultilinearScan: A consequence of the structure of the group algebra [136, 261] is that any monomial containing squared terms, e.g., $x_1^2 x_2$ evaluates to $\bar{0}$. Therefore, each polynomial $M_v(i,j)$ can only get non-zero contributions from multilinear terms. *Further, with positive probability, the multilinear terms do not get canceled out.* Therefore, if there is a connected subgraph of size i and weight j, $M(i,j) \neq \bar{0}$ with positive probability.

4.3.3 MultilinearScan: An optimal, but slow solution

Algorithm 4 MaxWeight$(G(V,E), \mathbf{w}, k)$.

1: **Input**: Instance $(G(V,E), \mathbf{w})$ and parameter k
2: **Output**: Polynomial M, such that $M(i,j)$ is non-zero if G has a subgraph S with size $i \leq k$ and weight $j \leq W(V)$
3: For each node v, pick a random vector $x_v \in Q[\mathbb{Z}_2^k]$
4: $M_v(i,j) = \bar{0}$ for $i \in K$, $j \in R$
5: **for** $v \in V$ **do**
6: $\quad M_v(1, w(v)) = x_v$
7: **for** $v \in V$, $i = 2$ to k, $j = 0$ to $W(V)$ **do**
8: $\quad M_v(i,j) = \sum_{u \in Nbr(v)} \sum_{i'=1}^{i-1} \sum_{j'=0}^{j} (M_v(i',j') \cdot M_u(i-i', j-j'))$
9: $M(i,j) = \sum_v M_v(i,j)$ for $i \in K$, $j \in R$
10: **return** M

In Algorithm 5, we present MultilinearScan for non-parametric scan statistic optimization. In the **for** loop in lines 4–6, we generate a weighted graph using the refinement procedure of [47], which depends on α, and then we invoke MaxWeight to obtain the polynomial M^α. The algorithm returns the best score over subgraphs with size at most k, weight at most $W(V)$, and over all possible values of $\alpha \leq \alpha_{\max}$, the significance level. Let $OPT(F,k) = \max_{S:i \leq k} F(W(S), i, \theta)$ be the optimal solution over connected subgraphs of G' with size $i \leq k$. Then, our algorithm returns $OPT(F,k)$ with constant probability.

Algorithm 5 MULTILINEARSCAN$((G(V,E),\mathbf{p},\alpha_{max}),k)$.

1: **Input**: Instance $(G(V,E),\mathbf{p},\alpha_{max})$, parameter k
2: **Output**: Score $OPT(F,k)$
3: Let A be the set of p-values of nodes in V below α_{max}
4: **for** $\alpha \in A$ **do**
5: Run refinement step to create a weighted graph $G'(V',E')$ on "super nodes". Let $w(v')$ denote the weight of node v'.
6: $M^{\alpha} =$ MAXWEIGHT$(G'(V',E'),\mathbf{w},k)$
7: **return** $\max_{\alpha\in A}\max_{i,j:M^{\alpha}(i,j)\neq\bar{0}} F(j,i,\alpha)$

As summarized in the following theorem, MULTILINEARSCAN finds a connected subgraph of maximal score $F(\cdot)$ over all subgraphs of size at most k, with high probability. However, MULTILINEARSCAN does not scale well, as it has quadratic running time on the maximum weight $W(V)$. For the BJ statistic, $W(V) \leq n$, so the running time is quadratic on the number of nodes. In the next section, we show how to reduce the complexity by grouping the possible weights into a small number of bins.

Theorem 9 *Let $F(\cdot)$ be a non-parametric scan statistic, as defined in Section 3.2. Algorithm* MULTILINEARSCAN *returns $OPT(F,k)$ defined above with probability at least $1/5$, in time $O(2^k|A|mk^2W(V)^2)$, and using space $O(knW(V)))$, where A and is defined in line 3 of Algorithm 5.*

Proof: The proof of correctness of the algorithm involves three parts.

(1) First, we have to argue that the dynamic program in MAXWEIGHT correctly constructs the polynomial $M_v(i,j)$, which we prove by induction on i. For $i = 1$, we can verify that $M(1,j)$ is correctly computed for all j in line 4 of MAXWEIGHT. Then, for $i \geq 2$, when we compute $M_v(i,j)$ (line 8), we have all the information we need from the inductive step.

(2) Then, we show that $M(i,j)$ is returned by MAXWEIGHT with probability at least $1/5$ if a subgraph of size i and weight j does exist. This follows from Theorem 8.

(3) And, lastly, we argue that our algorithm returns $OPT(F,K)$, the optimal solution. This follows because the algorithm tries every combination of i, j, and α, for which $M^{\alpha} \neq \bar{0}$—i.e., a subgraph of size i and weight j exists.

Now, we analyze the time and space complexity. The space complexity follows from the size of the dynamic programming table, which is $k \times W(V) \times n = O(knW(V))$. For the running time bound, notice that calculating $M_v(i,j)$ for some i and j requires summing over all $i' < i$ and all $j' \leq j$; thus the computation is quadratic in k and $W(V)$. Furthermore, we sum over all neighbors of node v. Therefore, the total running time of MAXWEIGHT is $O(mk^2 2^k W(V)^2)$, where the 2^k term is the time complexity of operations in the $Q[\mathbb{Z}_2^k]$ group algebra. Finally, MAXWEIGHT is invoked $|A|$ times by MULTILINEARSCAN, so the running time follows. ■

Remark: the success probability of MULTILINEARSCAN in Theorem 9 can be improved to $1-\epsilon$ for any $\epsilon \in (0,1)$ by simply running the algorithm $O(\log 1/\epsilon)$ times; we omit this step here to keep the discussion simple.

4.3.4 Scaling MultilinearScan with logarithmic binning

As discussed above, the quadratic dependence on $W(V)$ creates a bottleneck in MAXWEIGHT. To keep the computation scalable, we will group the weights of the input graph to MAXWEIGHT by considering powers of $(1+\epsilon)$, where $\epsilon > 0$ is an error parameter: if $w(v) \in [(1+\epsilon)^{j-1}, (1+\epsilon)^j)$, we say that v is in the weight group j; this definition is extended to the weight of a subgraph. For instance, if $\epsilon = 1$, nodes with weight in $[8, 16)$ are in group 4. The goal is to scale the weights of the input graph down by a logarithmic factor and run Algorithm 4 on this much smaller set of weights.

With a slight abuse of notation, let us redefine R as $\{0, 1, 2, \ldots, r\}$, where r is a *weight parameter*. Now the polynomial $M_v(i, j)$ corresponds to a subgraph (1) containing node v, (2) of size i, and (3) total weight in $[(1+\epsilon)^{j-1}, (1+\epsilon)^j)$. $M_v(i, 0)$ represents subgraphs of weight 0.

We need to modify MAXWEIGHT to satisfy condition (3). Algorithm 6 shows this modified version. In the base case (lines 4–6), we set $M_v(1, j)$ to be x_v if node v is in group j and 0 otherwise. For $i \geq 2$ (lines 7–15), we consider the following cases:
Case 1. For $j = 0$, a polynomial $M_v(i, 0)$ represents connected subgraphs of weight 0 only. For a graph of size i to have weight 0, its two parts, of size i' and size $i - i'$ must both have weight 0 as well.
Case 2. For $j = 1$, a polynomial $M_v(i, 1)$ represents connected subgraphs of weight 1 only. For a graph of size i to have weight 1, one of its two parts must have weight 1, and the other must have weight 0.
Case 3. For $j \geq 2$, a polynomial $M_v(i, j)$ represents connected subgraphs of weight between $(1+\epsilon)^{j-1}$ and $(1+\epsilon)^j$. We could obtain a subgraph of size i with this weight in one of two ways. The first is to combine two subgraphs, each of weights between $(1+\epsilon)^{j-2}$ and $(1+\epsilon)^{j-1}$. Thus, the resulting subgraph will have weight at least $(1+\epsilon)^{j-1}$ and at most $(1+\epsilon)^j$ (exclusive). The second way is to combine a subgraph with weight between $(1+\epsilon)^{j-1}$ and $(1+\epsilon)^j$ with one of weight 0, as in the case for $j = 1$.

Now, we present APPROX-MULTILINEARSCAN in Algorithm 7. This algorithm takes an extra parameter, ϵ, and calls APPROX-CONNECTEDSUBGRAPHSEARCH with weight parameter $r = \lceil \log_{(1+\epsilon)}(kw_{\max} + 1) \rceil$, for $w_{\max}$ defined in line 6. That way, r is the maximum (scaled down) weight of a subgraph of size k. Let $W_{1+\epsilon}(S) = (1+\epsilon)^{\lceil \log_{1+\epsilon}(W(S)+1) \rceil}$ be the approximate weight of S, and let $OPT(F, k, \epsilon) = \max_{S: i \leq k} F(W_{1+\epsilon}(S), i, \theta)$ be the optimal solution over connected subgraphs of G' with size $i \leq k$ and rounded weights. Then, we obtain the following result.

Algorithm 6 APPROX-CONNECTEDSUBGRAPHSEARCH$(G, \mathbf{w}, k, r)$

Input: Instance $(G(V, E), \mathbf{w})$ and parameters k, r
Output: Polynomial M, such that $M(i, j)$ is non-zero if G has a subgraph S with size $i \leq k$ and weight group $j \leq r$
For each node v, pick a random vector $x_v \in Q[\mathbb{Z}_2^k]$
for $v \in V$ **do**
  $j = \lceil \log_{(1+\epsilon)}(w(v) + 1) \rceil$
  $M_v(1, j) = x_v$
for $v \in V$, $i = 2$ to k, $j = 0$ to r **do**
  if $j = 0$ **then**
    $M_v(i, 0) = \sum_{u \in Nbr(v)} \sum_{i'=1}^{i} (M_v(i', 0) \cdot M_u(i - i', 0))$
  if $j = 1$ **then**
    $M_v(i, 1) = \sum_{u \in Nbr(v)} \sum_{i'=1}^{i} (M_v(i', 1) \cdot M_u(i - i', 0) +$
      $(M_v(i', 0) \cdot M_u(i - i', 1))$
  if $j \geq 2$ **then**
    $M_v(i, j) = \sum_{u \in Nbr(v)} \sum_{i'=1}^{i} (M_v(i', j - 1) \cdot M_u(i - i', j - 1) +$
      $M_v(i', j) \cdot M_u(i - i', 0) + M_v(i', 0) \cdot M_u(i - i', j))$
$M(i, j) = \sum_v M_v(i, j)$ for $i \in K$, $j \in R$
return M

Algorithm 7 APPROX-MULTILINEARSCAN$((G, \mathbf{p}, \alpha_{max}), k, \epsilon)$.

Input: Instance $(G(V, E), \mathbf{p}, \alpha_{max})$, parameters k, ϵ
Output: Score $OPT(F, k, \epsilon)$
Let A be the set of p-values of nodes in V below α_{max}
for $\alpha \in A$ **do**
  Run refinement step to create a weighted graph $G'(V', E')$ on "super nodes". Let $w(v')$ denote the weight of node v'
  Let $w_{\max} = \max_{v'} w(v')$ be the maximum weight
  Let $r = \lceil \log_{(1+\epsilon)}(k w_{\max} + 1) \rceil$
  M^α = APPROX-CONNECTEDSUBGRAPHSEARCH$(G', \mathbf{w}, k, r)$
return $\max_{\alpha \in A} \max_{i,j:M^\alpha(i,j) \neq \bar{0}} F((1 + \epsilon)^j, i, \alpha)$

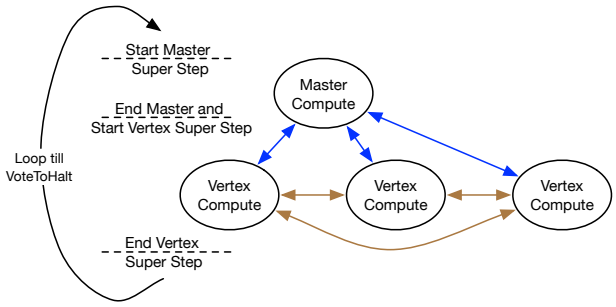


Figure 4.4: Pregel computation model as implemented in Giraph with master and vertex computations.

Theorem 10 *Let $F(\cdot)$ be a non-parametric scan statistic, as defined in Section 3.2. Algorithm* MultilinearScan *returns $OPT(F, k, \epsilon)$ with probability at least $1/5$, in time $O(2^k|A|mk^2 \log_{1+\epsilon}(kw_{\max}))$, and using space $O(kn \log_{1+\epsilon}(kw_{\max}))$, where A and $w_{\max}$ are defined in lines 3 and 6 of Algorithm 7.*

4.3.5 MultilinearScan in Pregel

Apache Giraph and GraphX implement the Pregel [96] model depicted in Figure 4.4. The brown arrows indicate neighbor communication. Master Compute is a special form of vertex computation in Giraph that happens before vertex computations in each super step. Vertices can communicate with the master vertex computation through collective operations such as aggregations, reductions, and broadcasts (blue arrows). Communication between any pair of vertices or between the master and any vertex takes effect only during the next super step. The computation stops when no vertex receives messages from its neighbors, and when they all agree to halt.

We describe ParConnectedSubgraphSearch, a parallel version of Approx-ConnectedSubgraphSearch in Giraph in Algorithm 8. As in the case of the sequential version, this is described without the more efficient matrix representation—this requires a for loop with 2^k steps around the Pregel call. We have also explored a GraphX [96] version of the algorithm, which has a slightly different structure but performs poorly.

Algorithm 8 exploits two levels of parallelism identifiable in the sequential implementation: (i) the outer **for** loop, which corresponds to the matrix representation (not shown here) can be easily parallelized; (ii) the **for** $v \in V$ loop in line 7 of Approx-ConnectedSubgraphSearch happens in parallel. Figure 4.5 elaborates the steps and vertex compute functions of Algorithm 8 for a sample graph of 4 nodes. A mapping of the elements in lines 7 to 15 of the sequential Approx-ConnectedSubgraphSearch algorithm to the Pregel model shown here is as follows. Each super step represents a step of the loop **for** $i = 2$ to k; each vertex within a super step represents the loop **for** $v \in V$;

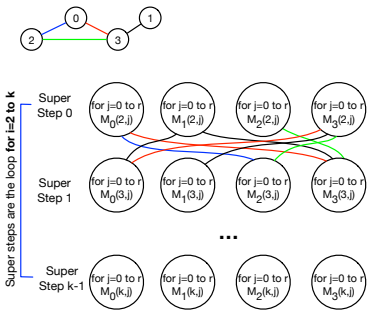


Figure 4.5: Pregel computation of ParConnectedSubgraphSearch for a sample graph of 4 nodes.

and the loop **for** $j = 0$ to r is carried out inside each vertex within a super step. The summation in line 16 of the Approx-ConnectedSubgraphSearch is carried out through an aggregate operation that collects results to the master vertex at the end of last super step. Note, in the Pregel implementation, a row of vertex computations within a super step occur in parallel, while the super steps follow one after the other.

Lemma 14 *The Giraph implementation using Algorithm 8 runs in* $O(2^k|A|k\log_{1+\epsilon}(kw_{\max}))$ *super steps and gives a solution with the same guarantees as in Theorem 10.*

4.4 Experiments

Our experiments address the following questions.
1. Scalability of Approx-MultilinearScan. How does our algorithm scale to networks with $\geq 10^5$ nodes compared with prior methods? How does it scale with the parameter k compared to the color-coding based algorithms from Chapter 3? (Section 4.4.2)
2. Approximation Error and Solution Quality. What is the effect of the error parameter ϵ on the solutions found by our algorithm? How close or far is it from optimal in practice, in terms of the objective value? (Section 4.4.3)
3. Parallel Evaluation. How does our Pregel algorithm scale with k for real graphs and what are the associated overheads? (Section 4.4.4)
4. Illustrative application. What are some applications of our proposed methods? Can we discover interesting clusters in real data? (Section 4.4.5)

4.4.1 Experimental Setup

Datasets. We evaluate our algorithms in datasets from various domains, such as social networks, citation networks, and road networks. For each dataset, we have snapshots taken

Algorithm 8 ParConnectedSubgraphSearch$(G(V,E), \mathbf{w}, k, r)$.

Input: Instance $(G(V,E), \mathbf{w})$ and parameters k, r
Output: Polynomial $M(i,j)$, such that $M(i,j)$ is non-zero if G has a subgraph of size $i \leq k$ and weight group $j \leq r$

For each node v, pick a random vector $x_v \in \mathbb{Z}_2^k$
return $M =$ **pregel**$(G(V,E), \mathbf{w}, \mathbf{x}, k, r,$ MCompute, VCompute$)$

Procedure MCompute(s,k)
  Input: Super step number s and parameter k
  if $s = k$ **then**
    $M =$ **GetAggregatedMessages**()
    for $v \in V$ **do**
      $M(i,j) = \sum_v M_v(i,j)$ for $i \in K$, $j \in R$
      Halt()
    return M

Procedure VCompute$(\mathcal{M}, s, w(v), x_v, k, r)$
  Input: Set of messages $\mathcal{M}$ from set $Nbr(v)$ of neighbors of node v, super step number s, node variables $w(v)$ and x_v, and parameters k, r.
  if $s = 0$ **then**
    $M_v(1,j) = 0$, for $j \leq r$
    $M_v(1, \lceil \log_{(1+\epsilon)}(w(v)+1) \rceil) = x_v$
  else
    Compute $M_v(i,j)$ using the recurrence as in Approx-ConnectedSubgraphSearch
  if $s < k-1$ **then**
    for $u \in Nbr(v)$ **do Send**(M_v, u)
  else
    Aggregate(M_v)
    VoteToHalt()

at different points in time, each with different activity in the nodes. We also use random networks with with planted anomalies for the scalability experiments. In Table 3.3, we show a summary of the networks used in this section.

Baseline methods. We compare our proposed methods to Algorithm ColCodeNP from Chapter 3 in addition to all the baseline methods in that chapter. We focus on the Berk-Jones (BJ) statistic [36] with $\alpha_{\max} = 0.15$.

Hardware. We test our parallel algorithms on two clusters: (1) An Intel Haswell HPC cluster, consisting of Intel(R) Xeon(R) E5-2698 CPU. Each compute node has two sockets totaling 36 cores, with 128GB memory per node, and a high-speed Infiniband interconnect; (2) an HPC cluster with Intel(R) Xeon(R) E5-2670 CPU, 24 cores per node over two sockets and similar memory and interconnect.

4.4.2 Scalability of Approx-MultilinearScan

In Figure 4.1 (top), we show the running time of our algorithms and the baseline methods as a function of the size of the graph. Approx-MultilinearScan is the only method besides ColCodeNP able to process the instance of 10^6 nodes. All the other algorithms run out of memory or do not finish running even after 24 hours.

We note, however, that the running time of Approx-MultilinearScan and ColCodeNP is quite high on the graph with one million nodes—around 15 hours. In contrast, our parallel algorithm, MultilinearScanGiraph, processes the same graph within 8 minutes, over 100x speedup. We also compare MultilinearScanGiraph to the distributed algorithm of Zhao et al. [269] for anomalous subgraph detection. This is, to the best of our knowledge, the only distributed algorithm for scan statistic optimization. The authors report an average running time of 1,743 seconds on the BWSN dataset using 4 parallel tasks, whereas MultilinearScanGiraph processes the same dataset in 1,234 seconds, a 1.4x speedup.

4.4.2.1 Scalability with k

We compare the time and space scalability of Approx-MultilinearScan to ColCodeNP in terms of the size parameter k. In Figure 4.6 a–b, we show the ratio of ColCodeNP to Approx-MultilinearScan with respect to time (a) and memory usage (b). Higher is better, and points above the dashed black line (ratio of 1) indicate that our proposed method improves over MultilinearScan, which is the case for $k \geq 6$. Our algorithm is up to 1,000 times faster for a solution size of 12 and uses as little as a tenth of memory compared to ColCodeNP. We also note that ColCodeNP did not complete running for the higher values of k after 24 hours. The dashed portion of the lines in the plot are an extrapolated estimate of the time to run this algorithm. On the other hand, Approx-MultilinearScan is able to run for all the values of the parameter k. The total running time (c) and memory

Table 4.1: Datasets used in our experiments

Dataset	**Description**	**Nodes**	**Edges**	**Snapshots**
CitHepPh	Citation network	11,895	76,284	4
NEast	Network of counties in Northeastern USA	245	683	10,000
Traffic	Traffic Network of Los Angeles Country, CA	1,870	1,993	1,488
Twitter	Follower network collected through Twitter API	2,645	17,108	182
BWSN	Battle of the Water Sensors	12,527	14,831	22
Random	Erdos-Renyi graphs with 100 to 1,000,000 nodes	100 to 10^6	$\sim$ 100 to $\sim 10^6$	5
soc-Live-Journal1-refined	The largest connected subgraph of soc-LiverJournal data from SNAP	4,843,864	42,843,302	1
as-Skitter-refined	The largest connected subgraph of as-Skitter data from SNAP	1,694,538	11,093,792	1
Census block group network for MN	Nodes consist of census block groups, and two blocks are connected by an edge if they share a border. Immunization records of 7th-grade students for 870 public and private schools in the academic year 2015-16 are used to infer event counts for the block group nodes	4084	12660	1

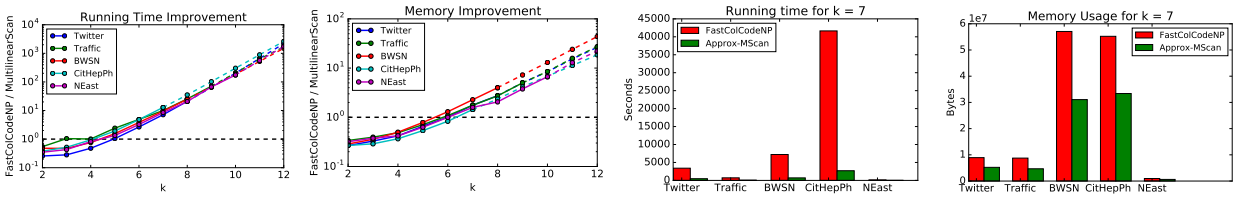


Figure 4.6: Running time (a) and memory improvement (b) when using Approx-MultilinearScan over ColCodeNP. For $k \geq 6$, our algorithm is orders of magnitude faster than MultilinearScan; thus, we can discover larger anomalous subgraphs in a fraction of the time. Memory usage also improves by up to an order of magnitude. We also show total running time (c) and memory usage (d) for $k = 7$.

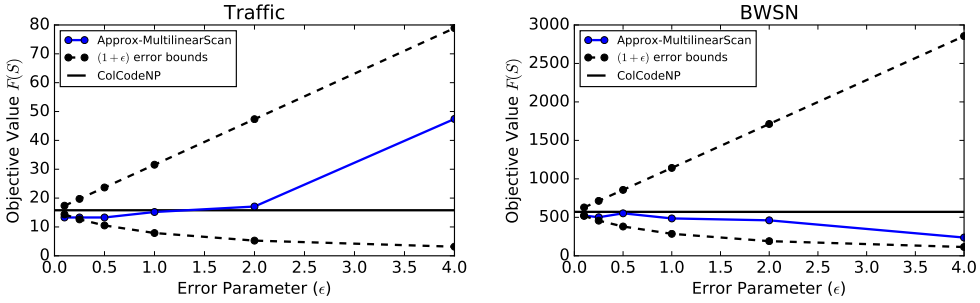


Figure 4.7: Effect of the error parameter (ϵ) on the quality of the solution discovered. MultilinearScan obtains close-to-optimal solutions (i.e., close to the horizontal line) despite the worst-case error bound (black dashed lines).

usage (d) for $k = 7$ are also shown.

4.4.3 Approximation Error and Solution Quality

We now study the effect of the ϵ parameter, which controls the approximation guarantee of Approx-MultilinearScan. Figure 4.7 shows the objective score for different values of ϵ, for $k = 5$. We compare this score to that obtained by ColCodeNP, which is optimal with high probability—this value would have been obtained using MultilinearScan. Note that the solutions discovered by our algorithm are much better than the worst case approximation bound from Theorem 10, and they are usually close to the optimal solution. Even for $\epsilon = 2$ (i.e, approximation guarantee of 3), Approx-MultilinearScan yields a good estimate while keeping the running time low. We show results for two datasets, but we have observed similar trends in the remaining ones.

Detection Power. We also evaluate Approx-MultilinearScan in terms of event detection power in the BWSN dataset. Table 4.2 reports the average performance of ColCodeNP and Approx-MultilinearScan on all the snapshots of the dataset. We improve on the precision, recall, and F1 score of the color-coding based method. We also note that

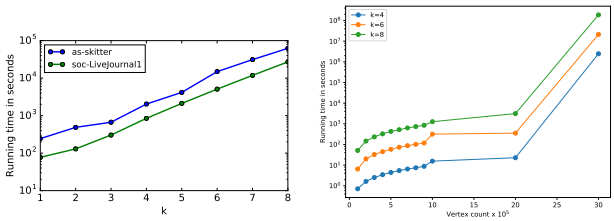


Figure 4.8: Performance variation with (a) parameter k for soc-LiverJournal1 and as-Skitter graphs from SNAP; (b) number of nodes in Erdos-Renyi graphs.

Approx-MultilinearScan has similar performance to ColCodeNP in terms of the objective score (i.e., BJ statistic) *while being almost 3 times faster*.

Table 4.2: Detection performance of ColCodeNP and Approx-MultilinearScan in the BWSN dataset.

	ColCodeNP	**Approx-MScan**
Precision	0.977	**0.985**
Recall	0.955	**0.957**
F1 Score	0.952	**0.961**
Accuracy	**0.973**	0.945
BJ Score	**602.164**	601.938
Running Time	987	**366**

4.4.4 Parallel algorithm evaluation

Figure 4.1 shows the parallel algorithm scaling to graphs of size up to 10^6 using just 24 parallel tasks. Figure 4.8(a) shows parallel runtime for two of the largest graphs found in SNAP. It shows the runtime grows exponentially with k as expected, yet is able to perform computations in a reasonable amount of time. Figure 4.8(b) shows the variation of the running time with the number of nodes for random networks, which is much more gradual.

Bottlenecks in parallel performance. The communication cost in Giraph and GraphX is the main bottleneck in our algorithm, which has a very different computing pattern from algorithms for problems such as PageRank—these stop sending messages to specific nodes as the algorithm proceeds, when certain criteria are met, leading to reduced communication over time. In contrast, in our algorithm, in each super step, vertices always send the same amount of data to neighbors, which contributes to significant overheads in Giraph and GraphX. In fact, initial experiments showed that GraphX has diminishing returns with increasing parallelism, because of its dataflow model. Further, our algorithm exchanges matrices instead of single primitive values as in common Pregel algorithms. Giraph uses an in-place peer-to-peer communication similar to MPI, which performed better than GraphX. Still, with large graphs such as the Skitter data, we observed over 80% overhead with 256 and 512 parallel tasks over 16 machines.

4.4.5 Case study: finding undervaccinated clusters

We apply our algorithm to discover undervaccinated clusters in Minnesota, which have become a concern for public health officials in recent years. We use school immunization data from the Minnesota Department of Health[1]. This dataset contains immunization records of 7th-grade students for 870 public and private schools during the academic year 2015–16. For each school, the dataset provides the number of students enrolled in the school and the percentage of students who are vaccinated for various diseases. We focus on the Measles-Mumps-Rubella (MMR) vaccine, which has a 97.5% average vaccination rate statewide in Minnesota.

Our goal is to discover geographical clusters where the vaccination rate for MMR is much lower than the statewide average. We pose this problem as scan statistics optimization in the network of census block groups of Minnesota. We aggregate the school data at the block group level, and we construct a planar graph where nodes are block groups, and there is an edge between every pair of adjacent block groups. After suitably defining p-values for the nodes, we find a subgraph that maximizes the BJ scan statistic.

For a block group s, let $w(s)$ be the total number of **unvaccinated** children in the schools within s, and let $b(s)$ be the total enrollment. We model $w(s)$ as a sample from a Poisson distribution with parameter proportional to the number of enrolled children: $w(s) \sim Poisson(\lambda b(s))$, where $\lambda = 0.025$ is the statewide average unvaccination rate. Under this model, we derive a p-value for each block: $p(s) = 1 - \Pr(X \leq w(s) | \lambda b(s))$. We use ApproxMultilinearScan to find a subgraph with high anomaly score according to the BJ scan statistic.

Figure 4.1 shows the block group network and a connected subgraph discovered by our method. This cluster is to the south of University of Minnesota and downtown Minneapolis, and the unvaccination rate in the cluster is 6%, 2.5 times higher than expected. All prior work on finding undervaccinated clusters has only considered well rounded shapes, e.g., [166], which would not find such a cluster.

4.5 Conclusions

We present a novel approach for graph scan statistics based on algebraic techniques for multilinear detection, which gives rigorously provable tradeoffs between the running time and approximation guarantee. Our method leads to significant improvement in terms of both time and space over the state-of-the-art methods. Our algorithm can be used without any modification for a broad class of parametric and non-parametric functions, and can support a variety of constraints, in addition to connectivity.

[1] //www.health.state.mn.us/divs/idepc/immunize/stats/school/

Chapter 5

Graph Scan Statistics with Uncertainty

5.1 Introduction

Most existing work in anomaly detection in networks implicitly assumes that the reported data is either correct or has only a negligible amount of noise, which is often not a realistic assumption. The observed counts in data have uncertainty and do not exactly match the real world due to reporting errors, geocoding errors, missing entries, etc. All these sources of uncertainty would affect the problem formulations and algorithms for anomaly detection, but they are especially relevant when using scan statistics because the anomaly score is formalized in terms of the log likelihood of occurrence of observed data. One of the very few results on the impact of uncertainty in scan statistics is by [169], who observes that uncertainty exists and that it can affect the quality of the clusters discovered by scan statistics, but this study does not propose any methods for taking uncertainty into account. To the best of our knowledge, the only scan-statistics-based method that incorporates uncertainty is the Bayesian scan statistic proposed in [187] and later extended to the multivariate case [186]—though the authors do not explicitly motivate their work as a solution to uncertainty. Both papers assume simple conjugate priors and thus are able to derive closed-form expressions. However, in general, more complex distributional assumptions for uncertainty would lead to expressions that do not have a closed form and can only be approximately optimized via sampling. In this chapter, we use methods from the theory of stochastic optimization to formally characterize scan statistic maximization with uncertainty and develop novel algorithms and heuristics for this problem. Our contributions are summarized below.

1. **Scan Statistics with Uncertainty.** We propose two approaches for taking uncertainty into account: a sample average approximation (SAA) and a max-min formulation [37, 202, 222], and show that these are well motivated for scan statistics by ana-

lyzing instances with stochastic perturbations. When we account for uncertainty, the anomaly detection task becomes much more challenging. We show that, even without any connectivity constraints, finding clusters based on scan statistics that optimize the max-min objective function is NP-hard, whereas it can be done in linear time if there is no uncertainty.

2. **Rigorous Algorithms and Theoretical Bounds.** We develop rigorous algorithms and heuristics for optimizing the scan statistics in both these formulations. For the SAA formulation, our algorithm AggregateSAA gives rigorous bounds on the approximation guarantee for finding solutions of a given "effective" size k, with or without connectivity constraints, and is a fixed parameter tractable algorithm. For the max-min formulation, we present two algorithms: (1) MaxMinLpRound for the case of no connectivity constraints, using linear programming rounding, which yields rigorous approximation bounds, and (2) a heuristic, BestMax, for the case with connectivity. Both AggregateSAA and BestMax use algorithms for scan statistics that we proposed in Chapter 3.

3. **Numerical Evaluation of the Proposed Methods.** We evaluate our proposed methods in two popular benchmarks for scan statistics with known events. Both the SAA and max-min approaches lead to fairly robust scan statistics, and our algorithms have a clear improvement in performance, compared to a natural baseline, over a wide range of signal-to-noise ratio regimes. For regimes with high noise in the data, we see up to two-fold improvement in terms F1 score. Similarly, the objective score improves in all experimental settings, especially when the signal-to-noise ratio is low, where we see gains over the baseline by more than a factor of 2.

5.2 Preliminaries

It is known that scan statistics have been mostly applied to spatial data. Here we consider the more general version of scan statistic on networks, which has attracted broad attention recently [101, 144, 156, 173, 185, 231]. Let us define an undirected graph $G = (V, E)$ with $n = |V|$ nodes and $m = |E|$ edges. For each node $v \in V$, we have an *event* count $c(v)$ and a *baseline* count $b(v)$.

Using scan statistics, the detection of anomalous clusters can be posed as a hypothesis testing problem. The null hypothesis H_0 is that there is no anomalous cluster. That is, the event counts for all nodes are generated independently from the same distribution—proportionally to the baseline counts. Under the alternative hypothesis $H_1(S)$, there exists a small connected subset of nodes S, such that event counts in S are generated at a higher rate than counts elsewhere (i.e., in $V \setminus S$). A scan statistic is an *anomalousness* function $F : S \to \mathbb{R}$ that evaluates how much a subset S deviates from the null hypothesis. Here, we

focus on a commonly used scan statistic based on the likelihood ratio test as

$$F(S) = \frac{\Pr(H_1(S)|\text{Data})}{\Pr(H_0|\text{Data})} = \frac{\Pr(H_1(S)|C(S), B(S), C(V \setminus S), B(V \setminus S))}{\Pr(H_0|C(V), B(V))},$$

where $C(S) = \sum_{v \in S} c(v)$ is the total count of S, and $B(S) = \sum_{v \in S} b(v)$ is the baseline count.

Depending on the type of data (i.e., counts, positive real values, etc.) and the statistical assumptions about the process generating the event counts (i.e., Poisson, Normal, etc.), the scan statistics can be broadly summarized in two categories:

Parametric scan statistics. It assumes that observations follow a parameterized distribution, usually from the exponential family, such as Poisson or Normal. For example, the Kulldorff scan statistic for count data, commonly used in disease surveillance [72, 144, 145, 185], is defined as

$$F(S) = C(S) \log\left(\frac{C(S)}{B(S)}\right) + (C(V) - C(S)) \log\left(\frac{C(V) - C(S)}{B(V) - B(S)}\right) - C(V) \log\left(\frac{C(V)}{B(V)}\right)$$

if $C(S)/B(S) > C(V)/B(V)$ and 0 otherwise.

Nonparametric scan statistics. The observations are not assumed from a specified distribution. The main idea is to compute the empirical distribution of the counts based on multiple snapshots of the graph. First, one can define a p-value $p(v)$ for each vertex v by comparing the current values of $c(v)$ and $b(v)$ to past observations. Then, a hypothesis test can be performed to check whether the empirical p-values are uniformly distributed on $[0, 1]$ [185, 205, 223]. For example, in the Berk-Jones scan statistic (BJ) [36], a node is declared to be *significant* if $p(v) < \alpha$ for a given significance level α. The *weight* of a node with respect to α is $w(v, \alpha)$ is 1 if v is significant and 0 otherwise. We use $w(v)$ when α is clear from context. Then, the weight of a set S is $W(S) = \sum_{v \in S} w(v)$. The baseline count is $b(v) = 1$ for all nodes, so $B(S) = |S|$. The BJ scan statistic is defined as

$$F(S) = \max_{\alpha \leq \alpha_{max}} |S| \left[\frac{W(S)}{|S|} \log\left(\frac{W(S)/|S|}{\alpha}\right) + \left(1 - \frac{W(S)}{|S|}\right) \log\left(\frac{1 - W(S)/|S|}{1 - \alpha}\right)\right]$$

if $W(S)/|S| > \alpha$ and 0 otherwise.

Thus, the anomaly detection problem in the network can be posed as a constrained optimization problem. Given a graph $G = (V, E)$, a scan statistic $F(\cdot)$, and the associated counts for vertices, C and B, the objective is to find a connected subset $S \subseteq V$, such that $F(S) = F(C(S), B(S))$ is maximized.

Example. Consider an instance of lung cancer cases in a population within a state. We consider a graph $G = (V, E)$ with the set V representing the counties in the state, and E consisting of edges between counties sharing a boundary. The baseline count $b(v)$ is the number of inhabitants in county v, and $c(v)$ is the number of lung cancer cases in v. A cluster of counties in which the incidence counts are significantly different from what is expected based on the population would have a high Kulldorff score.

5.3 Network Scan Statistics With Uncertainty

As shown in the above example, the observed data could contain different sources of uncertainty including unreported cases, inaccurate data collection, measurement error, etc. The observed data is just some realization of the underlying network. It is a crucial question on how to account such uncertainty in the scan statistic for anomaly detection. Let $c(v)$ be the (unobserved) real count for a node v and $x(v)$ be the (observed) estimate count. Then, we can characterize our uncertainty about $c(v)$ by making $x(v)$ a sample from a distribution θ parameterized by $c(v) : x(v)|c(v) \sim \theta(c(v))$. More generally, let $X = (x(v_1), \ldots, x(v_n))$ and $C = (c(v_1), \ldots, c(v_n))$ be the vectors of observed and real counts, respectively. Then, $X|C \sim \theta(C)$. In most of the existing work, the observed counts are taken as the ground truth; that is, $X = C$ deterministically.

5.3.1 Problem Formulations

We consider two approaches to account for uncertainty. The first method is based on the sample average approximation (SAA) method [222]. The second is based on a max-min formulation [37], which is referred to as the worst-case scenario in the stochastic optimization [202].

5.3.1.1 Scan Statistics Under Sample Average Approximation.

The key idea is taking the expectation of $F(S)$ to account for the uncertainty in scan statistics. That is, given a graph $G = (V, E)$, a scan statistic $F(\cdot)$, and the associated counts for vertices-X and B, the objective is to find a connected subset $S \subseteq V$ that maximizes

$$\begin{aligned}\mathbb{E}[F(S)|X] &= \int_{F(S)} (F(S)|X)p(F(S)|X)dF(S) \\ &= \int_{C} (F(S)|X)\frac{p(X|C)p(C)}{p(X)}dC.\end{aligned}$$

Equivalently, we want to maximize $\int_C (F(S)|X)p(X|C)p(C)dC$, since $P(X)$ is a constant.

The idea of SAA is to generate samples from the distribution for the input and optimize over the set of samples [240]. Specifically, maximizing $\mathbb{E}[F(S)|X]$ can be approximated by maximizing the average of N samples in the following manner:

1. for $i = 1$ to N:
   (a) Sample the i^{th} vector of real counts C_i with probability proportional to $p(C_i)$
   (b) Sample the i^{th} vector of estimate counts X_i from $\theta(C_i)$

(c) Compute $F_i(S) = F(X_i(S), B(S))$ for all connected S

2. Return S^* that maximizes the sample average: $S^* = \arg\max_S \frac{1}{N} \sum_{i=1}^{N} F_i(S)$

Problem 5 *Given a graph $G = (V, E)$, a scan statistic $F(\cdot)$, and N sets of counts for vertices—X_i for $i = 1$ to N—find a connected subset $S \subseteq V$ that maximizes the average score,*

$$\frac{1}{N} \sum_{i=1}^{N} F_i(S) = \frac{1}{N} \sum_{i=1}^{N} F(X_i(S), B(S)).$$

5.3.1.2 Scan Statistics Under the Max-Min Formulation.

Rather than taking the average for accounting the uncertainty, an alternative approach of address uncertainty in scan statistics is to consider the minimum score over all the samples. This gives us the following formulation.

Problem 6 *Given a graph $G = (V, E)$, a scan statistic $F(\cdot)$, and N sets of counts for vertices—X_i for $i = 1$ to N—find a connected subset $S \subseteq V$, such that maximizes the minimum over all samples,*

$$\min_{i=1,\ldots,N} F_i(S) = \min_{i=1,\ldots,N} F(X_i(S), B(S)).$$

5.4 Motivation for the Two Formulations and Challenges Arising From Uncertainty

We use a simple stochastic perturbation model to show that the two formulations, Problem 5 and 6, are well motivated. We consider a simple model using the BJ statistic on an instance $G = (V, E)$ constructed in the following manner. Let V be partitioned into $V = V_1 \cup V_2$, with $w(v) = 1$ for $v \in V_1$, and $w(v') = 0$ for $v' \in V_2$. We also have $|V_1| = k$, so that V_1 is the optimal solution maximizing the BJ-statistic for this instance. Next, we assume a simple noise model, in which each non-anomalous node (i.e., those with weight 0) becomes anomalous with a small probability p.

