Algorithms for Learning Networks and Learning from Networks

by

Mano Vikash Janardhanan
Integrated MS, Indian Institute of Science Education and Research, Thiruvananthapuram, 2014

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Mathematics
in the Graduate College of the
University of Illinois at Chicago, 2019

Chicago, Illinois

Defense Committee:

Lev Reyzin, Chair and Advisor
György Turán
Dhruv Mubayi
Bhaskar DasGupta, Department of Computer Science
Anastasios Sidiropoulos, Department of Computer Science

To my parents

ACKNOWLEDGMENT

I am deeply indebted to my advisor Lev Reyzin for all his help during my time at UIC. I am grateful to him for teaching me how to be passionate about research. From picking a problem that I find interesting to publishing the results, I can not understate the value of his guidance throughout the process. I am grateful for his incredible guidance and endless patience. I am inspired by his passion for research and honesty in everything he does. I also want to thank Lev for encouraging me to collaborate with other research groups within academia and outside because this showed me different styles of research and more importantly helped me find what I want to do. I feel I have been extremely fortunate to have him as my advisor and I hope to emulate these qualities in my career.

I would also like to extend my deepest gratitude to all my collaborators in UIC- Mohsen Aliabadi, Bhaskar DasGupta, Yi Huang and Farzane Yahyanejad. Their contribution has had a major role in this dissertation and it was a pleasure working with them. I also want to thank Bhaskar DasGupta, Dhruv Mubayi, Anastasios Sidiropoulos and Gyuri Turan for teaching the graduate classes that gave me a solid mathematical background and for always being available to answer my questions. I am extremely grateful to the subset of people mentioned above for being in my thesis committee.

I would like to extend my sincere thanks to Ádám Lelkes for helping me explore research in industry. I did not know much about industry when I applied but he helped me navigate through the entire process and found an amazing team for me to work with. It was an great experience which deeply influenced my perspective on research.

ACKNOWLEDGMENT (Continued)

Before coming to UIC, I spent five wonderful years in Indian Institute of Science Education and Research, Thiruvananthapuram which laid the foundations for my graduate study. I am grateful to my Masters thesis advisor, Sujith Vijay for helping me take my first steps in mathematics research which ultimately led to my first paper.

I thank all my friends in UIC for their encouragement and support. Special thanks to Maryam Emami for her marathon running tips and Sarthak Chimni for his insightful analysis of every soccer game we played and watched. I thank Arunkumar Muthusamy and Ajai Pulianmackal (my friends from undergrad) for visiting me once in a while and forcing me to host them every time. I am also grateful to Ramasubramonian Deivanayagam for his unwavering support right from my undergraduate days.

This thesis would not be possible without the support of my family. While every other student from my high school class went on to study engineering, my family encouraged me to pursue my passion. I dedicate this dissertation to the trust they placed in me.

MVJ

CONTRIBUTIONS OF AUTHORS

- Chapter 2 represents the paper *Graph verification using a betweenness oracle* by Mano Vikash Janardhanan (Janardhanan, 2017).
- Chapter 3 represents the paper *Network construction with ordered constraints* by Yi Huang, Mano Vikash Janardhanan and Lev Reyzin (Huang et al., 2017). This work also appears in the thesis of Yi Huang (Huang, 2017).
- Chapter 4 represents the preprint *How did the shape of your network change? (on detecting anomalies in static and dynamic networks via change of non-local curvatures)* by Bhaskar DasGupta, Mano Vikash Janardhanan and Farzane Yahyanejad (DasGupta et al., 2018). The results in this preprint are split between this thesis and Farzane Yahyanejad's thesis (Yahyanejad, 2019).

All content in these chapters, including introduction, formulation of definitions, theorems, algorithms, and writing of the various manuscripts were done jointly with the co-authors.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

SUMMARY

There are many different networks we encounter in every day life. Social networks, internet (short form of interconnected network) and biological networks are some examples. With the enormous amount of data this is being generated in recent times, the size of these networks have grown substantially. Hence it is important to develop new models and algorithms for studying networks on a large scale.

This thesis focuses on problems in graph theory that are related to learning and algorithms. We will explore how to formalise the question "How to learn a graph?" in Chapters 2 and 3. In Chapter 2, we give the learner the power to query a graph and the objective is to learn the edges (or connections) of the graph with the least number of queries. In Chapter 3, the learner gets to see some information about the graph that is generated by some process and the objective is to learn (or approximately learn) the edges. We consider various models for how the information is given to the algorithm. Finally, in Chapter 4, suppose we observe some dynamic graph at two different times, we ask what changes in the edges caused the maximum change in the overall structure of the graph. The intuition here is that these changes correspond to anomalies.

In this thesis, we have built models for various learning and algorithmic questions on graphs. For each of these models, we pick the quantities of most significance and give algorithms and bounds for them.

CHAPTER 1

INTRODUCTION

Recent advances in technology has made it easier to collect, store and process data about the world around us. Interactions among entities in our world play a significant role in the data that is generated. Hence, it is extremely important to consider these interactions when building models for the data. These interactions are captured mathematically by graphs (also called networks). As the amount of data available increases, so does the size of these graphs. This motivates the need to develop new models and algorithms for handling various problems over graphs.

The basic structure of a graph is a collection of elements and connections between pairs of elements to denote that those pairs interact with each other in some way. Graphs arise naturally in many different scenarios. In computer science, graphs can be used to represent a network of communications. The internet can be thought of as a graph where the elements of the graph are computers and two computers interact if there is a direct channel of communication between them. In molecular biology, an interactome is modelled by a graph which captures the whole set of molecular interactions in a particular cell. In a protein-protein interactome, the elements are proteins and connections represent whether two proteins interact. In a social network, individuals form the elements and connections may represent whether two individuals are friends.

Suppose a scientist observes some data and wants to construct a graph from the data. This data may be a result of experiments run by the scientist or it may be generated by some natural process that the scientist has no control over. In the former case (studied in Chapter 2), the scientist has the power to

choose experiments to run in order to find the graph. Hence, the question becomes what is the best experiments to run. In the later case (studied in Chapter 3), the interesting question is what method does the scientist use to deduce the graph from the data. Now, suppose a scientist used some data and inferred a graph. It is natural to ask what can the scientist conclude from it. In Chapter 4, we look at the problem of finding the interactions which are most important for the overall structure.

This thesis considers a theoretical framework for each of these problems. We start by giving a brief introduction to standard models in theoretical computer science.

1.1 Complexity theory

The P versus NP problem is one of the biggest unsolved problems in computer science. This section discusses the essence of the problem without many of the technical definitions. A formal treatment of the problem can be found in (Arora and Barak, 2009).

Turing Machines are models of computation used to define computational complexity problems. Intuitively, a Turing machine is equivalent to any reasonable model of computation (like a computer or laptop). Decision problems are problems for which the output is yes or no. The class P contains decision problems that can be solved by a deterministic Turing machine in time polynomial in the size of the input. The class NP contains decision problems for which the yes instances of the problem can be verified by a deterministic Turing machine in time polynomial in the size of the input. When a problem can be solved by a deterministic Turing machine in time polynomial in the size of the input, it is considered to be “easy”.

Given a set of integers A with $|A| = n$, consider the problem of checking whether the total sum of integers in A is positive or not. This problem is in class P because it is easy to sum all the numbers. Given

a set of integers A with $|A| = n$, consider the problem of checking whether there is a subset $S \subseteq A$ such that the sum of integers in S is equal to zero. If the subset S which is a candidate for summing to zero is given, it is easy to verify whether that the numbers in S adds up to zero or not. Hence, this problem is in NP. But it is not clear whether this problem is in P or not. This motivates the general question of whether a problem that can be verified in polynomial time can be solved in polynomial time. This is the essence of the P versus NP problem. Formally, the P versus NP problem is the question whether P=NP or P≠NP? While the problem is unsolved, it is reasonable to believe that in general, problems in NP can be harder than problems in P and hence it is conjectured that P≠NP. The hardest problems in class NP are called NP-complete problems. If P≠NP, then NP-complete problems can not be solved in polynomial time.

A polynomial time reduction from a decision problem A to a decision problem B is an algorithm that, on input instance x of problem A, outputs an instance y on problem B in polynomial time, such that the answer to y is yes if and only if the answer to x is yes. If such a polynomial time reduction exists from problem A to problem B, it is denoted as $A \leq_p B$. When studying the complexity of a problem X, if $X \leq_p Y$ and Y is in class P then, X is also in class P. On the other hand if $X \leq_p Y$ and Y is NP-complete, then X does not have a polynomial time algorithm assuming P≠NP. These type of reductions are standard methods in complexity theory to study the computation complexity of a problem.

SAT is the problem of deciding whether a Boolean formula is satisfied. A formula is satisfiable if there exists an assignment of true or false values to the variables such that the value of the formula becomes true. SAT is the first problem that was proven to be NP-complete. 3-SAT is the problem of deciding whether a Boolean formula in 3-conjunctive normal form is satisfiable. 3-SAT is also NP-complete. This means that if P≠NP, 3-SAT can not be decided in polynomial time. The Exponential Time Hypothesis

(ETH) introduced in (Impagliazzo and Paturi, 2001) postulates that 3-SAT can not be solved in $2^{o(n)}$ time in the worst case. Hence, ETH is a stronger assumption than P$\neq$NP and leads to stronger lower bounds for many problems. The unique games conjecture (UGC) introduced in (Khot, 2002) is another standard assumption that is stronger than P$\neq$NP. It postulates that finding the approximate value of a certain type of game is hard.