Observation 1 *For the above stochastic model, both the SAA and Max-Min formulations correctly identify the optimal cluster V_1, if the number of samples $N \geq \frac{2 \ln n}{\ln 1/p}$. In contrast, the optimal solution for any sample is a subset of V_2, with high probability.*

Uncertainty makes scan statistics much harder. Neill [183] observed that many scan statistics (without any uncertainty), including the BJ-statistic can be solved in linear time if there

are no connectivity constraints, because of the "linear time subset scanning" property. As a result, the optimum solution of a given size can be found by considering the items in the order of their counts. In contrast, we show below that under uncertainty, the Max-Min formulation of Problem 6 is NP-complete. We note that maximizing scan statistic score with connectivity constraints was shown to be NP-Hard in [47].

Lemma 15 *For any non-parametric scan statistic $F(\cdot)$, finding a subset $S \subset V$ without any connectivity requirement, and of size at most k, that maximizes $\min_i F_i(S)$ is NP-complete.*

Proof: It is easy to see that the problem is in NP, since a solution can be verified efficiently. We prove the NP-hardness below. As shown in [47], the functions $F(S) = \phi(\sum_{v \in S} w(v), N(S), \alpha)$ are monotone increasing functions of $\sum_{v \in S} w(v)$, when $|S|$ is fixed. Recall the notation $w_i(v)$ defined in Section 3.2. Suppose $\min_i F_i(S) = F_{i^*}(S)$, and $\sum_{v \in S} w_{i^*}(v) = z^*$. By the monotonicity property, it follows that for all $i \neq i^*$, $\sum_{v \in S} w_i(v) \geq z^*$.

We now reduce the problem from the following generalization of setcover, referred to as a multicover: an instance consists of a set system $(U, \mathcal{S} \subseteq 2^U)$ and a parameter r. The objective is to pick the smallest number of sets $S_1, \ldots, S_j \in \mathcal{S}$, such that each element in U is covered at least r times. We construct an instance of the scan statistics problem with $V = \mathcal{S}$, i.e., with a node v corresponding to a set $S_v \in \mathcal{S}$. We have $N = |U|$ samples, with $w_i(v) = 1$ if $i \in U$ is an element of the set S_v; else $w_i(v) = 0$.

We argue below that there exists a feasible solution $S_{v_1}, \ldots, S_{v_k}$ to the multi-cover instance if and only if there exists a subset $V' = \{v_1, \ldots, v_k\}$ with score function $\min_i F_i(V') = \phi(r, k, \alpha)$. Suppose there exists a feasible solution $S_{v_1}, \ldots, S_{v_k}$ which ensures each element $i \in U$ is covered r times. Then, the subset $V' = \{v_1, \ldots, v_k\}$ ensures that for each $i = 1, \ldots, N$, $\sum_{v \in V'} w_i(v) \geq r$, which implies $F_i(V') \geq \phi(r, k, \alpha)$. Conversely, suppose there exists a solution V' of size k with max-min objective at least $\phi(r, k, \alpha)$. Then, from the above argument, we have $\sum_{v \in V'} w_i(v) \geq r$ for each i. ■

5.5 Proposed Methods

We describe our algorithms for the SAA and Max-Min objectives. We focus on non-parametric functions here, though our methods extend to parametric functions in a natural manner. The approximation bounds depend on the specific functions, and here we derive those for the BJ-statistic (see Section 5.2). For the rest of the section, we use $w_i(v, \alpha)$ to denote the weight of node v—defined in Section 5.2—in the i^{th} replicate.

Algorithm 9 AGGREGATESAA$((G(V,E),\alpha_{max}),k,\epsilon)$.

Input: Instance $(G(V,E),\{\mathbf{w_i} : i = 1,\dots,N\},\alpha_{max})$, parameters k, ϵ
Output: Set S^* of size at most k
Let A be the set of p-values of nodes in V below α_{max}
for $\alpha \in A$
  Let $\mathbf{w}$ be a weight vector with $w(v) = \sum_{i=1}^{N} w_i(v,\alpha)$
  $\{S_j^*(\alpha) : j = 1,\dots,k\} =$ MAXWEIGHT$(G(V,E),\mathbf{w},k,\frac{\epsilon}{n^2})$
$S^* = argmax_{j\in K,\alpha\in A}\frac{1}{N}\sum_{i=1}^{N} F_i(S_j^*(\alpha))$
return S^*

procedure MAXWEIGHT$(G(V,E),\mathbf{w},k,\epsilon')$
Input: Instance $(G(V,E),\mathbf{w})$ and parameters k,ϵ'
Output: $\{S_j^* : j \in K\}$, such that S_j^* has weight ψ_j
Let $\psi_j = -\infty$ for all $j \in K$
for $\ell = 1$ to $e^k \log(1/\epsilon')$
  For each node v, pick random color $col(v) \in K$
  for $v \in V, s \in K$
    $M(v,\{s\}) = w(v)$ if $col(v) = s$; $-\infty$ otherwise
  for $v \in V$ and $T \subseteq K$, with $|T| \geq 2$
    $M(v,T) = \max_{\substack{u\in Nbr(v)\\ T_1,T_2\subseteq T}}\{M(v,T_1) + M(u,T_2)\}$
    If $M(v,T) > \psi_{|T|}$ **update** $\psi_{|T|} = M(v,T)$
return $\{S_j^* : \sum_{v\in S_j^*} w(v) = \psi_j, \text{for } j \in K\}$

5.5.1 Algorithm for the Sample Average Approximation Formulation

Algorithm 9 describes AggregateSAA for finding a solution to Problem 5. Our method builds on our work from Chapter 3 and uses the color-coding technique of Alon et al. [12]. This algorithm finds an **optimal** subgraph of size at most k, where k is a parameter, in time $O(a^k poly(n,m))$, for some constant a—i.e., the running time is polynomial on the size of the graph, but exponential on the solution size. In contrast, a "brute force" approach would need $\binom{n}{k} = O(n^k)$ time to examine every possible connected subset of nodes of size at most k.

Overview of Algorithm AggregateSAA.

- The for loop in lines 4-6 tries out each potential value of α. For each candidate α, the aggregate weight vector $\mathbf{w}$ is computed following the process described in Section 5.2.
- The subroutine MaxWeight (from [47]) is called in line 6, and it returns a candidate solution $S_j^*(\alpha)$ for each α, and each size $j \leq k$. It uses color coding to find the optimal solution by dynamic programming.

Theorem 11 *Suppose the solution S^* computed by Algorithm* AggregateSAA *corresponds to $S_j^*(\alpha)$ for some $j \in K$. Suppose $\sum_{i=1}^{N} \sum_{v \in S_j^*(\alpha)} w_i(v) \geq c\alpha jN$, for a constant $c > 1$. Then, the score of S^* is within a factor of $\frac{c\alpha \log 1/\alpha}{c\alpha \log c + (1-c\alpha)\log \frac{1-c\alpha}{1-\alpha}}$ of the optimum score, with probability at least $1-\epsilon$ for any $\epsilon \in (0,1)$. The total running time and space used are $O(2^k e^k |A| Nm \log(n^2/\epsilon))$, and $O(2^k n)$, respectively.*

Proof: The proof relies on the convexity of the function $F(\cdot)$ for non-parametric functions. Since $\sum_{i=1}^{N} \sum_{v \in S_j^*(\alpha)} w_i(v) \geq c\alpha jN$, the minimum value $\sum_i F_i(S_j^*(\alpha))$ can take is when the weight in each replicate is the average, namely $c\alpha j$. Therefore,

$$\sum_i F_i(S_j^*(\alpha))/N \geq c\alpha \log c + (1-c\alpha)\log \frac{1-c\alpha}{1-\alpha}$$

Since $S_j^*(\alpha)$ maximizes the total weight $\sum_{i=1}^{N} \sum_{v \in S_j^*(\alpha)} w_i(v)$, it follows that for any other set S', $\sum_{i=1}^{N} \sum_{v \in S_j^*(\alpha)} w_i(v) \geq \sum_{i=1}^{N} \sum_{v \in S'} w_i(v)$. The maximum value that can be achieved by $\sum_i F_i(S')$ is when the total weight in some replicates is close to j and 0 in the rest (See Supplementary Material). Therefore,

$$\sum_i F_i(S')/N \leq \frac{\sum_{i=1}^{N} \sum_{v \in S'} w_i(v)}{jN} log 1/\alpha = c\alpha \log 1/\alpha$$

The approximation factor is therefore bounded by the ratio of these, which proves the theorem. ■

Performance guarantee in practice. The performance guarantee in Theorem 11 depends on how far the average weight of the sets is from α, over the samples. Empirically, we find that the approximation bound decreases with both c and α, and is very close to 1 in our experiments in Section 5.6.

5.5.2 Algorithm for the Max-Min Formulation

The basic idea of our algorithm is to "guess" the total weight z^* of the anomalous nodes in the optimal solution in the minimum sample, and then find a solution that has at least weight z^* in each sample—this corresponds to a multi-cover problem, as in the reduction in the proof of Lemma 15. We use the linear programming rounding method of Kolliopoulos et al. [134] for finding such an approximate solution. Algorithm 10 describes MaxMinLpRound for this problem.

Algorithm 10 MaxMinLpRound$(V, \mathbf{w}, k)$ for Max-Min formulation without connectivity constraints.

Input: Instance $V, \mathbf{w}_i, i = 1, \ldots, N$ and parameter k
Output: Instance $S \subseteq V$ that maximizes $\min_i F_i(S)$ with $|S| = k$
for $z \in [1, \max_i \sum_v w_i(v)]$ in powers of $(1 + \epsilon)$
    Construct matrix $A \in \mathbb{R}^{N \times |V|}$ with $A_{iv} = w_i(v)$ and $b \in \mathbb{R}^N$ with $b_i = z$
    Use the algorithm of [134] to find an integral solution $x \in \{0, 1\}^V$ that minimizes $\sum_i x_i$ and satisfies $Ax \geq b$
    if a solution x exists, let $S_z = \{i : x_i = 1\}$
return $S_{z^*} = \{i : x_i = 1\}$ for the maximum z^* for which a solution exists

Lemma 16 *Let S_{z^*} be the solution returned by Algorithm 10. Then,* $\min_i F_i(S_{z^*}) \geq \frac{KL(z^*/(k \log n), \alpha)}{KL(z^*(1-\epsilon)/k, \alpha)}$ *for any* $\epsilon \in (0, 1)$.

Next, we describe the heuristic BestMax for problem 6 with connectivity constraints. Proving its approximation guarantee remains an open problem; here, we analyze its running time.

Algorithm 11 BestMax$((G(V, E), \alpha_{max}), k, \epsilon)$ for Max-Min formulation with connectivity constraints.

Input: Instance $(G(V, E), \{\mathbf{w_i} : i = 1, \ldots, N\}, \alpha_{max})$, parameters k, ϵ
Output: Set S^* of size k
Let A be the set of p-values of nodes in V below α_{max}
for $\alpha \in A$
    $\{S^*_j(i, \alpha) : j = 1, \ldots, k\}$ = MaxWeight$(G(V, E), \mathbf{w_i}, k, \epsilon/n^2)$
$S^* = argmax_{j \in K, i, \alpha \in A} \min_{i'=1}^{N} F_{i'}(S^*_j(i, \alpha))$
return S^*

Lemma 17 *Algorithm* BestMax *takes time* $O(2^k e^k |A| Nm \log(n^2/\epsilon))$ *and uses space* $O(2^k n)$*, where* A *is the set defined in line 3 of the algorithm.*

5.6 Experiments

Our experiments are motivated by the following questions:

- **Detection and optimization power.** Does accounting for uncertainty improve detection of anomalous clusters compared to the deterministic case? How do our algorithms AggregateSAA and BestMax perform for the objective functions in Problems 1 and 2?
- **Effect of the number of replicates** How does detection power vary with the number of replicates N?
- **Approximation guarantee in practice** What are the practical implications of Theorem 1? What is the empirical approximation bound of AggregateSAA?

We evaluate our algorithms for scan statistics with uncertainty on the Kulldorff statistic [144] and the BJ statistic [36], which are examples of parametric and non-parametric scan statistics, respectively. We implement the AggregateSAA and BestMax algorithms from Section 5.5 for the sample average approximation (SAA) and Max-Min formulations. We evaluate the event detection power in terms of accuracy, precision, recall, and the F1 score. Let R be the set of nodes in the anomalous subgraph we discover and let S be the detected subgraph; then, we define
(1) $\text{Accuracy}(R,S) = \frac{|R \cap S|}{|R \cup S|}$,
(2) $\text{Precision}(R,S) = \frac{|R \cap S|}{|S|}$,
(3) $\text{Recall}(R,S) = \frac{|R \cap S|}{|R|}$, and
(4) $\text{F1 score} = 2\left(\frac{\text{Precision}_{(R,S)} \cdot \text{Recall}_{(R,S)}}{\text{Precision}_{(R,S)} + \text{Recall}_{(R,S)}}\right)$.

Baselines. For the evaluation, we compare our algorithms to the following baseline: Given N count vectors as in Problems 5 and 6, we select one of these vectors uniformly at random and return the connected set S that maximizes the scan statistic in the selected counts. The baseline for Problem 5, referred to as B-SAA, returns the SAA objective value for S. Similarly, the baseline for Problem 6, referred to as B-Max-Min, returns the Max-Min objective value for S. These baselines reflect the current practice of ignoring uncertainty.

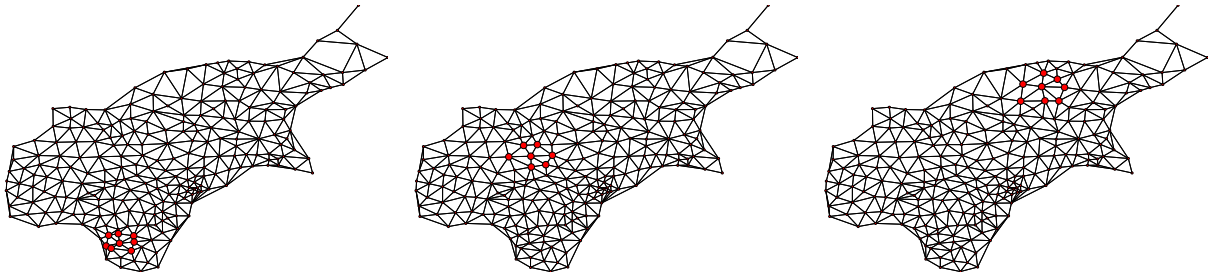


Figure 5.1: Simulated clusters on the NEast dataset in increasing order of difficulty to detect from left to right.

5.6.1 Datasets

The Northeastern USA Benchmark (NEast). This dataset [145] corresponds to occurrences of cancer in a network of 245 counties in the Northeastern part of USA. Under the null model (i.e., no significant cluster), each node v has a count $c(v) \sim Poisson(pb(v))$, where $p = 2.03 \times 10^{-5}$. We simulate anomalous clusters in this network as follows: A cluster consists of a node selected at random and all its neighbors. We generated three such clusters, which we show in Figure 5.1. These clusters vary in difficulty of detection—easy, medium, and hard. The counts for a node v inside the cluster are sampled from $Poisson(qb(v))$. We perform experiments with values of q of the form $q = \beta p$, where $\beta > 1$ is a parameter that we call *signal strength*. Intuitively, nodes in the anomalous clusters have β times as many expected counts as nodes outside the cluster. In Section 5.6.2.1, we discuss the effect of this parameter.

Then, we perturb the counts generated above using Gaussian noise. That is, for each node v, we sample and round down a count $x(v) \sim \mathcal{N}(c(v), \sigma^2)$, where σ^2 is a *noise* parameter. Notice that one could have a different noise parameter for each node, but we only consider uniform noise in our experiments.

Battle of the Water Sensor Networks (BWSN) This dataset [195] was originally used to evaluate different sensor network designs in terms of early detection of contaminants in a water system. The dataset includes "ground truth" subgraphs representing parts of the network that are contaminated, which we use for evaluation on the BJ scan statistic. We control the noise on the sensors with a parameter ϵ. With probability ϵ, the real p-value of a sensor in the network is replaced by a random p-value uniformly sampled from the interval $[0, 1]$. We show results for three clusters; these are typical for other clusters as well.

5.6.2 Evaluation

5.6.2.1 Detection and optimization performance.

First, we discuss the performance of our methods on the NEast benchmark. We evaluate performance for a wide range of signal (β) and noise (σ^2) parameters (defined in Section 5.6.1). In Figure 5.2, we show results for SAA (left), Max-Min (center), and Baseline (right) in the medium-difficulty instance. The heatmaps correspond to different combinations of signal and noise, with darker colors indicating a higher F1 score. We observe that our algorithms for SAA and Max-Min obtain higher scores than the baseline for a larger range of β-σ^2 combinations.

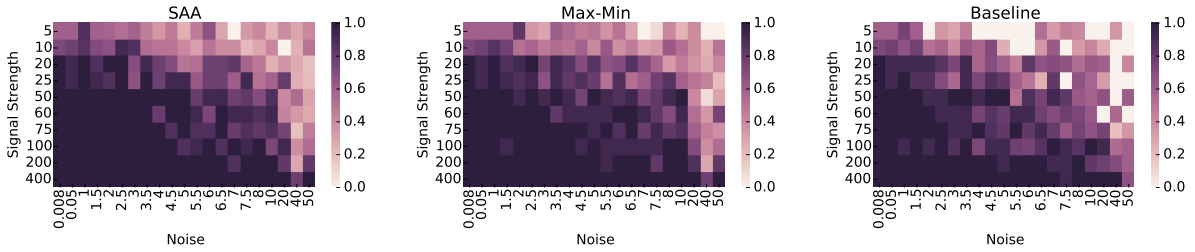


Figure 5.2: F1 score for various combinations of signal strength and noise for one of the clusters in the NEast dataset, using the AggregateSAA and BestMax algorithms, and the baseline (darker color means better performance).

In Figure 5.3, we provide an alternative way to summarize the performance gain for both formulations. The plots show the fraction of signal-noise combinations for which our algorithms have a given percentage of improvement over the baseline. For example, with the SAA formulation (left plot), we obtain at least a 20% improvement (x-axis) over the baseline on 25% (y-axis) of the signal-noise combinations from Figure 5.2 for the medium-difficulty cluster (green line). The two leftmost plots in Figure 5.3 reveal that we have larger improvement on F1 score over the baseline on the medium-difficulty instance than in the other two. If a subgraph is easy to discover, accounting for uncertainty may not offer a significant advantage. On the other hand, if an instance is hard to discover, performance is affected in all methods. However, we observe an improvement in performance in all clusters. Finally, here, we can see that the Max-Min formulation has higher improvement for the same instance. The same trends are observed for the objective score (two rightmost plots).

Next, we discuss results for the BJ statistic. In Figure 5.4 (top), we show the detection performance on the BWSN dataset with the BJ statistic as a function of noise, for three clusters. As expected, performance degrades for all methods as noise increases. However, our algorithms have a better performance than B-SAA and B-Max-Min for all levels of noise, and they degrade with the level of noise gradually. In contrast, the baselines show a very inconsistent performance. AggregateSAA and BestMax also show improved objective score compared to the baseline. In particular, for Problem 2, BestMax performs significantly better, typically giving over 20% improvement over B-Max-Min.

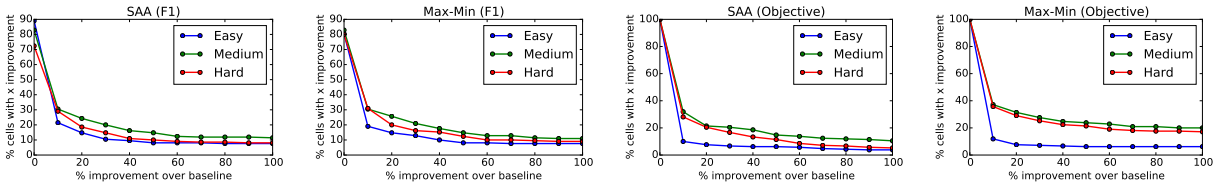


Figure 5.3: Relative improvement of AggregateSAA and BestMax over the baseline on the F1 score (left plots) and objective score (right plots) in the NEast dataset. The y-axis corresponds to the fraction of cells in Figure 5.2, for which our algorithms have a certain level of improvement over the baselines (x-axis). Higher is better.

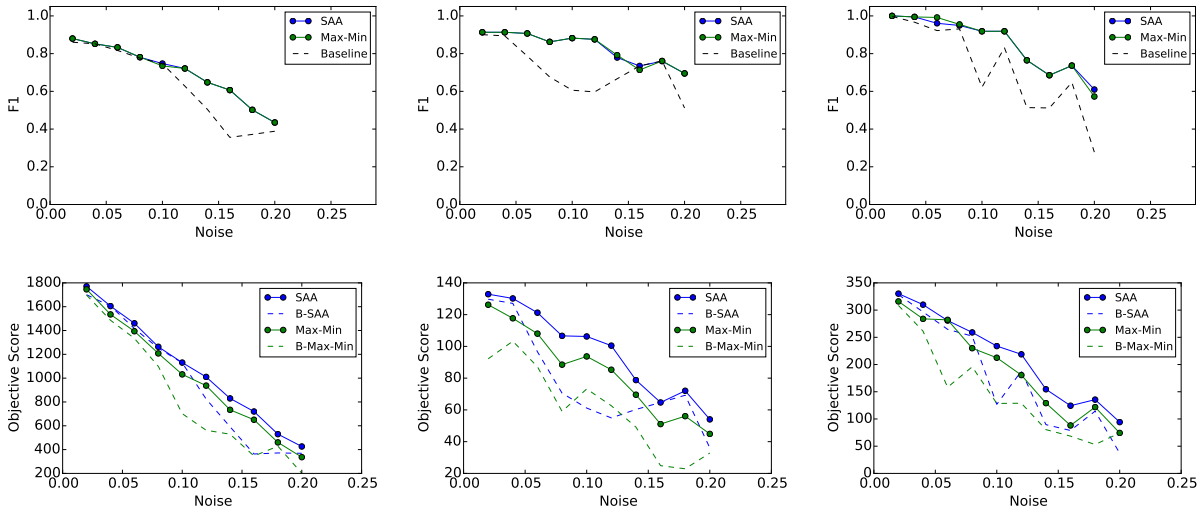


Figure 5.4: F1 score (top) and objective score (bottom) of AggregateSAA and BestMax, as a function of the noise level, compared to the baseline for three clusters in the BWSN dataset. Higher is better.

5.6.2.2 Effect of N on performance.

In Figure 5.5, we show the F1 score as a function of N, the number of replicates for the BWSN dataset. We observe that the detection power improves as we use more samples for both the SAA and the Max-Min formulation. However, the baseline does not benefit from a larger N.

5.6.2.3 Approximation bound in practice.

We analyze the empirical performance of AggregateSAA, compared with the worst case bound derived in Theorem 11. For each snapshot of the BWSN dataset (as described in Section 5.6.1), we compute the approximation ratio given in Theorem 11 for the subgraphs

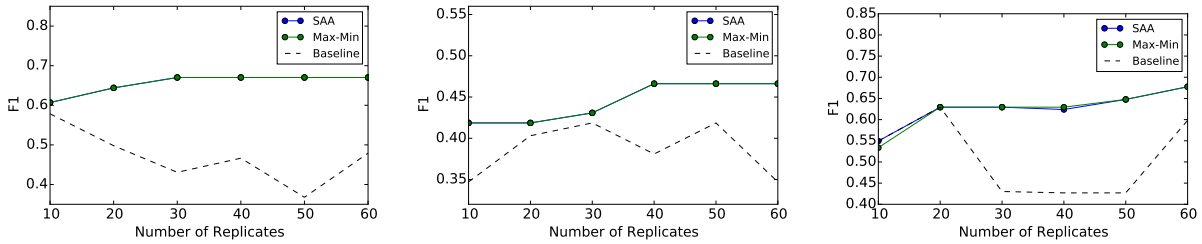


Figure 5.5: F1 score as a function of N on the BWSN dataset. Detection power increases with the number of replicates for our algorithms, but not for the baseline.

discovered by our algorithm. Figure 5.6 shows the empirical worst-case guarantee for different noise levels (i.e., each box in the plot) and each snapshot of the dataset (i.e., data points used to draw the box). We see that, for almost all the cases, the approximation guarantee is at least 80%. Further, we observe that the approximation generally becomes worse and has higher variance as noise increases. The increase in the two highest levels of noise may seem counterintuitive, but it is explained by the fact that the bound derived in Theorem 11 is with respect to the size of the solution discovered. As noise increases, the heuristic discovers smaller solutions, which are easier to approximate.

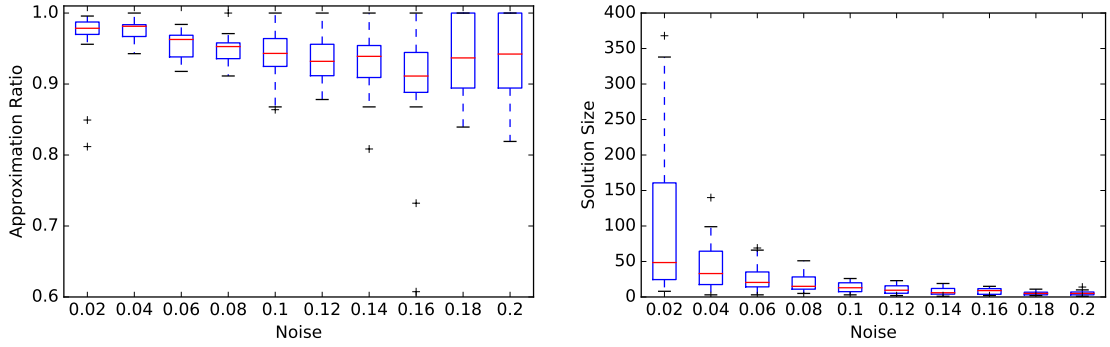


Figure 5.6: Empirical approximation guarantee of AggregateSAA for the BJ statistic (left) and size of the discovered subgraph (right) on the BWSN dataset.

5.7 Related Work

Our work is related to the broad area of anomaly detection for network data, and we refer to Akoglu et al. [9] for a comprehensive survey on this topic. For brevity, we only discuss work on scan statistics.

There are a number of parametric scan statistics, depending on the specific assumption about the observations, e.g., Positive Elevated Mean Scan Statistic [205], Expectation-based

Poisson Scan Statistic [185], and Expectation-based Gaussian Scan Statistic [185], in addition to the Kulldorff Scan Statistic [144] discussed in Section 5.2. In general, optimizing these functions is challenging in the presence of network constraints, and a number of heuristics have been proposed, e.g., branch-and-bound methods [231], Additive GraphScan [233] based on shortest paths in the graph, semi-definite programming [205] and Steiner tree heuristics [212]. The color-coding based algorithm of [47] gives rigorous results for all these functions. Similarly, in addition to the Berk-Jones [36] described in Section 5.2, there are a number of non-parametric scan statistics, such as Higher Criticism [67], Kolmogorov-Smirnov [260] and Anderson-Darling [75]. There are several heuristics to optimize such functions, [54, 173, 183, 188]. The approach of [47] extends to these functions as well.

However, as discussed in Section 5.1, none of the above methods deal with data uncertainty. Malizia [169] is one of the few papers considering the effect of uncertainty on scan statistics. This work only considers the baseline used in this chapter. The only method that we are aware of for incorporating uncertainty is the Bayesian scan statistic proposed in [186, 187]—though the authors do not explicitely motivate their work as a solution to uncertainty. The authors propose a Bayesian extension of the Kulldorff statistic assuming a Gamma-Poisson conjugate model, which allows them to derive a closed-form scan statistic. However, the methods that we develop are more general because we don't require a closed-form expression.

5.8 Conclusions

Scan statistics are used extensively in anomaly detection, but most previous works do not incorporate data uncertainty. We propose the first characterization of the effects of uncertainty on scan statistics using two formulations from stochastic optimization, and we design rigorous algorithms and heuristics for these problems. Our evaluation shows that both approaches give clear improvement on the detection power relative to a natural baseline. We expect our methodology can help incorporate the effects of uncertainty in other problems as well.

Chapter 6

Distributed Algorithms for Finding Subgraphs Using Algebraic Fingerprints

6.1 Introduction

Many problems in graph mining and social network analysis can be reduced to questions about different kinds of subgraphs; two important classes of such problems are (1) *Detecting subgraphs*, such as paths and trees, of a given size k—these are used for characterizing different kinds of networks, especially in biological models [11, 114]. (2) *Anomaly detection in network data using graph scan statistics*, which involves finding connected subgraphs of size k, optimizing different kinds of objectives [54, 101, 156, 182, 231].

Both problems are computationally very challenging. For instance, exact detection of paths is NP-hard and the corresponding counting problem is #P-Hard. Development of parallel algorithms for these problems has been an active area of research. Many parallel algorithms exist for counting local subgraphs, such as triangles [19, 22, 71, 218]. Finding trees is much harder, and a number of heuristics have been developed. One of the few techniques that gives rigorous approximation guarantees is *color coding* [11, 12, 114]. Parallel adaptations have been developed using MapReduce [270] and MPI [228, 229]. The MPI based FASCIA algorithm [228, 229] is the state-of-the-art in terms of counting trees in massive networks, scaling to finding trees with up to 12 vertices in networks with one billion edges. However, it seems very challenging to scale the color coding method to larger subgraphs, even on small networks. The main reason is that the time and space complexity of the color coding technique both scale as 2^k, where k denotes the subgraph size. We take the first steps towards beating this bound, which has remained a significant open problem since [229]. Our approach involves parallelization of a powerful algebraic technique for detecting multilinear

terms in a multivariate polynomial, developed by Koutis [136] and Williams [261].

Optimizing network scan statistics leads to challenging optimization problems as well. In Chapter refchapter:scan-stats, we use the color coding technique to develop the first method with rigorous approximation guarantees; however, one big challenge of color coding is the high memory overhead, since it scales as 2^k, where k denotes the subgraph size, which limits further scaling to larger graphs or subgraphs. In Chapter 4, we have developed a parallel adaptation of the multilinear detection technique using both GraphX and Giraph. However, none of these scaled beyond networks with 40 million edges.

Here, we develop a distributed algorithm for multilinear detection, which immediately leads to highly scalable algorithms for both paths and trees, and network scan statistics. Our contributions are

1. **MIDAS: Distributed algorithm for multilinear detection and applications to subgraph analysis.** We develop MIDAS, a distributed MPI based algorithm for finding paths and trees through detection of multilinear terms with k variables of the form $x_{i_1}x_{i_2}\ldots x_{i_k}$ (i.e., a term in which all variables have exponent 1) in a multivariate polynomial $P(x_1,\ldots,x_n)$. The sequential algorithm uses a matrix representation of a group algebra [136, 261], and its structure lends itself to a natural parallelization. We give rigorous bounds on the performance in terms of the time and space complexity, which scale as $O(2^k)$ and $O(k)$, respectively, compared with $O(2^k e^k)$ and $O(2^k)$ for color coding, respectively [228,229,270]. For random graphs, we show a rigorous scaling with N, the number of processors for $N \leq 2^k$.

2. **Cache optimization and weak scalability**. Our algorithm partitions the graph G into N_1 parts. The computation involves 2^k iterations, and N_1 processors together perform one iteration—this allows us to schedule N/N_1 such computations to occur in parallel. The total compute time exhibits good weak scaling. Additionally, our data structures for supporting Galois field operations during the iterations support a temporal cache locality, which actually leads to a reduction in the compute time as N_1 increases. On the other hand, the communication cost increases with N_1, leading to an optimal value of N_1 for the best performance.

3. **Experimental results**. We evaluate our results on a number of real and synthetic networks with up to 250 million edges and subgraph sizes up to $k = 18$. The reduced memory footprint allows us to scale to paths of size 18, which has not been done before. Our algorithms for both problems show reasonable scaling up to 512 processors, supporting our theoretical analysis. The running time grows linearly with the network size and as 2^k for the subgraph size k.

4. **Comparison with prior methods.** Our algorithm for finding paths gives over two orders of magnitude improvement in time compared to FASCIA, the state of the art method based on color coding [228, 229]. Our algorithm for scan statistics improves on

the Giraph based implementation (Chapter 4) by over an order of magnitude, and it scales to significantly larger networks.

One additional advantage is that our parallel algorithm based on multilinear detection is conceptually much simpler than color coding based algorithms, though it requires the language of algebra. It also requires far less bookkeeping than color coding. As we discuss later in Section 6.4, the obvious ways to try to parallelize multilinear detection do not perform well; instead, we find that a careful interplay between the degree of partitioning, as well as batching a set of iterations and the data structures help yield the optimal results.

6.2 Preliminaries

6.2.1 Problem Formulation

We will focus primarily on the following two classes of problems.

6.2.1.1 Finding Paths and Trees

Given a graph $G = (V, E)$ with $n = |V|$, $m = |E|$, and a subgraph $H = (V_H, E_H)$, with $k = |V_H|$, the basic subgraph isomorphism problem involves finding a mapping $f : V_H \to V$, such that $(i, j) \in E_H$ if and only if $(f(i), f(j)) \in E$.

Problem 7 *(k-Tree) Given a weighted graph $G = (V, E)$ with a weight vector $\mathbf{w}$, and a tree denoted by $H = (V^H, E^H)$ with $|V^H| = k$, the objective is to determine if there exists an embedding of H in G.*

Other common variants of this problem are: (1) counting all embeddings, and (2) finding a maximum weight embedding in a weighted version of the graph; our approach can be extended to all these variants.

6.2.1.2 Anomaly Detection Using Graph Scan Statistics

We consider the optimization problem from Chapter 3.

Problem 8 (Most Anomalous Connected Subgraph) *Given a graph $G = (V, E)$, a scan statistic $F(\cdot)$, the associated counts for vertices—$\mathbf{w}$ and $\mathbf{b}$—and a parameter k, find a connected subset $S \subseteq V$ that maximizes $F(S) = F(W(S), B(S), \theta)$ with $B(S) \leq k$.*

6.3 k-Multilinear Detection and Sequential Algorithms

We describe the sequential multilinear detection algorithm. This will start with some introduction to group algebras and Galois fields and end with the sequential algorithm for finding k-paths.

Let $X = x_1, \ldots, x_n$ be a set of variables, and let $P(X)$ be a polynomial, which is a sum of monomials on X. We will denote $P(X) = \sum_S \Pi_{i \in S} x_i$ as a monomial, where the sum is over multisets S. An example of a polynomial on six variables is $P(x_1, x_2, x_3, x_4, x_5, x_6) = x_1^2 x_2 + x_2 x_3 x_4 + x_3 x_4 x_5 + x_5 x_6$. A monomial is called *multilinear* or *square-free* if all its variables have exponent 1, and its *degree* is the sum of the exponents of all its variables. For instance, in the example above, $x_2 x_3 x_4$, $x_3 x_4 x_5$, and $x_5 x_6$ are multilinear monomials, but $x_1^2 x_2$ is not multilinear. Given variables $X = x_1, \ldots, x_n$ and a polynomial $P(X)$, the goal in the k-Multilinear Detection (k-MLD) problem is to decide whether or not $P(X)$ has a multilinear monomial of degree exactly k.

Problem 9 *(k-MLD problem) Given a polynomial $P(\cdot)$ defined recursively, in which each monomial has degree at most k and weight weight w_S, determine: (1) if $P(\cdot)$ has a multilinear term of degree k, and (2) the maximum weight of any multilinear term, if one exists.*

6.3.1 Group Algebras

Let $\mathbb{Z}_2^k$ be the group formed by all the k-dimensional binary vectors, and define the group multiplication operation as entry-wise XOR. For example, $\mathbb{Z}_2^2$ consists of the vectors $v_0 = (0,0), v_1 = (0,1), v_2 = (1,0), v_3 = (1,1)$. We note that v_0 is the multiplicative identity of the group, and each element is its own multiplicative inverse: $v_i \cdot v_i = v_0$. Now, we define a group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$. Each element in the group algebra is a sum of elements from $\mathbb{Z}_2^k$ with coefficients from $\mathbb{Z}_2$ (i.e., either 1 or 0): $\sum_{v \in \mathbb{Z}_2^k} a_v v$, where $a_v \in \{0, 1\}$. The addition operator of the group algebra is

$$\sum_{v \in \mathbb{Z}_2^k} a_v v + \sum_{v \in \mathbb{Z}_2^k} b_v v = \sum_{v \in \mathbb{Z}_2^k} (a_v + b_v) v,$$

where the addition of the coefficients is modulo 2, and the multiplication is defined as

$$\left(\sum_{v \in \mathbb{Z}_2^k} a_v v \right) \left(\sum_{u \in \mathbb{Z}_2^k} b_u u \right) = \sum_{v \in \mathbb{Z}_2^k} (a_v \cdot b_u)(v \cdot u).$$

Example. For $k = 2$, the group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$ has $2^{2^2} = 16$ elements, such as

$$x_1 = 0 \cdot \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 1 \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0 \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \text{ which we also write as } \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$x_2 = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}$$

We have

$$x_1 + x_2 = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} + 2\begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix}$$

$$x_1 x_2 = \left(\begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}\right) \cdot \left(\begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix}\right) = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}0\\1\end{bmatrix} + \begin{bmatrix}1\\0\end{bmatrix} + \begin{bmatrix}1\\1\end{bmatrix}$$

It is easy to check that

$$x_1^2 x_2 = 0 \cdot \begin{bmatrix}0\\0\end{bmatrix} + 0 \cdot \begin{bmatrix}0\\1\end{bmatrix} + 0 \cdot \begin{bmatrix}1\\0\end{bmatrix} + 0 \cdot \begin{bmatrix}1\\1\end{bmatrix} = \bar{\mathbf{0}} \text{ (additive identify)}$$

6.3.2 Sequential algorithm for Multilinear Detection

An important property is that for any $v_i \in \mathbb{Z}_2^k$, the square of the term $(v_0 + v_i) \in \mathbb{Z}_2[\mathbb{Z}_2^k]$ evaluates to 0:

$$(v_0 + v_i)^2 = v_0^2 + 2(v_0 \cdot v_i) + v_i^2 = v_0 + (0 \mod 2)v_i + v_0 = 2v_0 = 0.$$

We can show that, if we choose a $v_i \in \mathbb{Z}_2^k$ uniformly at random and set $x_i = v_0 + v_i$, then a multilinear monomial does **not** evaluate to $\bar{0}$ with high probability, whereas a monomial with squares is always $\bar{0}$ (as in the box above). The algorithm was later refined in [261] by evaluating the polynomial over the group algebra $GF(2^{3+\log_2 k})[\mathbb{Z}_2^k]$, where $GF(p)$ is the Galois field of order p [180], and this is the version that we implement. But, to simplify the discussion below, we assume that we are working on the group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$.

A polynomial $P(x_1, \ldots, x_n)$ with variables from $\mathbb{Z}_2[\mathbb{Z}_2^k]$ can be evaluated in time $O(2^k poly(n))$ and space $O(2^k poly(n))$, resulting in Theorem 8.

Theorem 12 (Koutis [136] and Williams [261]) *There exists an algorithm that, given an instance $P(x_1, \ldots, x_n)$ of the k-*MLD *problem, correctly returns "no" if $P(X)$ does not contain a k multilinear term. Otherwise, if $P(X)$ has a k multilinear term, it returns "yes" with probability at least 1/5. The algorithm has time complexity $O(2^k poly(n))$ and space complexity $O(2^k poly(n))$.*

6.3.3 Implementation Using a Matrix Representation of $\mathbb{Z}_2[\mathbb{Z}_2^k]$

Theorem 8 performs operations in the group algebra $\mathbb{Z}_2[\mathbb{Z}_2^k]$, which takes $O(2^k poly(n))$ space. Koutis [136] showed that the space complexity can be reduced to $O(kpoly(n))$ by using the idea of matrix representations. The main idea is that every element of the group algebra can be represented as a $2^k \times 2^k$ matrix, and the polynomial $P(X)$ evaluates to $\bar{0}$ if and only if the trace of its corresponding matrix representation is $0 \mod 2^{k+1}$. We can compute the trace

by evaluating the polynomial over the group of all integers 2^k times, once for each element of the diagonal. For each variable $x_i = v_0 + v_i$, the tth diagonal element in the corresponding matrix representation is $1 + (-1)^{v^T t_{bin}}$, where t_{bin} is the k-bit binary representation of t.

6.3.4 Application to k-Path

As an example of the multilinear detection technique, we now describe a sequential algorithm for the k-Path problem, which is a special case of Problem 7. We are given a graph $G(V, E)$ and a parameter k, and the algorithm decides whether or not the graph has a path of length k. First, we reduce k-Path to a k-MLD instance (this follows from [136,261]). Given a graph $G(V, E)$, let x_i denote a variable associated with each node $i \in V$. We define poynomials $P(i, j)$ for all $i \in V$, $j \leq k$ in the following manner.

- $P(i, 1) = x_i$ for all $i \in V$
- For $j > 1$, $P(i, j) = \sum_{u \in \text{NBR}(i)} P(i, 1)P(u, j - 1)$
- Define the polynomial $P(x_1, \ldots, x_n) = \sum_i P(i, k)$

Intuitively, a polynomial $P(i, j)$ encodes all the possible walks of length j ending at node i. Each monomial in $P(i, j)$ corresponds to one walk. It can be verified that the graph G has a path of length k if and only if the polynomial $P(x_1, \ldots, x_n)$ has a multilinear term—i.e., a walk with no repeated vertices.

Algorithm 12 presents the full procedure. With the matrix representation, the polynomial for k-Path is evaluated over 2^k iterations (lines 6—12). In each iteration, we first initialize $P(i, 1)$ (lines 7–8). From there, we compute recursively $P(i, j)$, a polynomial where each term contains x_i and has degree j (lines 9–11). The computation of $P(i, j)$ for a node i uses data from the immediate neighbors of i and all the polynomials of degree $j - 1$, which have already been computed at this point. The two applications that we consider in Section 6.5 have this structure.

6.4 Proposed parallel algorithm MIDAS for k-path

Opportunities and challenges for parallelization. Part of the outer for loop in lines 6–14 involves iterations which are uncoupled, in the sense that they can be done separately, as long as we are able to sum up the result P from each iteration, modulo 2^{k+1}. This gives us an easy source of parallelism, namely, run each iteration in parallel. However, this would not work if the graph does not fit in one processor's memory. The computation in the inductive step of Algorithm 12 has a *local* structure: a vertex only needs to data from its

Algorithm 12 MULTILINEARDETECTPATH($G(V, E), k$).

Input: Graph $G(V, E)$ and parameter k
Output: "Yes" if G has a k-Path, "No" otherwise.

For each node i, pick a random vector $v_i \in \mathbb{Z}_2^k$
$P = 0$
for $t = 0$ to 2^{k-1}
  ▶ **Base case**
  for $i \in V$ **do**
    $P(i, 1) = 1 + (-1)^{v_i^T \cdot t_{\text{bin}}}$
  ▶ **Inductive step**
  for $i \in V$, $j = 2$ to k **do**
    $P(i, j) = \sum_{u \in \text{NBR}(i)} P(i, 1)P(u, j - 1)$
  $P(k) = \sum_i P(i, k)$ for $i \in V$
  $P = P + P(k) \mod 2^{k+1}$
return "Yes" if $P \neq 0$, else "No"

immediate neighbors in the graph. An alternative approach is to partition the graph into N parts, and then try to parallelize the local computation. However, neither extreme works well, and instead, MIDAS partitions the graph into N_1 parts, and runs N/N_1 iterations in parallel. This approach can lead to significant savings on the computation time, but has a high communication overhead. Since the values exchanged between nodes are small, we introduce an idea of combining the communications of multiple iterations together as a way to reduce the overhead.

6.4.1 Overview of Algorithm MIDAS

Let N denote the total number of processors or parallel units available. Quantities N_1 and N_2 are parameters for controlling the parallelism in different parts of the algorithm. We assume $2^k/N_2$ and N/N_1 are integers, in order to avoid cluttering the notation using ceiling and floor of these quantities. The algorithm involves solving a dynamic program 2^k times; these 2^k loops are independent, and we divide them into *phases* of size N_2 each, so that a total of $2^k/N_2$ phases have to be run. These are run in $2^k/(N_2N/N_1)$ *batches*, where each batch involves running N/N_1 phases. A phase involves a call to the subroutine PARALLELEVALUATEPOLYNOMIALPATH. See Figure 6.1 for an illustration of this structure.

We partition the graph G into N_1 parts, denoted by $\mathcal{P} = \{G^1, \ldots, G^{N_1}\}$; desirable properties of the partition will be discussed later. For a partition j, let DEG(j) be the *degree* of j, defined as the number of edges connecting nodes in j to nodes in some other partition:

$$\text{DEG}(j) = |\{(u, v) : (u, v) \in G, u \in G^i, v \notin G^i\}|,$$

and let MAXDEG $= \max_j \text{DEG}(j)$. Also, let MAXLOAD $= \max_j |G^j|$ be the maximum "load"

Table 6.1: Summary of notation

Symbol	**Description**
N	Total number of processors
N_1	Number of parts in graph partitioning
N_2	Size of each phase
Phase	Group of N_2 iterations for which communication is done simultaneously
Batch	A set of N/N_1 phases
$\mathcal{P}$	Partition of G in N_1 parts, $G^1, \ldots, G^{N_1}$

or number of vertices on any partition. We will analyze the performance of our algorithm in terms of MAXLOAD and MAXDEG.

We describe the main steps of MIDAS below.

1. The algorithm starts with the partitioning $\mathcal{P}$ of the graph G.

2. The algorithm runs $\log 1/\epsilon \log 5/4$ rounds, each of which involves 2^k iterations. Here, $\epsilon \in (0, 1)$ is a parameter, which governs the success probability[1]. Each such round with 2^k iterations is partitioned into $2^k/N_2$ phases in the while loop in lines 8–12 of MIDAS, which are completely independent of other phases.

3. In the tth phase, Algorithm ParEvaluatePolynomialPath uses a vector of size N_2 to store polynomials $\langle P(i, tN_2, j), \ldots, P(i, (t+1)N_2 - 1, j)\rangle$ for each node i and $j \in [1, k]$. $P(i, q, j)$ corresponds to the polynomial of node i and degree j for the q^{th} diagonal element in the matrix representation.

4. In the tth phase, for each node i, we use the vector for each neighbor u of i to compute $\langle P(i, tN_2, j), \ldots, P(i, (t+1)N_2 - 1, j)\rangle$. If u is in the same partition, then its data is available on that processor. For every neighbor u in a different partition, u has to send a message with $\langle P(u, tN_2, j-1), \ldots, P(u, (t+1)N_2 - 1, j-1)\rangle$, introducing a communication overhead.

5. We use $\text{SUM}_t^\ell = \sum_{i \in V} P(i, tN_2, k) + \ldots + \sum_{i \in V} P(i, (t+1)N_2 - 1, k)$ to denote the sum of the polynomial evaluations for phase t, within round ℓ. These are summed up over all the phases within round ℓ to compute the total, denoted by P^ℓ.

[1] As per Theorem 8, Algorithm MultilinearDetectPath succeeds with probability 1/5, so we need to run it multiple times

Algorithm 13 MIDAS$(G, k, \epsilon, N_1, N_2)$.

Input: Graph $G = (V, E)$, parameter k, confidence parameter $\epsilon \in (0, 1)$, parameters N_1 and N_2, which guide the parallelism.
Output: "Yes" if G has a k-Path, "No" otherwise.

Let $v_i \in \mathbb{Z}_2^k$ be a random vector for each node i
Let $P = 0$ be the polynomial
Let N_1 denote the number of processors used for each iteration. Let $\mathcal{P} = \{G^1, \ldots, G^{N_1}\}$ denote the corresponding partition of the graph into N_1 parts.
for $\ell = 1$ to $(\log 1/\epsilon)/(\log 5/4)$
  $P^\ell = 0$
  while $s \leq \frac{2^k/N_2}{N/N_1}$ **do**
    for $t = sN/N_1$ to $(s+1)N/N_1$ **do in parallel**
      $\text{SUM}_t^\ell = \text{ParEvaluatePolynomial}(G, k, \mathbf{v}, t, N_2, N_1, \mathcal{P})$
    MpiBarrier
    $P^\ell = P^\ell + \sum_{t=sN/N_1}^{(s+1)N/N_1} \text{SUM}_t^\ell \mod 2^{k+1}$ using MpiReduce
if $P^\ell \neq 0$ for some ℓ
  return "Yes"
else
  return "No"

Algorithm 14 ParEvaluatePolynomialPath$(G, k, \mathbf{v}, t, N_2, N_1, \mathcal{P})$

Input: Graph G, parameter k, random assignment $\mathbf{v}$, phase number t, number of iterations within phase N_2, number of partitions N_1, and partitioning $\mathcal{P}$
Output: The value of the polynomial corresponding to k-path in the iterations within a phase

for each processor s **do in parallel**
  Base case
  for node $i \in G^s$ and iteration $q \in [tN_2, (t+1)N_2 - 1]$ **do**
    $P(i, q, 1) = 1 + (-1)^{v_i^T \cdot q_{\text{bin}}}$
  Recursive step
  for $j = 2$ to k **do**
    for node $i \in G^s$ **do**
      for all q set $P(i, q, j) = 0$
      for each incoming message $\langle u, P(u, q, j-1)\rangle$ **do**
        $P(i, q, j) = P(i, q, j) + P(i, q, 1)P(u, q, j-1)$
      Send result to neighbors
      for $u \in \text{Nbr}(i) \setminus G^s$ **do**
        Send $\langle i, P(i, q, j)\rangle$
MpiBarrier
return $\sum_q \sum_i P(i, q, k)$

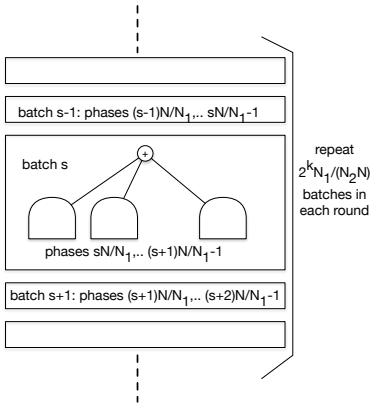


Figure 6.1: Schematic structure of MIDAS: we run $(\log 1/\epsilon)/(\log 5/4)$ rounds. Each round is partitioned into $2^k N_1/(N_2 N)$ batches, and each batch involves N/N_1 phases being run simultaneously. Each phase involves an evaluation of the polynomial on N_2 iterations in algorithm ParEvaluatePolynomial, which are then summed up.

6.4.2 Computation and Communication Complexity

Recall the definition of maxdeg corresponding to the partitioning $\mathcal{P}$. Further, let c_1 and c_2 denote the time for unit computation at any node in G and the unit communication along any edge, respectively, in the Algorithm ParEvaluatePolynomialPath. The time and communication complexity of algorithm MIDAS is summarized below in terms of these parameters.

Theorem 13 *For any $\epsilon \in (0,1)$, Algorithm* MIDAS *solves the k-Path problem for an instance G, k with probability at least $1-\epsilon$. The total time for computation and communication are $O\left(c_1 \frac{2^k N_1}{N} k\text{MAXLOAD} \log 1/\epsilon\right)$ and $O\left(c_2 \frac{2^k N_1}{N N_2} k\text{MAXDEG} \log 1/\epsilon\right)$, respectively.*

Proof: (Sketch) First, we argue the correctness. The call to ParEvaluatePolynomialPath evaluates the polynomial bottom up in parallel for all iterations in the tth phase, namely iterations $tN_2, \ldots, (t+1)N_2 - 1$. The vector $\langle P(tN_2, k), \ldots, P((t+1)N_2 - 1, k)\rangle$ is the final evaluation of the polynomial for each iteration in this phase. Each call to ParEvaluatePolynomialPath in Algorithm MIDAS returns the sum of these values for phase t. Each round of Algorithm MIDAS, corresponding to the value of ℓ in the outer for loop in lines 6–12 in the sequential algorithm, goes over all phases, and calls ParEvaluatePolynomialPath. Therefore, within round ℓ, P^ℓ denotes the sum of the polynomial evaluation over all the 2^k iterations within that round. If $P(\cdot)$ has a multilinear term, from [136, 261], $P^\ell \neq 0 \bmod 2^{k+1}$ with probability at least $1/5$. This implies that $\Pr[P^\ell = 0,\ \forall \ell] = (\frac{4}{5})^{(\log 1/\epsilon)/(\log 5/4)} \leq \epsilon$, so that with probability at least $1-\epsilon$, MIDAS returns "Yes" if G has a k-path. On the other hand, if $P(\cdot)$ has no multilinear term, then with probability 1, $P^\ell = 0$ for all ℓ. Therefore, MIDAS correctly solves the k-Path problem with probability at least $1-\epsilon$.

Next, we consider the computation and communication time complexity. The algorithm ParEvaluatePolynomialPath computes the polynomial for each degree up to k within each iteration. Therefore, the computation time in a phase t is $O(c_1 k \max_j |G^j| N_2) = O(c_1 k \textsc{maxload} N_2)$, which is the maximum time for any processor. Therefore, the total compute time over all the rounds is $O(\frac{2^k/N_2}{N/N_1} c_1 k \textsc{maxload} N_2)$, which corresponds to the bound in the theorem. After the evaluation in the recursive step, the results have to be sent on all neighbors, for every pair of processors s, s'. Therefore, the maximum number of messages in one iteration of the loop in lines 8–15 is maxdeg, and the total number of messages, over all the rounds, is $O(\frac{2^k/N_2}{N/N_1} \textsc{maxdeg}) = O(\frac{2^k N_1}{N N_2} \textsc{maxdeg})$. ■

Memory Access: The recursive step in ParEvaluatePolynomialPath has some interesting properties of a highly memory bound region. Recall that polynomial multiplication terms are summed up for each of the incoming messages. This computation loop may be subjected locality effects of the memory sub-system and pipe-lining by the logical processor. Therefore, selecting appropriate values for N_2 is important to leverage fast memory access[2] and achieve desired parallel performance.

Lemma 18 *For a graph $G = (V, E)$ drawn from the Erdös-Renyi model, $G(n, p)$, the computation and communication times for a random partition are $O(c_1 \frac{2^k nk}{N} \log 1/\epsilon)$ and $O(c_2 \log 1/\epsilon \frac{2^k mk}{N N_2})$, respectively, with high probability.*

Proof: (Sketch) For a random partition into N_1 parts of equal size, we have maxload $= n/N_1$, and the bound for the total compute time follows from Theorem 13. Since $G \in G(n, p)$, it follows that maxdeg $= O(\frac{n}{N_1}(n - \frac{n}{N_1}p)) = O(m/N_1)$, with high probability, and the Lemma follows. ■

6.5 Parallel Algorithms for k-Tree and Scan Statistics

We now describe how MIDAS for the k-path problem can be extended to parallel algorithms for finding trees and optimizing scan statistics. We discuss here how the corresponding polynomials are constructed recursively and evaluated in the subroutines ParEvaluatePolynomialTree and ParEvaluatePolynomialScanStat; the main Algorithm MIDAS remains unchanged. We recall the notation from Section 3.2.

6.5.1 k-Tree

We describe how an instance of k-Tree with graph $G = (V, E)$ and tree $H = (V^H, E^H)$ is reduced to a k-MLD instance. We consider the tree H to be rooted, and let root(H) be

[2]cache affinity in-terms of spatial and temporal locality results in fast memory access

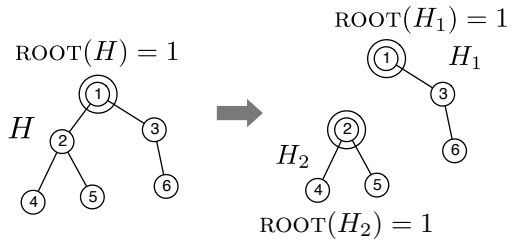


Figure 6.2: Tree H with $\text{ROOT}(H) = 1$. It is decomposed into trees H_1 and H_2 by removing the edge $(1, 2)$. $\text{ROOT}(H_1) = 1$ and $\text{ROOT}(H_2) = 2$.

the root node, selected arbitrarily. We consider a hierarchical structure among subtrees of H in the following manner: consider any node $u \in \text{NBR}(\text{ROOT}(H))$. Let H_1 and H_2 denote the subtrees or *children* obtained upon deleting the edge $(u, \text{ROOT}(H))$, with $\text{ROOT}(H) \in H_1$ and $u \in H_2$. We set $\text{ROOT}(H_1) = \text{ROOT}(H)$ and $\text{ROOT}(H_2) = u$. This process is illustrated in Figure 6.2. The subtrees H_1 and H_2 are further partitioned in a recursive manner until all trees have a single node. We define the polynomials $P(i, H')$, which will correspond to all layouts (not necessarily isomorphisms) of H' with $\text{ROOT}(H') = i$ for all nodes $i \in V$, in the following manner:

- If H' consists of a single node, $P(i, H') = x_i$
- Else, $P(i, H') = \sum_{u \in \text{NBR}(i)} P(i, H'_1)P(u, H'_2)$, where H'_1 and H'_2 are the children of H'.
- Finally, we have $P(x_1, \ldots, x_n) = \sum_{i \in V} P(i, H)$

By using ideas from [11], it can be verified that the tree H is a subgraph of G if and only if the polynomial $P(x_1, \ldots, x_n)$ has a multilinear term. Algorithm ParEvaluatePolynomialTree evaluates this polynomial analogous to Algorithm 15 from Section 6.4. The performance of MIDAS, using ParEvaluatePolynomialTree is summarized below.

Lemma 19 *For any $\epsilon \in (0, 1)$, Algorithm* MIDAS*, using* ParEvaluatePolynomialTree*, solves the k-*Tree *problem for an instance G, H with probability at least $1 - \epsilon$. The total time for computation and communication are $O\left(c_1 \frac{2^k N_1}{N} |\mathcal{T}| \text{MAXLOAD} \log 1/\epsilon\right)$ and $O\left(c_2 \frac{2^k N_1}{N N_2} |\mathcal{T}| \text{MAXDEG} \log 1/\epsilon\right)$, respectively.*

6.5.2 Scan Statistics

Let $W(V) = \sum_{i \in V} w(i)$ be the total weight of the nodes in G. For each node i, we define a variable x_i, and we construct a polynomial over the set of variables $\{x_i : i \in V\}$. Every term—i.e., monomial—in this polynomial will represent a connected subgraph of size at most

Algorithm 15 ParEvaluatePolynomialTree$(G(V,E), H(V^H, E^H), \mathbf{v}, t, N_2, N_1, \mathcal{P})$

Input: Graph $G(V,E)$, tree $H(V^H, E^H)$ with k vertices, random assignment $\mathbf{v}$, phase number t, number of iterations within phase N_2, number of partitions N_1, and partitioning $\mathcal{P}$
Output: The value of the polynomial corresponding to k-tree in the iterations within a phase

Let $\mathcal{T}$ be a collection of subtrees of H sorted by size
for each processor s**do in parallel**
  for each subtree $j \in \mathcal{T}$ **do**
    for node $i \in G^s$ and iteration $q \in [tN_2, (t+1)N_2 - 1]$ **do**
      if $|j| = 1$ **then**
        $P(i,q,j) = 1 + (-1)^{v_i^T \cdot q_{\text{bin}}}$
      else
        set $P(i,q,j) = 0$
        let j' and j'' be the children of subtree j
          for each incoming message $\langle u, P(u,q,j'')\rangle$ **do**
            $P(i,q,j) = P(i,q,j) + P(i,q,j')P(u,q,j'')$
      Send result to neighbors
      for $u \in \text{Nbr}(i) \setminus G^s$ **do**
        Send $\langle i, P(i,q,j)\rangle$
MpiBarrier
return $\sum_q \sum_i P(i,q,H)$

k and weight at most $W(V)$. For $j \leq k$ and $z \leq W(V)$, let $P(i,j,z)$ be the polynomial corresponding to a subgraph (1) containing node i, (2) of size j, and (3) total weight z. The following recurrence relations describe how the polynomials $P(i,j,z)$ are computed:

- $P(i,1,z) = x_i$ for all $i \in V$, $z = w(v)$
- For $i \in V$, $j = 2$ to k, $z = 0$ to $W(V)$, $P(i,j,z) = \sum_{u \in \text{Nbr}(i)} \sum_{j'=1}^{j-1} \sum_{z'=0}^{z} (P(i,j',z') \cdot P(u, j-j', z-z'))$
- $P(j,z) = \sum_i P(i,j,z)$ for $j \leq k$, $z \leq W(V)$

Algorithm 16 maintains variables $P(i,q,j,z)$ for every node i, $j \leq k$, $z \leq W(V)$, and iteration q within phase t. The input graph G has a connected subgraph S of size j and weight z if and only if the corresponding polynomial $P(j,z)$ has a multilinear term. We have the following lemma.

Lemma 20 *For any $\epsilon \in (0,1)$, Algorithm* MIDAS*, using* ParEvaluatePolynomialScan-Stat*, solves the* Scan Statistics *problem for an instance $G, k, \mathbf{w}$, with probability at least $1-\epsilon$. The total time for computation and communication are $O\left(c_1 \frac{2^k N_1}{N} W(V)^2 k^2 \text{MAXLOAD} \log 1/\epsilon\right)$ and $O\left(c_2 \frac{2^k N_1}{N N_2} W(V)^2 k^2 \text{MAXDEG} \log 1/\epsilon\right)$, respectively.*

We note that the performance for scan statistics can be improved significantly by rounding the weights, as in Chapter 4.

Algorithm 16 ParEvaluatePolynomialScanStat$(G(V,E), k, \mathbf{w}, \mathbf{v}, t, N_2, N_1, \mathcal{P})$

Input: Graph $G(V,E)$, parameter k, node weights $\mathbf{w}$, random assignment $\mathbf{v}$, phase number t, number of iterations within phase N_2, number of partitions N_1, and partitioning $\mathcal{P}$
Output: The value of the polynomial corresponding to the scan statistics in the iterations within a phase

for each processor s **do in parallel**
  for node $i \in G^s$ and iteration $q \in [tN_2, (t+1)N_2 - 1]$ **do**
    $P(i, q, 1, w(v)) = 1 + (-1)^{v_i^T \cdot q_{\text{bin}}}$
  for $j = 2$ to k and $z = 0$ to $W(V)$ **do**
    for node $i \in G^s$ **do**
      for all q set $P(i, q, j, z) = 0$
      for each incoming message $\langle u, P(u, q, j - j', z - z') \rangle$ **do**
        $P(i, q, j, z) = P(i, q, j, z) + P(i, q, j', z')P(u, q, j - j', z - z')$
      Send result to neighbors
      for $u \in \text{Nbr}(i) \setminus G^s$ **do**
        Send $\langle i, P(i, q, j, z) \rangle$
MpiBarrier
return $\sum_q \sum_i P(i, q, k, z)$ for all $z \leq W(V)$

6.6 Experiments

We evaluate the performance of the proposed parallel algorithms on parallel speedup and scalability. In particular, our experiments address the following questions:

1. **Effect of partition size** investigates the performance variation with partition size as N_1 is varied (Section 6.6.2)
2. **Scalability with subgraph size** depicts the total runtime as subgraph size is increased. (Section 6.6.3)
3. **Strong scaling** looks at the total runtime as the number of parallel processors is increased, thereby reducing the computing workload per process. (Section 6.6.4)
4. **MIDAS vs. FASCIA** presents the runtime comparison of our implementation compared to FASCIA. (Section 6.6.5)
5. **Performance of Scan Statistics and its applications** provides performance results for the parallel Scan Statistics implementation and presents a real life application of it. (Section 6.6.6)

6.6.1 Experimental Setup

6.6.1.1 Hardware

Experiments were conducted on Juliet, an Intel Haswell HPC cluster. Up to 32 nodes were used for the evaluation, where each node has 36 cores (2 sockets x 18 cores each). A node consists of 128GB of main memory and 56Gbps Infiniband interconnect. We also tested on another HPC cluster, Shadowfax-Haswell, where we used 32 nodes each with 32 cores (2 sockets x 16 cores each). Memory and interconnect of this cluster are similar to those of Juliet.

6.6.1.2 Datasets

A summary of the datasets is provided in Table 6.2. In addition, we perform experiments in two Erdos-Renyi networks of 1 and 10 million nodes with an expected number of edges of $n \log n$, where n is the number of nodes.

Table 6.2: Datasets used in our experiments

Dataset	**Nodes** ($\times 10^6$)	**Edges** ($\times 10^6$)
miami	2.1	51.5
com-Orkut	3.1	234.3
random-1e6	1	13.8
random-1e7	10	161.8

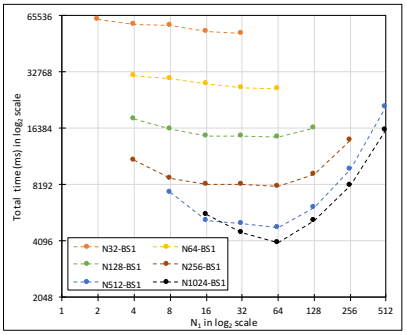


Figure 6.3: k-path total runtime for random-1e6 dataset and varying N_1. Note. $BS1 = N_2 = 1$

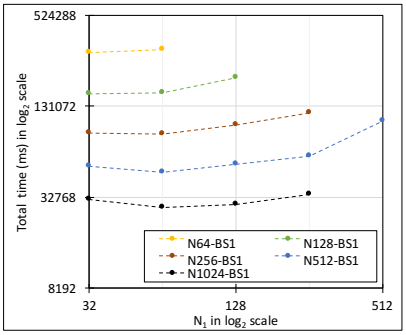


Figure 6.4: k-path total runtime for com-Orkut dataset and varying N_1. Note. $BS1 = N_2 = 1$

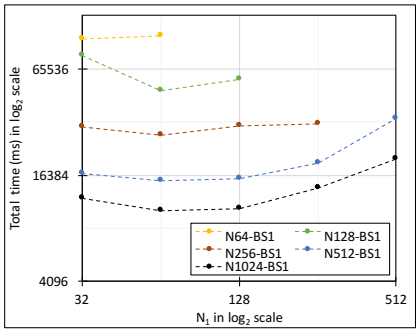


Figure 6.5: k-path total runtime for miami dataset and varying N_1. Note. $BS1 = N_2 = 1$

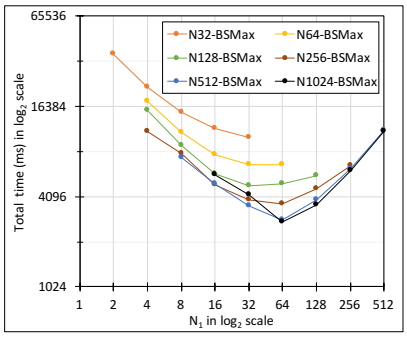


Figure 6.6: k-path total runtime for random-1e6 dataset and varying N_1. Note. $BSMax = N_2 = 2^k N_1/N$

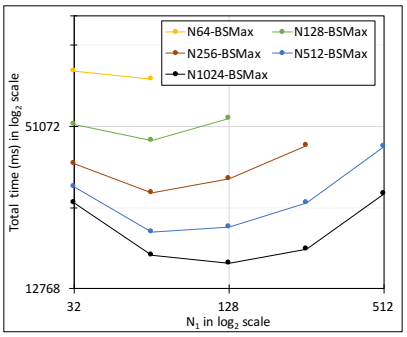


Figure 6.7: k-path total runtime for com-Orkut dataset and varying N_1. Note. $BSMax = N_2 = 2^k N_1/N$

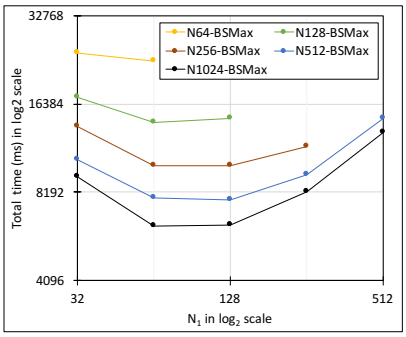


Figure 6.8: k-path total runtime for miami dataset and varying N_1. Note. $BSMax = N_2 = 2^k N_1/N$

6.6.2 Effect of Partition Size

The multilinear detection based parallel k-Path and Multilinear Scan algorithms exhibit two levels of parallelism: vertex and iterations. On one hand, the 2^k iterations are pleasingly parallel except for a global reduction at the end. The parallel vertex computation within each iteration, on the other hand, requires message passing between neighbors for $k-1$ steps (in the case of trees, this would be the number of sub templates instead of $k-1$).

Given these two levels of parallelism and a total of N processes, we can split the 2^k iterations among $a = N/N_1$ parallel phases. Each phase decomposes the graph across N_1 processes and performs the k-Path computation in parallel for $2^k/a$. To reduce the communication over computation cost, the algorithm packs a user defined N_2 number of iterations into one computation step, so each parallel phase only has to perform $2^k/(a * N_2)$ compute and communication phases.

To illustrate this with an example, consider the case of $k = 6$, $N = 128$, $N_1 = 32$, and $N_2 = 8$. The total number of iterations is $2^k = 64$. The number of parallel phases corresponding to $N_1 = 32$ is $128/32 = 4$. Each phase only needs to run $64/4 = 16$ iterations. Since $N_2 = 8$, the 16 iterations can be completed in just $16/8 = 2$ batches.

Increasing N_2, for example $N_2 = 16$ in the previous case, would allow us to finish the entire program in one compute and communicate batch. This results in higher parallel efficiency as the overhead of communication to computation is reduced. However, it increases the message size by N_2 factor[3]. Depending on the number of total processes, MPI may fail to accommodate very large message sizes requiring to reduce N_2 such that some form of chunking method may be required.

[3]Increasing message size for a communication step is not necessarily a bad occurrence. Reducing number of small messages in communication may lead to increased network performance. [135]

Figures 6.3, 6.4, 6.5 illustrate the performance of MIDAS on three different datasets when $N_2 = 1$. We observed that running times of MIDAS when N_2 is scaled for a fixed value of N for each problem size—this effectively tested our parallel algorithm for a large range of configurations. Our observations confirm the existence of an optimal point (i.e., a minimum) between two levels of parallelism discussed before. The communication cost gradually increases when moving from one extreme end of parallelism to the other because of the increase in number of messages exchanged [4]. However, at the optimal point, the cost of communication can be sufficiently amortized by the amount of parallelism gained. In other words, the optimal solution for MIDAS algorithm can be found in a point between vertex level and iteration based parallelism. Interestingly, when N_2 is increased (Figures 6.6, 6.7, 6.8) we observe further relative performance gains on the same set of experiments. The observed speedup (between ∼1x - ∼2x) is due to cache affinity effects on the main loop (Section 6.4.2) and reduction of communication phases by increasing the message size. Furthermore, speedups are evident for each experiment instance of the k-path problem when problem size is scaled.

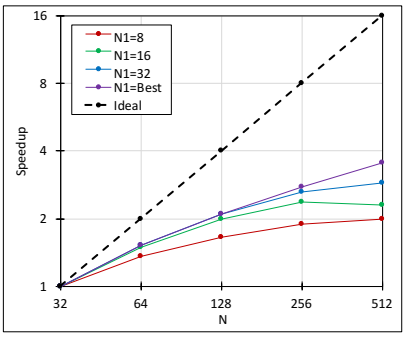


Figure 6.9: MIDAS strong scaling for the k-path problem with increasing N, while N_1 is fixed.

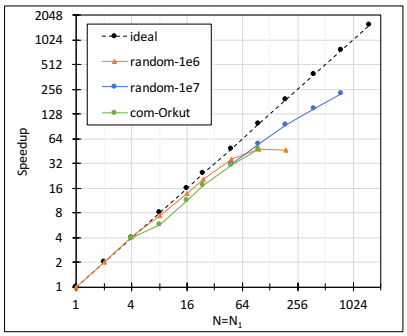


Figure 6.10: MIDAS strong scaling for the k-path problem with increasing N and $N_1 = N$.

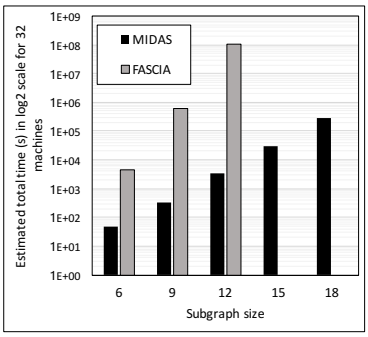


Figure 6.11: MIDAS runtime compared to FASCIA for varying subgraph sizes of the k-path problem.

6.6.3 Scalability with subgraph size

In Figure 6.11, we increase the subgraph size, k, in both FASCIA and MIDAS, while keeping N and N_1 fixed. Note, Figure 6.6 through Figure 6.8 suggest it is best to keep N_2 as high as possible to leverage the cache locality benefits discussed above. However, the total message size communicated out of a process increases with N_2 leading to diminishing returns. Therefore, we've kept $N_2 < 1024$.

[4]Number of messages exchanged can be approximated to $O(\log N_1)$ for small message sizes where N_1 ranges from $N_1 = 1$ to $N_1 \to N$

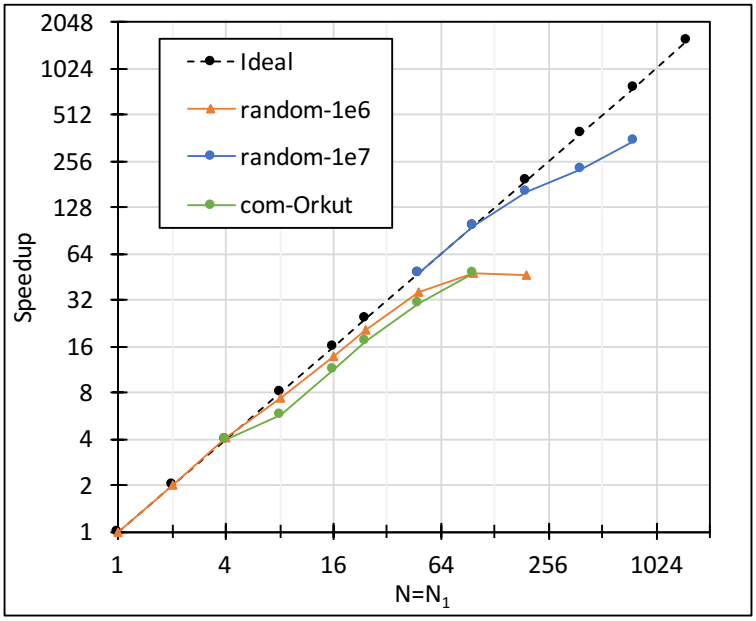


Figure 6.12: MIDAS strong scaling for the Scan Statistics problem with increasing N and $N_1 = N$

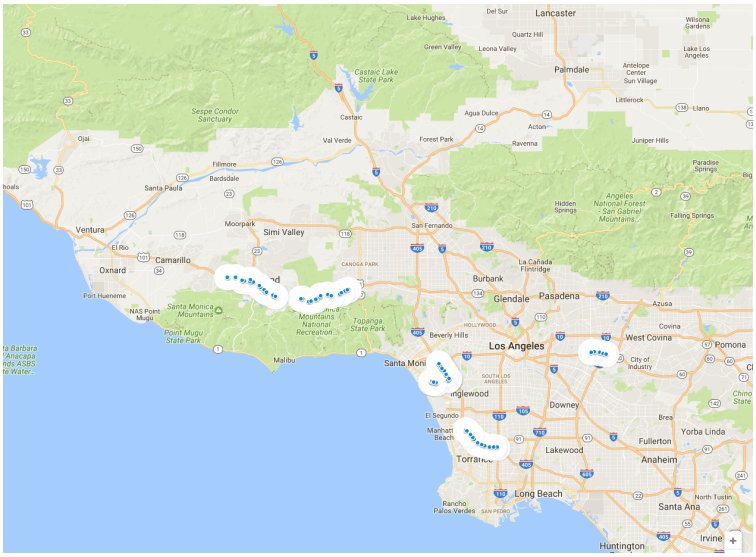


Figure 6.13: Discovering highway segments with unexpected congestion in the Los Angeles road network.

6.6.4 MIDAS Strong Scaling

Strong scaling of MIDAS can be investigated in two ways. The first is to fix N_1 and change N, thereby increasing the parallel phases to split the 2^k iterations. We could observe the effect of this behavior by examining the values along a fixed N_1 value in Figures 6.3 through 6.8. Dividing the runtime corresponding to the minimum N (the top most line) by the given N gives speedup indicating the strong scalability of MIDAS.

Figure 6.9 presents such speedup for a set of N_1 values over varying N. We observe the results do not necessarily scale linearly due to the fact that communication within a phase is dominant. We get the best speedup by going along points in Figures 6.3 through 6.8 that gave the minimum runtime. This is shown as the $N1 = Best$ line in Figure 6.9.

The other form of strong scalability we can test is by setting $N1 = N$. This produces a single phase and is the classic strong scaling of parallel graph algorithms. Figure 6.10 presents the speedp values for different datasets. While, the speedups are less than ideal, they still scale well up to a considerable number of processes.

6.6.5 MIDAS vs. FASCIA

Figure 6.11 compares the running time of FASCIA to MIDAS for varying subgraph sizes. We see FASCIA fails to support beyond subgraphs of size 12 on this random-1e6 dataset, whereas MIDAS scales to well over 18. Also, MIDAS shows a significant improvement over FASCIA in runtime.

6.6.6 Performance of Scan Statistics and Its Applications

In Figure 6.12, we present strong scaling results for Scan Statistics problem where N_1 is set to N. We do this for multiple datasets and observe they show considerable strong scalability similar to k-Path problem in Figure 6.10.

Congested Highways Clusters in Road Networks We apply our algorithm for scan statistics to find clusters with unexpectedly low-moving traffic in the highway network of Los Angeles County[5]. Nodes in the graph are sensors next to the road that record the average speed and the number of vehicles passing through. We have 30-minute snapshots for May 2014. We assume that the average speed recorded by each sensor follows a normal distribution. Then, the p-value of a node i is the cumulative distribution function of a normal distribution with mean $\mu_i^{[1,t-1]}$ and standard deviation $\sigma_i^{[1,t-1]}$, where $\mu_i^{[1,t-1]}$ and $\sigma_i^{[1,t-1]}$ are, respectively, the sample mean and standard deviation for node i from snapshots 1 to $t-1$.

We use our algorithm with $k = 12$ on this dataset. In Figure 6.13, we show with blue

[5] `http://pems.dot.ca.gov/`

dots highway segments that our algorithm identifies as having *unexpectedly* low average speed during rush hour (16:00 to 19:00) on Friday May 9, 2014. These segments are not necessarily the ones with most congestion. For instance, the center of Los Angeles city has higher congestion; however, such activity is normal on Friday afternoons according to the previous snapshots. The clusters shown in the map are selected because they have significantly lower average speeds than in previous observations.

6.7 Related Work

There is a large literature on a variety of subgraph analysis problems, arising out of a number of applications, such as bioinformatics, security, social network analysis, epidemiology and finance (see [9] for a survey). We discuss some of three main directions here: subgraph isomorphism and clique enumeration (for which parallel algorithms exist), and anomaly detection (for which there has been limited work on parallel algorithms).