Optimization problem occur naturally in various scenarios. Given a set of integers A, let X be problem of finding the maximum value that can be attained by summing the elements of a subset $S \subseteq A$. It is possible to get a decision version of the optimization problem X by introducing an extra variable k. In particular, let X' be following the decision problem: given A and k, does there exists a subset $S \subseteq A$ such that the sum of integers in S is greater than k. It is usually easy to convert an optimization problem to a decision problem. A polynomial time algorithm for X implies a polynomial time algorithm for X'. A simple binary search over the values of k along with a polynomial time algorithm for X' implies a polynomial time algorithm for X. Such relationships between optimization problems and the corresponding decision problems are usually obvious.

1.2 Learning theory

For a formal introduction to learning theory see (Mohri et al., 2012). There are many models to study the theoretical frameworks for learning. The PAC learning model is arguably the most popular one for real world scenarios. Looking at what scenarios it captures naturally leads to some of the other frameworks studied in this thesis.

In 1984, Valiant introduced PAC (Probably Approximately Correct) learning model (Valiant, 1984). This model captured the essence of what it means to learn from data that is generated by the real world.

In this model, training examples are generated by a fixed distribution, labeled by a concept and given to the learner. A concept class is said to be PAC learnable if for any concept in the class, and any distribution, a learner can produce with probability at least $(1 - \delta)$, a hypothesis that has at most ϵ error on sufficiently large number of future examples drawn from the same distribution where sufficiently large means polynomial in the relevant parameters. If the learner can produce the hypothesis in polynomial time in the relevant parameters, then the concept class is said to be efficiently PAC-learnable.

While the PAC-learning model captures most settings where the data is generated by some real world process, it does not model the ability of the learner to interact with the world and choose the training data. This gap was filled by active learning models introduced by (Angluin, 1988). In active learning, the learner has the ability to query an oracle. In Angluin's model, the learner could ask the oracle to label an example according to the unknown concept (called membership query) or the learner could ask whether a proposed concept is the correct one (called equivalence query).

In the online learning scenario, the examples appear one at a time (usually adversarially) and the learner must produce labels on the spot. This model captures settings where the data is generated over a large period of time or when the dataset is too large for the learner to store all the previously seen data. There are various models for the adversary and the amount of power the adversary has determines how well an algorithm can perform against it.

1.3 Preliminaries

This section starts with some of the basic definitions which may be found in standard textbooks in graph theory (van Lint et al., 2001), then proceeds to discuss some standard results in complexity of graph theoretic problems, definitions of problems in query learning of graphs and definition of offline

and online problems. While the subsequent chapters use the basic definitions and results in this section, the background and previous work related to the specific models are discussed within the chapters.

A graph is an ordered pair $G = (V, E)$ comprising of a set V of n vertices and a set E of m edges where each element $e \in E$ is a 2-element subset of V. If $\{u, v\} \in E$, u and v are said to be adjacent to each other. An edge e is incident on a vertex v if $v \in e$. A weighted graph is a graph together with a function associating a real number $w(e)$ to each edge $e \in E$. If no such function is provided, the graph is assumed to be unweighted and all edges are assumed to have unit weight. It will be made clear in each chapter whether the problem being considered is on weighted or unweighted graphs.

It is possible to define the edge set by 2-tuples of vertices instead of 2-element subsets. However, in the 2-tuple definition, if (u, v) is an edge, it is interpreted as a directed edge from u to v. This leads to the definition of a directed graph. It will be made clear in each chapter whether the graph being considered is directed or undirected.

A path is a sequence of vertices and edges $v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_{k-1}, v_k$ where $e_i = \{v_i, v_{i+1}\}$ for $1 \leq i \leq k-1$ and $e_1, e_2, \ldots, e_{k-1}$ are distinct elements. A path is called simple if the vertex terms $v_1, v_2, \ldots, v_k$ are distinct. The length of a path is the sum of weights of edges in the path. A cycle is a path where $v_1 = v_k$ and all other vertices are distinct. A graph is a path if $V = \{v_1, v_2, \ldots, v_k\}$ and $E = \{e_1, e_2, \ldots, e_{k-1}\}$. A graph is connected, if for any pair of vertices u and v, there is a path connecting them. An acyclic graph is a graph without any cycle. A tree is a connected acyclic graph. A leaf is a vertex in a tree with only one edge incident on it. A star graph is a tree where every edge is incident on a single vertex v_0. A path can also be denoted by an ordered sequence of vertices as it is clear what edges are present between them.

The distance between two vertices u and v in a graph G is the length of a shortest path connecting u and v. It is denoted by $d_G(u, v)$ or $dist_G(u, v)$. The subscript G is dropped when clear from context. A shortest path is always simple. Given a graph G and two vertices, a shortest path between the two vertices may not be unique. A shortest path between u and v is denoted by $\overline{u, v}$. A shortest path between v_1 and v_k passes through $v_2, v_3, \ldots, v_{k-1}$ if $v_1, e_1, v_2, e_2, \ldots, v_{k-1}, e_{k-1}, v_k$ is a shortest path. The v_i's for $2 \leq i \leq k-1$ are said to lie on a shortest path between v_1 and v_k.

A degree of a vertex in a graph is the number of edges incident on it. A graph has maximum degree Δ if all its vertices have degree $\leq \Delta$. A subgraph of a graph G is a graph H such that the vertex set of H is a subset of vertex set of G and the edge set of H is a subset of edge set of G that contains only edges e such that both ends of e are in H. The subgraph of G induced by a subset S of vertices of G is the subgraph whose vertex set is S and whose edges are all the edges of G with both ends in S.

1.3.1 Complexity of graph theoretic problems

Most of the results stated in this subsection can be found in standard textbooks (Papadimitriou and Steiglitz, 1982; Garey and Johnson, 1990).

Given an undirected unweighted graph $G = (V, E)$, a cut is a partition of V into two disjoint non-empty subsets. A cut can be denoted by a tuple $(S, V \backslash S)$ for $S \subset V$. The size of a cut $(S, V \backslash S)$ is the number of edges in G such that one end of the edge lies in S and the other end of the edge lies in $V \backslash S$.

Consider the following decision problem. Given a graph G and an integer k, does there exist a cut of size at most k. Let this problem be MIN-CUT. The Stoer-Wagner algorithm is a polynomial time algorithm for solving the problem and hence MIN-CUT$\in$P (Stoer and Wagner, 1997). Let MAX-CUT be

the following decision problem: Given a graph G and an integer k, does there exist a cut of size at least k. MAX-CUT is NP-complete. It is surprising that such a small change in the statement of the problem makes it difficult to solve efficiently.

The vertex cover problem is a classical problem in complexity theory. A vertex cover is a set of vertices $S \subset V$ such that each edge of the graph is incident on at least one vertex in S. The optimization problem of finding the vertex cover of minimum size is called the minimum vertex cover problem. The decision version of the problem is called the vertex cover problem and it is NP-complete.

Another classical NP-complete problem in graph theory is the Hamiltonian path problem. A Hamiltonian path is a path that visits every vertex in the graph exactly once. The Hamiltonian path problem is the decision problem of whether there exists a Hamiltonian path in a graph.

Consider the minimum vertex cover problem. If P≠NP, there does not exist a polynomial time algorithm to find the minimum vertex cover S_{OPT}. However, there exists a polynomial time algorithm that finds a set S_{ALG} such that $\frac{|S_{ALG}|}{|S_{OPT}|} \leq 2$. Such an algorithm is called an approximation algorithm and the upper bound 2 on the factor $\frac{|S_{ALG}|}{|S_{OPT}|}$ is called the approximation ratio of the algorithm.

If P≠NP, then the minimum vertex cover problem can not be approximated within a factor of 1.3606 (Dinur and Safra, 2005b). If unique games conjecture is true, then 2 is the best possible approximation ratio for the minimum vertex cover problem (Khot and Regev, 2008b).

1.3.2 Query learning of graphs

Let $G = (V, E)$ be a undirected, unweighted graph such that the vertex set is known and edge set is not known to the learner. The problem is to find the set E. The learner has access to the graph through an oracle. Suppose a learner has access to G through a distance oracle, then a learner can query (u, v) and

receive the distance between u and v in G. The objective of the learner is to find E exactly using the minimum number of queries to the oracle. This is one of the standard models in query learning of graphs and is called the graph learning problem with a distance oracle (Mathieu and Zhou, 2013).

Consider the following problem. Let $G = (V, E)$ be a undirected, unweighted graph such that the vertex set is known and edge set is not known to the learner. Let $\hat{G} = (V, \hat{E})$ be a undirected, unweighted graph on the same vertex set for which the learner knows both the vertices and edges. Suppose the learner has access to G through a distance oracle. The objective of the learner is to check whether $E = \hat{E}$ using the minimum number of queries to the oracle. In other words, the problem is to verify whether G is the same as $\hat{G}$. This is called the graph verification problem with a distance oracle.