Given a graph $G = (V, E)$ with $n = |V|$, $m = |E|$, and a subgraph $H = (V_H, E_H)$, with $k = |V_H|$, the basic subgraph isomorphism problem involves finding a mapping $f : V_H \to V$ such that $(i, j) \in E_H$ if and only if $(f(i), f(j)) \in E$. This is a very computationally challenging problem. The frequent subgraph detection problem involves finding subgraph isomorphisms (also referred to as embeddings) having frequency higher than a threshold; other variations involve finding non-overlapping embeddings. Parallel approaches for this problem involve a "bottom-up" candidate generation approach, combined with careful pruning, which builds embeddings of larger subgraphs using all possible embeddings of smaller subgraphs [2, 76]. While these results allow scaling to very large networks with millions of nodes, they give no guarantees on the performance. Our work is more closely related to the use of the color coding technique for finding tree-like subgraphs [11, 114], which guarantees a fully polynomial time approximation to the number of embeddings with running time and space of $O(2^k e^k m \log n)$ and $O(2^k m)$, respectively. This has been parallelized using MapReduce [270] and OpenMP [228,229], enabling subgraph counting in graphs with tens of millions of nodes with rigorous guarantees. Slota et al. [228,229] use threading and techniques for reducing the memory footprint of the color coding dynamic programming tables, in order to scale. Our approach uses algebraic methods for which the memory scales as $O(k)$ instead of $O(2^k)$, and the worst case running time scales as $O(2^k)$ instead of $O(2^k e^k)$, which gives the improved performance.

Another area where parallel algorithms have been developed is for dense subgraph enumeration. This is a very challenging problem, since finding the largest clique is NP-hard to approximate even within an $O(n^{1-\epsilon})$ factor for any $\epsilon > 0$. There are several implementations for finding maximal cliques in parallel by careful partitioning, pruning and backtracking heuristics [19, 57, 71, 218, 269]. Our results do not extend to the clique enumeration problem.

Finally, anomaly detection is a broad topic, and there has been some work on parallel algo-

rithms [221]. Our work is most related to the approach known as graph scan statistics, which involves finding connected subgraphs that optimize specific functions that model underlying processes about the data. While there exist a number of heuristics, the methods that we propose in Chapter 3 give the first rigorous methods for optimizing most scan statistics using the color coding technique. However, all these methods are sequential.

6.8 Conclusions

State of the art parallel algorithms for various subgraph detection problems are based on the color coding technique, which yields algorithms with running time and space complexity proportional exponential on a solution size k. Here, we have presented algorithms based on a more recent technique from the parameterized complexity literature, multilinear detection. This methodology gives us improved bounds on memory and time over color coding. We propose an MPI algorithm for multilinear detection for general polynomials, and we show applications to two important problems, k-path and anomaly detection via scan statistics. We also show that finding a partitioning with minimum cost for the problem discussed here is NP-Hard, and we leave the development of partitioning heuristics as a topic of future work.

Chapter 7

Dense Subgraph Mining in Signed Networks

7.1 Introduction

A common problem that arises in various network analysis tasks is that of dense subgraph mining. An example from computational biology is the problem of finding regulatory motifs [83], which can be formulated as finding a dense subgraph. In social network analysis, dense subgraphs have been used for discovering stories in Twitter [17] and improving the throughput of content shown to users in social-networking sites [91]. [17], and community detection [230]. One area where dense subgraph mining has had less of an impact is event and anomaly detection in networks. These problems are commonly formalized using changes in different types of network features, such as distances, density, community structure and spectral properties—see [9] for a detailed discussion on these problems in different domains, such as computer security [66,73], social networks [6], and finance and insurance [146]. Most of the previous work in this area has focused only on the simpler case of unsigned networks. However, many event detection settings require considering the more difficult case of signed networks, in which edge weights can be positive or negative [39, 58].

For unsigned networks (i.e., networks where all edge weights are non-negative), there are many notions of density, such as the average degree, edge density and triangle density [14, 23, 24, 51, 250, 251]. The two latter ones can be computed more efficiently; however, as discussed in [251], these notions might not give the densest subgraph in real networks, and they propose a different notion called the Optimal QuasiClique, which does much better at finding dense networks.

One limitation of existing methods for finding dense subgraphs is that they are restricted to unsigned networks. Even though there is extensive work on dense subgraph mining for unsigned networks, the problem has been relatively under-studied in signed networks.

Furthermore, the algorithms developed for the unsigned setting do not extend to the signed case. In this chapter, we propose methods for finding dense subgraphs in signed networks and present its application to event detection. Our contributions are summarized as follows.

1. *Formalizing density problems and event detection in signed networks*: We introduce the Generalized Optimal Quasiclique (GOQC) problem for dense subgraph mining in signed networks and show that the problem is NP-complete. We also show that event detection in network streams can be naturally formalized using GOQC.

2. *Algorithms for GOQC*: We develop an algorithm called DenSDP for GOQC using semidefinite programming (SDP) based rounding, which gives an $O(\log n)$ approximation to the optimal solution under certain conditions (see Theorem 15); in practice, the approximation factor is much better. Our method is based on the approach of [52]; however, we find that a different rounding approach based on [79] performs better in practice. Furthermore, DenSDP can be easily modified to admit additional constraints of practical interest, such as finding dense subgraphs containing a specific set of query nodes. Although DenSDP runs in polynomial time, it does not scale very well to large networks. Motivated by the low diameter of dense subgraphs, we design another algorithm, EgoScan, with significantly faster running time, by modifying DenSDP using heuristics for pruning and scanning neighborhoods (ego-networks) of bounded size.

3. *Detection of dense subgraphs in signed and unsigned networks*: We evaluate DenSDP and EgoScan in more than 20 real networks and observe that they find solutions with very high density, significantly improving on adaptations of the best prior methods for both signed and unsigned networks. With a variant of EgoScan that optimizes the edge density instead of the GOQC score, we obtain EgoScan-δ, an algorithm that gives solutions with much higher edge and triangle density than all earlier methods. The improvement in edge density over previous methods is as much as 85% and usually over 50%. These results are consistent across signed and unsigned networks in different domains. The improvement in performance is even more significant for the constrained version involving finding subgraphs containing a subset of query nodes.

4. *Event detection using GOQC*: We use our approach for event detection in three real datasets (ICEWS, Traffic, and Enron), for which we have suitable ground truth events (described in Section 7.4.4). We find that our method based on EgoScan significantly outperforms existing approaches and baseline methods in terms of the precision-recall tradeoff (by as much as 25-50% in some instances).

7.2 Preliminaries

We assume that we have a signed network stream, which is defined as a time series of signed networks $\mathcal{G} = \{G^{(1)}, G^{(2)}, \cdots, G^{(T)}\}$. The network at time t is denoted by $G^{(t)}(V, E^{(t)})$, where V is the set of nodes—this is constant across time steps—and $E^{(t)}$ is the set of edges at time t. Each edge $e = (u, v)$ in $E^{(t)}$ has a weight $w^{(t)}(u, v)$ indicating the strength of the interaction between u and v at that time step—this can be positive or negative. If e is not present at a given time instant, its weight is 0.

We focus on detecting *surprising* interactions between the nodes of a network compared to historical interactions. Let $\alpha^{(t)}(u, v)$ be the *expected weight* of the edge at time t— this is inferred from the already-observed snapshots (i.e, 1 to $t-1$). We are interested in detecting a subset of nodes of $G^{(t)}$ whose total weight is much higher than expected. We formalize this problem below.

Problem 10 (Event Detection in Signed Networks (EDSN)) *Given a signed network stream $G^{(t)}(V, E)$ and values $w^{(t)}(u, v)$ and $\alpha^{(t)}(u, v)$ for each pair of nodes $e = (u, v)$, find a subset of nodes $S \subseteq V$ that maximizes*

$$f^{(t)}(S) = \sum_{u,v \in S} (w^{(t)}(u, v) - \alpha^{(t)}(u, v)). \tag{7.1}$$

Our approach for event detection is to solve Problem 10 for each time step t. If there exists a subset S with $f^{(t)}(S)$ above a threshold level, we say that there is an event. For a single time step, Problem 10 is a generalization of the Optimal Quasiclique (OQC) problem proposed by [251]. For a fixed time step t, we drop the superscript and denote the weight and expected weight of edge $e = (u, v)$ by $w(u, v)$ and $\bar{\alpha}(u, v)$, respectively, which gives us the GOQC problem:

Problem 11 (Generalized Optimal Quasiclique (GOQC)) *Given a signed network $G(V, E)$, a weight function $w(\cdot)$ and a penalty function $\bar{\alpha}(\cdot)$, the goal is to find a subset of nodes S that maximizes $f_{\bar{\alpha}}(S) = \sum_{u,v \in S} w(u, v) - \bar{\alpha}(u, v)$.*

When we have a parameter α such that $\bar{\alpha}(u, v) = \alpha$ and $w(u, v) = 1$, for all edges $(u, v) \in E$, the above function becomes

$$\begin{aligned} f_\alpha(S) &= \sum_{u,v \in S} (w(u, v) - \alpha(u, v)) \\ &= \sum_{u,v \in S} w(u, v) - \sum_{u,v \in S} \alpha && \text{from (2)} \\ &= |E[S]| - \alpha \left(\frac{|S| \cdot (|S| - 1)}{2} \right) && \text{from (1)} \end{aligned}$$

Here, $E[S]$ denotes the edges in the subgraph induced by S. Thi is precisely the OQC function of [251] restricted to uniform α and edge weights of 1. We also consider a variant of the EDSN problem based on finding interactions that are either too high or too low compared to the expectation, as defined below.

Problem 12 (Event Detection in Signed Networks using Total Deviation (EDSN-TD)) *Given a network $G^{(t)}(V, E)$ and values $w^{(t)}(u, v)$ and $\alpha^{(t)}(u, v)$ for each pair of nodes $e = (u, v)$, find a subset of nodes $S \subseteq V$ that maximizes*

$$f^{(t)}(S) = \left| \sum_{u,v \in S} (w^{(t)}(u, v) - \alpha^{(t)}(u, v)) \right|. \tag{7.2}$$

The EDSN-TD problem differs from EDSN in that it considers the absolute value of the score. This variant is useful when we are interested in activity that is either too high or too low compared to historical observations.

We use the following definitions in the rest of the chapter. For a subset S, we define the *average degree* as $deg(S) = \frac{|E[S]|}{|S|}$, *density* as $\delta(S) = \frac{|E[S]|}{\binom{|S|}{2}}$, and triangle density as $\tau(S) = \frac{\#\text{triangles in } S}{\binom{|S|}{3}}$.

7.3 Proposed Methods

We start by observing that the GOQC problem is computationally hard in general.

Theorem 14 *The GOQC problem is NP-complete. Further, it is NP-hard to approximate the GOQC value within a factor of $O(n^{1/2-\epsilon})$.*

Proof: It is easy to see that GOQC is in NP. We now show that the k-Clique problem is polynomial-time reducible to GOQC. Let $G = (V, E)$ be an instance of the k-Clique problem. We construct an instance $(G, w, \bar{\alpha})$ of GOQC in the following manner on graph G. We define $w(u, v) = 2$ for all $e = (u, v) \in E$ and $\bar{\alpha}(u, v) = 1$ if $(u, v) \in E$, else $\bar{\alpha}(u, v) = N$, where $N > n(n-1)$. This implies that for a subset $S \subseteq V$, if there exists $u, v \in S$ with $(u, v) \notin E$, we would have $f_{\bar{\alpha}}(S) < 0$. On the other hand, if S is a clique, $w(u, v) - \bar{\alpha}(u, v) = 1$ for all $u, v \in S$. Therefore, $f_{\bar{\alpha}}(S) \geq 0$ if and only if S is a clique. Furthermore, if S is a clique, $f_{\bar{\alpha}}(S) = \binom{|S|}{2}$. Therefore, G has a clique of size at least k if and only if there is a solution in the GOQC instance $(G, w, \bar{\alpha})$ of value at least $k(k-1)/2$. This completes the proof. ■

In contrast, we note that the complexity of the OQC problem is not known [250]. In light of this hardness, we focus on approximation algorithms.

7.3.1 Algorithm DenSDP for the GOQC problem

We start with the following quadratic programming formulation for an instance of GOQC with inputs $G = (V, E)$, w and $\bar{\alpha}$.

$$\begin{aligned}
\text{(QP) max} \quad & \sum_{e=(u,s)\in E} w(u,s)\left(\frac{1 + x_u x_0 + x_s x_0 + x_u x_s}{4}\right) \\
& - \sum_{u,s\in V, u\neq s} \bar{\alpha}(u,s)\left(\frac{1 + x_u x_0 + x_s x_0 + x_u x_s}{4}\right) \\
\text{Subject to} \quad & \\
& x_u \in \{-1, 1\} \qquad \text{for all } u \in V
\end{aligned}$$

Here, each variable x_u, except for x_0, corresponds to a node $u \in V$. Lemma 21 shows that the above program solves the GOQC problem.

Lemma 21 *The program (QP) is equivalent to the GOQC problem.*

Proof: Given a set of nodes $S \subseteq V$, we obtain a feasible solution $\mathbf{x}$ to the quadratic programming problem above by setting: $x_u = 1$ for all $u \in S$, $x_0 = 1$, and $x_v = -1$ for all $v \notin S$. We observe below that the objective value of this solution $\mathbf{x}$ equals the GOQC score for this subset. For this definition, $1 + x_u x_0 + x_s x_0 + x_u x_s$ takes the value of either 4 or 0; it has the value 4 if and only if $u \in S$. Therefore, an edge $(u, s) \in E$ contributes to the first sum if and only if $u, s \in S$; also, all the $\binom{|S|}{2}$ possible pairs of nodes in S contribute to the second sum, and, thus, the value of the objective function in the quadratic program is equivalent to $f_\alpha(S)$. The converse follows by considering the set $S = \{u : x_u = x_0\}$. ■

We use a semidefinite relaxation of the problem:

$$\begin{aligned}
\text{(SDP) max} \quad & \sum_{e=(u,s)\in E} w(u,s)\left(\frac{1 + v_u v_0 + v_s v_0 + v_u v_s}{4}\right) \\
& - \sum_{u,s\in V, u\neq s} \bar{\alpha}(u,s)\left(\frac{1 + v_u v_0 + v_s v_0 + v_u v_s}{4}\right) \\
\text{Subject to} \quad & \\
& v_u^T \cdot v_u = 1 \qquad \text{for all } u \in V \\
& v_u \in \mathbb{R}^{n+1} \qquad \text{for all } u \in V
\end{aligned}$$

In this case, each v_u is an $(n + 1)$-dimensional vector constrained to have unit norm. It is easy to show that the optimal solution to this relaxation, OPT_{SDP}, is an upper bound on the optimal solution of the corresponding instance of GOQC. However, the solution to SDP

is *high-dimensional* and needs to be *rounded* in order to get a solution to QP (and therefore, GOQC). We use the rounding approach of [52]. Once we obtain a subset of nodes S' from rounding, we refine this solution by using the local search algorithm proposed by [251] for OQC. We take S' and add a node u to the set if $f_{\bar{\alpha}}(S' \cup \{u\}) > f_{\bar{\alpha}}(S')$. When no more nodes can be added, we remove a node u from S' if $f_{\bar{\alpha}}(S' - \{u\}) > f_{\bar{\alpha}}(S')$. These two steps are repeated until there is no improvement in the score. Our algorithm DenSDP is summarized in Algorithm 17.

Algorithm 17 DenSDP$(G(V, E), w, \bar{\alpha})$.

Input: Signed network $(G(V, E)$, weight function w, and penalty function $\bar{\alpha})$
Output: $S \subseteq V$, a solution to GOQC

(1) SDP Step
Construct an instance of (SDP) using G, w, $\bar{\alpha}$
Solve (SDP), obtaining a vector v_u for each $u \in V$
(2) Rounding Step
Sample $r \sim \mathcal{N}(0_{(n+1)}, I_{(n+1)\times(n+1)})$
For each i, let $z_i = v_i \cdot r/T$, where $T = \sqrt{4 \log n}$.
For each i, if $|z_i| > 1$, set $y_i = z_i/|z_i|$, else $y_i = z_i$.
For each i, set $x_i = 1$ with probability $\frac{1+y_i}{2}$ and $x_i = -1$ with probability $\frac{1-y_i}{2}$
Let $S' \leftarrow \{u | x_u = x_0\}$
$S \leftarrow$ LocalSearch$(G, w, \bar{\alpha}, S')$
return S

Theorem 15 *If $w(\cdot)$ and $\bar{\alpha}(\cdot)$ are symmetric and $\sum_{e=(u,v)} w(u, v) - \bar{\alpha}(u, v) \geq 0$, then, the set S returned by algorithm* DenSDP *satisfies $f_{\bar{\alpha}}(S) = \Omega(OPT/\log n)$.*

Proof: The program (SDP) is equivalent to maximizing

$$\phi(v_0, \ldots, v_n) = \sum_{e=(u,s)\in E} w(u, s) \left(\frac{v_u v_0 + v_s v_0 + v_u v_s}{4} \right) - \sum_{u,s\in V, u\neq s} \bar{\alpha}(u, s) \left(\frac{v_u v_0 + v_s v_0 + v_u v_s}{4} \right).$$

We assume that $V = \{1, \ldots, n\}$, and we consider an $(n + 1) \times (n + 1)$-dimensional matrix $A = (a_{ij})$ defined in the following manner: (1) for all $u \in V$, $a_{0u} = a_{u0} = \sum_{v\in N(u)} w(u, v)/4 - \sum_{s\in V} \bar{\alpha}(u, s)/4$, (2) for all $(u, s) \in E$, $a_{us} = (w(u, s) - \bar{\alpha}(u, s))/4$, (3) for all $u, s \in V$ such that $(u, s) \notin E$, $a_{us} = -\bar{\alpha}(u, s)/4$, (4) for all $u \in V$, $a_{uu} = 0$. Then, maximizing the function $\phi(\cdot)$ above is equivalent to maximizing $\sum_{ij} a_{ij} v_i \cdot v_j$, with the matrix A having zeros on all diagonal entries—this corresponds precisely to the SDP formulation of [52]. Therefore, for $T = \sqrt{4 \log n}$, it follows that $\sum_{ij} a_{ij} x_i x_j = \Omega(OPT_{SDP}/\log n)$, where $x \in \{-1, +1\}^{n+1}$ is the

vector resulting from the rounding step in Algorithm 17, and OPT_{SDP} denotes the optimum SDP objective value. Since $\sum_e w(e) - \bar{\alpha}(e) \geq 0$ and $OPT_{SDP} \geq OPT$, it follows that this solution x gives an $O(\log n)$ approximation to (QP). Finally, by Lemma 21, $f_{\bar{\alpha}}(S)$ equals the value of (QP), and the theorem follows. ■

Remark. We note that the condition $\sum_{e=(u,v)}(w(u,v) - \bar{\alpha}(u,v)) \geq 0$ in Theorem 15 does not imply that the solution is trivially the entire graph. Consider a graph $G(V,E)$ with $V = \{1,2,3,4,5\}$, where $\{1,\cdots,4\}$ form a clique while 5 is an isolated node. Also, assume a fixed $\alpha = 1/3$ for all edges. In this case, $f_\alpha(V) = 6-(1/3)(10) = 2.6667 > 0$, but the optimal solution is given by the clique $S = \{1,2,3,4\}$ and has value $f_\alpha(S) = 6 - (1/3)(6) = 4$.

7.3.2 Alternative rounding approach

An alternative rounding approach is based on [79]. The $(n+1)$-dimensional vector r is chosen as before, so that each component is normally distributed. We include a node u in S if and only if the corresponding vector v_u satisfies: $sgn(v_0^T \cdot r) = sgn(v_u^T \cdot r)$, where $sgn(\cdot)$ is the sign function. The algorithm DenSDP-FW is summarized in Algorithm 18, and performs much better than DenSDP in our experiments.

Algorithm 18 DenSDP-FW$(G(V,E), w, \bar{\alpha})$.

Input: Signed network ($G(V,E)$, weight function w, and penalty function $\bar{\alpha}$)
Output: $S \subseteq V$, a solution to GOQC

(1) SDP Step
Construct an instance of (SDP) using G, w, and $\bar{\alpha}$
Solve (SDP), obtaining a vector v_u for each $u \in V$
(2) Rounding Step
Sample $r \sim \mathcal{N}(0_{(n+1)}, I_{(n+1)\times(n+1)})$
Let $S' \leftarrow \{u | sgn(v_0^T \cdot r) = sgn(v_u^T \cdot r)\}$
$S \leftarrow$ LocalSearch$(G, w, \bar{\alpha}, S')$
return S

7.3.3 EgoScan: A scalable SDP-based approach

Though Theorem 15 gives a rigorous guarantee for algorithm DenSDP, and it runs in polynomial time, it does not scale very well. Using the observation that dense subgraphs typically have low diameter, we propose EgoScan, a local-search approach that divides a large network into subgraphs of small size for which we can run DenSDP quickly. For a node u, we define G_u^d to be the subgraph induced by u and its neighbors within d hops (i.e., the d-neighborhood of u). We also refer to this subgraph as the *ego network* of u. Given a graph G and a parameter d, our algorithm EgoScan iterates over all the nodes of G; for

each node u, we solve GOQC on G_u^d, which is much smaller than the entire graph. Our final solution is the subgraph with highest GOQC score over all the G_u^d networks; as before, we use local search to refine the solution. EgoScan is described in detail in Algorithm 19. In the pseudocode, $UB(G_u^d)$ is an upper bound on the GOQC score of G_u^d, which we describe below.

Algorithm 19 EgoScan$(G(V, E), w, \bar{\alpha}, d)$.

Input: Signed network $(G(V, E)$, weight function w, penalty function $\bar{\alpha})$, and parameter d
Output: $S \subseteq V$, a solution to GOQC

```
Let S' ← ∅
for u ∈ V do
    Compute G_u^d
    if UB(G_u^d) ≥ f_ᾱ(S) then
        Let S_u ← DenSDP-FW(G_u, w, ᾱ)
        if f_ᾱ(S_u) > f_ᾱ(S) then
            S' ← S_u
        end if
    end if
end for
S ← LocalSearch(G, w, ᾱ, S')
return S
```

Pruning and parallelization. We can further speed up EgoScan by reducing the number of calls to DenSDP. Given the subgraph G_u^d for a node u, the sum of the weights of the positive edges gives an upper bound on the optimal GOQC score for G_u^d. We call this upper bound UB. In our algorithm, if the UB value for G_u^d is less than the best solution seen so far, we can discard G_u^d without any loss in quality. As we show in our experiments, pruning helps significantly in removing large portions of the search space. Moreover, each ego network can be processed independently, so that the algorithm can be parallelized easily.

7.3.4 GOQC With Membership Constraints

A natural extension of the GOQC problem is the *Constrained Generalized Optimal Quasi-Clique (CGOQC)*. The objective here is to find a solution that contains a specific set of query nodes.

Problem 13 (Constrained Generalized Optimal Quasiclique (CGOQC)) *Given a signed network $G(V, E)$, a weight function $w : E \to \mathbb{R}$, a penalty function $\bar{\alpha} : E \to \mathbb{R}$, and a subset of nodes $Q \subset V$, the goal is to find a subset of nodes $S \supseteq Q$ that maximizes $f_{\bar{\alpha}}(S) = \sum_{u,v \in S} w(u, v) - \bar{\alpha}(u, v)$.*

Algorithms DenSDP and EgoScan can be naturally modified to solve the CGOQC problem by adding a constraint $x_u x_0 = 1$ for all $u \in Q$ to the program (QP). Intuitively, the constraint forces the node to be in the solution returned by the quadratic program. The program (QP) can be rewritten as follows:

$$\begin{aligned}
\text{Maximize} \quad & \sum_{e=(u,s)\in E} w(u,s)\left(\frac{1+x_u x_0 + x_s x_0 + x_u x_s}{4}\right) \\
& - \sum_{u,s\in V, u\neq s} \bar{\alpha}(u,s)\left(\frac{1+x_u x_0 + x_s x_0 + x_u x_s}{4}\right) \\
\text{Subject to} \quad & \\
& x_u x_0 = 1 \qquad \text{for all } u \in Q \\
& x_u \in \{-1, 1\} \qquad \text{for all } u \in V
\end{aligned}$$

As before, we obtain a semidefinite relaxation to the problem, which we call C-SDP.

(C-SDP)

$$\begin{aligned}
\text{Maximize} \quad & \sum_{e=(u,s)\in E} w(u,s)\left(\frac{1+v_u v_0 + v_s v_0 + v_u v_s}{4}\right) \\
& - \sum_{u,s\in V, u\neq s} \alpha\left(\frac{1+v_u v_0 + v_s v_0 + v_u v_s}{4}\right) \\
\text{Subject to} \quad & \\
& v_u^T \cdot v_u = 1 \qquad \text{for all } u \in V \\
& v_u^T \cdot v_0 = 1 \qquad \text{for all } u \in Q \\
& v_u \in \mathbb{R}^{n+1} \qquad \text{for all } u \in V
\end{aligned}$$

Our algorithm for CGOQC is analogous to Algorithm 17, but we solve (C-SDP) instead of (SDP). The rounding step remains unchanged, and we obtain the following approximation guarantee.

Theorem 16 *If $w(\cdot)$ and $\bar{\alpha}(\cdot)$ are symmetric, and $\sum_{e=(u,v)} w(u,v) - \bar{\alpha}(u,v) \geq 0$, then, Algorithm 17 finds a solution S for the CGOQC problem, satisfying $f_{\bar{\alpha}}(S) = \Omega(OPT/\log n)$ in polynomial time in n, for any given query set Q.*

7.4 Experiments

Our experimental analysis addresses the following questions:

1. Detection of dense subgraphs in signed and unsigned networks. Does EgoScan find dense subgraphs? Are the results consistent across signed and unsigned networks in different domains? How does EgoScan compare to existing methods?

2. Approximation guarantee in practice. How far from optimal is the DenSDP solution in real networks, compared to the worst case bound of $O(\log n)$ we prove in Theorem 15? (Section 7.4.1).

3. Comparing DenSDP and EgoScan. How much speedup does EgoScan give us over DenSDP? How effective is the pruning in reducing the search space? (Section 7.4.3).

4. Event Detection Performance. What is the precision-recall tradeoff of our methods for event detection in real datasets? How does it compare to existing methods? (Section 7.4.4).

5. Constrained GOQC. What is the quality (defined below) of our SDP-based algorithm for CGOQC? How does it compare with existing methods? (Section 7.4.6).

7.4.1 Finding Dense Subgraphs

We compare EgoScan to the greedy algorithms proposed in [251], which we call Greedy and LS below. There are no algorithms for dense subgraph mining in signed networks. Therefore, for comparison purposes, we use the local search algorithm from [251]. To select the seed set, we find a node v that maximizes the average degree of the ego-network of v: $\frac{\sum_{u,s \in ego(v)} w(u,s)}{|ego(v)|}$, where $ego(v)$ is the set of nodes in the ego-network of v.

Table 7.1 shows our results on real-world networks from different domains, using a uniform penalty of $\alpha = 1/3$—as justified by [251]—for all the edges in most of the graphs, with the exceptions discussed there. Networks marked with an asterisk (*) are signed and networks marked with two asterisks (**) are signed with non-uniform α. This last set of networks and the penalties used are described in Section 7.4.4 below. We also show results for a variant of EgoScan in which the algorithm returns the subgraph with highest density instead of highest GOQC score; we call this variant EgoScan-δ.

We report the size of the subgraph found ($|S|$), its density (δ), triangle density (τ), and GOQC objective (f_α). We find that, in most cases, EgoScan discovers subgraphs of higher density and similar size compared to Greedy and LS. EgoScan-δ is able to find subgraphs with notably higher density, more than 0.70 for most networks. The difference in triangle density is even more pronounced, with EgoScan-δ achieving scores above 0.70 in most cases, even in large instances.

Table 7.1: **Optimal quasicliques extracted from real networks by greedy methods and our SDP-based algorithms.** We report the size ($|S|$) of the discovered subgraph, its density (δ), triangle density (τ), and GOQC objective (f_α). EgoScan and EgoScan-δ are able to find denser subgraphs than existing greedy methods (δ). These results are consistent over networks of different sizes and different domains. One asterisk (*) denotes that the network is signed. Two asterisks (**) denote that the network is signed and has different penalties for each edge; otherwise, the penalty is 1/3 for all edges.

	$\lvert S\rvert$				δ				τ				f_α			
	Greedy	LS	EgoScan	EgoScan-δ	Greedy	LS	EgoScan	EgoScan-δ	Greedy	LS	EgoScan	EgoScan-δ	Greedy	LS	EgoScan	EgoScan-δ
Dolphins	12	9	8	5	0.50	0.64	0.64	1.00	0.13	0.26	0.38	1.00	11.0	11.0	11.0	6.7
Polbooks	14	17	15	7	0.63	0.58	0.58	0.90	0.24	0.21	0.25	0.74	26.7	33.7	33.67	12.0
Adjnoun	15	12	15	4	0.50	0.58	0.48	0.83	0.13	0.20	0.13	0.50	18.0	16.0	18.0	3.0
Football	9	9	9	9	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	24.0	24.0	24.0	24.0
Jazz	57	59	50	30	0.55	0.54	0.62	1.00	0.23	0.24	0.33	1.00	348.0	351.7	354.67	290.0
Celeg. M	26	27	28	7	0.57	0.55	0.54	0.95	0.22	0.21	0.19	0.86	76.7	77.0	77.0	13.0
Wiki-Vote	132	133	130	60	0.48	0.48	0.48	0.59	0.13	0.13	0.13	0.24	1265.0	1250.0	1264.0	451.0
ca-AstroPh	57	81	98	56	1.00	0.75	0.62	1.00	1.00	0.51	0.34	1.00	1064.0	1357.0	1377.67	1026.7
AS-24july06	63	63	58	6	0.52	0.52	0.52	0.93	0.17	0.17	0.18	0.80	372.0	372.0	372.0	9.0
email-Enron	111	106	96	8	0.48	0.50	0.51	0.96	0.14	0.15	0.18	0.89	890.0	914.0	902.0	17.7
web-Google	104	66	66	17	0.48	0.85	0.85	1.00	0.22	0.64	0.64	1.00	769.7	1103.0	1103.0	90.7
AS-Skitter	318	319	276	18	0.54	0.53	0.53	0.87	0.19	0.19	0.22	0.71	10200.0	10096.0	10196.0	82.0
wikiElec.ElecBs3*	-	87	103	43	-	0.46	0.49	0.63	-	0.27	0.32	0.72	-	480.0	804.0	272.0
soc-sign-Slashdot081106*	-	144	144	23	-	0.55	0.55	0.66	-	0.43	0.43	0.64	-	2259.0	2259.0	83.67
soc-sign-Slashdot090216*	-	147	148	6	-	0.55	0.55	0.73	-	0.42	0.42	0.80	-	2329.0	2329.0	6.0
icews-countries-201407*	-	40	40	24	-	0.51	0.51	0.66	-	0.44	0.44	0.93	-	137.0	137.0	91.0
icews-countries-201412*	-	43	43	20	-	0.51	0.51	0.73	-	0.42	0.43	0.90	-	159.0	161.0	74.67
wiki-rfa-2006*	-	108	105	79	-	0.48	0.49	0.50	-	0.31	0.32	0.35	-	836.70	836.70	505.50
wiki-rfa-2012*	-	51	50	45	-	0.50	0.50	0.53	-	0.29	0.29	0.35	-	214.0	214.0	195.0
icews-brazil**	-	24	15	15	-	0.03	0.08	0.08	-	0.05	0.15	0.15	-	7.55	7.83	7.83
icews-mexico**	-	24	18	18	-	0.03	0.06	0.06	-	0.07	0.12	0.12	-	8.92	9.42	9.42
icews-venezuela**	-	17	11	11	-	0.03	0.09	0.09	-	0.07	0.19	0.19	-	4.67	5.17	5.17

7.4.2 Approximation guarantee for DenSDP in practice

We compare the objective value $f_\alpha(S_{app})$ of the solution S_{app} from DenSDP with the *fractional* SDP solution, OPT_{SDP} in Table 7.2 for some of the networks considered in Table 7.1. This ratio $f_\alpha(S_{app})/OPT_{SDP}$ is a lower bound on the approximation quality of DenSDP, which is shown to be $\Omega(1/\log n)$ in Theorem 15. We observe that the ratios are at least 0.65 and above 0.70 for most networks, which implies that the algorithm performs better than the theoretical worst case bound.

Table 7.2: **Empirical approximation ratio.** We show the ratio $f_\alpha(S_{app})/OPT_{SDP}$, where S_{app} is the solution found by DenSDP, and OPT_{SDP} is the fractional SDP objective value. DenSDP finds solutions that have GOQC score *at least* 0.65 times the score of the optimal; in most cases, the approximation is above 0.70.

Dataset	S_{app}/S_{SDP}
Dolphins	0.70
Polbooks	0.70
Adjnoun	0.79
Football	0.65
Jazz	0.88
Celegans M.	0.82

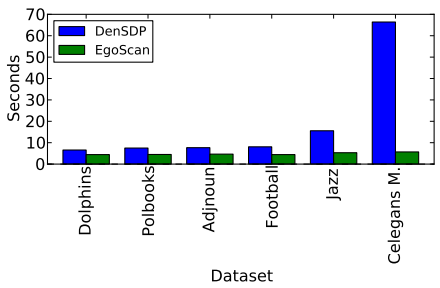


Figure 7.1: Running time of DenSDP and EgoScan for various datasets. For EgoScan, we plot the average time to process the ego networks of all nodes (without considering the pruned neighborhoods). The average running time for EgoScan remains almost constant for increasing graph sizes.

7.4.3 Speedup from EgoScan

We compare DenSDP and EgoScan in terms of scalability. In Figure 7.1, we show the running time of both algorithms for different instances. In the case of EgoScan, we plot the average time to solve the semidefinite program for one ego network. We see that for different instances the average time to evaluate an ego network remains almost constant, which allows EgoScan to process larger instances much faster, especially when we add pruning and parallelization, as discussed in Section 7.3. We also find that the pruning step is able to discard at least 50% of the search space for EgoScan in most cases, and more than 99% in some networks, such as Wiki-Vote.

We now analyze the pruning power of the UB upper bound. Table 7.3 reports the number of nodes of the networks we evaluated and the number of ego networks that are discarded. Even in the small networks, we are able to discard at least 50% of the search space for EgoScan, except for the Football network. In the larger networks (Wiki-Vote and below), we are able to prune more than 99% of the search space.

7.4.4 Event Detection

7.4.4.1 Datasets

We use the EDSN and EDSN-TD problems from Section 3.2 for event detection on real-world network time series. A summary of the datasets is provided in Table 7.4.

ICEWS. The Integrated Crisis Early Warning System (ICEWS) [88] is a dataset of political events around the world, automatically extracted from news sources. Every entry in the dataset corresponds to an interaction between two social *actors*. An actor may be as general as a country or an ethnic group, or as specific as a particular person. Each actor falls in one

Table 7.3: **Number of ego networks pruned using the UB upper bound.** With UB, we discard most of the search space. In the larger graphs, we are able to discard more than 99% of the network.

Dataset	**Total nodes**	**Pruned**
Dolphins	62	31
Polbooks	105	68
Adjnoun	112	76
Football	115	15
Jazz	198	123
Celegans M.	453	391
Wiki-Vote	7,115	6,872
ca-AstroPh	18,772	18,474
AS-24july06	22,602	22,527
email-Enron	36,692	36,430
web-Google	875,713	874,159

of 32 classes[1] defined in the CAMEO coding convention [89]. Additionally, each interaction has an *intensity* score. The score ranges from -10 to $+10$, where negative numbers indicate increasingly hostile events, and positive numbers indicate increasingly cooperative events. We use ICEWS to detect protest events in Latin American cities. We choose three Latin American capitals: Brasilia (ICEWS Brazil), Mexico City (ICEWS Mexico), and Caracas (ICEWS Venezuela). For each city, we build signed networks where the nodes are the 32 CAMEO actor classes. In a given week, an edge $e = (u, v)$ has weight $w^{(t)}(u, v) = c^{(t)}(u, v)$, where $c^{(t)}(u, v)$ is the number of events between nodes u and v. The historical weight, $\alpha^{(t)}(u, v)$, is the average number of events per week in a recent time window: $\alpha^{(t)}(u, v) = \sum_{i=t-W}^{t-1} \frac{c^{(i)}(u,v)}{W}$. We use $W = 12$ below, but we obtain similar results for $W = 8$ and $W = 16$.

For the evaluation, we use the Gold Standard Report (GSR) dataset presented in [207]. The GSR is a compilation of civil unrest events in 10 Latin American countries. This dataset also has information about which events are considered *surprising* or unexpected protests. For the evaluation, our goal is to correctly identify if there is an unexpected protest at a given week.

Traffic. We use the highway network of Los Angeles County, California[2] and its activity on May, 2014. Nodes in the graph correspond to sensors on the highway that measure the average speed and number of vehicles on the road. The sampling rate is 5 minutes, but we aggregate the data to intervals of 30 minutes. An edge in the graph represents a highway segment between two sensors. Our goal is to detect traffic congestion in this network. At time t, an edge $e = (u, v)$ has weight $w^{(t)}(u, v) = -s^{(t)}(u, v)$, where $s^{(t)}(u, v)$ is be the average speed recorded by sensors u and v at time t. The historical weight for

[1]Some example of these classes are GOV (Goverment), CVL (Civilian), MED (Media)

[2]http://pems.dot.ca.gov/

the edge is the average speed on (u, v) at the same time of the day in the last W days: $\alpha^{(t)}(u, v) = \sum_{i \in \{t-48W, t-48(W-1), \ldots, t-48\}} -\frac{s^{(i)}(u,v)}{W}$. Intuitively, an edge with a positive value indicates that the speed observed at the current timestamp is lower than in previous history, which is a signal of a traffic accident in the corresponding highway segment.

The ground truth consists of reports emitted by the California Highway Patrol in May 2014 (collected from the PEMS website). As events of interest, we consider the subset of congestion-sensitive events corresponding to traffic collisions, car fires, hit-and-run reports, wrong-way driver incidents, and closure of roads. We ignored other types of events in the data, such as traffic hazards, which are not likely to cause traffic congestion. Often accidents result in slow movement of traffic. By our choice of edge weights, dense subgraphs correspond to parts of the network with unexpectedly slow traffic. In a time window, there can be more than one accident at different parts of the network, so we find the top-30 densest subgraphs in each timestamp. We say that we detect an event if the reported location is within a radius of 2 miles of our dense subgraphs and occurred on the same 30 minute period. In figure 7.3, we show the precision-recall tradeoff for $k = 1$ to 30.

Enron. The Enron corpus[3] consists of the email directories of 151 Enron employees from May 1999 and July 2002 [133]. We consider each employee as a node in the graph; in a given week t, there is an edge between two employees if they exchanged emails during that week. The weight of the edge is the number of exchanged emails, $c^{(t)}(u, v)$, and the penalty is given by $\alpha^{(t)}(u, v) = \sum_{i=t-W}^{t-1} \frac{c^{(i)}(u,v)}{W}$. For the ground truth, we use a timeline of important events related to the Enron corporation[4].