The betweenness oracle returns information about whether a vertex lies on a shortest path between two other vertices. Chapter 2 gives matching bounds for the query complexity of the graph verification problem with a betweenness oracle.

1.3.3 Offline and online problems

This subsection motivates offline and online problems by considering the example of the minimum hitting set problem. A formal introduction to the topic can be found in (Borodin and El-Yaniv, 1998). The input to the minimum hitting set problem is a tuple $(U, \mathcal{S})$ where $U = \{u_1, \ldots, u_n\}$ is the universe, and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a set of subsets of U. A subset $H \subset U$ is called a hitting set if $H \cap S_i \neq \emptyset$ for $i = 1, \ldots, m$. The objective is to find a hitting set of minimum cardinality. The decision version of the minimum hitting set problem is NP-complete. In the standard version of the minimum hitting set problem, the input is assumed to be given all at once. This standard version is called the offline minimum hitting set problem. The online version of the problem is an instance where the input U is given all at

once but $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ arrives one by one. This can be interpreted as the algorithm seeing the set S_i at time i. As the sets arrive one by one, the algorithm has to update the H but it is not allowed to delete elements from H which were added in the past. A solution to the online hitting set sequence of sets $H_1 \subset H_2 \subset \cdots \subset H_m$ such that for all $1 \leq i \leq m$, H_i is a hitting set for the first i constraints. The competitive ratio for an online algorithm is the approximation ratio achieved by the algorithm. In other words, it is the worst-case ratio between the cost of solution obtained by the algorithm to the cost of an optimal solution. For the online hitting set problem, it is the worst-case value of $\frac{|H_m|}{|H_{OPT}|}$ over all the instances of the problem where H_m is the solution produced by the algorithm for an instance and H_{OPT} is the optimal solution in hindsight for the instance.

For the online hitting set problem, let the sets S_i be produced by an adversary. There are various models for the adversary like the oblivious adversary and the adaptive adversary. For a deterministic algorithm, these models are equivalent. But for a randomized algorithm, the competitive ratio may depend on the adversary model. Both oblivious adversary and adaptive adversary know the algorithm completely. The adaptive adversary gets to see the randomization of the algorithm at each step before giving the next S_i. Hence, its S_i can depend on the random choices made by the algorithm before time i. The oblivious adversary does not get to see these random choices. An adaptive adversary is stronger and hence the competitive ratio for the online hitting set problem against an adaptive adversary is at least the competitive ratio for the online hitting set problem against an oblivious adversary.

CHAPTER 2

NETWORK VERIFICATION WITH BETWEENNESS ORACLE

This chapter was previously published as Graph Verification using a Betweenness Oracle by Mano Vikash Janardhanan (Janardhanan, 2017).

2.1 Introduction and previous work

Graph learning and graph verification problems arise in various situations. Consider the internet graph where vertices correspond to routers and edges correspond to physical connections. It is often the case that one knows the set of vertices in the network (routers) but does not know the edges (physical connections). To learn the physical connections, one has to use computer network diagnostic tools (such as traceroute and mtrace) which give information about the shortest paths in the network. Assume one has access to the internet graph through such an oracle. A natural question to ask is what is the best way to use the oracle to find the physical connections between the routers. In other words, what is the minimum number of queries needed to learn the edge set of the graph?

Graph learning and graph verification problems are well-studied problems in the area of graph algorithms. In both these problems, there is a hidden graph which one has access to through a black-box oracle. In graph learning problems, the task is to use this oracle to learn the edge set of the graph. Graph learning problems are also referred to as the graph reconstruction problems. Graph learning problems has been studied extensively (Alon and Asodi, 2005a; Alon et al., 2004a; Angluin and Chen, 2008a; Beerliova et al., 2006; Dall'Asta et al., 2006; Hein, 1989; Reyzin and Srivastava, 2007a).

In graph verification problems, one is given another input graph and the task is to verify that the graph one has access to through an oracle is the same as the input graph. Graph verification problems have received a lot of attention recently (Beerliova et al., 2006; Erlebach et al., 2006; Kannan et al., 2015; Mathieu and Zhou, 2013; Reyzin and Srivastava, 2007a). Graph verification problems have many applications for Internet Service Providers (ISPs). The ISPs have knowledge about the structure of the network based on past information. And at any point of time, they might wish to verify that there is no fault in the network. In any internet protocol network, fault detection methods are critical for providing quality of service guarantees.

Some of the oracles studied in literature include the distance oracle by (Kannan et al., 2015), layered-graph oracle by (Beerliova et al., 2006), edge detection and edge counting oracle by (Reyzin and Srivastava, 2007a), etc. Among these, the most natural and well-studied one is the distance oracle (Erlebach et al., 2006; Kannan et al., 2015; Mathieu and Zhou, 2013; Reyzin and Srivastava, 2007c). The distance oracle takes as input two vertices and returns the distance between the two vertices. This oracle nicely captures applications in computational biology. One example in computational biology is the problem of learning evolutionary trees (Hein, 1989; King et al., 2003; Waterman et al., 1977). Researchers can obtain the distance between two species in an unknown evolutionary tree. This can be thought of as making a distance query. But each query requires a lot of research effort. Hence the objective is to learn the evolutionary tree with the minimum number of queries. In general, for both graph learning and graph verification problems, we assume that making queries is costly. Hence, we are concerned with optimizing the worst-case query complexity herein.

In this paper, we look at the query complexity of the graph verification problem with a betweenness oracle. The betweenness oracle, introduced by (Abrahamsen et al., 2016), returns whether a given vertex lies along a shortest path between two other vertices. When the graphs are connected, unweighted, and have bounded maximum degree Δ, we prove the worst-case query complexity has an upper bound of $n^{1+o(1)}$. We also prove a lower bound of $\Omega(n)$. The betweenness oracle also has many natural applications in the study of evolutionary trees. For evolutionary trees, the method for calculating evolutionary distance is error-prone. If we use the betweenness oracle approach, we only need to query whether one species lies in the shortest path connecting two other species. Such a query is more natural for evolutionary trees.

Intuitively, the betweenness oracle is expected to be much weaker than the distance oracle. Notice that a betweenness query can be simulated by three distance queries. Let x, y and z be vertices of a graph and $d(\cdot, \cdot)$ denote the distance between two vertices in the graph. Then, $d(x, y) + d(y, z) = d(x, z)$ if and only if y lies on a shortest path between x and z. Conversely, it is easy to see that in a path graph, one needs $\Omega(n)$ betweenness queries to simulate a single distance query.

For degree-bounded graphs, (Abrahamsen et al., 2016) showed that the graph learning problem with a betweenness oracle has the same worst-case query complexity as its analogue with a distance oracle. However, the problem of verifying a graph with a betweenness oracle remained open. In this paper, we give matching lower and upper bounds for this problem. The main result of (Abrahamsen et al., 2016) is the following:

Theorem 1. *Learning a graph can be done with $\tilde{O}(n^{3/2} \cdot \Delta^4)$ betweenness queries, where Δ is the maximum degree of the graph.*

2.1.1 The problem

The hidden graph to be verified is denoted as $G = (V, E)$. A 2-element subset of vertices $\{u, v\}$ is called a non-edge if $\{u, v\} \notin E$. This can be also denoted as $\{u, v\} \in NE$ where NE is the set of non-edges of G. Similarly, the given graph $\hat{G} = (V, \hat{E})$ has non-edge set $\hat{NE}$ defined in a similar way.

In graph learning problems, we are given an oracle access to G, and the task is to determine E. In graph verification problems we are given $\hat{G}$ and an oracle access to G, and asked to verify that $E = \hat{E}$.

Given a subset $U \subset V$, $G[U]$ is the subgraph induced by U. For the rest of the paper, we assume that the graph is connected, undirected, unweighted and has maximum degree Δ. We have access to G through a betweenness oracle.

Definition 1. *A **betweenness query** denoted as* $\mathrm{bet}_G(u, v, w)$ *is true if and only if there exists a shortest path in* G *between* u *and* w *that passes through* v*. Often, the subscript will be dropped when* G *is clear from context.*

We prove matching lower bound and upper bound for the query complexity of the graph verification problem with a betweenness oracle.

2.2 Lower bound

In this section, we consider an instance of the graph verification problem where the input graph $\hat{G}$ is a caterpillar tree and the hidden graph G is a slight modification of $\hat{G}$. Then, we show that $\Omega(n)$ betweenness queries are required to catch this modification.

Theorem 2. *Graph verification requires* $\Omega(n)$ *betweenness queries.*

Proof. Let $\hat{G}$ be the caterpillar tree with spine from v_1 to $v_{n/2}$ and one vertex connected to each vertex in the spine (Figure 1). $|\hat{G}| = n - 1$ and n is an even number. Consider a new graph H obtained from $\hat{G}$ by doing the following (Figure 2):

- Fix some $i \in [1, n/2 - 1]$
- Delete the edge between v_i and v_{i+1}
- Add an edge between $v_{n/2+i}$ and v_{i+1}

Suppose the hidden graph G is H (Figure 2). The betweenness queries that give away the difference between $\hat{G}$ and H contain $v_{n/2+i}$ as one of its three arguments in $\mathrm{bet}_G(\cdot, \cdot, \cdot)$. There are $n/2 - 1$ vertices of the form $v_{n/2+i}$ and each query can cover at most 3 of them. This gives the desired lower bound. □

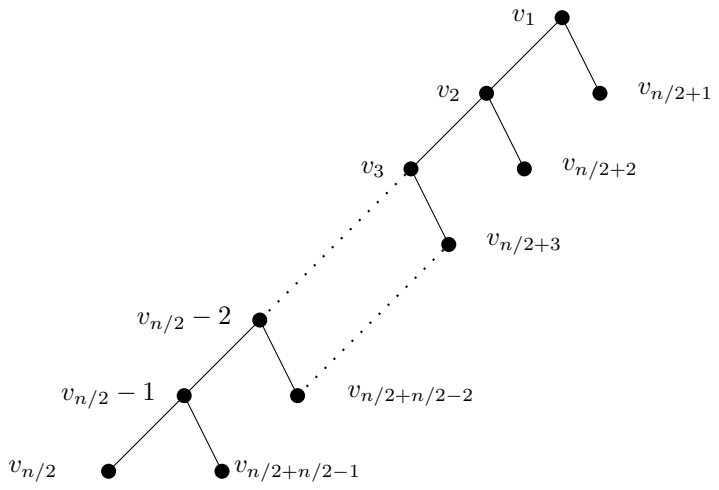


Figure 1: Representation of $\hat{G}$ for Theorem 2 in Chapter 2

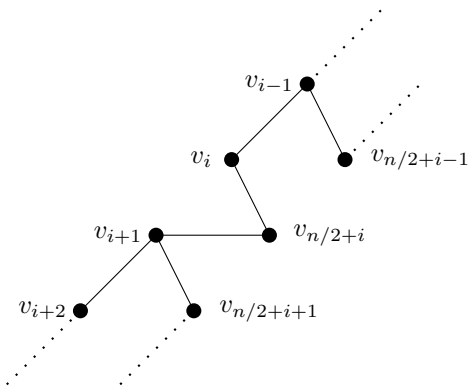


Figure 2: Representation of H for Theorem 2 in Chapter 2

Observation 1. *When the target graph G has maximum degree Δ, graph learning requires $\Omega(n\Delta \log n)$ betweenness queries. This is because the number of connected graphs with maximum degree Δ is $\Omega(n^{\Omega(\Delta n)})$ (McKay and Wormald, 1990). Hence, information theoretically, we get the desired lower bound.*

2.3 Definitions

The full generality of the following definitions are not necessary for this paper. But we do it to maintain consistency with (Abrahamsen et al., 2016).

Definition 2. *Let $G = (V, E)$ and $v \in V$. Let $N_i(v)$ denote the set of vertices that are distance i or less from v. Let $N(v) = N_1(v)$. Hence $N(v)$ is the set that contains v and all its neighbours. When $x, y \in V$, let $\delta(x, y)$ denote the distance between x and y in G.*

Definition 3. *Given a graph $G = (V, E)$, a subset $X \subset V$ is said to be **starshaped** with respect to centre $x \in X$ if for all $v \in X$, every shortest path from x to v is entirely contained in X.*

Definition 4. *Given a graph $G = (V, E)$ and a starshaped set $X \subset V$ with centre $x \in X$, a node $v \in X$ is said to be in layer i if $\delta(x, v) = i$. The set of nodes in layer i is denoted $L(x)_i^X$. When $X = V$, the superscript is dropped and written as $L(x)_i$.*

Definition 5. *Given a graph $G = (V, E)$ and a starshaped set $X \subset V$ with centre $x \in X$, a subgraph $\tau(x)_X$ is a **spanning tree** with respect to centre x if it is a tree such that for all $v \in X$, $\tau(x)_X$ contains a shortest path from x to v. Given a starshaped set $X \subset V$ with centre $x \in X$, the subgraph $S(x)_X$ obtained by removing all edges in the same layer $L(x)_i^X$ is called the **shortest path graph** with respect to centre x.*

Definition 6. *Given a starshaped set $X \subset V$ with centre $x \in X$, if $v \in L(x)_i^X$, then u is a **parent** of v with respect to centre x if $u \in N(v) \cap L(x)_{i-1}^X$. This can be written as $u \in p_x(v)$. Note that $p_x(v)$ is a set. Given a starshaped set $X \subset V$ with centre $x \in X$, if $v \in L(x)_i^X$, then u is a **child** of v with respect to centre x if $u \in N(v) \cap L(x)_{i+1}^X$.*

Definition 7. *The **ancestor** relation is the transitive closure of the parent relation and the **descendant** relation is the transitive closure of the child relation. The set of ancestors of a vertex v with respect to centre x is denoted $A_x(v)$ and the set of descendants is denoted $D_x(v)$. The subscript is dropped when the centres x is clear from context.*

For $\hat{G} = (V, \hat{E})$, we define $\hat{\delta}(x, y)$, $\hat{N}_i(v)$, $\hat{N}(v)$, $\hat{L}(x)_i^X$, $\hat{\tau}(x)_X$, $\hat{S}(x)_X$, $\hat{p}_x(v)$, $\hat{A}_x(v)$ and $\hat{D}_x(v)$ in a similar way.

2.4 Main result

The main result is Theorem 3 and its proof proceeds in two steps- edge verification and non-edge verification. The proof relies on some techniques developed earlier for graph verification with a distance oracle and graph learning with a betweenness oracle. For edge verification, we use some results from (Abrahamsen et al., 2016). For non-edge verification, we use some results from (Kannan et al., 2015). These results are stated without proof before being used.

Theorem 3. *Let $\hat{G}$ be a connected graph with maximum degree Δ. For graph verification using a betweenness oracle, there is a deterministic algorithm with a query complexity of $n^{1+O\left(\sqrt{(\log\log n+\log\Delta)/\log n}\right)}$. When $\Delta = n^{o(1)}$, this gives us a query complexity of $n^{1+o(1)}$.*

When $\Delta = n^{o(1)}$, (Kannan et al., 2015) devised a recursive algorithm that does non-edge verification using $n^{1+o(1)}$ distance queries. To simulate a distance query, we need $\Omega(n)$ betweenness queries. Hence, a brute force generalization of their approach can not give query complexity better than $n^{2+o(1)}$. The main contributions of this paper are the following:

- Edge verification can be done using $O(n\Delta^2)$ betweenness queries. This is proved in Lemma 5.
- We prove that the recursive approach for non-edge verification developed in (Kannan et al., 2015) can be implemented using $n^{1+o(1)}$ betweenness queries.

2.4.1 Edge verification

We start by proving a bound on the number of betweenness queries required for edge verification. Before doing that, we need the following three results (stated without proof) from (Abrahamsen et al., 2016).

Lemma 1 ((Abrahamsen et al., 2016)). *Every starshaped set X with centre x has a node $s \in X$ with the property*

$$\left\lceil \frac{|X|}{3\Delta} \right\rceil \leq |D(s)| \leq \left\lceil \frac{|X|}{3} \right\rceil$$

Further, $D(s)$ and $(X - D(s)) \cup \{s\}$ are both starshaped with centres s and x respectively.

Lemma 2 ((Abrahamsen et al., 2016)). *Let $X \subset V$ be a starshaped set with centre x. One can discover all edges in $G[X]$ using $O(|X|^2)$ betweenness queries.*

Lemma 3 ((Abrahamsen et al., 2016)). *Given a starshaped set X with centre x, and the shortest path graph of X, one can decide whether or not there exists an edge between any two nodes u and v in the hidden graph using $O(1)$ betweenness queries.*

For doing edge verification, we start by recursively applying Lemma 1 to partition the edge set of a spanning tree of $\hat{G}$ and then apply Lemma 2 to verify that all edges of the spanning tree of $\hat{G}$ is present in G. Then, we use Lemma 4 to show that G and $\hat{G}$ have the same layer structure. After this, we only need to verify edges within the same layer and edges between adjacent layers. These type of edges require $O(1)$ query per edge.

Lemma 4 (Layer Structure Verification). *Let $\hat{G} = (V, \hat{E})$ be a connected graph. Suppose $\hat{\tau}(x)_V$ is a spanning tree of $\hat{G}$ with respect to centre x and every edge in $\hat{\tau}(x)_V$ has been verified to be present in G. Then, n betweenness queries are sufficient to verify that $L(x)_i^V = \hat{L}(x)_i^V$ for all i.*

Proof. To show $L(x)_i^V = \hat{L}(x)_i^V$ for $i \leq k$, we need to establish there is no edge in G going from $\hat{L}(x)_i^V$ to $\hat{L}(x)_{i-s}^V$ for $i \leq k$ and $s > 1$. We prove this by induction on k.