Table 7.4: **Datasets used in our event detection experiments.**

Dataset	Nodes	Edges	Timestamps	Resolution
ICEWS Brazil	32	120,203	48	1 week
ICEWS Mexico	32	120,203	48	1 week
ICEWS Venezuela	32	120,203	48	1 week
Traffic	1870	2,965,584	1,488	30 minutes
Enron	151	7,444	162	1 week

7.4.4.2 Performance Evaluation

Our method outputs, for each timestamp, the subgraph with highest GOQC score found and the score of this subgraph. These scores induce a ranking of the likelihood of an event at each time t (i.e. higher scores indicate higher evidence for an event of interest). For our

[3]https://www.cs.cmu.edu/%7Eenron/

[4]http://www.agsm.edu.au/bobm/teaching/BE/Enron/timeline.html

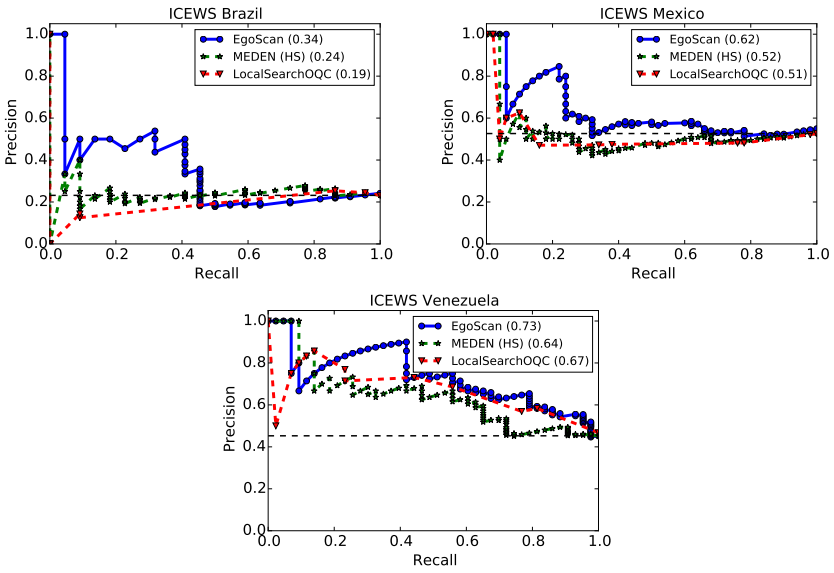


Figure 7.2: **Precision-Recall plots for ICEWS.** The number in parenthesis is the area under the curve for each method. Our algorithm achieves higher precision at the same levels of recall compared to existing methods and a baseline.

quantitative evaluation, we use the scores to generate a precision-recall curve for each dataset and compute the area under the curve of this plot—the average precision of the method.

To the best of our knowledge, the only methods for event detection in temporal networks that consider positive and negative weights are MEDEN [40] and its variant Netspot [178]. In both papers, the authors formulate the event detection task as the Heaviest Subgraph problem, which is based on Steiner connectivity instead of density. We compare our results to these methods. Additionally, to compare our results to existing methods based on density, we convert our datasets to unweighted networks and find the optimal quasiclique in each timestamp using the LocalSearchOQC algorithm of [251].

Figure 7.2 shows the precision-recall plots for ICEWS, with the black dotted line illustrating the precision that we would obtain by flipping a fair coin (null model). For the three countries that we consider, our algorithm achieves higher precision than the other two methods at the same level of recall. Furthermore, we point that in the case of Brazil and Mexico, both MEDEN and LocalSearchOQC have performance close to the null model for most levels of recall, whereas our method shows significant precision.

Figure 7.3 shows the precision-recall results for the Traffic and Enron datasets. For the Traffic dataset, all the methods have low absolute recall (below 0.10). A similar result is reported by [178] for this dataset. However, our algorithm improves significantly over Netspot and LocalSearchOQC. In Enron, we achieve precision well above the null model and improve over the other two models for different levels of recall.

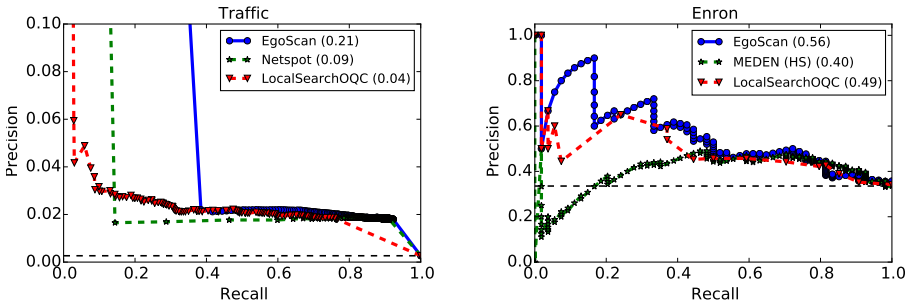


Figure 7.3: **Precision-Recall tradeoff in the traffic network.** EgoScan has better performance than the NetSpot method for temporal anomaly detection. The absolute precision and recall are low because traffic congestions can occur by events other than accidents, such as regular rush hour, sports games, concerts, etc.

7.4.5 Qualitative Analysis

ICEWS. During February 2014, Venezuela was in a state of heightened civil unrest due to lack of public safety and general dissatisfaction with the government of president Nicolas Maduro. Figure 7.4 shows a time series of the number of events in Caracas, Venezuela from December 15, 2013 to March 10, 2014. The dense subgraphs found by EgoScan are able to capture this increase in protest activity. Figure 7.5 shows the dense subgraph reported in the week of February 17, 2014. This week has many interactions between actors. In particular, CVL (civilians), GOV (Government), and OPP (opposition) are connected to almost every other node in the subgraph. In contrast, the same set of nodes in the previous month is disconnected.

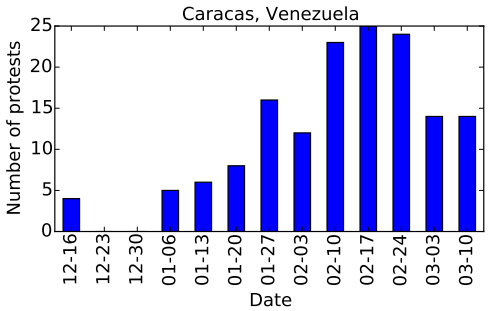


Figure 7.4: **Timeseries of protests in Caracas, Venezuela.**

Enron. In August 14, 2001, Enron CEO Jeff Skilling anounced his resignation from the company. One week prior to this event, the densest subgraph reported by EgoScan shows increased email activity among Enron officials, including Kenneth Lay (former Chairman and successor of Skilling as CEO), Dave Delainey (Chairman and CEO after Lay), Greg Whalley (former president and COO), Louise Kitchen (president of Enron Online), Mark Haedicke

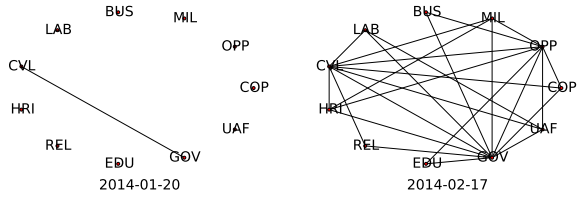


Figure 7.5: **Dense subgraph for February 17, 2014 in ICEWS Venezuela (right) compared with the corresponding graph from the prior month (left).** The increased activity between the actors during February is consistent with nationwide protests in Venezuela.

(Managing Director of Enron Wholesale), and Steven Kean (VP and Chief of Staff), among others.

Traffic. We computed the number of times that each node appears in a dense subgraph reported by our algorithm. We find that the nodes that are most often reported correspond to the intersection of Freeways 710 and 105 in the highway system—near Lynwood, CA. This road is a known hotspot for truck accidents, with 5.8 accidents per mile per year reported in 2015[5].

7.4.6 Constrained GOQC

Lastly, we evaluate our algorithm for constrained GOQC in different instances. For each instance, we generate a random query set of 5 to 7 nodes. As in Section 7.4.1, we compare our method to the local search algorithm proposed in [251] using the same four criteria. Results are reported in table 7.5. We notice that DenSDP finds subgraphs with notably higher GOQC score and density. The difference between the two algorithms is more pronounced in this variant compared to the unconstrained GOQC problem.

7.5 Related Work

We divide the related work into three categories: signed networks, dense subgraph mining, and graph-based event detection. Table 7.6 shows a comparative summary of our work with existing methods and formulations.

Signed Networks. In a signed network, edge weights are positive or negative, representing friendship or conflict, respectively. [106] proposed a theory of balance in relationships using

[5]http://www.latimes.com/local/california/la-me-california-commute-20150602-story.html

Table 7.5: **Performance of DenSDP for the constrained GOQC problem.** DenSDP finds subgraphs with GOQC score up to 2 times better than a local search algorithm. The edge and triangle density are also better for all network instances.

	$\|S\|$		δ		τ		f_α	
	LS	DenSDP	LS	DenSDP	LS	DenSDP	LS	DenSDP
Dolphins	6	13	0.13	0.31	0.00	0.07	-3.0	-2.0
Polbooks	12	16	0.41	0.42	0.14	0.16	5.0	11.0
Adjnoun	15	14	0.37	0.38	0.08	0.09	4.0	4.67
Football	14	14	0.54	0,54	0.21	0.21	18.67	18.67
Jazz	61	54	0.47	0.54	0.15	0.26	241.0	300.0
Celeg. M	30	30	0.39	0.41	0.12	0.13	25.0	33.0

Table 7.6: **Comparative summary of the related work**

Category	Method	Static	Temporal	Event Detection	Dense Subgraph	Signed	Unsigned	Weighted	Theoretical guarantees	Scalable
Dense Subgraph Mining	Densest Subgraph [23,51,95]	✓	✗	✗	✓	✗	✓	✓	✓	✓
	Densest-k Subgraph [24,79]	✓	✗	✗	✓	✗	✓	✓	✓	✗
	Densest At-Least (At-Most) k subgraph [14]	✓	✗	✗	✓	✗	✓	✓	✓	✓
	Optimal Quasiclique [3,250]	✓	✗	✗	✓	✗	✓	✓	✗	✓
Graph-based Event Detection	OddBall [8]	✓	✗	✓	✓	✗	✓	✓	✗	✓
	GraphScope, COM2 [20,237]	✓	✓	✓	✓	✗	✓	✓	✗	✓
	Meden, NetSpot [40,178]	✓	✓	✓	✗	✓	✓	✓	✗	✓
	Graph Scan Statistics [54,203,234]	✓	✓	✓	✗	✗	✓	✓	✗	✓
Signed Networks	Community Detection [39,248]	✓	✗	✗	✗	✓	✗	✗	✗	✓
	Spectral Clustering [148]	✓	✗	✗	✓	✓	✓	✓	✗	✓
	Low-Rank Modeling [58]	✓	✗	✗	✗	✓	✗	✗	✓	✓
Our contributions	DenSDP	✓	✓	✓	✓	✓	✓	✓	✓	✗
	EgoScan	✓	✓	✓	✓	✓	✓	✓	✗	✓

signed networks, which was formalized in terms of the structural balance theory [49]. The theory captures the colloquial notions of the "the friend of my friend is my friend" and "the enemy of my friend is my enemy", and it has been verified among tribal groups and countries [18, 209]. Leskovec et al. [158, 159] study variants of structural balance in social networks. Recent work has focused on link prediction [111, 165, 242], community detection [244, 248], and clustering [58,148]. Instead, we consider density and event detection problems. We refer the reader to [243] for a survey on signed networks.

Dense Subgraph Mining. Many formulations for finding dense subgraphs have been proposed, and we briefly touch upon them here. We refer the reader to [250, 251] for more details. In the *Densest Subgraph* problem, we are given a graph $G(V, E)$, and the goal is to find a subset of nodes $S \subseteq V$, such that the *average degree* of the graph induced by S, $\frac{E(S)}{|S|}$ is maximized. This problem can be solved in polynomial time using Goldberg's flow-based algorithm [95]. There is also a linear-time greedy algorithm that yields a $\frac{1}{2}$-

approximation [23, 51]; in practice, this algorithm finds subgraph with average degree close to optimal. In real-world networks, subgraphs with maximal average degree have been found to be large —sometimes trivially spanning the entire node set V— and not very dense [251]. When we want to control the size of the subgraphs discovered, we can add a constraint k to the densest subgraph formulation. In the *Densest-k Subgraph* problem [80], the goal is to find a subset S of size k with maximum number of edges. This problem is NP-Hard. Asahiro et al. propose a $O(k/n)$ greedy approximation algorithm for any value of k [24]; for particular values of k, Feige and Langberg are able to obtain better approximations using semidefinite programming [79]. When the constraint is to find a set S of size at least k, we obtain the *Densest-at-least-k Subgraph* problem; when the constraint is $|S| \leq k$, we have the *Densest-at-most-k Subgraph* problem. Both variants (also NP-Hard) were proposed by Andersen and Chellapilla [14]. Recently, [250] proposed the *k-Clique* densest subgraph as an extension to the classical *densest subgraph* problem. In this formulation, the goal is to find a set of nodes that maximizes $\frac{G_k(S)}{|S|}$, where G_k is the number of k-cliques induced by the nodes in S. For $k = 3$, we obtain the *triangle-densest subgraph* problem; the authors show this latter formulation discovers graphs that are denser than the ones found by maximizing the average degree.

Dense subgraphs also have connections with cliques; a k-clique is the densest graph of size k. Then, an alternative way to find dense subgraphs is to look for large cliques. However, there are two big challenges with this approach. First, the clique problem cannot be approximated to a constant factor, unless P = NP. Second, the clique definition is too restrictive because all of the edges have to be present. In [251], Tsourakakis et al. proposed to the *Optimal Quasiclique* problem, where the goal is to find a set of nodes that maximizes $f_\alpha(S) = E(S) - \alpha\binom{|S|}{2}$, where α is a parameter. The authors propose a greedy method based on [51] and a local search algorithm to optimize the objective. They also show that subgraphs with a high f_α score have high edge and triangle density, and have small diameter —all of these are desirable properties for dense subgraphs.

Despite the extensive literature in graph density, the problem of finding dense subgraphs in signed networks has not been explored yet. As we discuss later, signed networks bring a new set of challenges when it comes to density problems; the current methods and formulations assume the weights of the graph are non-negative, and there is no simple way to extend these methods to the signed case.

Graph-based Event Detection. [9] gives a good survey on graph-based anomaly and event detection. They classify the different approaches in dynamic graphs depending on the graph characteristic that is used and the kind of events that are detected. One line of work formalizes anomalous patterns in the graph within a time window, which is used for identifying anomalies in the entire stream [40, 178]. Information theoretic approaches have also been proposed [20, 237]. In these formulations, anomalies are subgraphs that have high encoding cost. A related approach examines changes in community structure, e.g., [6, 198]. However, as discussed in [243], there has been limited work on event detection in signed networks.

7.6 Conclusions

We propose the GOQC problem as the first formalization of dense subgraphs in signed networks. We also find an interesting connection between the GOQC problem and event detection in signed networks. This connection leads to a window-based method for event detection for both signed and unsigned network streams. We also develop the first efficient methods with rigorous approximation guarantees and good empirical performance for the GOQC problem. Our results show that semidefinite programming based methods are able to find dense subgraphs in many different domains. These methods can be scaled up to achieve practical algorithms by the ego-network exploration that we develop here.

Part II

Dynamics on Networks

Chapter 8

Critical Sets for Epidemic Spread

8.1 Introduction

Infectious diseases constitute one of the largest causes of human mortality worldwide and account for more than 13 million deaths a year. Despite significant strides in medicine and public health practices, such as high vaccination coverage, large outbreaks continue to be an important source of concern. Further, there is limited understanding of the dynamics of infectious disease spread through human communities. For instance, despite the fact that the average vaccination rate for measles is over 95% in the United States, there were 667 cases of the disease in 2014 and 187 in 2015 according to the Centers for Disease Control and Prevention (CDC) [1]. This poses the question: are there pockets of the population that, if left unvaccinated, will spread a disease in spite of high vaccination rates elsewhere? We formalize this problem as the k-Critical Set problem, where we want to find a cluster of k nodes in a population that will maximize spread if left unvaccinated, even if the rest of the population follows a fairly strict vaccination policy. Our contributions are the following:

1. **Critical sets for epidemic spread.** We formalize the notion of *criticality* of a subpopulation S—i.e., the expected number of infections if there is an outbreak in S and its inhabitants are not properly immunized. Then, we cast the problem of finding the most critical subpopulation as a constrained optimization problem. We propose the k-Critical Set (k-CSP) problem, which tasks us with finding a connected subpopulation of bounded size (at most k) that maximizes the spread of a disease. We show that k-CSP generalizes the Influence Maximization problem of Kempe et al. [128] and is an instance of submodularity maximization with size and connectivity constraints. We also propose the k-Critical Region (k-CRP) problem, which asks us to find a geographical area with high criticality. This problem formulation more readily translates to public policy than k-CSP.
2. **Algorithms for k-CSP and k-CRP.** We propose approximation algorithms and fast

heuristics for k-CSP and k-CRP. Even though maximizing criticality is NP-Hard in general, we show conditions for which the function can be optimized in polynomial time by exploiting *local* structure. Loosely, in populations where most people are vaccinated, except for small clusters—which is the case with measles, for instance—the criticality function is *approximately modular*, and we can derive a polynomial-time algorithm with rigorous bounds. However, the resulting approximation algorithm does not scale, so we also propose a nearly linear time heuristic with good empirical performance.

3. **Experimental Results.** We evaluate our algorithm on a realistic, activity-based population and mobility model for the state of Minnesota, where we simulate an outbreak of a highly-infectious disease. We find that the sets we discover have very high criticality compared to reasonable baselines and heuristics commonly considered in the public health community.

8.2 Preliminaries

Let V denote a population, and let $G = (V, E)$ be a contact graph on which a disease can spread; that is, node $v \in V$ can propagate the disease to its neighbors. In the social contact network datasets that we consider (Section 8.4.1.1), each person is associated with a geographical location corresponding to his/her home address. This allows us to consider a spatial decomposition $\mathcal{R} = \{R_1, \ldots, R_N\}$ of the entire region; the entire region could be, for instance, a state or province, and each R_i could be a block group or a census tract. We let $V(R_i)$ denote the set of people who live in the block group R_i. Let $n_{R_i}(t)$ denote the number of nodes infected in R_i at time t. Then, the sequence $(n_{R_i}(t), t = 0, 1, \ldots)$ is the *epicurve* restricted to region R_i. For a subset $R \subset \mathcal{R}$, we define $V(R) = \cup_{R_i \in \mathcal{R}} V(R_i)$; the quantity $n_R(t)$, and the epicurve restricted to R are also defined accordingly.

For the diffusion process governing the disease, we consider an SIR model of epidemic spread [131], where a node is in one of *three* states: Susceptible (S), Infected (I), Recovered/Removed (R); this approach is commonly used for a number of diseases, including flu, smallpox, and measles (see, e.g., [171,190]). An infected node v (i.e., in state I) spreads the infection to each susceptible neighbor $u \in N(v)$ with certain probability $p(u, v)$, which is referred to as the *transmissibility*—this is usually measured in terms of probability of infection per unit time. The infections spread independently from all the neighbors. An infected node v stays in the state I for an "infectious duration", and then moves to the R state. This is the simplest model, and many other variations, such as SEIR (Susceptible–Exposed–Infected–Recovered) and SIS (Susceptible–Infected–Susceptible) exist [171, 190].

We focus on vaccination interventions for controlling epidemic spread. Let $\mathbf{x}$ denote an intervention vector, where $x_i \in [0, 1]$ denotes the probability that node i is vaccinated. Let $\#\text{inf}(\mathbf{x})$ denote the expected number of infections for the intervention $\mathbf{x}$, with respect to some specific initial conditions. Different kinds of initial conditions are possible; the most

common one, which we assume here, is a random initial infection.

8.2.1 Critical Sets

For an intervention $\mathbf{x}$, let $\mathbf{x}^S$ denote the corresponding intervention where a subset $S \subset V$ of nodes is not vaccinated, and the remaining nodes being vaccinated with the same probability as in $\mathbf{x}$; that is $\mathbf{x}_i^S = \mathbf{x}_i$ for $i \notin S$ and $\mathbf{x}_i^S = 0$ for $i \in S$.

We define the *criticality* of a set $S \subset V$ with respect to an intervention $\mathbf{x}$ as

$$\text{crit}(S, \mathbf{x}) = \#\text{inf}(\mathbf{x}^S) - \#\text{inf}(\mathbf{x}),$$

which is the expected number of extra infections that occur if S is not vaccinated and there is an initial infection in S. We are interested in finding sets with high criticality.

Problem 14 (k-Critical Set Problem (k-CSP)) *Given a social contact network $G = (V, E)$, disease model $p(\cdot)$, intervention $\mathbf{x}$, and a parameter k, the objective is to find a set $S \subset V$ of size at most k that has the largest criticality:*

$$S = argmax_{S' \subset V, |S'| \leq k} crit(S', \mathbf{x})$$

The size constraint k in k-CSP is important because, without it, the subset $S = V$ would end up maximizing the criticality. Furthermore, only small critical sets are of practical interest. The size constraint is important because interventions and remedial action require site visits. When there is under-vaccination, a public health agency combats it by running immunization campaigns in the area; with limited time and human resources, connectivity helps to focus on a small part of the network. Additionally, the connectivity constraint helps to find subgraphs that are easier to interpret. A public health agency could correlate low vaccination with demographic properties or infrastructure in the target region—e,g., income or access to health services—which would be more difficult to do if the discovered nodes were disconnected and scattered over the entire graph.

For an intervention $\mathbf{x}$ and subset $R \subset \mathcal{R}$ of regions, we define $\mathbf{x}^R = \mathbf{x}^{V(R)}$ to be the intervention where the nodes in the set $S = V(R)$ do not get vaccinated. Our interest is in finding high-risk sets that are located within a small spatial region. Therefore, we define the *criticality* of a set $R \subset \mathcal{R}$, with respect to an intervention $\mathbf{x}$ as

$$\text{crit}(R, \mathbf{x}) = \#\text{inf}(\mathbf{x}^R) - \#\text{inf}(\mathbf{x}),$$

which is the expected number of extra infections that might be caused if the nodes in the regions R are not vaccinated.

Problem 15 (k-Critical Region Problem (k-CRP)) *Given a social contact network $G = (V, E)$, regions $\mathcal{R}$, disease model $p(\cdot)$, intervention $\mathbf{x}$, and a parameter k, the objective is to find a set $R \subset \mathcal{R}$ of size at most k that has the largest criticality:*

$$R = argmax_{R' \subset \mathcal{R}, |R'| \leq k} crit(R', \mathbf{x})$$

8.2.2 Submodularity

A set function $f : 2^V \to \mathbb{R}$ is said to be *submodular* if it satisfies the diminishing returns property: for any $T \subset S \subset V$ and $x \in V \setminus S$, we have that

$$f(T \cup x) - f(T) \geq f(S \cup x) - f(S).$$

That is, the marginal gain of adding x to a set is larger on a smaller context.

Submodularity has become popular in recent years in machine learning [141] because 1) the diminishing returns property applies naturally to many problems of interest (e.g., sensor placement) and 2) there exist simple algorithms for approximate constrained submodularity maximization [189]—though the problem is NP-Hard in general. For the problem of maximizing a submodular function over subsets of size at most k, Nemhauser et al. [189] proposes the following greedy algorithm. We start with the empty set $S_0 = \emptyset$. We construct a set S_i by adding an element to S_{i-1} that yields the highest increase in objective value:

$$S_i = S_{i-1} \cup \underset{x \in V \setminus S_{i-1}}{\operatorname{argmax}} \{f(S_{i-1} \cup \{x\}) - f(S_{i-1})\}$$

The algorithm returns a set S_k as the solution, with objective value at least $1 - \frac{1}{e}$ of the optimal. We refer to this procedure as *the greedy algorithm.*

We focus on submodular function maximization with connectivity constraints.

Problem 16 *Given a graph $G = (V, E)$, a non-negative submodular function $f : 2^V \to \mathbb{R}^+$, and a parameter k, the objective is to find a set $S \subset V$ of size at most k that maximizes f.*

As we discuss in Section 8.3, the k-CSP and k-CRP problems are equivalent to Problem 16.

8.3 Proposed Methods

We start this section by establishing the relationship between k-CSP and the well-studied problem of Influence Maximization. k-CSP can be seen as a generalization of Influence Maximization with connectivity constraints. This result also shows connections between our problem and constrained submodularity maximization. In the second part of this section, we propose algorithms for k-CSP.

8.3.1 Computational Complexity

8.3.1.1 k-CSP and Influence Maximization

In the Influence Maximization (InfMax) problem [128], we are given a directed graph $G = (V, E)$ and edge weights $p(u, v) \in [0, 1]$ indicating the probability that node u influences node v. The goal is to find a set $S \subset V$ of k *seed nodes* to infect, such that the expected number of influenced nodes or *spread*, $\sigma(S)$, is maximized.

Problem 17 (InfMax) *Given a directed social network $G = (V, E)$, spread probabilities $p : E \to [0, 1]$, and a parameter k, the objective is to find a set $S \subset V$ of size at most k with the largest spread:*

$$S = argmax_{S' \subset V, |S'| \leq k} \sigma(S').$$

For the influence process, we focus on the Independent Cascade model [128], where influence propagates in discrete time steps. At time $t = 0$, the seed nodes $S \subset V$ become *active*. At any time step $t \geq 0$ in the process, an active node u tries to activate or influence each neighbor v, succeeding with probability $p(u, v)$. If any of the active nodes is successful in activating v, v becomes active at time $t + 1$. A node u can only attempt to activate its neighbors once—during the time step where u becomes active—and the process ends when no more activations are possible.

Kempe et al. [128] consider an equivalent view of the diffusion process under the Independent Cascade model that is easier to analyze. For each edge $e = (u, v)$, we flip a coin with bias $p(u, v)$. Let $X(e)$ be a Bernoulli variable that is 1 with probability $p(u, v)$; then, we define the *live* graph G' as the subgraph of G induced by the edges for which $X(e) = 1$: $G' = (V, E' = \{e \in E | X(e) = 1\})$. In the live graph, a node v is activated by the end of diffusion process if and only if v is in the same component as one of the seeds in S. Then, the spread of S in L is the union of the components of the seeds, and the spread $\sigma(S)$ is the expected value over realizations of the live graph. Formally, let $\text{comp}_L(v)$ be the set of nodes in the component of node v in live graph L; then, the spread starting from seed set S is

$$\sigma(S) = \sum_L \Pr(L) | \bigcup_{s \in S} \text{comp}_L(s) |,$$

where $\Pr(L)$ denotes the probability of obtaining the live graph L with the edge probabilities $p(u, v)$.

8.3.1.2 k-CSP generalizes InfMax

k-CSP is NP-Hard; in fact, below we show that InfMax is a special case of k-CSP. First, we note that the Independent Cascade model is a special case of the SIR process where the

infectious period is one time step for all nodes, and there is no intervention–i.e., $x_i = 0$, for all nodes $i \in V$.

Theorem 17 *The k-Critical Set problem with the SIR model is NP-Hard.*

Proof: We reduce InfMax, which is NP-Hard, to k-CSP. Suppose we are given an arbitrary instance of Influence Maximization; that is, a directed graph $G = (V, E)$, edge probabilities $p(u, v)$ for every edge $e = (u, v) \in E$, and parameter k. We will create an instance of k-CSP with graph $H = (V_H, E_H)$, disease model $p'(\cdot)$, intervention $\mathbf{x}$, and parameter $k' = k$.

We construct H starting from G—i.e., G is a subgraph of H. For every node $v \in V$, we add a node v' and directed edge (v', v). Additionally, we add a *source* node u and directed edges from u to each v'. Then, H is a graph with node set $V_H = V \cup \{v' | v \in V\} \cup \{u\}$ and edge set $E_H = E \cup \{(u, v'), (v', v) |\ v \in V\}$.

For the disease model, we let $p'(u, v) = p(u, v)$, for all $e = (u, v) \in E$. Every edge from v' to v activates v with probability 1; that is, $p(v', v) = 1$, for all $v \in V$. Similarly, $p(u, v') = 1$, for all $v \in V$.

The intervention is defined as follows. First, all the nodes in V are left unvaccinated: $x_v = 0$, for all $v \in V$. Second, all the v' nodes are vaccinated with probability 1: $x_{v'} = 1$, for all $v \in V$. Lastly, u is unvaccinated. In fact, the diffusion process starts from u.

Let S be a solution to the instance of Influence Maximization, and let $S' = \{v' | v \in S\}$ be a solution in the corresponding instance of k-CSP constructed above. We consider an SIR process on the graph H in which the infectious period is 1 time step and node u is the only node infected initally. At time $t = 0$, the process starts at node u; then, at $t = 1$, all the nodes in S' are infected deterministically, and at $t = 2$, all the nodes in S—and only the nodes in S—are infected. From here on, the SIR process in H is stochastically equivalent to the Independent Cascade model in G with seed set S. Further, the expected spread of S in G is related to the criticality of S' in H by $\sigma(S) = \text{crit}(S') - (k + 1)$. Thus, any instance of Influence Maximization under the Independent Cascade model can be viewed as an instance of k-CSP for the SIR process; this completes the proof. ■

As a corollary, we have that k-CSP generalizes the Influence Maximization problem.

8.3.1.3 k-CSP and Submodularity

Here, we establish that the criticality function $\text{crit}(S, \mathbf{x})$ is submodular. The equivalence between Problem 16 and k-CSP follows from this fact. The analysis of the submodularity of $\text{crit}(S, \mathbf{x})$ is baesd on the proof of submodularity of the spread function by Kempet et al. [128].

Lemma 22 *The criticality function crit$(S, \mathbf{x})$ under the SIR model is submodular.*

Proof: Consider a random live graph $L(V, E')$ defined as above, and let $\text{crit}_L(S, \mathbf{x})$ be the criticality of $S \subset V$ on this graph. Without loss of generality, assume there is only one initial infection and let r be the initially infected node. Then, the criticality of S can be computed as

$$\text{crit}_L(S, \mathbf{x}) = |\left(\bigcup_{s \in S} \text{comp}_L(s)\right) \cup \text{comp}_L(r)| - |\text{comp}_L(r)|.$$

In words, the number of extra infections in L due to node set S is the size of the union of the components of nodes in S minus the infections that we get if only the seed is infected—i.e., the size of $\text{comp}_L(r)$.

Now, consider two sets S, T, such that $T \subseteq S$, and a node $v \notin S$. The quantity $\text{crit}_L(T \cup \{v\}, \mathbf{x}) - \text{crit}_L(T, \mathbf{x})$ is the number of extra infections from adding node v; that is, the number of elements in $\text{comp}_L(v)$ that are not already in $(\bigcup_{s \in T} \text{comp}_L(s)) \cup \text{comp}_L(r)$. This number is at least as large as the number of extra infections that we get by adding v to the bigger set S, which gives us $\text{crit}_L(T \cup \{v\}, \mathbf{x}) - \text{crit}_L(T, \mathbf{x}) \geq \text{crit}_L(S \cup \{v\}, \mathbf{x}) - \text{crit}_L(S, \mathbf{x})$. This is precisely the definition of submodularity, so the criticality function for a fixed graph is submodular. Over random realizations of L, we obtain

$$\text{crit}(S, \mathbf{x}) = \sum_L \Pr(L) \text{crit}_L(S, \mathbf{x}).$$

This is a convex combination of submodular functions, which is known to be submodular too, completing the proof. ■

8.3.2 Finding Critical Sets

8.3.2.1 Constant-factor approximation exceeding size constraint

We derive a polynomial-time approximation algorithm for k-CSP building on the work of Krause et al. [142]. The main idea of the authors is that a submodular function on subsets of nodes of a graph can be approximately modular if the nodes are *far enough*. Formally, let $d(S_1, S_2) = \min_{u \in S_1, v \in S_2} d(u, v)$ be the distance between two subsets of nodes; then, a submodular function $F : 2^V \to \mathbb{R}$ is (r, γ)-local [142] if $F(S_1 \cup S_2) \geq F(S_1) + \gamma F(S_2)$ whenever $d(S_1, S_2) > r$, for constants $r > 0$ and $0 < \gamma \leq 1$.

In general, it is unlikely that criticality is (r, γ)-local in a social network, since an epidemic process starting at a hub node could have a far-reaching cascading effect leading to infections everywhere in the network. However, in settings where most individuals in the social network are vaccinated, the graph is likely to break into small components—i.e., size at most $O(\log n)$—so a node will have only a local effect with high probability.

Lemma 23 *Let $G = (V, E)$ be a social contact network with n nodes and maximum node degree Δ_G. Let $\mathbf{x}$ denote an intervention vector for which every node $v \in V$ is vaccinated*

with probability $x_i =\geq 1 - 1/\Delta_G$*, and let* $L = (V, E')$ *be a random live graph as defined above. Then, the size of any component in* L *is at most* $\log n$*.*

Proof: If a vertex i is vaccinated with probability x_i, all incident edges are removed. Therefore, the connected components in this process are no larger than if each edge is independently removed with probability $q = 1 - 1/\Delta_G$. Let $p = 1 - q = 1/\Delta_G$. For a vertex i, the number of incident edges Y_i is a random variable with distribution $Bin(deg(i), p)$, which is upper bounded by $Bin(\Delta_G, p)$.

The proof follows on the same lines as Theorem 5.4 of [119]. Consider a component exploration from a vertex v. At each step, we examine the neighbors of a vertex i. A neighbor j is added to the component if the edge (i, j) is picked, which occurs with probability p. Therefore, the probability that a vertex v belongs to a component of size k in the live graph L is upper bounded by the probability that the sum of k random variables $\sum_{i=1}^{k} Y_i \sim Bin(k\Delta_G, p)$ is at least $k - 1$. Let $p\Delta_G = c < 1$. By a Chernoff bound, we have

$$\begin{aligned} n \Pr[\sum_{i=1}^{k} Y_i \geq k - 1] &= n \Pr[\sum_{i=1}^{k} Y_i \geq ck + (1-c)k - 1] \\ &\leq n\exp\big(-\frac{((1-c)k-1)^2}{2ck + (1-c)k/3}\big) \\ &\leq n\exp\big(-\frac{(1-c)^2 k}{2}\big) \\ &= o(1), \end{aligned}$$

for $k \geq 3\log n/(1-c)^2$. ∎

We note that this regime is useful in practice. For example, the vaccination rate for measles is close to 100% in the United States; however, there exists connected clusters of undervaccination.

Given the above, the following lemma follows.

Lemma 24 *Let* $\mathbf{x}$ *denote an intervention vector for which every node* $v \in V$ *is vaccinated with probability* $x \geq 1 - 1/\Delta_G$*, then the criticality function* $crit(S, \mathbf{x})$ *is* $(2\log n, 1)$*-local.*

Proof: Let S_1 and S_2 be two subsets of nodes, such that $d(S_1, S_2) > 2\log n$. We examine the behavior in a random live graph L. We need to show that the criticality of $S_1 \cup S_2$ in L is

$$\mathrm{crit}_L(S_1 \cup S_2, \mathbf{x}) = \mathrm{crit}_L(S_1, \mathbf{x}) + \mathrm{crit}_L(S_2, \mathbf{x}).$$

Abusing notation, let $\mathrm{comp}_L(S) = \bigcup_{s\in S} \mathrm{comp}_L(s)$ denote the union of components of nodes in set S. Because all the connected components of L have size at most $\log n$ (Lemma 23) and $d(S_1, S_2) > 2\log n$, the components that cover S_1 and S_2 do not overlap, so

$$|\mathrm{comp}_L(S_1 \cup S_2)| = |\mathrm{comp}_L(S_1)| + |\mathrm{comp}_L(S_2)|.$$

Now, we split $\text{crit}_L(S_1 \cup S_2, \mathbf{x})$ in three separate cases depending on how $\text{comp}_L(r)$ interacts with S_1 and S_2.

Case 1: $\text{comp}_L(r)$ does not overlap with $\text{comp}_L(S_1 \cup S_2)$. For any set S, if $\text{comp}_L(S)$ does not overlap with $\text{comp}_L(r)$, then

$$\text{crit}_L(S, \mathbf{x}) = |\text{comp}_L(S) \cup \text{comp}_L(r)| - |\text{comp}_L(r)| = |\text{comp}_L(S)| + |\text{comp}_L(r)| - |\text{comp}_L(r)| = |\text{comp}_L(S)|.$$

It then follows that

$$\text{crit}_L(S_1 \cup S_2, \mathbf{x}) = |\text{comp}_L(S_1 \cup S_2)| = |\text{comp}_L(S_1)| + |\text{comp}_L(S_2)| = \text{crit}_L(S_1, \mathbf{x}) + \text{crit}_L(S_2, \mathbf{x}).$$

Case 2: $\text{comp}_L(r)$ overlaps with either $\text{comp}_L(S_1)$ or $\text{comp}_L(S_2)$, but not both. Without loss of generality, assume that the overlap is with $\text{comp}_L(S_1)$. Then,

$$\begin{aligned}\text{crit}_L(S_1 \cup S_2, \mathbf{x}) &= |\text{comp}_L(S_1 \cup S_2) \cup \text{comp}_L(r)| - |\text{comp}_L(r)| \\ &= |\text{comp}_L(S_2)| + (|\text{comp}_L(S_1) \cup \text{comp}_L(r)| - |\text{comp}_L(r)|) \\ &= \text{crit}_L(S_1, \mathbf{x}) + \text{crit}_L(S_2, \mathbf{x}).\end{aligned}$$

Case 3: $\text{comp}_L(r)$ overlaps with both $\text{comp}_L(S_1)$ and $\text{comp}_L(S_2)$. By Lemma 23, this implies that $d(S_1, \{r\}) < \log n$ and $d(S_2, \{r\}) < \log n$, but then S_1 and S_2 can be connected by r with a total distance less than $2 \log n$, which contradicts our initial assumption about $d(S_1, S_2)$. Therefore, this case cannot occur.

Since these three cases exhaustively cover the domain of $\text{crit}_L(S_1 \cup S_2, \mathbf{x})$, and the result holds for any random live graph, the proof is complete. ■

In [142], the authors propose an algorithm for maximizing an (r, γ)-local function F on edge-weighted networks, subject to a weight budget constraint. The algorithm provides a constant-factor approximation on the objective value of the solution, but it violates the budget constraint by a factor that depends on the doubling dimension of the graph, $\dim(G, E)$—which is at most $\log n$. The algorithm has the following main steps:

1. **Padded decompositions.** First, the authors partition the graph into clusters $\mathcal{C}_1, \ldots \mathcal{C}_\ell$, each of diameter at most αr, where $\alpha = 64\dim(G, E)$. If a node v and all nodes at distance at most r of v are in the same cluster, we say that v is r-padded. After clustering, all the nodes that are not r-padded are removed; this occurs with probability $1/2$ for each node, and the best solution S of size k after removal has objective value $F(S) \geq \frac{1}{2}F(S^*)$, where S^* is the optimal subgraph of size k.

2. **Greedy solution in the clusters.** The purpose of the padded decomposition was to partition the graph into small clusters where we can ignore the connectivity cost. For each cluster, the authors now run the greedy algorithm to obtain an ordering of the nodes; the first j nodes in this ordering are approximately the most informative nodes in the cluster. The greedy algorithm degrades the quality of the optimal solution by a factor of at most $(1 - 1/\epsilon)$.

3. **Connecting solutions from different clusters.** The last step is to connect solutions across clusters. The authors cast this problem as a Budgeted Steiner Tree problem [122], essentially ignoring the submodularity of F. However, because the function is (r, γ)-local, this only degrades the solution by a factor of γ, times an extra $\kappa_b = 3$ factor from the approximation guarantee of the algorithm used for Budgeted Steiner Tree [122].

We can show the following bound and running time for k-CSP.

Theorem 18 *Given a graph $G = (V, E)$, intervention vector $\mathbf{x}$, such that $x_v \geq 1 - 1/\Delta_G$, for all $v \in V$, and a size constraint k, there is an algorithm for k-*CSP *that finds a connected subgraph S with expected criticality $crit(S, \mathbf{x}) \geq (1 - \frac{1}{\epsilon})\frac{1}{6} crit(S^*, \mathbf{x})$ containing at most $2(\alpha(\log n + 1) + 1)(k - 1) = O(k \log^2 n)$ vertices. Furthermore, let t_{crit} be the time it takes to evaluate crit once; then, the algorithm has time complexity $O(n^2 + t_{crit} n^2 + mn^4 \log n)$.*

Proof: As discussed above the approximation guarantee in [142] is $F(S) \geq (1-\frac{1}{\epsilon})\frac{1}{2}\frac{1}{\kappa_b}\gamma F(S^*)$. Because crit is $(2 \log n, 1)$-local, we obtain the claimed bound.

The number of nodes used follows from the more general bound derived in [142]. In general, the algorithm there violates the budget b by a factor of at most $\alpha(r + 2) + 2$. Since we are searching for a solution of k nodes, we can run the algorithm with unit weight for all edges and set the budget to $b = k - 1$. In that case, we obtain a solution with at most $(\alpha(r + 2) + 2)(k - 1) = 64\text{dim}(G, E)(2 \log n + 2 + 2)(k - 1)$ nodes. Since the doubling dimension is at most $\log n$, we get an upper bound of $O(k \log^2 n)$.

Finally, for the running time bound, we analyze the three main parts of the algorithm separately. First, the padded decomposition requires querying the distance of arbitrary pairs of nodes in the graph, which requires $O(n^2)$ operations in the worst case. Second, running the greedy algorithm in each cluster involves at most $|\mathcal{C}_i|^2$ evaluations of the crit function; since the size of a cluster is bounded above by n, the complexity of this step is $O(t_{\text{crit}} n^2)$. Third, the algorithm for Budgeted Steiner Tree has complexity $O(mn^4 \log n)$ [87, 122]. ■

We can derive a similar result for k-CRP. For this problem, the bound on the solution size is better by a factor of $\log n$ because the doubling dimension of the graph induced by planar regions, such as block groups or census tracts, is a constant.

Theorem 19 *Given a graph $G = (V, E)$, intervention vector $\mathbf{x}$, such that $x_v \geq 1-1/\Delta_G$, for all $v \in V$, and a size constraint k, there is an algorithm for k-*CRP *that finds a connected subgraph S with expected criticality $crit(S, \mathbf{x}) \geq (1 - \frac{1}{\epsilon})\frac{1}{6} crit(S^*, \mathbf{x})$ containing $O(k \log n)$ vertices.*

8.3.2.2 $O(\sqrt{k})$ Approximation

Kuo et al. [149] propose a $O(\sqrt{k})$ approximation algorithm to Problem 16. Their approach consists of first relaxing the connectivity constraint and using the greedy algorithm to find a set S' of size at most $\lfloor\sqrt{k}\rfloor$, over the $\lfloor\sqrt{k}\rfloor$-neighborhood of every node. The authors show that it is possible to make the set S' connected by adding at most $\lfloor\sqrt{k}\rfloor$ nodes to S' to obtain a final solution S with objective value $\frac{1}{O(\sqrt{k})}$ of the optimal. This result carries over to the k-CSP and k-CRP.

Theorem 20 (Kuo et al. [149]) *Given a graph $G = (V, E)$, intervention vector $\mathbf{x}$, and a size constraint k, there is an algorithm for k-*CSP *and k-*CRP *that finds a connected subgraph S of size at most k with criticality $crit(S, \mathbf{x}) \geq \frac{1}{O(\sqrt{k})} crit(S^*, \mathbf{x})$. The algorithm has time complexity $O(n^{2.376} + n(\sqrt{k} t_{crit} n))$.*

Proof: The approximation bound follows because k-CSP and k-CRP are special cases of Problem 16. The running time comes from the main two steps of the algorithm. First, we have to find the $\lfloor\sqrt{k}\rfloor$-neighborhood of every node, which can be done in time $O(n^{2.376})$ using an all-pairs-shortest-path algorithm [220]. The second step consists of running the greedy algorithm to find a subset S' of size $\lfloor\sqrt{k}\rfloor$, which takes at most $\sqrt{k}n$ evaluations of the crit function. Finally, we have to add nodes to S' to make it connected, for which we can use the previously-computed all-pairs-shortest-path data structure, giving the claimed time complexity. ■

8.3.2.3 A Heuristic Based on Steiner Connectivity

The two algorithms shown above do not scale well. One main challenge is that both algorithms rely on the greedy algorithm, which consists of many evaluations of the crit function. In practice, this function may be costly to compute, requiring computationally expensive simulations of an SRI process. For example, the simulation engine that we use in Section 8.4 to estimate criticality in the state of Minnesota takes 30 seconds to run per query. In a graph of 4,000 block groups, one single iteration of the greedy algorithm would take more than 33 hours. Even though parallelization and lazy evaluations [176] can be used to speed up the computation, running several iterations of the greedy algorithm may just not be feasible.

Here, we propose a heuristic based on Steiner connectivity that requires a single iteration of the greedy procedure. First, we compute the criticality of each node in the graph, and then we find a connected subgraph of size at most k that maximizes the sum of criticalities. This involves solving a node variant of the NP-Hard k-Maximum Spanning Tree problem [208].

Problem 18 (k-Maximum Spanning Tree (k-MaxST)) *Given a graph $G = (V, E)$ where each node v has a weight $w(v)$, we want to find a tree $H(V', E')$ with $|V'| \leq k$, such*

that the total weight of the nodes of the tree,

$$\sum_{v \in V'} w(v),$$

is maximized.

In order to solve k-MST, we adapt our color-coding algorithm for Prize-Collecting Steiner Tree from Chapter 3. Naively, one could find a solution to k-MaxST by exhaustively checking all the possible $\binom{n}{k}$ subgraphs of k nodes in time $O(n^k)$. The color coding technique allows us to reduce the search space to $O((2e)^k)$, thus keeping the computation feasible.

Let $K = \{1, \ldots, k\}$ be a set of k colors. Uniformly at random, we assign one of this k colors to each node in the graph. Let $col(v)$ be the color of node v and let $\text{OPT}(v, T)$ represent the optimal weight of a tree containing node v and colorful with respect to T. $\text{OPT}(v, T)$ can be computed using a dynamic program. When T is a singleton set, this is easy to compute:

$$\text{OPT}(v, \{s\}) = \begin{cases} w(v) & : col(v) = s \\ -\infty & : col(v) \neq s \end{cases}$$

Now, for a color set T of size greater than 1, we can compute $\text{OPT}(v, T)$ recursively:

$$\text{OPT}(v, T) = \max_{\substack{u \in Nbr(v) \\ T_1, T_2 \subset T}} (\text{OPT}(v, T_1) + \text{OPT}(v, T_2)),$$

where the maximum is over all possible partitions of T into two subsets, T_1 and T_2, and all possible neighbors u of v. The final answer is $\text{OPT} = \max_{v \in V} \text{OPT}(v, K)$.

We can verify that this dynamic program correctly returns the optimal weight over all connected subgraphs of size at most k. By repeating the algorithm for many random colorings, the optimal subgraph of size less than k will indeed be colorful with high probability; see Chapter 3 for details.

Theorem 21 *Given a graph $G = (V, E)$ with node weights, size constraint k, and error parameter ϵ, there is an algorithm for the k-MST problem that finds the optimal tree with probability $1 - \epsilon$ in time $O((2e)^k m \log(1/\epsilon))$.*

8.4 Experimental Results

8.4.1 Experimental Setup

8.4.1.1 Dataset and Simulation Engine

A study of epidemics that spread through physical proximity requires social contact networks in which an edge represents an actual physical contact between two people at some location

during the day. Such networks are not readily available and cannot be constructed easily because of the difficulty in tracking individuals' contacts over a day on a reasonable scale. We study a specific class of realistic population and social contact graphs developed using first principles based methods [32,78]. We briefly summarize how these networks are constructed, and refer to [32, 78] for the details. (i) A synthetic urban population model is constructed by integrating over a dozen public data (e.g., US Census, land use and activity surveys) and commercial data (e.g., from Dunn & BradStreet on location profiles). The resulting population is statistically equivalent to the census; (ii) A sequence of activities is constructed for each household and each person—this models the type of activity performed by each person, along with other attributes, such as duration. The activities of different members of the same household are correlated; (iii) Activity locations are assigned for each person, using land use data and activity choice models; and (iv) Individuals are routed through the road network, which in turn gives a social contact network based on co-location. These network models have been used in a number of studies on epidemic spread and public health policy planning—see, e.g., [32, 38, 56, 78, 213].

In particular, we focus on a synthetic population of Minnesota, which has $5,048,920$ agents in total. These agents are aggregated into 4,082 census block groups from the 2010 U.S. census. Using the EpiFast simulation engine [38], we simulate 400 days of the spread of a disease with an attack rate of 45%, following the SIR model. In order to estimate the criticality of a subgraph, we run the simulation 100 times.

8.4.1.2 Baseline Methods

We compare our k-MaxST based algorithm for k-CRP with two heuristics used in epidemiology and public health and a naive random baseline.

1. Population. The first baseline is based on population; we use our algorithm for k-MaxST to find a connected set of block groups of size k with the largest total population. The intuition behind this heuristic is leaving as many people as possible unvaccinated.

2. Vulnerability. The second baseline is based on the notion of vulnerability; the vulnerability of an individual is the probability that this person will get infected when the disease is left to propagate with no intervention—i.e., $x_v = 0$ for all nodes. We further define the vulnerability of a block group as the average vulnerability of its inhabitants, and we find a subgraph of k block groups with as large total vulnerability as possible—again, using our k-MaxST algorithm. This baseline tries to maximize spread by prioritizing individuals who are most likely to get infected.

3. Random. We obtain a connected subgraph of size k by performing a random walk starting from a randomly chosen node in the graph. For each size k, we obtain 10

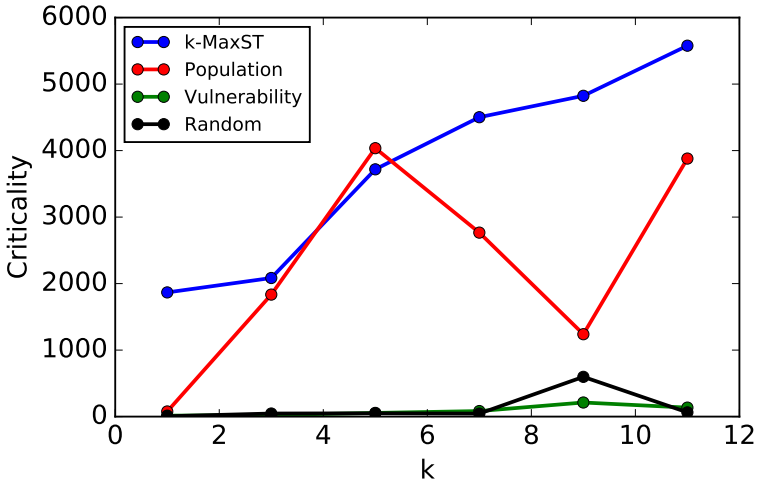


Figure 8.1: Comparison of algorithms for criticality as a function of the solution size k.

random such subgraphs, and the results that we show below is the average over those 10 runs.

8.4.2 Evaluation

In Figure 8.1, we show the criticality of our algorithm (k-MaxST) and the three baseline methods as a function of k. As expected, selecting subgraphs at random performs poorly and does not lead to large outbreaks. Surprisingly, the vulnerability-based heuristic does not perform better than random. It is also interesting that the population-based heuristic does not have monotonic improvement with k. Even though the subgraph of size 9 has 55,800 inhabitants, the smaller subgraph of size 5 with a population of 34,000 leads to a significantly larger outbreak. Overall, the population-based heuristic has better performance among the baselines, and it even surpasses our algorithm for $k = 5$; however, our k-MaxST algorithm that maximizes the sum of criticalities leads to the largest outbreaks for most choices of k.

In addition to comparing criticality, we also examine the performance of the algorithms in individual runs of the simulation. In Figure 8.2, we show the distribution of criticality values for each method over 100 simulations—i.e., samples of the live graph. We observe that even the largest outbreaks caused by Vulnerability and Random are much smaller than those of k-MaxST and Population. We also note that the population-based clusters have larger variance in criticality and can result in larger outbreaks than those from our k-MaxST algorithm. This suggests that if the goal for a public health department is to prevent the worst-case scenario, then intervening the most-populated areas is a good heuristic. However, in doing so, one could miss smaller regions that, on average, are likely to infect more people.

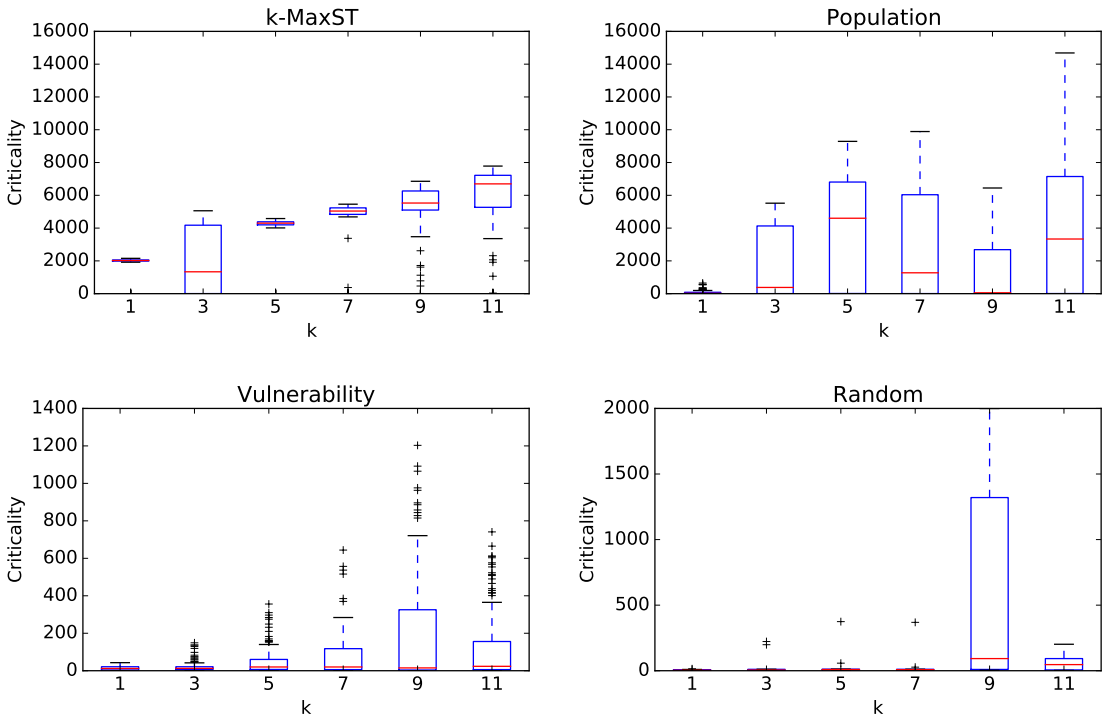


Figure 8.2: Criticality over 100 SRI simulations for each method evaluated.

8.5 Related Work

Mathematical models have played an important role in epidemiology for over a century (see, e.g., the classical text of May and Anderson [15]). Most of the work in epidemiology has been focused on analyzing homogeneous models which assume very simplistic mixing patterns of the underlying population. The importance of mathematical and computational modeling is further highlighted by the fact that controlled physical experiments used to understand scientific phenomena are almost impossible in epidemiology (e.g., it is not feasible to run randomized trials). Examples of fundamental questions studied by public health officials during epidemic outbreaks include: where did the disease start, how is it spreading, and how can it be controlled. In recent years, data-driven computational networked epidemiology has emerged as an interdisciplinary area that leverages computational models and big data to support epidemic planning and response—an example is the use of search queries by (now discontinued) Google's FluTrends for epidemic forecasting [90]. Numerous problems with this approach have been pointed out [154], and in recent years, more comprehensive ensemble methods have been found to be more effective [50].

There has been a lot of work on different kinds of detection problems related to outbreaks in networks. For instance, Christakis and Fowler [59] use the "friend of random people" approach to monitor a subset of people and infer characteristics of the epicurve for the entire population. Leskovec et al. [161] study the problem of early detection of different kinds of events—e.g., in water networks or social networks. However, these approaches have been focused on either just detecting that some event (e.g., start of an infection) has occurred or the epidemic characteristics for the entire region. Instead, we are interested in finding regions that would lead to a big number of infections if left unvaccinated.

Our work is also related to submodularity function maximization with connectivity constraints; this constraint makes the problem much harder than other constraints, such as cardinality or matroid constraints, which can be approximately optimized using a simple greedy procedure [189]. Recently, Kuo et al. [149] proposed a $O(\sqrt{k})$ approximation algorithm to this problem, which is applicable to our setting but potentially slow. Krause et al. [142] propose an approximatinon algorithm for budgeted submodularity maximization on graphs based on exploiting local structure. We can adapt these results to show theoretical bounds on our problem as well, but the resulting algorithms do not scale.

8.6 Conclusions

We study the problem of finding clusters that are critical for epidemic spread. We show that the notion of criticality (i.e., number of extra infections when a region is left unvaccinated) is related to Influence Maximization and submodularity. Finding the most critical region is a submodular maximization problem with connectivity constraints, which is computationally

hard. We show the existence of approximation algorithms for k-CSP and k-CRP; however, these algorithms are not scalable. We propose a fast heuristic based on the k-Maximum Steiner Tree problem: we find a subgraph that maximizes the sum of individual criticalities. Experimental results show that this modular approximation is quite effective and could be used by public health practitioners as an alternative to simple baselines based on population or vulnerability.

Chapter 9

Forecasting Social Unrest Using Activity Cascades

9.1 Introduction

Social media has become a window into happenings on the ground, from earthquakes [215] to specific news stories [217]. A key population-level event is civil unrest, i.e., protests, strikes, and occupy events, wherein civilian populations mobilize to raise awareness of key issues. As is evident from recent protests in many countries (e.g., Egypt, Turkey, Brazil), social media plays an important role in documenting, triggering, mobilizing, or even quelling such events [113, 175]. However, it is not clear when preliminary chatter observed on social media becomes a true precursor for a protest, and, thus, understanding the structure chatter behavior is a relevant problem.

While Twitter is used for many purposes (e.g., discontent expression, event reporting, planned protest recruitment), all of which can be used in a predictive model for civil unrest, we focus on modeling activity cascades (using a formulation of [42, 84, 97]) as a uniform precursor for civil unrest forecasting. Our goal is to design a model that forecasts the date of the event. Cascades, as is well known, help formalize the spread of influence and information [4, 28, 151, 157, 236]. Informally, cascades are subgraphs (often trees) that capture the spread of influence from a node to its newly activated/influenced neighbors and descendants.

There are many notions of cascades, which afford varying levels of formal characterization and utility. For instance, in Bakshy et al. [28], an edge $A \rightarrow B$ is included in a cascade only if it can be argued that some action (e.g., a posting or use of a URL) by B can be directly attributed to A, which makes the analysis intensive in terms of data and computation; also, the mathematical analysis under their model becomes challenging. Another common approach is to define cascades in terms of the (random) subgraph over which diffusion processes like linear threshold and independent cascades models spread [130, 157]. Here, we use

a simpler notion of cascades (referred to as "activity cascades"), introduced by [42, 84, 97]. Informally, such a cascade consists of a tweet emitted by a user u, and the tweets of the users who see/mention u's message (for instance, her direct followers or users who mention her) given that those tweets are sent within a small time interval (denoted by Δ), and so on (see Section 9.2 for a precise definition). This turns out to be a special case of Hawkes processes [63, 227, 271], which are based on mixtures of mutually exciting point processes. Although simpler to define, we demonstrate that this notion of cascades has good predictive power for modeling civil unrest, and is also amenable to rigorous analysis. Our key contributions are focused on the following three questions:

1. When do large activity cascades happen? A common empirical observation is that cascades seldom become very large. We rigorously prove necessary and sufficient conditions for large cascades in terms of spectral properties of the underlying graph; these also imply a similar characterization for a class of Hawkes processes. We find that this characterization closely matches our empirical observations for synthetic traces. Specifically, our analyses show that if the spectral radius of a cascade graph is below a particular constant, then large cascades are not possible. Our techniques build on approaches for analyzing the spread of epidemics [85, 199, 247], and are the first such results for cascades of this kind.

2. Are there critical subsets of users that contribute to cascades? We study the questions of identifying critical subsets of users responsible for formation and survival of cascades, and formalize these as two complementary problems: CriticalSetFormation (CSFP) and CriticalSetShattering (CSSP). We show both to be NP-complete, and we evaluate different greedy heuristics to approximate them empirically by studying large, monthly cascades for all (country, month) combinations for Brazil, Mexico, and Venezuela over a 15-month period. Our results for CSSP show that a very small set of users are critical for a cascade to exist—their non-participation causes the cascade to shatter. We also prove that a high degree strategy gives a constant factor approximation for CSSP in random power law graphs. On the other hand, the results for CSFP suggest that unless a large fraction of users participate a cascade cannot exist. Thus, one needs a reasonable fraction of users, plus some critical users for a cascade to exist.

3. Can we forecast protests using activity cascades? Since large cascades are not very common, their occurrence signals a big event. We analyze over 353 million tweets from three Latin American countries over a 1.5-year period, and consider activity cascades formed by a filtered set of tweets. These tweets contain at least 3 keywords from a dictionary that has over 900 words in English, Spanish, and Portuguese, related to civil unrest activities. This ensures that the resulting activity cascades will be relevant to the topic of civil unrest. Next, we build a feature set based on the structural properties of the cascades, to be used as predictors of social unrest. Statistical models are then used to remove redundancies among features and make predictions of events from the reduced feature set. The model is tested on multiple countries for robustness and compared against a baseline model. We show that our approach can 'beat the news,' i.e., contribute a lead time of one to two days over the

reporting of a protest in major news media, with an accuracy of over 0.75. It can even predict black swan events like the Brazilian Spring with an accuracy of 0.83.

Our work here helps explain the model and observations of [42, 84, 97] rigorously, especially conditions for occurrence of large cascades. Since their frequency is relatively low, their occurrence is a signal of significant events—-this corroborates with the observations of [97] and is the basis of our approach for forecasting protest events.

9.2 Preliminaries

Let $G = (V, E)$ denote a directed graph, with $N_o(u)$ and $N_{in}(u)$ denoting the set of out-neighbors and in-neighbors for a node $u \in V$, respectively. The nodes represent Twitter users. We consider two kinds of graphs—a *follower* graph and a combined *mentions* and *retweet* graph. An edge (u, v) has different interpretations depending on the graph, as discussed below. We assume each user u sends at most one tweet at a time (with one second granularity), so a node-time pair (u, t) identifies a tweet. For a node $u \in V$, time t and time interval Δ, we define a cascade $C(u, t, \Delta)$ to be a set of tweets/node-time pairs in the following recursive manner, using the formulation of [42, 84, 97].

- If there is no tweet *driven by* node u at time t, then $C(u, t, \Delta) = \phi$.
- Else, $C(u, t, \Delta) = \{(u, t)\} \cup \{x \in C(v, t', \Delta) : v \in N_o(u), t' \in (t, t + \Delta]\}$

The term *driven by* will be explained below. This general notion of cascade makes no prior assumptions about the nature of the edges connecting the nodes of the graph, which gives us the flexibility to define the neighborhood of a node in different ways. Though this definition does not explicitly look for any correlation between the messages of u and v (which is used by [4, 28, 151, 157, 236]), we use it on a set of tweets that are already filtered for protest-related keywords, as done by [42, 84, 97], which brings in some correlation. Also, note that this notion of cascades is different from the more commonly studied notion associated with diffusion processes [130, 157]—here, a cascade is a random subgraph on which the influence spreads.

We study two types of activity cascades: *follower* (F) and combined *mention* plus *retweet* (MRT) cascades, defined, respectively, by the *follower* and *mention* and *retweet* graphs, each of which models a different kind of interaction between users in the Twitter network. Our methodology is the same as [216], except that we also include retweets.

In a follower graph, for every node $u \in G$, $N_o(u)$ is the set of Twitter followers of u, and $N_{in}(u)$ is the set of Twitter friends of u (i.e. users followed by u). From this definition, follower cascades in Twitter emerge in the following manner: a user u posts a tweet at time t starting a cascade where she is the only participant. For each follower v of u who posts a

tweet at some time $t' \in (t, t + \Delta]$, (v, t') is added to this cascade, and so on, as illustrated in Figure 9.1. This process is repeated until no more users can be added to the cascade.

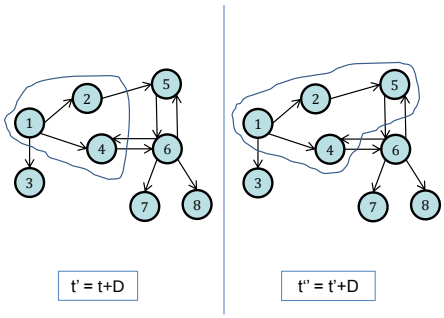


Figure 9.1: **Formation of cascades in the Twitter follower network.** At time t, node 1 posts a tweet. Nodes 2 and 4 post at times t_2 and t_4 between t and $t' = t + D$. Node 5, which follows 2, posts at some time t_5 between t' and $t'' = t' + D$. Therefore, the cascade $C(1, t, D)$ is $C(1, t, D) = \{(1, t), (2, t_2), (4, t_4), (5, t_5)\}$.

We combine mentions and retweets to form an MRT graph because both types of tweets indicate influence between pairs of users. Suppose a user w with name W composes a tweet at time t_1 that mentions another user u with name U, where W and U are sequences of characters of the form $[a - zA - z0 - 9\_]+$. In the following, we specify the concatenation of two sequences of characters, $A = (a_1, a_2, \ldots, a_q)$ and $B = (b_1, b_2, \ldots, b_r)$, using the operator $\oplus$, as $A \oplus B = (a_1, a_2, \ldots, a_q, b_1, b_2, \ldots, b_r)$. Let P_W be the payload or content of the tweet of W; P_W is a sequence of characters. Because w mentions u, we have $(@) \oplus U \subseteq P_W$, which produces the directed edge (u, w) of the MRT graph. This edge has the semantics that u influences w. Analogously, if x with name X retweets a message from user w at time t_2, where $(RT\ @) \oplus W \subseteq P_X$, then, we have the directed edge (w, x), where the semantics are the same as in the mentions edge: w influences x. If the two tweets occur such that $t_2 \in (t_1, t_1 + \Delta]$, then, the two edges link up to form a directed path of length 3, $u \to w \to x$, and the cascade $C(u, t_1, \Delta) = \{(u, t_1), (w, t_2)\}$.

We have several notes. The term *driven by* indicates the user that instigates the cascade. For a follower graph, the instigator is the user that sends the first tweet of a cascade. For an MRT graph, the instigator is the first influencer of a cascade. Second, users (nodes) in an MRT graph with zero out-degree (x in this example) are not included in the MRT cascade because there is no evidence that these nodes influence other users. Also, a single tweet can produce multiple edges in an MRT graph. Since retweets of an original tweet preserve the original tweeter, no matter how many times the original tweet is (sequentially) retweeted, a set of these retweets (without mentions) produces a star subgraph of the MRT graph. Finally, MRT cascades, unlike follower cascades, directly use tweet payloads; however, F cascades utilize the follower graph.

We provide additional definitions that will be useful in forecasting social unrest. We say that a (follower or mentions/retweet) cascade C is *active* on day d if there exists at least one message or tweet (u, t) by user u at time t, such that t is some time during day d and (u, t) is an element of C. The *size* of a cascade is the number of tweets comprising it. A *user* or *participant* is a tweeter.

9.3 Proposed Methods

9.3.1 Characterizing large cascades in terms of graph properties

Our empirical observations on Twitter data suggest that large and long cascades are rare, and they arise within communities of users. We now attempt to explain this behavior by relating it to the spectral properties of the graph, by considering a formulation based on a slight relaxation of the notion of cascades: We consider a cascade starting at a random initial node u_0 at time t_0; (i) $X(0)$ denotes the initial configuration. We say that (u_0, t_0) is active in the cascade at time t_0. Following our definition, we will think of a cascade as consisting of tweets indexed by user-time pairs (u, t); (ii) The number of tweets sent by each user u is a Poisson process with parameter α_u; (iii) If user u sends a message at time t, and some other user v with $u \in N_o(v)$ is active at time t, then (u, t) becomes active in the cascade at time t; (iv) A tweet (u, t) ceases to be active after a (random) time duration $D(u, t)$ drawn from an exponential distribution with parameter $\delta = 1/\Delta$; and (v) The cascade $C(u_0, t_0)$ dies when there are no more active tweets in it.

We now model this as a Markov process $\mathbf{X(t)}$ with values in $\mathbb{N}^V$. Let $X_u(t)$ denote the number of messages by user u that are active at time t. Then, the cascade evolves in the following manner:

$$X_u: \text{ increases by 1} \quad \text{at rate } \alpha(u) \text{ if } \{v \in N_{in}(u) : X_v > 0\} \neq \phi \tag{9.1}$$

$$X_u: \text{ decreases by 1} \quad \text{at rate } \delta X_u \tag{9.2}$$

Every cascade eventually becomes inactive, since $X_u = 0$ for all u is the unique absorbing state for this Markov process. The *lifetime* of the cascade, the duration for which it lasts, is precisely $T = \sup\{t : X_u(t) > 0, \text{ for some } u \in V\}$. We now derive necessary and sufficient conditions for obtaining large cascades.

9.3.1.1 Multivariate Hawkes Processes

Our formulation above makes it a special case of the multivariate Hawkes processes, as we now discuss. A Hawkes process $\mathbf{N_t}$ is a type of self-exciting counting process characterized

by a time-dependent intensity (rate) $\lambda(t)$ [63, 227, 271].

Let $\mathbf{N_d}(\mathbf{t})$ be a multidimensional counting process, where $d \in \{1, \ldots, D\}$ denotes a dimension (with D being the number of dimensions). Let $\lambda_d(t)$ denote the intensity of $N_d(t)$. The process is defined in the following manner:

$$\lambda_d(t) = \mu_d(t) + \sum_{d'=1}^{D} \int_{-\infty}^{t} \kappa_{d'd}(t-s) dN_{d'}(s),$$

where μ_d is a base intensity for dimension d, and $\kappa_{d'd}(\tau)$ is a kernel function describing the influence of the previous events in dimension d' on the current rate on d.

For our formulation, let each node $u \in V$ be a separate dimension, and let $\mathbf{N_u}(\mathbf{t})$ be the number of messages contributed to an ongoing cascade. We have $\mu_u(t) = 0$ and the kernel function as $\kappa_{vu}(\tau) = \alpha_{vu} \times \kappa(\tau)$, where: (i) $\alpha_{vu} = \alpha_u N_v(t)$ if $v \in N_{in}(u)$; otherwise, $\alpha_{vu} = 0$. Here, α_u is the (fixed) tweeting rate of u, and α_{vu} describes the fact that u's contributions to the cascade are proportional to her in-neighbors' contributions (i.e. her friends in the follower graph); and (ii) $\kappa(\tau) = 1$ if $0 < \tau \leq \Delta$; otherwise $\kappa(\tau) = 0$. As a result, we have:

$$\lambda_u(t) = \alpha(u) \sum_{v \in N_{in}(u)} (N_v(t) - N_v(t - \Delta))$$

We use the process $\mathbf{X}(\mathbf{t})$ below for our discussion, since it simplifies the analysis; our results hold for a class of Hawkes processes with the kind of kernel function mentioned above.

9.3.1.2 Conditions for Small Cascades

We now derive conditions when the maximum cascade size is $O(\log n)$, with high probability, where n is the number of nodes. The process $\mathbf{X}(\mathbf{t})$ is non-linear, making it quite complex to analyze; instead we consider the following relaxation $\mathbf{Y}(\mathbf{t})$:

$$Y_u: \text{ increases by 1} \quad \text{at rate } \alpha(u) \textstyle\sum_{v \in N_{in}(u)} Y_v \tag{9.3}$$

$$Y_u: \text{ decreases by 1} \quad \text{at rate } \delta Y_u \tag{9.4}$$

Lemma 25 The process $\mathbf{Y}(\mathbf{t})$ stochastically dominates $\mathbf{X}(\mathbf{t})$ so that $X(t) \leq Y(t)$ *for all* $t \geq 0$.

Proof: *Our proof is based on designing a coupling that ensures that* $X(t) \leq Y(t)$ *for all* $t \geq 0$, *and builds on [85]. Clearly,* $X(0) \leq Y(0)$. *We consider the process* $\mathbf{Y}(\mathbf{t})$ *and for each node* u, *we sample random variables* R_u^1 *and* R_u^2 *from exponential distributions with parameters* $\alpha(u) \sum_{v \in N_{in}(u)} Y_v$ *and* δY_u, *respectively. The first transition out of* $Y(0)$ *happens at time* τ,

which equals $\min_u\{R_u^1, R_u^2\}$. *Our coupling will specify the transition for the process* $\mathbf{X(t)}$ *in the following manner. Suppose the transition at time* τ *corresponds to* $Y_u(\tau) = Y_u(0) + 1$*; this would have happened with rate* $\alpha(u)\sum_{v\in N_{in}(u)} Y_v(0)$. *For the corresponding process* $\mathbf{X(t)}$*, the transition* $X_u(\tau) = X_u(0) + 1$ *is made with probability* $\frac{1}{\sum_{v\in N_{in}(u)} Y_v(0)}$*, if* $\{v \in N_{in}(u) : X_v > 0\} \neq \phi$*; otherwise* $X_u(\tau) = X_u(0)$. *This ensures that the transition* $X_u(\tau) = X_u(0)+1$ *happens with the correct rate. Similarly, the transition corresponding to* $Y_u(\tau) = Y_u(0) - 1$ *can be handled to get a coupling of the first jumps in* $\mathbf{X(t)}$ *and* $\mathbf{Y(t)}$. ■

Lemma 26 Let $\rho(A)$ denote the spectral radius of A, the adjacency matrix of G. Assume that G is a bi-directed graph and let $\alpha_{max} = \max_u \alpha(u)$. If $\alpha_{max}\rho(A) < \delta$, the duration of the cascade T satisfies $\Pr[T > t] \leq ne^{-(\delta-\alpha_{max}\rho(A))t}$ and $E[T] \leq \frac{\log n+1}{\delta-\alpha_{max}\rho(A)}$.

Proof: *Our proof is an adaptation of that of [85] for the SIS model; we describe it here completely for completeness. From equations 9.3 and 9.4, it follows that*

$$\begin{aligned} E[Y_u(t+dt) - Y_u(t)|\mathbf{Y(t)}] &= \alpha(u)\sum_{v\in N_{in}(u)} Y_v(t)dt - \delta Y_u(t)dt + o(dt) \\ &\leq \alpha_{max}\sum_{v\in N_{in}(u)} Y_v(t)dt - \delta Y_u(t)dt + o(dt), \end{aligned}$$

which implies $\frac{dE[\mathbf{Y}(t)]}{dt} \leq (\alpha_{max}A - \delta I)E[\mathbf{Y}(t)]$.

This has solution $E[\mathbf{Y}(t)] \leq e^{(\alpha_{max}A-\delta I)}\mathbf{Y}(0)$. *From Lemma 25, and since* $\mathbf{X}(0) = \mathbf{Y}(0)$*, we have* $E[\mathbf{X}(t)] \leq e^{(\alpha_{max}A-\delta I)}\mathbf{X}(0)$.

Let $N_t = \sum_v X_v(t) = \mathbf{1}^T\mathbf{X}(t)$ *denote the number of nodes infected at time* t. *Then,* $N_t \leq \mathbf{1}^T e^{(\alpha_{max}A-\delta I)}\mathbf{X}(0)$. *Since* A *is a symmetric matrix,* $e^{(\alpha_{max}A-\delta I)}$ *is also symmetric, and we have* $\| e^{(\alpha_{max}A-\delta I)}\mathbf{X}(0) \| \leq \rho(e^{(\alpha_{max}A-\delta I)}) \| \mathbf{X}(0) \| = e^{\alpha_{max}\rho(A)-\delta}\sqrt{n}$. *This implies* $E[N_t] \leq ne^{\alpha_{max}\rho(A)-\delta} = ne^{-(\delta-\alpha_{max}\rho(A))}$*, since* $\| \mathbf{X}(0) \| \leq \sqrt{n}$. *The first part of the lemma follows since* $\Pr[T > t] = \Pr[N_t \geq 1] \leq E[N_t]$.

For the second part of the lemma, we have

$$\begin{aligned} E[T] &= \int_0^\infty \Pr[T > t]dt \\ &\leq \int_0^\infty \min\{1, ne^{\alpha_{max}\rho(A)-\delta}\} \\ &\leq \int_0^{\log n/(\delta-\alpha\rho(A))} 1dt + \int_{\log n/(\delta-\alpha_{max}\rho(A))}^\infty ne^{-(\delta-\alpha_{max}\rho(A))}dt \\ &\leq \frac{\log n + 1}{\delta - \alpha_{max}\rho(A)} \end{aligned}$$

■

Lemma 26 implies that when $\alpha_{max}\rho(A) < \delta$, any cascade has size $O(\log n)$. We are able to prove Lemma 26 only when G is symmetric because the proof relies on all eigenvalues being real.

9.3.1.3 Conditions for Large Cascades

We now consider the conditions for having a large cascade (of size c^m, where c is a constant larger than 1, and m is a parameter). We need the following version of the isoperimetric constant, which captures node expansion.

$$\hat{\eta}(G,m) = \min_{S\subseteq V, |S|\leq m} \frac{\sum_{v\in V-S:N_{in}(v)\cap S\neq\phi}\alpha_v}{|S|}.$$

We sometimes omit the reference to the graph G in $\hat{\eta}(G,m)$, and just use $\hat{\eta}(m)$ when G is clear from the context. We now consider a Markov process $\mathbf{Z}(t)$ with state space $\{0,\ldots,m\}$, defined in the following manner:

$$\begin{aligned} Z(t) &= Z(t)+1 \quad \text{at rate } \hat{\eta}(m)Z, \text{ if } Z<m \\ Z(t) &= Z(t)-1 \qquad \text{at rate } \delta Z, \text{ if } Z>0 \end{aligned}$$

Lemma 27 *$Z(t)$ is stochastically dominated by $\sum_u X_u(t)$, i.e., $Z(t) \leq \sum_u X_u(t)$ for all $t \geq 0$.*

Proof: *The proof is also by designing a coupling, as in Lemma 25. We assume that $Z(0) \leq \sum_u X_u(0)$, and prove the statement by induction. We consider the process $\mathbf{X}(t)$ and for each node u, we sample random variables R_u^1 and R_u^2 from exponential distributions with parameters $\alpha(u)1_{\{v\in N_{in}(u):X_v>0\}\neq\phi}$ and $\delta X_u(0)$, respectively. The first transition out of $X(0)$ happens at time τ, which equals $\min_u\{R_u^1, R_u^2\}$. Our coupling will specify the transition for the process $\mathbf{Z}(\mathbf{t})$ in the following manner. Let $S = \{w : X_w(0) > 0\}$. Let $N^+(S) = \{v \in V - S : N(v) \cap S \neq \phi\}$.*

Suppose the transition at time τ corresponds to a transition $X_u(\tau) = X_u(0)+1$ for some node u (which increases the number of active messages). The total rate at which such an increase happens equals $\sum_u \alpha(u)1_{\{v\in N_{in}(u):X_v>0\}\neq\phi} = \sum_{u\in N^+(S)}\alpha(u)$. First, suppose that $|S| < m$. The transition $Z = Z+1$ is now made at time τ with probability

$$\frac{\hat{\eta}(m)Z}{\sum_{u\in N^+(S)}\alpha(u)}.$$

This fraction is in $[0,1]$, because $Z(0)\hat{\eta}(m) \leq \sum_{u\in N^+(S)}\alpha(u)$, by definition of $\hat{\eta}(m)$, and because $|S| < m$, so that the transition happens with the correct rate. Second, if $|S| \geq m$, Z is unchanged, which is the correct rate.

Next, we consider the case that the transition at time τ corresponds to a transition $X_u(\tau) = X_u(0) - 1 = 0$ for some node u. In this case, the transition $Z(\tau) = Z(0) - 1$ is made with probability $\frac{Z(0)}{\sum_u X_u(0)}$, which is well defined since this is in $[0, 1]$. Also, note that there is some probability that $\sum_u X_u(0)$ decreases by 1, but $Z(0)$ does not— this does not violate the property, because in this case $Z(0) < \sum_u X_u(0)$.