Query $\mathrm{bet}(x, p, v)$ for $v \in \hat{L}(x)_k^V$ and some $p \in \hat{p}_x(v)$. Because every edge in $\hat{\tau}(x)_V$ has been verified to be present in G, $\mathrm{bet}(x, p, v)$ will return false if and only if there is an edge in G from v to some vertex in $\hat{L}(x)_{k-s}^V$ for $s > 1$. Hence, it takes $|\hat{L}(x)_k^V|$ queries to show there is no edge in G from $\hat{L}(x)_k^V$ to $\hat{L}(x)_{i-s}^V$ for $s > 1$. Continuing for all layers takes at most n queries. □

Lemma 5. *Given $\hat{G}$ and access to G through a betweenness oracle, verifying all edges of $\hat{G}$ are present in G can be done with $O(n\Delta^2)$ betweenness queries.*

Proof. We start by proving that every edge of a spanning tree of $\hat{G}$ is present in G. Fix a centre x in $\hat{G}$ and let $\hat{\tau}(x)_V$ be a spanning tree with respect to centre x. Now, we partition the edge set of $\hat{\tau}(x)_V$ by doing the following. Recursively apply Lemma 1 to obtain starshaped sets $\{S_1, S_2, \ldots, S_h\}$ that satisfy the following properties:

- $\frac{k}{3\Delta} \leq |S_i| \leq k$ for all i where $k = c\Delta$ and c is a constant.
- Every edge of $\hat{\tau}(x)_V$ is present inside exactly one S_i.

In other words, S_i's partition the edge set of $\hat{\tau}(x)_V$. Note that the S_i's are sets of vertices that are not disjoint.

Verifying all edges inside a S_i takes $O(k^2)$ queries by Lemma 2. The total number of starshaped sets is h which is at most $3\Delta n/k$. Hence, the total number of queries to verify the edges within every S_i is $O(n\Delta^2)$. Note that we have not verified the shortest path graph as there may be edges from layer i to layer $i + 1$ that are not contained in any starshaped set. However, since every edge of $\hat{\tau}(x)_V$ is inside some S_i, we have shown that every edge of $\hat{\tau}(x)_V$ is present in G.

Using Lemma 4, we get that G has the same layer structure as $\hat{G}$ by making at most n queries. Now, use Lemma 3 to verify all edges in the same layer using $O(1)$ query per edge. Finally, to verify edges $e = (y, z)$ from layer i to layer $i + 1$, that are not contained in $\hat{\tau}(x)_V$, note that $\mathrm{bet}(x, y, z)$ is true if and only if e is present in G. This takes 1 query per edge. The total number of such edges is at most $n\Delta$. Hence, we get the desired bound. □

2.4.2 Non-edge verification

The algorithm for non-edge verification proceeds as follows. We start with V and split it into cells $U_1, U_2, \ldots, U_k$ such that every edge in $\hat{G}$ is completely contained in some U_i. These U_i's are called extended Voronoi cells and will be defined soon. Using Lemma 7, we can verify (with few queries) that every edge in G is completely contained in some U_i. Then, we can recursively apply the splitting technique to each U_i.

Given a cell U, the algorithm for splitting U into extended Voronoi cells first selects a set of centres $\{a_1, a_2, \ldots, a_k\} = A \subseteq V$ using Algorithm 1. Then, it builds Voronoi cells around these centres.

Definition 8. *Given $A \subseteq V$ and $w \in V$, the Voronoi cell of w with respect to A is defined as*

$$\hat{C}_A(w) = \{v \in V : \hat{\delta}(w, v) < \hat{\delta}(A, v)\}$$

We expand the Voronoi cells slightly so that every edge of $\hat{G}$ contained in U is completely contained in one of the extended Voronoi cells produced by splitting U. Note that the extended Voronoi cells are not disjoint.

Definition 9. *Let $A \subseteq V$ be the set of centres and $U \subseteq V$. Define for each $a \in A$, its extended Voronoi cell $\hat{D}_a \subseteq U$ as*

$$\hat{D}_a^U = \left(\bigcup \{\hat{C}_A(b) : b \in \hat{N}_2(a)\} \cup \hat{N}_2(a)\right) \cap U$$

The superscript U is dropped when clear from context. The following lemma as stated in (Kannan et al., 2015) guarantees that the splitting done using the centres algorithm does not return too many cells and the size of each cell goes down significantly compared to the size of U.

Lemma 6. *Given a graph $\hat{G} = (V, \hat{E})$, a subset of vertices $U \subseteq V$, and an integer $s \in [1, n]$, Algorithm 1 computes a subset of vertices $A \subseteq V$ such that the following conditions hold:*

- *The expected size of the set A is at most $2s \log n$*
- *For every vertex $w \in V$, we have $|\hat{C}_A(w) \cap U| \leq 4|U|/s$*

Remark 1. *Lemma 6 does not hold for arbitrary graphs. The bounded degree condition is necessary for its proof to go through. One obvious example where the bounded degree condition is not satisfied and the conclusion of the lemma is not true is the star graph.*

Also note that Algorithm 1 is randomized. (Thorup and Zwick, 2001) showed that it is possible to derandomize it and the running time is still polynomial.

Now, we can recursively apply this technique for each extended Voronoi cell $U \in \{\hat{D}_{a_1}, \hat{D}_{a_2}, \ldots, \hat{D}_{a_s}\}$. When applying the centres algorithm recursively, the following definitions are useful. Each node of the recursion tree is a subset of V. The root is V and it is in level 1 of the recursion. We use N_k to denote the set of nodes in level k. Hence $N_1 = \{V\}$. Let $U \in N_i$ be a node in level i. If the centres algorithm

Function *SUBSET-CENTRES*$(\hat{G}, U, s)$

$A \leftarrow \emptyset$;

while *there exists* $w \in V$ *such that* $|\hat{C}_A(w) \cap U| > 4|U|/s$ **do**

$W \leftarrow \{w \in V : |\hat{C}_A(w) \cap U| > 4|U|/s\}$;

Add each element of W to A with probability $\min(s/|W|, 1)$

end

return A

Algorithm 1: Finding Centres for a Subset

returns $A(U) = \{a_1, a_2, \dots a_k\}$ when run on U, then $C(U) = \{\hat{D}^U_{a_1}, \hat{D}^U_{a_2}, \dots, \hat{D}^U_{a_k}\}$ are the children of U.

Definition 10.

$$S^U_{\{a\}} = \{(a', p, u) : a' \in \hat{N}_2(a), u \in U, p \in \hat{p}_{a'}(u)\}$$

$$S^U_A = \bigcup_{a \in A} S^U_{\{a\}}$$

Lemma 7 (Recursion step)**.** *Assume that* $\hat{E} \subseteq E$*. Let* $U \in N_k$ *and* A *be the centres returned by the algorithm on* U*. If* $\mathrm{bet}(a, u, v) = \hat{\mathrm{bet}}(a, u, v)$ *for all* $(a, u, v) \in S^U_A$*, then every edge of* G *contained in* U *is contained in some* $\hat{D}^U_{a_i} \in C(U)$*.*

Proof. Suppose there exists an edge $e = (v_1, v_2)$ in G that is not completely contained in any of the $C(U)$. Let $v_1 \in \hat{D}_{a_1}$ and $v_2 \in \hat{D}_{a_2}$. By assumption, $v_1 \notin \hat{D}_{a_2}$ and $v_2 \notin \hat{D}_{a_1}$.

If $v_1 \in \hat{N}_2(a_1)$, then $\mathrm{bet}(v_1, p, v_2) \neq \hat{\mathrm{bet}}(v_1, p, v_2)$ for $p \in \hat{p}_{v_1}(v_2)$. Hence, $v_1 \notin \hat{N}_2(a_1)$. By the same argument, $v_2 \notin \hat{N}_2(a_2)$. For the rest of the proof, we assume $v_1 \notin \hat{N}_2(a_1)$ and $v_2 \notin \hat{N}_2(a_2)$. The definitions below are also represented in Figure 3.

1. $\hat{\delta}(a_1, v_1) = m_1$
2. $\hat{\delta}(a_1, v_2) = m_2$
3. $\hat{\delta}(a_2, v_1) = l_1$
4. $\hat{\delta}(a_2, v_2) = l_2$

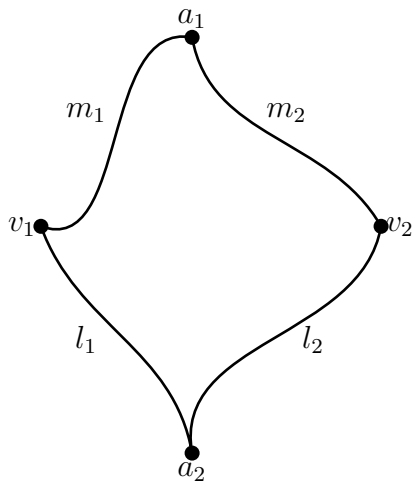


Figure 3: Pictorial representation of the definitions for Lemma 7 in Chapter 2

Let b be a vertex at distance 2 from a_2 in the shortest path from a_2 to v_1. Then, $\hat{\delta}(b, v_1) \geq \hat{\delta}(a_1, v_1)$ because $v_1 \in \hat{D}_{a_1}$ and $v_1 \notin \hat{D}_{a_2}$. Hence, we get $l_1 - 2 \geq m_1$. Similarly, $m_2 - 2 \geq l_2$.

We claim that at least one of the following statements is true:

1. $v_1 \in \hat{L}(a_1)_k$ and $v_2 \in \hat{L}(a_1)_{k+s}$ for $s > 1$.

2. $v_2 \in \hat{L}(a_2)_k$ and $v_1 \in \hat{L}(a_2)_{k+s}$ for $s > 1$.

Suppose the first statement is false. Then, $m_1 - 1 \leq m_2 \leq m_1 + 1$. Using $l_1 \geq m_1 + 2$ and $m_2 \geq l_2 + 2$, we get that $l_2 + 3 \leq l_1$. Hence, the second statement is true.

If statement i is true, then with a_i as centre, v_i and v_j belong to layers that are far apart in $\hat{G}$ (where $i \in \{1, 2\}$, $j \neq i$ and $j \in \{1, 2\}$). But they are close in G because there is an edge between v_i and v_j. We exploit this to get a contradiction. Let $v_i \in \hat{L}(a_i)^V_k$ and $v_j \in \hat{L}(a_i)^V_{k+s}$ for $s > 1$. Because there is an edge between v_i and v_j, $v_j \in L(a_i)^V_t$ for some $t \leq k + 1$. Hence, with a_i as center, the shortest path to v_j has changed in G. This changes the output of some betweenness query in G. In particular, if P denotes the set of vertices along a shortest path from a_i to v_j in $\hat{G}$, $\mathrm{bet}(a_i, p_v, v) \neq \hat{\mathrm{bet}}(a_i, p_v, v)$ for some $v \in P$ and $p_v \in \hat{p}_{a_i}(v)$. This contradicts $\mathrm{bet}(a, u, v) = \hat{\mathrm{bet}}(a, u, v)$ for all $(a, u, v) \in S^U_A$ concluding the proof.

□

Finally, we have all the machinery required to prove the main result. We need Lemma 5 for edge verification and Lemma 7 for recursion.

Proof of Theorem 3. First we do the edge verification using Lemma 5. For non-edge verification, the proof follows closely the recursive verification analysis done in (Kannan et al., 2015).

Algorithm 2 is the recursive algorithm for non-edge verification. It starts with $U = V$ and queries every $(u, v, w) \in S^U_A$ where A is the set of centres returned by the centres algorithm. Then, it repeats the process for each $\hat{D}_a$. The tree interpretation of the recursion process discussed earlier will be useful

for the rest of the proof. In Algorithm 2, Q UERY$\left(S^U_{\{a\}}\right)$ means querying every $(u, v, w) \in S^U_{\{a\}}$ and Q UERY(X, Y, Z) means querying every 3-tuple of the form (x, y, z) such that $x \in X$, $y \in Y$ and $z \in Z$. If the queries in Q UERY$\left(S^U_{\{a\}}\right)$ returns the expected result for all $a \in A(U)$, by Lemma 7, we conclude that every edge of G contained in U is contained in some $\hat{D}^U_{a_i} \in C(U)$. Now, we need to fix the constants in Algorithm 2 and compute its query complexity.

Procedure *VERIFY-SUBGRAPH($\hat{G}, U$)*

if $|U| > n_0$ **then**
  $A \leftarrow$ SUBSET-CENTRES$(\hat{G}, U, s)$
  for $a \in A$ **do**
    QUERY$\left(S^U_{\{a\}}\right)$
    VERIFY-SUBGRAPH$(\hat{G}, \hat{D}_a)$
  end
else
  QUERY(U, U, U)
end

Algorithm 2: Recursive Verification

Define

$$k_0 = \left\lfloor \sqrt{\frac{\log n}{\log(\log n \cdot 128(\Delta^2 + 1)^3)}} \right\rfloor$$

Define $s = n^{1/k_0}$, $n_0 = (4(\Delta^2+1))^{k_0}$. n_0 is going to be a threshold on —U—. If the size of $|U|$ falls below this, we stop the recursion. If $|U| > n_0$, the number of centres returned by the algorithm is $|A(U)| \leq 2s \log n$ and for any $W \in C(U)$, $|W| \leq (\Delta^2+1) \cdot \max(4|U|/s, 1)$. By induction, we get for any $U \in N_k$, $|U| \leq n(4(\Delta^2+1)/s)^{k-1}$ for all $1 \leq k \leq k_0+1$.

Consider the queries made by the leaf nodes of the tree (i.e. $|U| \leq n_0$). The depth of the tree is at most k_0+1 . Hence, there are at most $(2s \log n)^{k_0}$ leaves. Each leaf node makes at most $|U|^3 \leq (4(\Delta^2+1))^{3k_0}$ queries. Hence, the total number of queries in this step is at most $n(\log n \cdot 128(\Delta^2+1)^3)^{k_0} \leq n^{1+1/k_0}$.

Now consider the recursive calls made by non-leaf nodes (i.e. $|U| > n_0$). Here, $k \in [1, k_0]$. For a fixed k, there are at most $|N_k| = (2s \log n)^{k-1}$ calls at level k. Each such call takes at most $|A(U)||S^U_{\{a\}}| = (\Delta^2+1)\Delta|A(U)||U|$ queries where $U \in N_k$. Hence, the total number of queries for a fixed k is at most $\Delta \cdot n^{1+1/k_0}(\log n \cdot 8(\Delta^2+1))^k$. Summing over $k \in [1, k_0]$, we find the total number of queries made by non-leaf nodes is at most $2\Delta \cdot n^{1+1/k_0}(\log n \cdot 8(\Delta^2+1))^{k_0} \leq 2\Delta \cdot n^{1+2/k_0}$. This completes the proof. □

2.5 Open problems

For the graph learning problem with a distance oracle, there is an upper bound of $\tilde{O}(n^{3/2}\Delta^4)$ for graphs with constant Δ and the lower bound is $\tilde{\Omega}(n\Delta)$. The main open problem is to find an algorithm with an upper bound of $n^{1+o(1)} \cdot f(\Delta)$ where $f(\Delta)$ is some function of Δ. For the graph verification problem with a distance oracle, there is an upper bound of $n^{1+O\left(\sqrt{(\log\log n + \log\Delta)/\log n}\right)}$ and a lower bound of $\Omega(n)$. It would be interesting to find an algorithm where the dependence on Δ is of the form $\tilde{O}(nf(\Delta))$.

For the betweenness oracle, the state of the art for degree-bounded graphs is almost the same as that of the distance oracle stated above. The main difference comes from the dependence on Δ. Because betweenness queries are weaker, the dependence on Δ becomes worse. But the fact that the weaker query can get us almost the same upper bound gives motivation for improving the upper bounds for distance oracle. This paper shows how the partitioning technique developed for the graph verification problem in (Kannan et al., 2015) is robust enough to be extended to a betweenness oracle. It would be interesting to see if this technique can be extended to other oracles.

CHAPTER 3

NETWORK RECONSTRUCTION WITH ORDERED CONSTRAINTS

This chapter was previously published as Network Reconstruction with Ordered Constraints by Yi Huang, Mano Vikash Janardhanan and Lev Reyzin (Huang et al., 2017).

3.1 Introduction

In this paper, we study the problem of recovering a network after observing how information propagates through the network. Consider how a tweet (through "retweeting" or via other means) propagates through the Twitter network – we can observe the identities of the people who have retweeted it and the timestamps when they did so, but may not know, for a fixed user, via whom he got the original tweet. So we see a chain of users for a given tweet. This chain is semi-ordered in the sense that, each user retweets from some one before him in the chain, but not necessarily the one directly before him. Similarly, when a virus such as Ebola spreads, each new patient in an outbreak is infected from some one who has previously been infected, but it is often not immediately clear from whom.

In a graphical social network model with nodes representing users and edges representing links, an "outbreak" illustrated above is captured exactly by the concept of an **ordered constraint** which we will define formally below. One could hope to be able to learn something about the structure of the network by observing repeated outbreaks, or a sequence of ordered constraints.

Formally we call our problem **Network Construction with Ordered Constraints** and define it as follows. Let $V = \{v_1, \ldots, v_n\}$ be a set of vertices. An **ordered constraint** $\mathcal{O}$ is an ordering on a subset

of V of size $s \geq 2$. The constraint $\mathcal{O} = (v_{k_1}, \ldots, v_{k_s})$ is satisfied if for any $2 \leq i \leq s$, there exists at least one $1 \leq j < i$ such that the edge $e = \{v_{k_j}, v_{k_i}\}$ is included in a solution. Given a collection of ordered constraints $\{\mathcal{O}_1, \ldots, \mathcal{O}_r\}$, the task is to construct a set E of edges among the vertices V such that all the ordered constraints are satisfied and $|E|$ is minimized.

We can see that our newly defined problem resides in a middle ground between path constraints, which are too rigid to be very interesting, and the well-studied subgraph connectivity constraints (Angluin et al., 2015; Korach and Stern, 2003; Korach and Stern, 2008), which are more relaxed. The established subgraph connectivity constraints problem involves getting an arbitrary collection of connectivity constraints $\{S_1, \ldots, S_r\}$ where each $S_i \subset V$ and requires vertices in a given constraint to form a connected induced subgraph; we will occasionally refer to these as **unordered** or **general constraints**. The task is to construct a set E of edges satisfying the connectivity constraints such that $|E|$ is minimized.

We want to point out one key observation relating the ordered constraint to the connectivity constraint – an ordered constraint $\mathcal{O} = (v_{k_1}, \ldots, v_{k_s})$ is equivalent to $s - 1$ connectivity constraints $S_2, \ldots, S_s$, where $S_i = \{v_{k_1}, \ldots, v_{k_i}\}$. We note that this observation plays an important role in several proofs in this paper which employ previous results on subgraph connectivity constraints – in particular, upper bounds from the more general case can be used in the ordered case (with some overhead), and our lower bounds apply to the general problem.