Therefore, in either case, we have $Z(\tau) \leq \sum_u X_u(\tau)$, and the lemma follows. ■

Lemma 28 Suppose $r = \frac{\delta}{\hat{\eta}(m)} < 1$. Then, we have $\Pr[T > \frac{r^{-m+1}}{2m}] \geq \frac{1-r}{e}(1 + O(r^m))$.

Proof: *The proof of the above lemma follows by observing that the process $\mathbf{Z}(t)$ is a one-dimensional random walk, defined in the following manner. Consider the discrete time Markov chain associated with Z. Let $p(i, j)$ denote the probability that Z switches to value j from i. Then, we have:*

$$\begin{aligned} p(i, i+1) &= \frac{\hat{\eta}(m)}{\hat{\eta}(m) + \delta},\ i = 1, \ldots, m-1, \\ p(i, i-1) &= \frac{\delta}{\hat{\eta}(m) + \delta},\ i = 1, \ldots, m-1 \\ p(0, 0) &= 1, \\ p(m, m-1) &= 1. \end{aligned}$$

Then, the duration of the cascade, T, is the time before the process hits 0. As in [85], this is the standard gambler's ruin probability, and the rest of the proof follows exactly as in [85]. ■

Spectral connection. Vertex expansion is related to the graph spectrum. If G is a d-regular graph, and if its spectral gap, i.e., the difference between the smallest and second smallest eigenvalue, is μ, the vertex expansion for sets of size at most m is $\frac{1}{(1-m/n)\mu^2 + m/n}$.

9.3.2 Identifying Critical Sets in a Cascade

We now consider the following questions: What is the critical subset of users whose tweets are responsible for the cascade to survive? What is the critical subset of users whose removal would cause the cascade to disintegrate? These are related and complementary problems, which can help explain the conditions for cascade formation. We consider a slightly more general notion of cascades than the one defined in Section 3.2— for a set S of nodes, we define $C(S, t, \Delta) = \cup_{u \in S} C(u, t, \Delta)$ to be the union of cascades starting at nodes in S. As a result, any directed acyclic graph can be seen as a cascade formed by its sources.

CriticalSetShattering Problem CSSP$(G, \mathcal{C}, k)$:
Input: A set of cascades $\mathcal{C}$ in a graph $G = (V, E)$ and parameter k.

Goal: Determine the smallest set $S \subseteq V$ of users, such that the sub-cascades of all $C \in \mathcal{C}$ in $G[V \setminus S]$ are of size at most k.

Thus, the goal in CSSP is to find the subset S whose removal causes all cascades in $\mathcal{C}$ to be "shattered".

CriticalSetFormation Problem, CSFP$(G, \mathcal{C}, \alpha, k)$:
Input: A set of cascades $\mathcal{C}$ in a graph $G = (V, E)$, tweet rate α and parameter k.
Goal: Determine the smallest set $S \subseteq V$ of users, such that for every $C \in \mathcal{C}$, a sub-cascade of size at least k exists in the graph $G[S]$ with tweet rate α.

Thus, CSFP quantifies the number of users needed to cause large cascades. While CSSP and CSFP are closely related and seem to be complementary problems, they are quite different from a computational perspective.

9.3.2.1 Complexity and algorithms for CSSP

Lemma 29 CSSP(C, G, k) is NP-complete.

Proof: *It is easy to verify that* CSSP *is in NP. The rest of the proof is by a reduction from the balanced graph partitioning problem [86]—this problem involves finding the smallest subset $S \subset V'$ of nodes in an undirected graph $H = (V', E')$ so that all components in $H[V' - S]$ have size at most b, which is a given parameter.*

Let C be a DAG formed by orienting the edges of H arbitrarily, so that it forms a DAG. Let $B \subset V'$ whose removal splits C into weakly-connected components $H_1, \ldots, H_r$, each of size at most $k = b$; as discussed earlier, each component H_i is a cascade formed by the sources in that DAG. If we ignore the directions of the edges, we get components of size at most $k = b$. This implies that the solution to CSSP *in C corresponds to a solution to the separator problem on H. The converse also holds similarly.* ■

Whenever the condition in Lemma 26 is tight, i.e., it gives both necessary and sufficient conditions, CSSP can be solved by simply attempting to reduce the spectral radius $\rho(A)$. We consider the special case of the Chung-Lu random graph model [60]: given a weight sequence $\mathbf{w} = (w(v_1), w(v_2), ..., w(v_n))$ for nodes $v_i \in V$, the random graph $G \in G(\mathbf{w})$ is obtained by choosing each edge (u, v) with probability $\frac{w(u)w(v)}{\sum_{v_i \in V} w(v_i)}$. We use the following result from [214].

Lemma 30 *[214] If $G = G(\mathbf{w})$ is a random graph in the Chung-Lu model with the weight sequence being a power law with exponent $\beta > 2$, removal of the $\Theta(n/T^{2(\beta-1)})$ nodes with the highest weight ensures that the spectral radius of the residual graph is at most T, almost surely.*

Motivated by Lemma 30, we study heuristics for CSSP based on degree and the core number in the underlying graph. Since $\rho(A) \geq \sqrt{\max_v deg(v, G)}$, a natural heuristic for CSSP is to reduce the maximum degree $\max_v deg(v, G)$. Also, since $\rho(A) \geq \frac{2|E(H)|}{|V(H)|}$ for any subgraph H of G, another natural heuristic for CSSP is to reduce the density of every subgraph H. Motivated by these bounds on the spectral radius of a graph, we consider the following heuristics for CSSP: (i) *high degree heuristic*: remove nodes in decreasing order of degree in G; and (ii) *high core number heuristic*: remove nodes in decreasing order of their core-number in G.

9.3.2.2 Complexity and algorithms for CSFP

Lemma 31 CSFP(C, G, α, k) is NP-complete.

Proof: *It is easy to verify that* CSFP *is in NP. The rest of the proof is by a reduction from the Set Cover problem, an instance of which consists of a set B of elements, a set A of subsets of B; the goal is to select the smallest subset $A' \subset A$ such that each element in B is covered by a set in A'.*

We construct an instance of CFP *in the following manner. We set ϵ to be a large integer. We construct a graph $G = (\{r, r'\} \cup A \cup B, E)$, where E consists of the following edges: edges (j, i) if $j \in B, i \in A$ and j is contained in set i, edges $(r, i), (r', i)$ for all $i \in A$, and edge (r, r'). We have $\alpha_r = \alpha_{r'} = \epsilon$, while $\alpha_u = 1/n$ for all $u \in A \cup B$. We note that $\eta(G, \hat{1}, \{u\}) \geq \epsilon$ for all $u \in \{r, r'\} \cup A$.*

Suppose $A' \subseteq A$ is a minimum set cover. Then, increasing $\alpha_u = \epsilon$ for all $u \in A'$ will ensure that $\eta(G, \hat{1}, \{j\}) \geq \epsilon$ for all $j \in B$. Similarly, suppose S is the optimum solution to the CFP *problem. Clearly, $S \subseteq A$; if $S \cap \{r, r'\} \neq \phi$, we can drop r, r' from S without affecting the feasibility of the solution. For each $j \in B$, there must be at least one neighbor in S; else, we cannot have $\eta(G, \hat{1}, \{j\}) \geq \epsilon$. This implies S is a set cover, completing the reduction.* ■

We consider a greedy algorithm for CSFP: pick nodes in non-increasing order of degree until the cascade on the graph induced by these nodes has size at least k.

We note that the maximum cascade size can be estimated for a given rate assignment within a factor of $1 \pm \epsilon$, with high probability, in time $O(|E| \log n/\epsilon^2)$ by a standard Chebyshev bound.

9.3.3 Forecasting Social Unrest using Cascades

Social media is believed to be responsible for facilitating critical communication often required to fuel momentum preceding the events of civil unrest. Here, we explore the ability of Twitter data to act as a predictive signal of future civil unrest. Specifically, we study the

prediction of civil unrest events (e.g., protests, strikes) by using properties of the activity cascades in Twitter data.

We employ a regression model to predict the probability of a civil unrest event in a given day by using features based on the structural properties of the activity cascades described earlier. We hypothesize that, in general, unusually large and long cascades are likely to be indicative of future events of interest. These could be sport events, concerts, revolutions, elections, etc. However,, given that our tweets are filtered by civil unrest related keywords, we expect the events detected will be of civil unrest type.

Starting from May 1, 2012 to November 30, 2013, for each day, we compute the total number and size of cascades, number of participants and duration (in days) of cascades, change in the number of participants and tweets, average growth rate of tweets and average growth rate of participants. These features are collected daily for each active follower cascade and MRT cascade. For each of the features described above, we also compute the minimum, maximum, median, and average of the cascade size, duration, and users, as well as the average value of the 1st, 2nd, 3rd, and 4th quartile of their distribution and add them to the feature set. Therefore, our initial feature set consists of 114 attributes: (7 cascade properties $\times$ 8 aggregate statistics $\times$ 2 types of cascades + (daily cascade count $\times$ 2 types of cascades)).

Our initial list of features is expected to be highly correlated, resulting in unstable estimates if used in regression modeling. For example, the cascade size of an MRT cascade is almost equivalent to the number of users in that cascade, since people tend to retweet a message only once. In addition, the majority of these cascades last for one day (duration=1), which makes the average growth equivalent to the change in the cascade size on the last day, which in turn gives the change in the number of users, and so on.

In order to address the problem of multi-colinearity, we compute the correlation between every pair of features (i.e. the correlation matrix) and remove highly correlated features. Specifically, if the correlation between two variables is greater than 0.7, we only keep one of the two. This methodology reduced the size of our feature set from 114 to 13. Next, we use a regression methodology called LASSO (Least Absolute Shrinkage and Selection Operator) to further remove the redundant features and to shrink coefficients [246]. The LASSO based logistic regression model uses cascade features from both the follower and the MRT models.

9.4 Experiments

Our experimental results are focused on answering the following questions.

1. Do our theoretical conditions for large cascades from Section 9.3.1 hold true in real datasets? (Section 9.4.2)

2. What level of increase in user tweeting initiates large cascades? Does the solution to

the CFP problem using our greedy heuristic suggest that a few important users are sufficient or is large number of users necessary? (Section 9.4.3)

3. Which cascade features yield the best forecasting performance? Are these features consistently better across multiple countries? (Section 9.4.4)

4. Are cascade models for forecasting protests significantly better than baseline models? Is there value in building features from different Twitter networks (i.e. follower and MRT)? (Section 9.4.5)

5. Can our methods help forecast 'black swan' events like the Brazilian Spring? (Section 9.4.6)

9.4.1 Data

9.4.1.1 Twitter Dataset

For event prediction, we use a set of over 353 million tweets collected for Brazil, Mexico, and Venezuela for the period of May 2012 through November 2013. Our dataset constitutes a 10% sample of the tweets for these countries during the above time period.

Our analysis is done separately for each country, and, as a pre-processing step, the tweets are filtered by country (using geolocation codes, place identifiers, language detection, author identification, and other enrichment processes), ignoring tweets for which a country of origin could not be determined. Next, we organized a vocabulary of 614 protest-related words (such as march, riot, strike, organize, democracia, conflicto, revolucion, criminalidade), 192 keyphrases (such as "right to work", "marcha por la paz"), and 105 country-specific key players (which include important public figures, political parties, labor unions), collectively referred to henceforth as keywords. Compiled by social scientists who are experts in the region, the keywords include English, Spanish, and Portuguese translations. We then subselect tweets which contain at least 3 keywords from our vocabulary. Tweet volumes before and after filtering are shown in Table 9.1.

Table 9.1: Number of tweets between May 2012 and November 2013

Country	**Raw**	**Filtered**
Mexico	97,873,616	3,524,695
Venezuela	105,938,438	6,683,834
Brazil	150,147,141	1,575,041

9.4.1.2 Follower Data

We obtained the follower network for a subset of the users in our dataset, for each country, by using the Twitter API. The size of the graphs for the three countries are the following: Mexico has 79,598 nodes and 1,437,687 edges, Venezuela has 312,241 nodes and 21,119,120 edges, and Brazil has 142,176 nodes and 6,854,368 edges.

9.4.1.3 Gold Standard Report (GSR) Datasets

We use the Gold Standard Report (GSR) dataset presented in [207]. The GSR was compiled by an independent group, selected by IARPA (Intelligence Advanced Research Projects Activity), which is comprised of social scientists and experts on Latin America. A small set of well-reputed newspapers for each country are used to identify the instances of civil unrest events to be included in the GSR. For each event, the GSR captures the when, where, who and why of the event, i.e. the date of the event, its geographic location, the population protesting (e.g. labor, medical workers, general population) and the reason for the protest (i.e., the event type, e.g., economic, political, resource).

9.4.1.4 Operational issues

All the Twitter data used in this study is in compliance with the Terms of Use and all website conditions of Twitter. We use data from May 2012 to infer the activity cascades. The GSR data is available only from Nov 2012. Hence, data from Nov 2012 to May 2013 is used for training the forecasting model and the held-out June 2013 data is used for testing in case of Brazilian Spring. This training/test split is actually a tough test for the forecasting algorithm because June 2013 depicted very significant changes in rates of occurrences of protests in Brazil (more on this below), and our approach was nevertheless able to capture this variation in its forecasts.

9.4.2 Validation: Two Regimes for Cascade Sizes

We empirically verify the conditions for large cascades uncovered using our theoretical analysis. We find ten of the largest follower cascades in Mexico between June 27 and Sep 7, 2012, and the subgraphs induced by these users (also referred to as the cascade graphs). We consider synthetic twitter traffic generated using a Poisson process (as in Section 9.3.1) for the users in these cascade graphs with rate $\alpha_u = \alpha$, and then compute the cascades induced by this for $\Delta = 4$ hours. Figure 9.2 shows the maximum cascade size n_α, as a function of α for each of these cascade graphs. The y-axis is normalized by the number of nodes n in the respective cascade. For most of the cascades, we observe a clear phase transition for α somewhere in the range $[0.05, 0.15]$. For these particular graphs, we find that $\rho(\hat{A})$ (in the

notation of Lemma 26) is below δ when α is in the range $[0.10, 0.15]$. We also observe in Figure 9.2 that the cascades die out when $\alpha \leq 0.05$, which is consistent with the condition in Lemma 26. Note that some of the cascades die out even for higher values of α, which is consistent with the gap between the necessary and sufficient conditions in Lemmas 26 and 28.

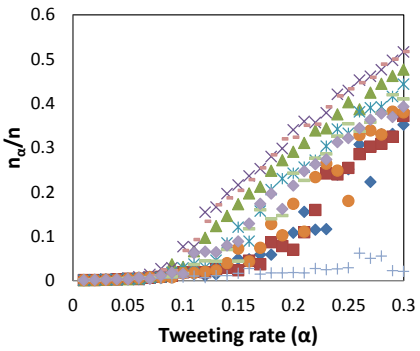


Figure 9.2: **Follower cascade size as a function of tweeting rate for ten follower cascades in Mexico (produced between June 27, 2012 and September 7, 2012), with synthetic traffic.** As the tweet rate of the users increases, we observe a sudden transition from a regime of very low user participation to a higher-activity regime.

9.4.3 Identifying Critical Sets in Cascades: CSSP and CSFP

Empirical analysis of heuristics for CSSP. We start with collections of tweets from Brazil, Mexico, and Venezuela that form cascades, in monthly intervals, from May 2012 through July 2013. We use reciprocal follower graphs (i.e., two users must follow each other to form an edge in the reciprocal follower graph, which implies a stronger association between users [97]) to determine which users follow each other. We use $\Delta = 4$ hours for the maximum duration that may separate a user's and a follower's tweets in forming edges in the cascade graph. The reciprocal follower graphs for Brazil, Mexico, and Venezuela have 1.9, 0.5, and 4.9 million edges, respectively, and 123,409, 69,226, and 253,423 nodes.

We select nodes (users) from the cascade graphs based on node properties in the *follower graph*. Specifically, we successively remove nodes (i) from greatest degree to least, and (ii) from the greatest k-shell to the least, from the follower graphs. We then remove these nodes from cascade graphs and compute the numbers of nodes and the sizes of the largest weakly connected components that remain in them (Section 9.3.2). Recall that nodes in a cascade graph are (user,time) pairs. Results are provided in Figure 9.3 for the largest cascades of Brazil and Venezuela, respectively. Results for other cascades, across countries and months, show the same behavior.

Removing relatively small fractions of high degree nodes and high k-shell nodes are both

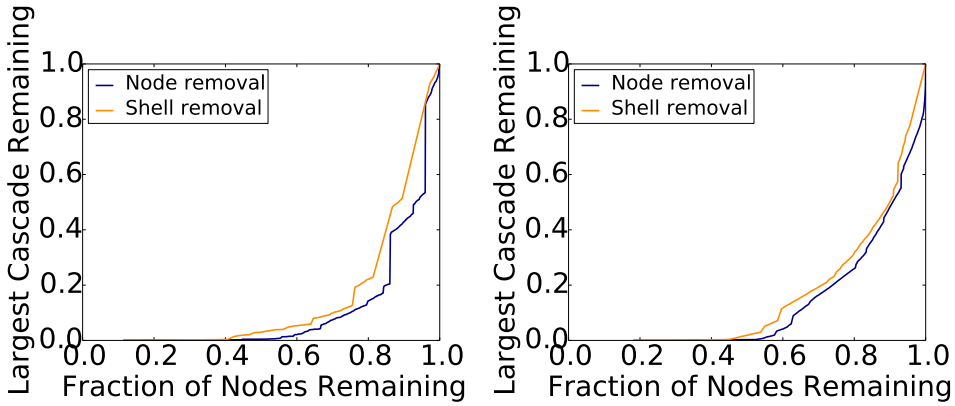


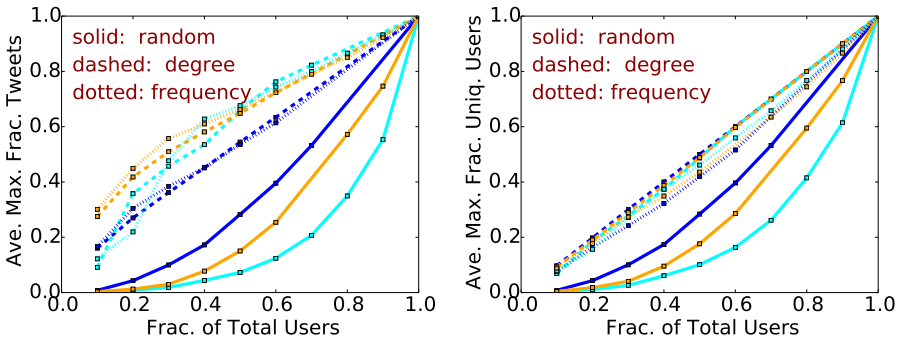
Figure 9.3: **Critical Set Shattering.** The largest remaining sub-cascade size in terms of numbers of tweets (normalized by the original size) as a function of numbers of remaining nodes in the cascade graph (normalized by the original number of nodes): 9.3a Brazil, June 2013, original cascade size is 15,791 tweets: 9.3b Venezuela, April 2013, original cascade size is 226,179 tweets.

effective in reducing the sizes of cascades. For all (country, month) combinations, the high degree heuristic is more effective than the high k-shell heuristic. Differences between the methods can be significant, particularly for small numbers of removed nodes.

Empirical analysis of heuristics for CSFP. We now solve the CSFP problem for selected large cascades in Mexico, Brazil, and Venezuela using the greedy heuristic in Section 9.3.2, in order to approximate the change in the level of tweeting that caused the cascade. We examine the differences in aggregate level of tweets and user participation, as well the characteristics of cascades that might result at lower levels of participation. When we consider the largest cascades and retain either the tweets or the users involved with probability p, the resulting sub-cascade size varies quite gradually with p, instead of showing a clear phase transition (in contrast with the results in Section 9.4.2). It is possible that the more gradual change is due to the non-uniform rates α_u for users u in the large cascades, which cause a higher level of weighted vertex expansion, even for moderate values of p.

We now consider the effect of a greedy choice of users from the original cascades using variants of the greedy heuristic described for CSFP. The first (structural) heuristic selects k nodes $\{v_1, \ldots, v_k\}$ with the greatest values $|N'_o(v)|$, where $|N'_o(v)|$ is the number of out-neighbors of v appearing in the maximum cascade for a (country, Δ) pair; this is a high-degree heuristic. The second (dynamical) heuristic simply chooses the k nodes with the greatest frequency of occurrence in a cascade. In both heuristics, $k = pN_c$, where p is the probability of selecting a node (cf. previous subsection) and N_c is the number of nodes in an original cascade, making it consistent with the earlier analysis. We compare these with a random selection of users

with probability p. Figure 9.4a shows the (normalized) maximum size of a cascade for each of the above heuristics (labeled "degree", "frequency" and "random", respectively), averaged over 50 trials. Figure 9.4b shows the corresponding normalized maximum number of unique users in the cascades. The normalization constant in each plot is the empirically determined maximum cascade size and maximum number of users, respectively. We find that the high degree heuristic generally produces the largest cascades in terms of tweets and users. For ordinate values in the range 0.2 to 0.4, the maximum sizes for the high degree heuristic are 2× to 10× those of the random heuristic. These data indicate that large cascades are tenuous; e.g., even with the high degree heuristic, 80% of the original users are required to produce a cascade that is 80% of the maximum measured size. Thus, it is not the case that a few users drive cascade formation. However, for CSSP, removal of a smaller fraction ($\sim$ 10–20%) of users can significantly reduce cascade size.



(a) (b)

Figure 9.4: **Critical Set Formation.** The (normalized) maximum 9.4a cascade size and 9.4b number of unique users vs the fraction of users selected for some of the largest cascades in different countries. Data are: blue (Mexico, $\Delta = 1$ hour); light blue (Brazil, $\Delta = 4$ hours); and orange (Venezuela, $\Delta = 4$ hours).

9.4.4 Cascade Feature Utility for Forecasting

Figure 9.5 illustrates descriptive statistics of selected features of mention and follower graph cascades in Brazil. Figure 9.6 shows the variables selected by the LASSO based logistic regression model. The LASSO based model finds that the probability of an event depends upon the duration and the slope of the follower and MRT graphs. These selected features are used as explanatory variables in a generalized linear regression model [103], which confirms their significance and relevance.

Retweet-Mention Cascades	mean	stdev	median	min	max	skew	kurtosis	s.err
RT_Cascade.Duration.Median	1.02	0.13	1	1	2	6.95	48.49	0.01
RT_Cascade.Duration.Max	1.97	0.16	2	1	2	-5.74	30.97	0.01
RT_User.Slope.All.Average	5.05	1.46	4.79	1.9	16.13	2.39	10.59	0.06
RT_User.Slope.All.Max	31.53	64.97	22	4	1450	18.25	391.39	2.7
RT_User.Slope.All.Median	3.93	0.66	4	1	7	-0.73	6.67	0.03
RT_Users.In.Cascade.Average	5.8	1.55	5.53	1.9	17.53	2.03	8.64	0.06
RT_Users.In.Cascade.Max	36.36	68.81	25	4	1450	15.71	308.8	2.86
RT_Users.In.Cascade.Median	4.29	0.65	4	1	7	0.01	6.58	0.03
Total.Cascades	53.63	78.23	30	8	1077	7.44	75.49	3.26
Follower Graph Cascades								
F_Cascade.Duration.Max	2.93	2.33	2	1	18	3.44	13.38	0.1
F_Cascade.Duration.Median	1.6	1.29	1	1	10	3.39	13.64	0.05
F_User.Slope.Last.Day.Average	70.4	131.08	9.96	1.77	1361.1	4.12	25.75	5.46
F_User.Slope.Last.Day.Max	174.69	267.86	51	4	2662	3.51	19.43	11.15
F_User.Slope.Last.Day.Median	69.18	150.59	4	1	1736	4.75	34.47	6.27
F_Users.In.Cascade.Average	240.14	650.57	15.84	1.95	4798.1	4.9	26.77	27.08
F_Users.In.Cascade.Max	577.97	1289.6	78	4	9120	4.37	22.02	53.69
Total.Cascades	1106.2	2517.2	239	22	20651	5.1	30.21	104.8

Figure 9.5: **Descriptive statistics of selected features (Brazil) for the MRT and F models.** The names in the first column consist of the name of the structural feature (i.e., cascade size, duration or slope, which is the incremental increase in the size per day), and the statistical operations (i.e. median, average etc.).

Lasso Variables and Coefficients	
(Intercept)	-0.4622
MRT_Cascade.Duration.Max	0.0143
MRT_User.Slope.All.Average	0.1253
F_User.Slope.Last.Day.Max	0.0009
F_User.Slope.Last.Day.Median	0.0008

Figure 9.6: **Variables selected by LASSO in the cascade model** for Brazil, for a training period of November 2012 through May 2013.

9.4.5 Comparison Against Baseline Models

9.4.5.1 Baseline model

We build a baseline model in order to set a benchmark and to measure the added predictability provided by the Twitter data. The baseline model uses no external input in the regression model. It uses an autoregressive logistic model of order 1, since we have a binary dependent variable. It uses lagged values of itself as the predictor. Formally, we estimate $Y_t = \alpha + \beta Y_{t-1} + \varepsilon$ where Y_t is the binary variable: 1, if there is an event on day t in the GSR; 0 otherwise. The fitted values of Y_t, which give the likelihood of future events, are compared against the actual events in the GSR for measuring the model's performance.

		#Event	Threshold	Match	TPR	FPR	ACC.	Brier	ROC Area
Brazil	Baseline	4	0.25	15	0.25	0.18	0.71	0.27	0.54
	Model - Tweet Volume	4	0.3	10	0.75	0.59	0.48	0.2	0.44
	Model - Cascade	4	0.33	16	0.5	0.17	0.76	0.24	0.74
Mexico	Baseline	19	0.71	17	0.89	1	0.81	0.13	0.45
	Model - Tweet Volume	19	0.61	15	0.68	0	0.71	0.1	0.79
	Model - Cascade	19	0.49	20	1	0.5	0.95	0.1	0.92
Venezuela	Baseline	6	0.3	10	0.16	0.40	0.48	0.3	0.383
	Model - Tweet Volume	6	0.38	15	0.66	0.27	0.71	0.26	0.75
	Model - Cascade	6	0.37	16	0.83	0.26	0.76	0.26	0.81

Threshold: The probability above which an event is predicted to occur
Match: TP+TN
TPR,FPR: True positive rate, false positive rate
Accuracy (ACC): Match/#Days

Figure 9.7: **Performance of the models.** The cascades model has notably better predictive performance than the baseline and the model based only on tweet volume.

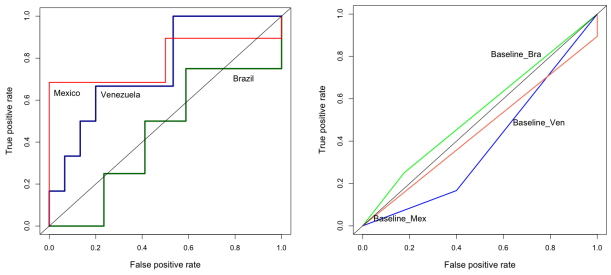


Figure 9.8: **Performance compared to baselines.** ROC curves for (a) volume-based model, (b) baseline model. Training period November 1, 2012 to November 9, 2013; test period November 10, 2013 to November 30, 2013.

9.4.5.2 Evaluation metrics

Once the probabilities are estimated for the test days, a threshold t is used to determine whether or not the probability exceeds t for an event to occur. The optimal threshold t^* is determined by cross-validation, especially maximizing the area under the ROC (receiver operating characteristic) curve. Once learned, t^* is further used to separate events from non-events given the estimated probabilities. We evaluate our models against two settings—a lead time of 1 day and a lead time of 2 days—and compare the results against the GSR using standard measures such as precision, recall, and the misclassification rate.

Figure 9.7 compares the performance of the baseline model, volume-based model and the cascade model for the three countries. For each model, we report the threshold used, true positive rate (TPR), false positive rate (FPR), accuracy (ACC), brier score, and the area under the ROC curve. The results in Figure 9.7 report the threshold that results in the highest accuracy in prediction.

Note that the cascade model outperforms both the baseline model and the volume-based model. Figure 9.8 shows the ROC for these models. Each point in the line represents a different threshold for the model.

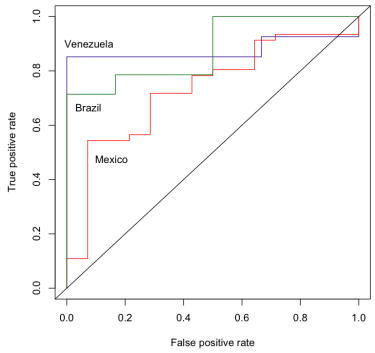


Figure 9.9: **Performance across different countries.** ROC curves for Mexico, Brazil and Venezuela for the cascade model. Training period November 1, 2012 to November 9, 2013; test period November 10, 2013 to November 30, 2013.

Model Robustness Across Countries: Figure 9.7 shows the performance of the cascade model for Brazil, Venezuela and Mexico. For Mexico, there are 20 matches out of 21 prediction days which results in 95% accuracy. On the other hand, the cascade model results in 76% accuracy for Venezuela and Brazil. Figure 9.9 illustrates the ROC plots for each of the countries at various thresholds confirming Brazil and Venezuela's performance to be worse than Mexico.

9.4.6 Forecasting the Unexpected: The Brazilian Spring

In a recent wave of uprisings in Brazil, known as the Brazilian spring, demonstrations were organized to protest increases in bus, train, and metro ticket prices in some Brazilian cities, which quickly grew to become Brazil's largest unrest since 1992. These events involved the "General Population." We test the performance of our cascade-based prediction model by making a retrospective forecast for the events occurred in the month of June 2013 in Brazil. In the training period (November 01, 2012 to May 30, 2013), there were 131 days (out of 212) with events that involved the general population. In the test period of June 2013, there were events almost every day (29 days out of 30). The total number of events was more than 29, since there were multiple events on some days.

For this experiment, we collected 83 million tweets between November 2012 and June 2013 from Brazil. The keyword-based filtering (select if a tweet has at least 3 keywords present) resulted in 890,000 tweets which were further used to generate the graphs and the cascades.

The graph-based features were extracted for each of the cascade-based models. Figure 9.10 displays the performance of the cascade model for Brazil in June 2013. The model results in an area of 0.86, showing good performance. However, ROC does well when the number of events is very high. Therefore, we also plot the probabilities obtained from the regression model for the test period. Note that the peaks correspond to the days when the events

become nationwide and violent. Figure 9.11 highlights the sudden surge in the structural features of the cascades. The cascade model results in 25 matches out of 26 alerts (when the best threshold is chosen as 0.6), a performance accuracy of 0.83 and TPR of 0.86.

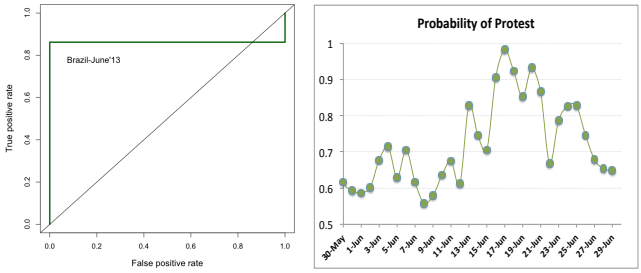


Figure 9.10: **Performance of the cascades model on the Brazilian Spring.** Our model is able to capture the rise in the number of protests during June 2013, indicating a high probability of protest for most days of the month.

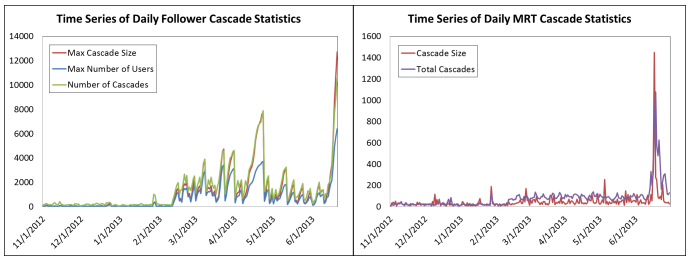


Figure 9.11: **Time series of cascade properties.** Cascade size, number of users, and number of cascades for Follower and MRT cascades in Brazil for the period November 2012 - June 2013.

9.5 Related work

Analyzing traffic trends in Twitter and other social media sites is a very active topic of research. Some of the specific applications include identifying specific news stories [34, 217], tracking natural disasters [215], predicting stock market moves [177, 201], and understanding political or cultural events [42, 97, 179, 249]. Yang et al. [265] predict temporal patterns in the usage of specific hashtags in social media data. Hutto et al. [115] show that increases in followers on Twitter are predicated on social behavior, message content, and social network structure variables in roughly equal proportions. Hsieh et. al. [110] demonstrated that experts could not match the crowd in identifying future interesting news stories. Most of these works have focused on counts of keywords and hashtags, and they do not capture peer influence in the use of such terms. Peer influence is often modeled by diffusion processes, such as linear threshold and SI/SIS/SIR epidemic models [130, 157]. In this context, cascades

are used to refer to the (random) subgraph on which the diffusion spreads. There has also been a lot of work on using semantic information for attributing influence more carefully [4, 28, 151, 236].

A simpler notion of cascades is studied by [42, 97] in Twitter follower graphs and by [84] in the mentions graphs. Large cascades involving protest-related hashtags are found to occur infrequently. Our formulations extend point process models, which have been studied extensively. Two closely related approaches are by [227, 271]. Simma et al. [227] consider a model in which a Poisson process triggers other Poisson processes. They develop an EM algorithm to infer the random forest of events, which captures the cause of each event. Zhou et al. [271] use a multi-dimensional Hawkes process.

There are other works that utilize models for characterization and prediction. Linear regression models using average tweet rates and tweet rate time series (per-day tweet rates over a 7-day period), have been used to predict box-office revenues from movies [25]. A classifier and hidden Markov model have been used with tweet content to establish the onset and end of identified events (versus event prediction) [117]. Natural language processing and topic modeling have been used to identify topics that capture collections of events found in tweets; a linear regression model is then used to predict crimes [257].

With respect to forecasting social unrest, [153] provides empirical data showing that increases in food prices correlate with protests in 2008 and 2011. Rather than predicting specific unrest events, [45] uses a 2-parameter dynamic model to predict the distributions of numbers of unrest events per year for many regions of the world. Disease outbreaks, deaths, and riots are forecasted with topic detection and tracking using news articles, and a Bayes scheme to compute the probability of some event, given other events occurring beforehand [206]. A tension parameter, based on hashtag usage, was shown to correlate well with clashes in Egypt between secularists and Islamists [259]. A generalized least squares model of political violence [35] is used to predict the overall level of violent activity in a country, by year. By contrast, we are interested in violent and non-violent protests.

Network characteristics and spectral bounds have been used for analyzing epidemic spread in networks. Ganesh et al. [85] develop necessary and sufficient conditions for the duration of an SIS process; our analysis strongly builds on this approach, but our model requires the use of a variant of the node expansion, instead of edge expansion. Similar spectral radius bounds are also considered in [199, 247] for the SIS process.

Our work can be differentiated from the above studies in the following ways. This is the first work of which we are aware that predicts daily civil unrest events in multiple countries using a combination of different graph cascade characteristics. Further, we explain theoretically and demonstrate empirically conditions that delineate small and large cascade regimes, using spectral properties of the underlying graphs.

9.6 Conclusions

Our main contributions in this chapter include: (i) a detailed analysis of activity cascades arising from protest-related tweets, (ii) use of cascade features for a predictive model for protest events, (iii) a rigorous formulation to explain the regimes for small and large cascades, in terms of the spectral radius and the node expansion, and (iv) characterizing critical sets for cascades, by means of the CSSP and CSFP formulations.

Our results suggest that, despite their simplified notion, activity cascades are useful in characterizing and predicting civil unrest events. Our rigorous characterization of the conditions for having large cascades highlights the role of the overall network structure; this corroborates with other recent work on influence cascades [28].

Chapter 10

Conclusions

Finding interesting subgraphs with constraints is a primitive that appears in many different forms when analyzing network data. Among other examples, graph anomaly detection can be posed as finding a subgraph that is unusual or unexpected compared to past data or some assumed baseline process. Dense subgraph mining in its many variants involves finding a subgraph with as high edge density as possible. And frequent subgraph mining is the problem of finding a subgraph that appears many times in a network database.

However, finding constrained subgraphs in complex networks is computationally harder than in other datasets, such as spatial data or transaction records, where one can exploit certain properties, such as low dimensionality or geometry. In turn, addressing these difficult graph problems has been at the forefront of the algorithm design and theoretical computer science community for several decades with many research hours devoted to discovering algorithms with good theoretical bounds. However, scalability and ease-of-implementation have not always been primary considerations when searching for strong theoretical bounds. On the other hand, for most practical applications, we need algorithms that scale to networks of more than a few thousand nodes—much bigger datasets with millions of nodes are common nowadays. This has led to a line of work in heuristics with fast running times and good empirical performance.

Our work addresses this gap between scalability and approximation guarantees for various subgraph finding problems. We have proposed algorithms for graph anomaly detection—scan statistics optimization, in particular—dense subgraph mining, motif finding in parallel, finding critical sets for epidemic spread, and characterizing information diffusion. We leverage on techniques from parameterized complexity—color-coding and multilinear detection—semidefinite programming, submodularity optimization with connectivity, and spectral graph theory. Traditionally, these techniques have been under-explored in data mining problems partly because of their limited scalability. For example, FPT algorithms scale exponentially with the size of the subgraph discovered, and SDP algorithms have super-quadratic running time and memory footprint. In order to make these techniques scalable, additional steps

need to be taken. For scan statistics optimization, we exploit a monotonicity property of the functions to compress parts of the network and discover large solutions while keeping the exponential parameter small. For other problems, it has been possible to exploit local structure. For example, in our dense subgraph mining work, we solve a semidefinite program on ego-networks, which are usually much smaller than the whole graph. Similarly, we exploit locality on our work on finding critical sets. Parallelization is yet another tool to scale graph algorithms to larger datasets.

10.1 Open Questions

We have touched only a tiny subset of problems that involve finding subgraphs, but we hope that this dissertation motivates future work on algorithms with guarantees for network analysis and mining. Even in the problems discussed in this dissertation, there are still many avenues for future work. We mention just a few below.

1. **Parameterized complexity:** We have used fixed-parameter tractability to obtain fast algorithms with good bounds for many scan statistics from the literature. The widely used Kulldorff statistic [144] and the functions proposed by Neill [181, 182, 183, 186, 188] that satisfy the LTSS property can be optimized over graphs with our algorithms. However, there are other functions in the literature that are not amenable to these methods. Challenges arise if the scan statistic does not satisfy the monotonicity property or if we cannot decompose the computation into subproblems. In particular, functions that rely on an ordering of the nodes in the subgraph or require an "all-pairs" computation cannot be optimized with our current algorithms. For example, we have the Simes' modified Bonferroni test [226]:

   $$F(S) = \min_{i=1}^{|S|} \frac{|S|}{i} p(v_i),$$

   where v_i is the vertex with the i^{th} smallest p-value in S. Computing this function requires keeping track of the order of the elements in S with respect to their p-values, which severely hinders scalability. We leave the existence of scan statistics that are not fixed-parameter tractable as an open question.

   On a more general note, our scan statistics work here is one of the few to apply parameterized complexity to a data mining problem in practice—the other being motif finding [11, 228]. An interesting question is what other problems are amenable to FPT algorithms. There are positive results for the aforementioned motif finding problem and negative results for finding cliques [70].