In the offline version of the Network Construction with Ordered Constraints problem, the algorithm is given all of the constraints all at once; in the **online** version of the problem, the constraints are given one by one to the algorithm, and edges must be added to satisfy each new constraint when it is given. Edges cannot be removed.

An algorithm is said to be **c-competitive** if the cost of its solution is less than c times OPT, where OPT is the best solution in hindsight (c is also called the competitive ratio). When we restrict the underlying graph in a problem to be a class of graphs, e.g. trees, we mean all the constraints can be satisfied, in an optimal solution (for the online case, in hindsight), by a graph from that class.

3.1.1 Past Work

In this paper we study the problem of network construction from ordered constraints. This is an extension of the more general model where constraints come unordered.

For the general problem, Korach and Stern (Korach and Stern, 2003) had some of the initial results, in particular for the case where the constraints can be optimally satisfied by a tree, they give a polynomial time algorithm that finds the optimal solution. In subsequent work, in (Korach and Stern, 2008) Korach and Stern considered this problem for the even more restricted problem where the optimal solution forms a tree, and all of the connectivity constraints must be satisfied by stars.

Then, Angluin et al. (Angluin et al., 2015) studied the general problem, where there is no restriction on structure of the optimal solution, in both the offline and online settings. In the offline case, they gave nearly matching upper and lower bounds on the hardness of approximation for the problem. In the online case, they give a $O(n^{2/3}\log^{2/3} n)$-competitive algorithm against oblivious adversaries; we show that this bound can be drastically improved in the ordered version of the problem. They also characterized special classes of graphs, i.e. stars and paths, which we are also able to do herein for the ordered constraint case. Independently of that work, Chockler et al. (Chockler et al., 2007) also nearly characterized the offline general case.

In a different line of work Alon et al. (Alon et al., 2006) explore a wide range of network optimization problems; one problem they study involves ensuring that a network with fractional edge weights has a flow of 1 over cuts specified by the constraints. Alon et al. (Alon et al., 2009) also study approximation algorithms for the Online Set Cover problem which have been shown by Angluin et al. (Angluin et al., 2015) to have connections with Network Construction problems.

In related areas, Gupta et al. (Gupta et al., 2012) considered a network design problem for pairwise vertex connectivity constraints. Moulin and Laigret (Moulin and Laigret, 2011) studied network connectivity constraints from an economics perspective. Another motivation for studying this problem is to discover social networks from observations. This and similar problems have also been studied in the learning context (Angluin et al., 2010c; Angluin et al., 2010a; Gomez-Rodriguez et al., 2012; Saito et al., 2008).

Finally, in query learning, the problem of discovering networks from connectivity queries has been much studied (Alon and Asodi, 2005b; Alon et al., 2004b; Angluin and Chen, 2008b; Beigel et al., 2001; Grebinski and Kucherov, 1998; Reyzin and Srivastava, 2007b). In active learning of hidden networks, the object of the algorithm is to learn the network exactly. Our model is similar, except the algorithm only has the constraints it is given, and the task is to output the cheapest network consistent with the constraints.

3.1.2 Connection to network inference

This model is also known to have connections to network inference (Angluin et al., 2010b; Reyzin, 2009). Let $p_{(u,v)}$ be the a priori probability of an edge appearing between nodes u and v. If $p_{u,v}$'s are

$\leq 1/2$ and are independent, the maximum likelihood social network given the constraints is a set of edges E that satisfies all of the constraints and maximises the following quantity

$$\prod_{\{u,v\}\in E} p_{(u,v)} \prod_{\{u,v\}\notin E} \left(1 - p_{(u,v)}\right) = \prod_{\{u,v\}} \left(1 - p_{(u,v)}\right) \prod_{\{u,v\}\in E} \frac{p_{(u,v)}}{\left(1 - p_{(u,v)}\right)}$$

Taking the logarithm, we want a set of edges E that minimizes the sum

$$\sum_{\{v,u\}\in E} -\log\left(\frac{p_{(u,v)}}{\left(1 - p_{(u,v)}\right)}\right).$$

The assumption that for all u, v, $p_{(u,v)} \leq 1/2$ implies that each term, or cost, in the sum is non-negative.

3.1.3 Our results

In Section 3.2, we examine the offline problem, and show that the Network Construction problem is NP-Hard to approximate within a factor of $\Omega(\log n)$. A nearly matching upper bound comes from Theorem 2 of Angluin et al. (Angluin et al., 2015).

In Section 3.3, we study online problem. For problems on n nodes, for r constraints, we give an $O\left((\log r + \log n)\log n\right)$ competitive algorithm against oblivious adversaries, and an $\Omega(\log n)$ lower bound (Section 3.3.1).

Then, for the special cases of stars and paths (Sections 3.3.2 and 3.3.3), we find asymptotic optimal competitive ratios of $3/2$ and 2, respectively. The proof of the latter uses a detailed analysis involving PQ-trees (Booth and Lueker, 1976). The competitive ratios are asymptotic in n.

3.2 The offline problem

In this section, we examine the Network Construction with Ordered Constraints problem in the offline case. We are able to obtain the same lower bound as Angluin et al. (Angluin et al., 2015) in the general connectivity constraints case.

Theorem 4. *If P≠NP, the approximation ratio of the* ***Network Construction with Ordered Constraints*** *problem is* $\Omega(\log n)$.

Proof. We prove the theorem by reducing from the Hitting Set problem. Let $(U, \mathcal{S})$ be a hitting set instance, where $U = \{u_1, \ldots, u_n\}$ is the universe, and $\mathcal{S} = \{S_1, \ldots, S_m\}$ is a set of subsets of U. A subset $H \subset U$ is called a hitting set if $H \cap S_i \neq \emptyset$ for $i = 1, \ldots, m$. The objective of the Hitting Set problem is to minimize $|H|$. We know from (Feige, 1998; Raz and Safra, 1997) that the Hitting Set problem cannot be approximated by any polynomial time algorithm within a ratio of $o(\log n)$ unless P=NP. Here we show that the Network Construction problem is inapproximable better than an $O(\log n)$ factor by first showing that we can construct a corresponding Network Construction instance to any given Hitting Set instance, and then showing that if there is a polynomial time algorithm that can achieve an approximation ratio $o(\log n)$ to the Network Construction problem, then the Hitting Set problem can also be approximated within in a ratio of $o(\log n)$, which is a contradiction.

We first define a Network Construction instance, corresponding to a given Hitting Set instance $(U, \mathcal{S})$, with vertex set $U \cup W$, where $W = \{w_1, \ldots, w_{n^c}\}$ for some $c > 2$. Note that we use the elements of the universe of hitting set instance as a part of the vertex set of Network Construction instance. The ordered constraints are the union of the following two sets:

- $\{(u_i, u_j)\}_{1 \leq i < j \leq n}$;
- $\{(S_k, w_l)\}_{S_k \in \mathcal{S}, 1 \leq l \leq n^c}$,

where by (S_k, w_l) we mean an ordered constraint with all vertices except the last one from a subset S_k of U, while the last vertex w_l is an element in W. The vertices from S_k are ordered arbitrarily.

We note that the first set of ordered constraints forces a complete graph on U, and the second set of ordering demands that there is at least one edge going out from each S_k connecting each element in W. Let $\mathcal{A}$ be an algorithm solving the Network Construction problem, and let E_l denote the set of edges added by $\mathcal{A}$ incident to w_l. Because of the second set of ordered constraints, the set $H_l = \{u \in U | \{u, w_l\} \in E_l\}$ is a hitting set of $\mathcal{S}$!

Let $H \subset U$ be any optimal solution to the hitting set instance, and denote by $\mathrm{OPT_H}$ the size of H. It is easy to see the two sets of ordered constraints can be satisfied by putting a complete graph on U and a complete bipartite graph between H and W. Hence the optimal solution to the Network Construction instance satisfies

$$\mathrm{OPT} \leq \binom{n}{2} + n^c \, \mathrm{OPT_H},$$

where OPT is the minimum number of edges needed to solve the Network Construction instance. Let us assume that there is a polynomial time approximation algorithm to the Network Construction problem that adds ALG edges. Without loss of generality we can assume that the algorithm adds no edge among vertices in W, because any edge within W can be removed without affecting the correctness of the

solution, which implies that $\mathrm{ALG} = \binom{n}{2} + \sum_{l=1}^{n^c} |E_l|$. Now if ALG is $o\left(\log n \cdot \mathrm{OPT}\right)$, from the fact that $|H_l| = |E_l|$, we get

$$\begin{aligned} \min_{1 \le l \le n^c} |H_l| \le \frac{\mathrm{ALG} - \binom{n}{2}}{n^c} &= \frac{o\left(\log n \left(\binom{n}{2} + n^c \, \mathrm{OPT_H}\right)\right) - \binom{n}{2}}{n^c} \\ &= o\left(\log n \cdot \mathrm{OPT_H}\right), \end{aligned}$$

which means by finding the smallest set H_{l_0} among all the H_ls, we get a hitting set that has size within an $o(\log n)$ factor of the optimal solution to the Hitting Set instance, which is a contradiction.

□

We also observe that the upper bound from the more general problem implies a bound in our ordered case. We note the upper and lower bounds match when $r = poly(n)$.