2. **Dense subgraph mining:** We have proposed a generalized notion of the Optimal Quasiclique problem to find dense subgraphs in signed networks—i.e., when edges

can have negative weights. One advantage of this formulation is that negative edges decrease density; an interpretation inspired by physical density would be that these edges represent "negative" mass. However, it is somewhat unsatisfying that we consider a graph to be denser if it has less negative edges even if the graph has a lower edge count. For example, a subgraph with no edges at all would be much denser than a clique of negative-weighted edges according to our definition. We posit that there could be other more "natural" definitions of signed density that avoid this problem.

Additionally, the hardness of the OQC problem on unsigned unweighted networks remains open.

3. **Submodularity with connectivity constraints:** This is a difficult problem that remains open. The current best approximation in general graphs is a $O(\sqrt{k})$ factor [149], and our conjecture is that any improvement would require new theoretical ideas. It seems more likely to make progress on particular instances by exploiting the structure of particular functions or graphs, similar to how Krause et al. [142] exploit locality. In general, if the diminishing returns property in the submodular function is not very strong—i.e., the function is nearly modular—then, algorithms based on Steiner connectivity should provide good bounds. One promising angle to explore is combining the notion of *curvature* [255] of a submodular function with Steiner connectivity to derive rigorous bounds.

4. **Uncertainty:** An important direction of work is on handling constraints when there is uncertainty on the input data. How do we preserve connectivity in the solution if we are not sure that the connecting edges are there? A recent survey of Kassiano et al. [127] gives an overview on methods for mining uncertain graphs. One approach to model uncertainty is assigning probabilities to the edges; the weight of an edge is the probability that the graph has that edge. Then, one could extend problems like dense subgraph mining or clustering to this setting by finding a graph of maximal probability. We consider a different setting to incorporate uncertainty in scan statistics optimization (Chapter 5). We assume that the node observations are noisy—represented as samples from some distribution—but there is no uncertainty in graph structure. Assuming that we can sample from the observation distribution, we optimize the scan statistic on expectation or optimize the lowest possible score. These models from stochastic optimization could be used in other settings as well.

We end the dissertation on that note, expecting to see continuous advances on finding interesting subgraphs with guarantees in this relatively young field of network analysis and mining.

Bibliography

[1] Measles (rubeola). `https://www.cdc.gov/measles/cases-outbreaks.html`. Accessed: 2017-11.

[2] E. Abdelhamid, I. Abdelaziz, P. Kalnis, Z. Khayyat, and F. Jamour. Scalemine: Scalable parallel frequent subgraph mining in a single large graph. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, page 11. ACM, 2016.

[3] J. Abello, M. G. Resende, and S. Sudarsky. Massive quasi-clique detection. In *LATIN 2002: Theoretical Informatics*, pages 598–612. Springer, 2002.

[4] E. Adar and L. Adamic. Tracking information epidemics in blogspace. In *IEEE/WIC/ACM International Conference on Web Intelligence*, 2005.

[5] D. Agarwal, A. McGregor, J. M. Phillips, S. Venkatasubramanian, and Z. Zhu. Spatial scan statistics: approximations and performance study. In *KDD*, 2006.

[6] C. Aggarwal, Y. Zhao, and P. Yu. Outlier detection in graph streams. In *ICDE*, 2011.

[7] L. Akoglu and C. Faloutsos. Event detection in time series of mobile communication graphs. In *Army Science Conference*, pages 77–79, 2010.

[8] L. Akoglu, M. McGlohon, and C. Faloutsos. Oddball: Spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining*, pages 410–421. Springer, 2010.

[9] L. Akoglu, H. Tong, and D. Koutra. Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, 2014.

[10] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning. In *SDM06: workshop on link analysis, counter-terrorism and security*, 2006.

[11] N. Alon, P. Dao, I. Hajirasouliha, F. Hormozdiari, and S. C. Sahinalp. Biomolecular network motif counting and discovery by color coding. *Bioinformatics*, 2008.

[12] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM (JACM)*, 1995.

[13] G. Alves and Y.-K. Yu. Accuracy evaluation of the unified p-value from combining correlated p-values. *PLoS ONE*, 2014.

[14] R. Andersen and K. Chellapilla. Finding dense subgraphs with size bounds. In *Algorithms and Models for the Web-Graph*, pages 25–37. Springer, 2009.

[15] R. Anderson and R. May. *Infectious Diseases of Humans*. Oxford University Press, Oxford, 1991.

[16] R. M. Anderson, R. M. May, and B. Anderson. *Infectious diseases of humans: dynamics and control*, volume 28. Wiley Online Library, 1992.

[17] A. Angel, N. Sarkas, N. Koudas, and D. Srivastava. Dense subgraph maintenance under streaming edge weight updates for real-time story identification. *Proceedings of the VLDB Endowment*, 5(6):574–585, 2012.

[18] T. Antal, P. L. Krapivsky, and S. Redner. Social balance on networks: The dynamics of friendship and enmity. *Physica D: Nonlinear Phenomena*, 224(1):130–136, 2006.

[19] D. Aparicio, P. Ribeiro, and F. A. da Silva. Parallel subgraph counting for multicore architectures. In *IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA)*, 2014.

[20] M. Araujo, S. Papadimitriou, S. Günnemann, C. Faloutsos, P. Basu, A. Swami, E. E. Papalexakis, and D. Koutra. Com2: fast automatic discovery of temporal (?comet?) communities. In *Advances in Knowledge Discovery and Data Mining*, pages 271–283. Springer, 2014.

[21] E. Arias-Castro, E. J. Candès, and A. Durand. Detection of an anomalous cluster in a network. *The Annals of Statistics*, pages 278–304, 2011.

[22] S. Arifuzzaman, M. Khan, and M. Marathe. Patric: A parallel algorithm for counting triangles in massive networks. In *Proc. CIKM*, 2013.

[23] Y. Asahiro, R. Hassin, and K. Iwama. Complexity of finding dense subgraphs. *Discrete Applied Mathematics*, 121(1):15–26, 2002.

[24] Y. Asahiro, K. Iwama, H. Tamaki, and T. Tokuyama. Greedily finding a dense subgraph. *Journal of Algorithms*, 34(2):203–221, 2000.

[25] S. Asur and B. A. Huberman. Predicting the future with social media. In *WI-IAT*, 2010.

[26] E. Awini, P. Mattah, O. Sankoh, and M. Gyapong. Spatial variations in childhood mortalities at the dodowa health and demographic surveillance system site of the indepth network in ghana. *Tropical Medicine & International Health*, 15(5):520–528, 2010.

[27] N. T. Bailey et al. The mathematical theory of epidemics. 1957.

[28] E. Bakshy, J. Hofman, W. Mason, and D. Watts. Everyone's an influencer: Quantifying influence on twitter. In *Proc. of the fourth ACM international conference on Web search and data mining (WSDM)*, 2011.

[29] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st international conference on World Wide Web*, pages 519–528. ACM, 2012.

[30] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[31] S. Bansal, J. Read, B. Pourbohloul, and L. A. Meyers. The dynamic nature of contact networks in infectious disease epidemiology. *Journal of Biological Dynamics*, 4(5):478–489, 2010.

[32] C. L. Barrett, R. J. Beckman, M. Khan, V. S. Anil Kumar, M. V. Marathe, P. E. Stretz, T. Dutta, and B. Lewis. Generation and analysis of large synthetic social contact networks. In *Winter Simulation Conference*, pages 1003–1014. Winter Simulation Conference, 2009.

[33] A. Barron, J. Rissanen, and B. Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

[34] H. Becker, M. Naaman, and L. Gravano. Beyond trending topics: Real-world event identification on twitter. In *ICWSM*, 2011.

[35] S. Bell, D. Cingranelli, A. Murdie, and A. Caglayan. Coercion, capacity, and coordination: Predictors of political violence. *Conflict Management and Peace Science*, 2013.

[36] R. H. Berk and D. H. Jones. Goodness-of-fit test statistics that dominate the kolmogorov statistics. *Z. Wahrsch. Verw. Gebiete*, 1979.

[37] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 1995.

[38] K. Bisset, J. Chen, X. Feng, A. Vullikanti, and M. Marathe. EpiFast: A fast algorithm for large-scale realistic epidemic simulations on distributed memory systems. In *Proceedings of the 23rd ACM International Conference on Supercomputing (ICS)*, 2009.

[39] P. Bogdanov, N. D. Larusso, and A. Singh. Towards community discovery in signed collaborative interaction networks. In *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pages 288–295. IEEE, 2010.

[40] P. Bogdanov, M. Mongiovì, and A. Singh. Mining heavy subgraphs in time-evolving networks. In *ICDM*, 2011.

[41] I. M. Bomze, M. Budinich, P. M. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of combinatorial optimization*, pages 1–74. Springer, 1999.

[42] J. Borge-Holthoefer, A. Rivero, and Y. Moreno. Locating privileged spreaders on an online social network. *Physical Review E*, 2012.

[43] C. Borgelt and M. R. Berthold. Mining molecular fragments: Finding relevant substructures of molecules. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 51–58. IEEE, 2002.

[44] F. P. Boscoe, T. O. Talbot, and M. Kulldorff. Public domain small-area cancer incidence data for new york state, 2005-2009. *Geospatial health*, 11(1), 2016.

[45] D. Braha. Global civil unrest: Contagion, self-organization, and prediction. *PLoS One*, 2012.

[46] N. Buchbinder, M. Feldman, J. Seffi, and R. Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5):1384–1402, 2015.

[47] J. Cadena, F. Chen, and A. Vullikanti. Near-optimal and practical algorithms for graph scan statistics. In *SIAM Data Mining (SDM)*, 2017.

[48] J. Callut, K. Françoisse, M. Saerens, and P. Dupont. Semi-supervised classification from discriminative random walks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 162–177. Springer, 2008.

[49] D. Cartwright and F. Harary. Structural balance: a generalization of heider's theory. *Psychological review*, 63(5):277, 1956.

[50] P. Chakraborty, P. Khadivi, B. Lewis, A. Mahendiran, J. Chen, P. Butler, E. Nsoesie, S. Mekaru, J. Brownstein, M. Marathe, and N. Ramakrishnan. Forecasting a moving target: Ensemble models for ILI case count predictions. In *Proc. SIAM International Conference on Data Mining (SDM)*, pages 262–270, 2014.

[51] M. Charikar. Greedy approximation algorithms for finding dense components in a graph. In *APPROX*, 2000.

[52] M. Charikar and A. Wirth. Maximizing quadratic programs: extending grothendick's inequality. In *IEEE FOCS*, 2004.

[53] F. Chen and D. Neill. Non-parametric scan statistics for disease outbreak detection on twitter. *Online jl public health informatics*, 2014.

[54] F. Chen and D. Neill. Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs. In *KDD*, 2014.

[55] F. Chen and D. B. Neill. Human rights event detection from heterogeneous social media graphs. *Big Data*, 3(1):34–40, 2015.

[56] J. Chen, A. Marathe, and M. V. Marathe. Coevolution of epidemics, social networks, and individual behavior: A case study. In S.-K. Chai, J. J. Salerno, and P. L. Mabry, editors, *SBP*, volume 6007 of *Lecture Notes in Computer Science*, pages 218–227. Springer, 2010.

[57] J. Cheng, L. Zhu, Y. Ke, and S. Chu. Fast algorithms for maximal clique enumeration with limited memory. In *Proc. SIGKDD*, 2012.

[58] K.-Y. Chiang et al. Prediction and clustering in signed networks: a local to global perspective. *The Journal of Machine Learning Research*, 15(1):1177–1213, 2014.

[59] N. Christakis and J. Fowler. Social network sensors for early detection of contagious outbreaks. *PloS one*, 5(9):e12948, 2010.

[60] F. Chung and L. Lu. Connected components in random graphs with given expected degree sequences. *Annals of Combinatorics*, 6:125–145, 2002.

[61] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[62] D. M. Conforth, T. C. Reluga, E. Shim, C. T. Bauch, A. P. Galvani, and L. A. Meyers. Erratic flu vaccination emerges from short-sighted behavior in contact networks. Under review, August 2010.

[63] R. Crane and D. Sornette. Robust dynamic classes revealed by measuring the response function of a social system. *PNAS*, 2008.

[64] J. Dai, F. Chen, S. Sahu, and M. Naphade. Regional behavior change detection via local spatial scan. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 490–493. ACM, 2010.

[65] L. Dehaspe, H. Toivonen, and R. D. King. Finding frequent substructures in chemical compounds. In *KDD*, volume 98, page 1998, 1998.

[66] Q. Ding, N. Katenka, P. Barford, E. D. Kolaczyk, and M. Crovella. Intrusion as (anti)social communication: characterization and detection. In *Proc. of the 18th ACM SIGKDD*, pages 886–894, 2012.

[67] D. Donoho and J. Jin. Higher criticism for detecting sparse heterogeneous mixtures. *The Annals of Statistics*, 2004.

[68] D. Donoho and J. Jin. Higher criticism for large-scale inference, especially for rare and weak effects. *Statist. Sci.*, 30(1), 2015.

[69] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 2012.

[70] R. G. Downey and M. R. Fellows. *Parameterized complexity*. Springer Science & Business Media, 2012.

[71] N. Du, B. Wu, L. Xu, B. Wang, and P. Xin. Parallel algorithm for enumerating maximal cliques in complex network. *Mining Complex Data*, pages 207–221, 2009.

[72] L. Duczmal, M. Kulldorff, and L. Huang. Evaluation of spatial scan statistics for irregularly shaped clusters. *Journal of Computational and Graphical Statistics*, 2006.

[73] W. Eberle and L. Holder. Graph-based approaches to insider threat detection. In *Proc. of CSIIRW*, 2009.

[74] E. Edgington. An additive method for combining probability values from independent experiments. *The Journal of Psychology*, 1972.

[75] F. Eicker. The asymptotic distribution of the suprema of the standardized empirical processes. *The Annals of Statistics*, 1979.

[76] M. Elseidy, E. Abdelhamid, S. Skiadopoulos, and P. Kalnis. Grami: Frequent subgraph and pattern mining in a single large graph. In *Proc. International Conference on Very Large Data Bases*, 2014.

[77] K. A. Eriksen, I. Simonsen, S. Maslov, and K. Sneppen. Modularity and extreme edges of the internet. *Physical review letters*, 90(14):148701, 2003.

[78] S. Eubank, H. Guclu, V. S. Anil Kumar, M. Marathe, A. Srinivasan, Z. Toroczkai, and N. Wang. Modelling disease outbreaks in realistic urban social networks. *Nature*, 429:180–184, 2004.

[79] U. Feige and M. Langberg. Approximation algorithms for maximization problems arising in graph partitioning. *J. Algorithms*, 2001.

[80] U. Feige, M. Seltser, et al. *On the densest k-subgraph problem*. Citeseer, 1997.

[81] R. Fisher. *Statistical methods for research workers*. Edinburgh Oliver & Boyd, 1925.

[82] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[83] E. Fratkin, B. T. Naughton, D. L. Brutlag, and S. Batzoglou. Motifcut: regulatory motifs finding with maximum density subgraphs. *Bioinformatics*, 22(14):e150–e157, 2006.

[84] W. Galuba, K. Aberer, D. Chakraborty, Z. Despotovic, and W. Kellerer. Outtweeting the twitterers - predicting information cascades in microblogs. In *WOSN*, 2010.

[85] A. Ganesh, L. Massoulie, and D. Towsley. The effect of network topology on the spread of epidemics. *Proceedings of INFOCOM*, 2005.

[86] M. Garey and D. Johnson. *Computers and Intractibility*. 1979.

[87] N. Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 396–402. ACM, 2005.

[88] D. J. Gerner et al. Machine coding of event data using regional and international sources. *Inter. Studies Quarterly*, pages 91–119, 1994.

[89] D. J. Gerner et al. Conflict and mediation event observations (cameo): A new event data framework for the analysis of foreign policy interactions. *International Studies Association*, 2002.

[90] J. Ginsberg et al. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2008.

[91] A. Gionis, F. Junqueira, V. Leroy, M. Serafini, and I. Weber. Piggybacking on social networks. *Proceedings of the VLDB Endowment*, 6(6):409–420, 2013.

[92] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM Journal on Computing*, 24(2):296–317, 1995.

[93] M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[94] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. *SIAM Journal of Computing*, 1997.

[95] A. V. Goldberg. *Finding a maximum density subgraph*. University of California Berkeley, CA, 1984.

[96] J. Gonzalez et al. Graphx: Graph processing in a distributed dataflow framework. In *Proc OSDI*, 2014.

[97] S. González-Bailón, J. Borge-Holthoefer, A. Rivero, and Y. Moreno. The dynamics of protest recruitment through an online network. *Scientific Reports*, 1, 2011.

[98] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, pages 491–501, 2004.

[99] S. Guillemot and F. Sikora. Finding and counting vertex-colored subtrees. *Algorithmica*, 65(4):828–844, 2013.

[100] A. Gupta and A. Kumar. Online steiner tree with deletions. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 455–467. Society for Industrial and Applied Mathematics, 2014.

[101] T. Hansen and F. Vandin. Finding mutated subnetworks associated with survival in cancer. *arXiv preprint arXiv:1604.02467*, 2016.

[102] J. Hastad. Clique is hard to approximate within n 1-&epsiv. In *Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on*, pages 627–636. IEEE, 1996.

[103] T. J. Hastie and R. J. Tibshirani. *Generalized additive models*. London: Chapman & Hall, 1990.

[104] M. B. Hastings. Community detection as an inference problem. *Physical Review E*, 74(3):035102, 2006.

[105] N. A. Heard, D. J. Weston, K. Platanioti, D. J. Hand, et al. Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics*, 4(2):645–662, 2010.

[106] F. Heider. Attitudes and cognitive organization. *The Journal of psychology*, 21(1):107–112, 1946.

[107] H. W. Hethcote and P. Van den Driessche. Some epidemiological models with nonlinear incidence. *Journal of Mathematical Biology*, 29(3):271–287, 1991.

[108] J. M. Hofman and C. H. Wiggins. Bayesian approach to network modularity. *Physical review letters*, 100(25):258701, 2008.

[109] L. B. Holder, D. J. Cook, S. Djoko, et al. Substucture discovery in the subdue system. In *KDD workshop*, pages 169–180, 1994.

[110] C.-C. Hsieh, C. Moghbel, J. Fang, and J. Cho. Expert vs the crowd: Examining popular news prediction performance on twitter. In *WWW*. ACM, 2013.

[111] C.-J. Hsieh et al. Low rank modeling of signed networks. In *Proc. of ACM SIGKDD*, pages 507–515, 2012.

[112] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*, pages 549–552. IEEE, 2003.

[113] C. Huang. Facebook and Twitter key to Arab Spring uprisings: report. In *The National*. 2011.

[114] F. Hüffner, S. Wernicke, and T. Zichner. Algorithm engineering for color-coding with applications to signaling pathway detection. *Algorithmica*, 52(2):114–132, 2008.

[115] C. J. Hutto, S. Yardi, and E. Gilbert. A longitudinal study of follow predictors on twitter. In *CHI*, 2010.

[116] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer, 2000.

[117] A. Iyengar, T. Finin, and A. Joshi. Content-based prediction of temporal boundaries for events in twitter. In *IEEE Int. Conf. on Social Computing*. IEEE, 2011.

[118] L. Jager and J. A. Wellner. Goodness-of-fit tests via phi-divergences. *The Annals of Statistics*, 2007.

[119] S. Janson, T. Luczak, and A. Rucinski. *Random graphs*. Wiley, 2000.

[120] C. Jiang, F. Coenen, and M. Zito. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, 28(01):75–105, 2013.

[121] J. Jin and T. Ke. Rare and weak effects in large-scale inference: methods and phase diagrams. *arXiv preprint arXiv:1410.4578*, 2014.

[122] D. Johnson, M. Minkoff, and S. Phillips. The prize collecting steiner tree problem: Theory and practice. In *ACM SODA*, 2000.

[123] D. S. Johnson, J. K. Lenstra, and A. Kan. The complexity of the network design problem. *Networks*, 8(4):279–285, 1978.

[124] E. Jonckheere and P. Lohsoonthorn. Geometry of network security. In *American Control Conference, 2004. Proceedings of the 2004*, volume 2, pages 976–981 vol.2, 2004.

[125] I. Jung, M. Kulldorff, and O. J. Richard. A spatial scan statistic for multinomial data. *Statistics in medicine*, 2010.

[126] H. Kashima and A. Inokuchi. Kernels for graph classification. In *ICDM Workshop on Active Mining*, volume 2002, 2002.

[127] V. Kassiano, A. Gounaris, A. N. Papadopoulos, and K. Tsichlas. Mining uncertain graphs: An overview. In *International Workshop of Algorithmic Aspects of Cloud Computing*, pages 87–116. Springer, 2016.

[128] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[129] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[130] D. Kempe, J. M. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. In *ICALP 2005*, 2005.

[131] W. O. Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proc. R. Soc. Lond. A*, 115:700–721, 1927.

[132] S. Khuller and B. Saha. On finding dense subgraphs. In *International Colloquium on Automata, Languages, and Programming*, pages 597–608. Springer, 2009.

[133] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.

[134] S. G. Kolliopoulos and N. E. Young. Approximation algorithms for covering/packing integer programs. *Journal of Computer and System Sciences*, pages 495–505, 2005.

[135] M. J. Koop, T. Jones, and D. K. Panda. Reducing connection memory requirements of mpi for infiniband clusters: A message coalescing approach. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 495–504. IEEE, 2007.

[136] I. Koutis. Faster algebraic algorithms for path and packing problems. In *Proc. ICALP*, 2008.

[137] I. Koutis. Constrained multilinear detection for faster functional motif discovery. *Information Processing Letters*, 112(22):889–892, 2012.

[138] I. Koutis and R. Williams. Limits and applications of group algebras for parameterized problems. In *International Colloquium on Automata, Languages, and Programming*, pages 653–664. Springer, 2009.

[139] I. Koutis and R. Williams. Algebraic fingerprints for faster algorithms. *Communications of the ACM*, 59(1):98–105, 2015.

[140] D. Koutra, E. E. Papalexakis, and C. Faloutsos. Tensorsplat: Spotting latent anomalies in time. In *Informatics (PCI), 2012 16th Panhellenic Conference on*, pages 144–149. IEEE, 2012.

[141] A. Krause and C. Guestrin. Beyond convexity: Submodularity in machine learning. *ICML Tutorials*, 2008.

[142] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10. ACM, 2006.

[143] S. Krishnan, P. Butler, R. Tandon, J. Leskovec, and N. Ramakrishnan. Seeing the forest for the trees: new approaches to forecasting cascades. In *Proceedings of the 8th ACM Conference on Web Science*, pages 249–258. ACM, 2016.

[144] M. Kulldorff. A spatial scan statistic. *Communications in Statistics: Theory and Methods*, 1997.

[145] M. Kulldorff, T. Tango, and P. J. Park. Power comparisons for disease clustering tests. *Computational Statistics & Data Analysis*, 2003.

[146] M. Kumar et al. Data mining to predict and prevent errors in health insurance claims processing. In *Proc. of ACM SIGKDD*, pages 65–74, 2010.

[147] V. S. A. Kumar, R. Rajaraman, Z. Sun, and R. Sundaram. Existence theorems and approximation algorithms for generalized network security games. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 348–357. IEEE, 2010.

[148] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. W. De Luca, and S. Albayrak. Spectral analysis of signed graphs for clustering, prediction and visualization. In *SDM*, volume 10, pages 559–559. SIAM, 2010.

[149] T.-W. Kuo, K. C.-J. Lin, and M.-J. Tsai. Maximizing submodular set function with connectivity constraint: Theory and application to networks. *IEEE/ACM Transactions on Networking*, 23(2):533–546, 2015.

[150] M. Kuramochi and G. Karypis. Frequent subgraph discovery. In *Proc. of IEEE Int. Conf. on Data Mining*, page 313, 2001.

[151] H. Kwak, C. Lee, H. Park, and S. Moon. What is twitter, a social network or a news media? In *WWW*, 2010.

[152] V. Lacroix, C. G. Fernandes, and M.-F. Sagot. Motif search in graphs: application to metabolic networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 3(4):360–368, 2006.

[153] M. Lagi, K. Z. Bertand, and Y. Bar-Yam. The food crises and political instability in north africa and the middle east, 2011. arXiv:1108.2455v1: 15 pages.

[154] D. Lazer, R. Kennedy, G. King, and A. Vespignani. The parable of Google Flu: Traps in big data analysis. *Science*, 343:1203–1205, 2014.

[155] V. E. Lee, N. Ruan, R. Jin, and C. Aggarwal. A survey of algorithms for dense subgraph discovery. In *Managing and Mining Graph Data*, pages 303–336. Springer, 2010.

[156] M. Leiserson et al. Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes. *Nature genetics*, 47(2):106–114, 2015.

[157] J. Leskovec, L. Adamic, and B. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 2007.

[158] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Predicting positive and negative links in online social networks. In *Proc. of WWW*, pages 641–650, 2010.

[159] J. Leskovec, D. Huttenlocher, and J. Kleinberg. Signed networks in social media. In *Proc. of SIGCHI*, pages 1361–1370, 2010.

[160] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[161] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance. Cost-effective outbreak detection in networks. In *KDD*, pages 420–429, 2007.

[162] J. Leskovec and A. Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, 2014.

[163] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *Proceedings of the 17th international conference on World Wide Web*, pages 695–704. ACM, 2008.

[164] L. Li, D. Alderson, W. Willinger, and J. Doyle. A first-principles approach to understanding the internet's router-level topology. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–14, New York, NY, USA, 2004. ACM Press.

[165] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[166] T. A. Lieu, G. T. Ray, N. P. Klein, C. Chung, and M. Kulldorff. Geographic clusters in underimmunization and vaccine refusal. *Pediatrics*, 135(2):280–289, 2015.

[167] L. Lü and T. Zhou. Link prediction in complex networks: A survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

[168] S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 8(May):935–983, 2007.

[169] N. Malizia. Inaccuracy, uncertainty and the space-time permutation scan statistic. *PLoS ONE*, 2013.

[170] F. D. Malliaros and M. Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, 533(4):95–142, 2013.

[171] M. Marathe and A. Vullikanti. Computational epidemiology. *Communications of the ACM*, 56(7):88–96, 2013.

[172] F. Margai and N. Henry. A community-based assessment of learning disabilities using environmental and contextual risk factors. *Social Science & Medicine*, 56(5):1073–1085, 2003.

[173] E. McFowland, S. Speakman, and D. B. Neill. Fast generalized subset scan for anomalous pattern detection. *JMLR*, 14(1), 2013.

[174] B. Miller, N. Bliss, and P. J. Wolfe. Subgraph detection using eigenvector l1 norms. In *Advances in Neural Information Processing Systems*, pages 1633–1641, 2010.

[175] S. Milne. Egypt, Brazil, Turkey: without politics, protest is at the mercy of the elites. In *The Guardian*. 2013.

[176] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques*, pages 234–243, 1978.

[177] H. S. Moat, C. Curme, A. Avakian, D. Y. Kenett, H. E. Stanley, and T. Preis. Quantifying wikipedia usage patterns before stock market moves. *Scientific reports*, 3, 2013.

[178] M. Mongiovì, P. Bogdanov, R. Ranca, A. Singh, E. Papalexakis, and C. Faloutsos. Netspot: Spotting significant anomalous regions on dynamic networks. In *SDM*, 2013.

[179] A. Morales, J. Losada, and R. Benito. Users structure and behavior on an online social network during a political protest. *Physica A*, pages 5244–5253, 2012.

[180] G. L. Mullen and C. Mummert. Finite fields and applications, volume 41 of student mathematical library. *American Mathematical Society, Providence, RI*, 3:19–20, 2007.

[181] D. Neil, A. Moore, and G. Cooper. A Bayesian spatial scan statistic. In *NIPS*, 2005.

[182] J. Neil, C. Hash, A. Brugh, M. Fisk, and C. B. Storlie. Scan statistics for the online detection of locally anomalous subgraphs. *Technometrics*, 55(4):403–414, 2013.

[183] D. B. Neill. Fast and flexible outbreak detection by linear-time subset scanning. *Advances in Disease Surveillance*, 2008.

[184] D. B. Neill. An empirical comparison of spatial scan statistics for outbreak detection. *International Journal of Health Geographics*, 2009.

[185] D. B. Neill. Fast subset scan for spatial pattern detection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2012.

[186] D. B. Neill and G. F. Cooper. A multivariate bayesian scan statistic for early event detection and characterization. *Machine learning*, 2010.

[187] D. B. Neill, G. F. Cooper, K. Das, X. Jiang, and J. Schneider. Bayesian network scan statistics for multivariate pattern detection. In *Scan Statistics*. 2009.

[188] D. B. Neill and J. Lingwall. A nonparametric scan statistic for multivariate disease surveillance. *Advances in Disease Surveillance*, 2007.

[189] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming*, 14(1):265–294, 1978.

[190] M. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[191] M. E. Newman. Spectral methods for community detection and graph partitioning. *Physical Review E*, 88(4):042822, 2013.

[192] M. E. Newman and E. A. Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences*, 104(23):9564–9569, 2007.

[193] S. Nijssen and J. N. Kok. The gaston tool for frequent subgraph mining. *Electronic Notes in Theoretical Computer Science*, 127(1):77–87, 2005.

[194] C. C. Noble and D. J. Cook. Graph-based anomaly detection. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636. ACM, 2003.

[195] A. Ostfeld et al. The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management*, 2008.

[196] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[197] A. Pati, C. Vasquez-Robinet, L. S. Heath, R. Grene, and T. M. Murali. Xcisclique: Analysis of regulatory bicliques. *BMC Bioinformatics*, 7:14 pages, 2006.

[198] L. Peel and A. Clauset. Detecting change points in the large-scale structure of evolving networks, 2014. CoRR, abs/1403.0989.

[199] B. A. Prakash, D. Chakrabarti, M. Faloutsos, N. Valler, and C. Faloutsos. Threshold conditions for arbitrary cascade models on arbitrary networks. In *ICDM*, 2011.

[200] B. A. Prakash, J. Vrekeen, and C. Faloutsos. Spotting culprits in epidemics : How many and which ones? In *Proc. ICDM*, 2012.

[201] T. Preis, H. S. Moat, and H. E. Stanley. Quantifying trading behavior in financial markets using google trends. *Scientific reports*, 3, 2013.

[202] A. Prékopa. *Stochastic programming*, volume 324. Springer Science & Business Media, 2013.

[203] C. E. Priebe, J. M. Conroy, D. J. Marchette, and Y. Park. Scan statistics on enron graphs. *Computational & Mathematical Organization Theory*, 11(3):229–247, 2005.

[204] N. Pržulj. Biological network comparison using graphlet degree distribution. *Bioinformatics*, 23(2):e177, 2007.

[205] J. Qian, V. Saligrama, and Y. Chen. Connected sub-graph detection. In *AISTATS*, 2014.

[206] K. Radinsky and E. Horvitz. Mining the web to predict future events. In *WSDM*, 2013.

[207] N. Ramakrishnan et al. 'beating the news' with embers: Forecasting civil unrest using open source indicators. In *Proc. of ACM SIGKDD*, pages 1799–1808, 2014.

[208] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi. Spanning treesshort or small. *SIAM Journal on Discrete Mathematics*, 9(2):178–200, 1996.

[209] K. E. Read. Cultures of the central highlands, new guinea. *Southwestern Journal of Anthropology*, pages 1–43, 1954.

[210] J. Reichardt and S. Bornholdt. Partitioning and modularity of graphs with arbitrary degree distribution. *Physical Review E*, 76(1):015102, 2007.

[211] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, 2009.

[212] P. Rozenshtein, A. Anagnostopoulos, A. Gionis, and N. Tatti. Event detection in activity networks. In *KDD*, 2014.

[213] S. Saha, A. Adiga, B. A. Prakash, and A. K. S. Vullikanti. Approximation algorithms for reducing the spectral radius to control epidemic spread. In *Siam Data Mining (SDM)*, 2015.

[214] S. Saha, A. Adiga, and A. K. S. Vullikanti. Equilibria in epidemic containment games. In *The 28th AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[215] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: Real-time event detection by social sensors. In *WWW*, 2010.

[216] J. B.-H. Sandra Gonzlez-Bailn and Y. Moreno. Broadcasters and hidden influentials in online protest diffusion. *American Behavioral Scientist*, pages 943–965, 2013.

[217] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: News in tweets. In *ACM GIS*, 2009.

[218] M. Schmidt, N. Samatova, K. Thomas, and B. Park. A scalable, parallel algorithm for maximal clique enumeration. *Journal of Parallel and Distributed Computing*, 69(4):417–428, 2009.

[219] J. Scott, T. Ideker, R. M. Karp, and R. Sharan. Efficient algorithms for detecting signaling pathways in protein interaction networks. *Journal of Computational Biology*, 13(2):133–144, 2006.

[220] R. Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of computer and system sciences*, 51(3):400–403, 1995.

[221] S. Shanbhag and T. Wolf. Massively parallel anomaly detection in online network measurement. In *Proc. of 17th International Conference on Computer Communications and Networks (ICCCN)*, 2008.

[222] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on stochastic programming: modeling and theory*. SIAM, 2009.

[223] J. Sharpnack, A. Krishnamurthy, and A. Singh. Near-optimal anomaly detection in graphs using lovasz extended scan statistic. In *NIPS*, 2013.

[224] J. Sharpnack, A. Singh, and A. Rinaldo. Sparsistency of the edge lasso over graphs. In *AISTATS*, 2012.

[225] J. Sharpnack, A. Singh, and A. Rinaldo. Changepoint detection over graphs with the spectral scan statistic. In *AISTATS*, 2013.

[226] R. J. Simes. An improved Bonferroni procedure for multiple tests of significance. *Biometrika*, 1986.

[227] A. Simma and M. I. Jordan. Modeling events with cascades of poisson processes. In *UAI*, 2010.

[228] G. M. Slota and K. Madduri. Fast approximate subgraph counting and enumeration. In *Proc. ICPP*, 2013.

[229] G. M. Slota and K. Madduri. Complex network analysis using parallel approximate motif counting. In *Proc. IPDPS*, 2014.

[230] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 939–948. ACM, 2010.

[231] S. Speakman et al. Scalable detection of anomalous patterns with connectivity constraints. *Jl Comp Graphical Stat*, 2015.

[232] S. Speakman and D. B. Neill. Fast graph scan for scalable detection of arbitrary connected clusters. *Advances in Disease Surveillance*, 2010.

[233] S. Speakman, Y. Zhang, and D. B. Neill. Dynamic pattern detection with temporal consistency and connectivity constraints. In *ICDM*, 2013.

[234] S. Speakman, Y. Zhang, and D. B. Neill. Dynamic pattern detection with temporal consistency and connectivity constraints. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 697–706. IEEE, 2013.

[235] S. A. Stouffer, E. A. Suchman, L. C. DeVinney, S. A. Star, and R. M. Williams. *The American Soldier. Adjustment During Army Life.* Princeton University Press, 1949.

[236] E. Sun, I. Rosenn, C. Marlow, and T. Lento. Modeling contagion through facebook news feed. In *ICWSM*, 2009.

[237] J. Sun, C. Faloutsos, S. Papadimitriou, and P. S. Yu. Graphscope: parameter-free mining of large time-evolving graphs. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 687–696. ACM, 2007.

[238] J. Sun, H. Qu, D. Chakrabarti, and C. Faloutsos. Neighborhood formation and anomaly detection in bipartite graphs. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 8–pp. IEEE, 2005.

[239] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Sparse graph mining with compact matrix decomposition. *Statistical Analysis and Data Mining*, 1(1):6–22, 2008.

[240] C. Swamy and D. Shmoys. Algorithms column: Approximation algorithms for 2-stage stochastic optimization problems. *SIGACT News*, 2006.

[241] K. Takahashi, M. Kulldorff, T. Tango, and K. Yih. A flexibly shaped space-time scan statistic for disease outbreak detection and monitoring. *International Journal of Health Geographics*, 2008.

[242] J. Tang, S. Chang, C. Aggarwal, and H. Liu. Negative link prediction in social media. In *Proc. of ACM WSDM*, pages 87–96, 2015.

[243] J. Tang, Y. Chang, C. Aggarwal, and H. Liu. A survey of signed network mining in social media. *ACM Comput. Surv.*, 49(3):42:1–42:37, Aug. 2016.

[244] L. Tang and H. Liu. Community detection and mining in social media. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2(1):1–137, 2010.

[245] A. Terras. *Fourier analysis on finite groups and applications*. Number 43. Cambridge University Press, 1999.

[246] R. Tibshirani. Regression shrinkage and selection via the lasso: a retrospective. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(3):273–282, 2011.

[247] H. Tong, B. A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *CIKM*, 2012.

[248] V. Traag and J. Bruggeman. Community detection in networks with positive and negative links. *Physical Review E*, 80(036115):1–6, 2009.

[249] M. Tremayne. Anatomy of protest in the digital era: A network analysis of twitter and occupy wall street. *Social Movement Studies: Journal of Social, Cultural and Political Protest*, pages 110–126, 2013.

[250] C. Tsourakakis. The k-clique densest subgraph problem. In *Proc. of WWW*, pages 1122–1132, 2015.

[251] C. Tsourakakis et al. Denser than the densest subgraph: extracting optimal quasi-cliques with quality guarantees. In *Proc. of ACM SIGKDD*, pages 104–112, 2013.

[252] J. R. Ullmann. An algorithm for subgraph isomorphism. *Journal of the ACM (JACM)*, 23(1):31–42, 1976.

[253] 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.

[254] P. Vaneckova, P. J. Beggs, and C. R. Jacobson. Spatial analysis of heat-related mortality among the elderly between 1993 and 2004 in sydney, australia. *Social science & medicine*, 70(2):293–304, 2010.

[255] J. Vondrák. Submodularity and curvature: The optimal algorithm (combinatorial optimization and discrete algorithms). 2010.

[256] B. Wang, J. M. Phillips, R. Schreiber, D. M. Wilkinson, N. Mishra, and R. Tarjan. Spatial scan statistics for graph clustering. In *SDM*, pages 727–738. SIAM, 2008.

[257] X. Wang, M. S. Gerber, and D. E. Brown. Automatic crime prediction using events extracted from twitter posts. In *SBP*, 2012.

[258] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[259] I. Weber, V. R. K. Garimella, and A. Batayneh. Secular vs. islamist polarization in egypt on twitter. In *ASONAM*, 2013.

[260] R. Wilcox. Kolmogorov–smirnov test. *Encyclopedia of biostatistics*, 2005.

[261] R. Williams. Finding paths of length k in o(k2) time. *Information Processing Letters*, 109(6):315–318, 2009.

[262] D. P. Williamson and D. B. Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[263] M. Wörlein, T. Meinl, I. Fischer, and M. Philippsen. A quantitative comparison of the subgraph miners mofa, gspan, ffsm, and gaston. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 392–403. Springer, 2005.

[264] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 721–724. IEEE, 2002.

[265] J. Yang and J. Leskovec. Patterns of temporal variation in online media. In *WSDM*, 2011.

[266] J. Yang, J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th International Conference on Data Mining*, pages 1151–1156. IEEE, 2013.

[267] M. J. Zaki and W. Meira Jr. *Data mining and analysis: fundamental concepts and algorithms*. Cambridge University Press, 2014.

[268] A. Zeoli et al. Homicide as infectious disease: Using public health methods to investigate the diffusion of homicide. *Justice quarterly*, 31(3):609–632, 2014.

[269] J. Zhao, J. Li, B. Zhou, F. Chen, P. Tomchik, and W. Ju. Parallel algorithms for anomalous subgraph detection. *Concurrency and Computation: Practice and Experience*, 2016.

[270] Z. Zhao, G. Wang, A. R. Butt, M. Khan, V. A. Kumar, and M. V. Marathe. Sahad: Subgraph analysis in massive networks using hadoop. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pages 390–401. IEEE, 2012.

[271] K. Zhou, L. Song, and H. Zha. Learning social infectivity in sparse low-rank networks using multi-dimensional hawkes processes. In *AISTATS*, 2013.