Corollary 1 (of Theorem 2 from Angluin et al. (Angluin et al., 2015))**.** *There is a polynomial time $O(\log r + \log n)$-approximation algorithm for the Network Construction with Ordered Constraints problem on n nodes and r ordered constraints.*

Proof. Observing that r ordered constraints imply at most nr unordered constraints on a graph with n nodes, we can use the $O(\log nr)$ upper bound from Angluin et al. (Angluin et al., 2015).

□

3.3 The online problem

Here, we study the online problem, where constraints come in one at a time, and the algorithm must satisfy them by adding edges as the constraints arrive.

3.3.1 Arbitrary graphs

Theorem 5. *There is an* $O\left((\log r + \log n)\log n\right)$ *upper bound for the competitive ratio for the* ***Online Network Construction with Ordered Constraints*** *problem on* n *nodes and* r *ordered constraints against an oblivious adversary.*

Proof. To prove the statement, we first define the **Fractional Network Construction** problem, which has been shown by Angluin et al. (Angluin et al., 2015) to have an $O(\log n)$-approximation algorithm. The upper bound is then obtained by applying a probabilistic rounding scheme to the fractional solution given by the approximation. The proof heavily relies on arguments developed by Buchbinder and Naor (Buchbinder and Naor, 2009), and Angluin et al. (Angluin et al., 2015).

In the **Fractional Network Construction** problem, we are also given a set of vertices and a set of constraints $\{S_1, \ldots, S_r\}$ where each S_i is a subset of the vertex set. Our task is to assign weights w_e to each edge e so that the maximum flow between each pair of vertices in S_i is at least 1. The optimization problem is to minimize $\sum w_e$. Since subgraph connectivity constraint is equivalent to requiring a maximum flow of 1 between each pair of vertices with edge weight $w_e \in \{0, 1\}$, the fractional network construction problem is the linear relaxation of the subgraph connectivity problem. Lemma 2 of Angluin et al. (Angluin et al., 2015) gives an algorithm that multiplicatively updates the edge weights until all the flow constraints are satisfied. It also shows that the sum of weights given by the algorithm is upper bounded by $O(\log n)$ times the optimum.

As we pointed out in the introduction, an ordered constraint $\mathcal{O}$ is equivalent to a sequence of subgraph connectivity constraints. So in the first step, we feed the r sequences of connectivity constraints, each one is equivalent to an ordered constraint, to the approximation algorithm to the fractional network

construction problem and get the edge weights. Then we apply a rounding scheme similar to the one considered by Buchbinder and Naor (Buchbinder and Naor, 2009) to the weights. For each edge e, we choose t random variables $X(e,i)$ independently and uniformly from $[0,1]$, and let the threshold $T(e) = \min_{i=1}^{t} X(e,i)$. We add e to the graph if $w_e \geq T(e)$.

Since the rounding scheme has no guarantee to produce a feasible solution, the first thing we need to do is to determine how large t should be to make all the ordered constraints satisfied with high probability.

We note that an ordered constraint $\mathcal{O}_i = \{v_{i1}, v_{i2}, \ldots, v_{is_i}\}$ is satisfied if and only if the $(s_i - 1)$ connectivity constraints $\{v_{i1}, v_{i2}\}, \ldots, \{v_{i1}, \ldots, v_{is_i-1}, v_{is_i}\}$ are satisfied, which is equivalent, in turn, to the fact that there is an edge that goes across the $(\{v_{i1}, \ldots, v_{ij-1}\}, \{v_{ij}\})$ cut, for $2 \leq j \leq s_i$. For any fixed cut C, the probability the cut is not crossed equals $\prod_{e \in C}(1 - w_e)^t \leq \exp\left(-t \sum_{e \in C} w_e\right)$. By the max-flow min-cut correspondence, we know that $\sum_{c \in C} w_e \geq 1$ in the fractional solution given by the approximation algorithm for all cuts $C = (\{v_{i1}, \ldots, v_{ij-1}\}, \{v_{ij}\})$, $1 \leq i \leq r$, $2 \leq j \leq s_i$, and hence the probability that there exists at least one unsatisfied $\mathcal{O}_i$ is upper bounded by $rn \exp(-t)$. So $t = c(\log n + \log r)$, for any $c > 1$, makes the probability that the rounding scheme fails to produce a feasible solution approaches 0 as n increases.

Because the probability that e is added equals the probability that at least one $X(e,i)$ is less than w_e, and hence is upper bounded by $w_e t$, we get the expected number of edges added is upper bounded by $t \sum w_e$ by linearity of expectation. Since the fractional solution is upper bounded by $O(\log n)$ times the optimum of the fractional problem, which is upper bounded by any integral solution, our rounding scheme gives a solution that is $O((\log r + \log n) \log n)$ times the optimum.

□

Corollary 2. *If the number of ordered constraints $r = \mathrm{poly}(n)$, then the algorithm above gives an $O\left((\log n)^2\right)$ upper bound for the competitive ratio against an oblivious adversary.*

Remark 2. *We can generalize Theorem 5 to the weighted version of the Online Network Construction with Ordered Constraints problem. In the weighted version, each edge $e = (u, v)$ is associated with a cost c_e and the task is to select edges such that the connectivity constraints are satisfied and $\sum c_e w_e$ is minimised where $w_e \in \{0, 1\}$ is a variable indicating whether an edge is picked or not and c_e is the cost of the edge. The same approach in the proof of Theorem 5 gives an upper bound of $O\left((\log r + \log n) \log n\right)$ for the competitive ratio of the weighted version of the Online Network Construction with Ordered Constraints problem.*

For an lower bound for the competitive ratio for the Online Network Construction with Ordered Constraints problem against an oblivious adversary, we study the **Online Permutation Hitting Set** problem defined below: Let k be a positive integer, and let π be a permutation on $[k]$. Define sets

$$P_\pi^i = \{\pi(1), \pi(2), \ldots, \pi(i)\} \quad \text{for } i = 1, \ldots, k,$$

Definition 11 (**Online Permutation Hitting Set**). *Let π be a permutation on $[k]$ and let $\left(P_\pi^k, P_\pi^{k-1}, \ldots, P_\pi^1\right)$ arrive one at a time. A solution to the Online Permutation Hitting Set problem is a sequence of set $H_1 \subset H_2 \subset \cdots \subset H_k$ such that $H_i \cap P_\pi^{k+1-i} \neq \emptyset$ for $i = 1, \ldots, k$.*

Lemma 8. *The expected size of H_k for any algorithm that solves the **Online Permutation Hitting Set** problem over the space of all permutations on $[k]$ under the uniform distribution is lower bounded by $h_k = \sum_{i=1}^{k} \frac{1}{i}$.*

Proof. We first show that the lower bound is achieved by a randomized algorithm $\mathcal{A}^0$, and then we show that no other randomized algorithm could do better than $\mathcal{A}^0$.

We start by describing $\mathcal{A}^0$. Upon seeing P_π^{k+1-i}, $\mathcal{A}^0$ sets H_i using the following rule:

$$H_i = \begin{cases} H_{i-1} & \text{if } H_{i-1} \cap P_\pi^{k+1-i} \neq \emptyset \\ H_{i-1} \cup \{a\} & \text{if } H_{i-1} \cap P_\pi^{k+1-i} = \emptyset \end{cases},$$

where a is a random point in P_π^{k+1-i}. Let E_i denote the expected size of the final output of $\mathcal{A}^0$ on permutations on $[i]$ for all $i = 1, \ldots, k$. By symmetry, without loss of generality, we assume that $\mathcal{A}^0$ add 1 upon receiving $P_\pi^k = [k]$, then we have

$$E_k = 1 + \sum_{i=1}^{k} E_i \mathbb{P}\left(\pi(i) = 1\right) = 1 + \frac{1}{k} \sum_{i=1}^{k-1} E_i$$

The first equality is because $\mathcal{A}^0$ doesn't need to add more points until it receives P_π^{k-i} if $\pi(i) = 1$, and the second equality is because that all i has the same probability to be mapped to 1. To show that $E_k = h_k$, we first verify that the harmonic series satisfies the same recursive relation. In fact, we have

$$\begin{aligned} 1 + \frac{1}{k} \sum_{i=1}^{k-1} h_i &= \frac{1}{k} \left(k + \sum_{i=1}^{k-1} \sum_{j=1}^{i} \frac{1}{j} \right) = \frac{1}{k} \left(k + \sum_{i=1}^{k-1} (k-i) \cdot \frac{1}{i} \right) \\ &= \frac{1}{k} \left(k + \sum_{i=1}^{k-1} \left(k \cdot \frac{1}{i} - 1 \right) \right) = \frac{1}{k} \left(1 + k \sum_{i=1}^{k-1} \frac{1}{i} \right) = h_k. \end{aligned}$$

Since $E_1 = h_1 = 1$, we have $E_k = h_k$ for all $k \in \mathbb{N}^+$.

Next, we show that $\mathcal{A}^0$ is in fact the best randomized algorithm in expectation. To this end, we need to show two things:

i. when $H_{i-1} \cap P_\pi^{k+1-i} \neq \emptyset$, adding point(s) is counter-productive;

ii. when $H_{i-1} \cap P_\pi^{k+1-i} = \emptyset$, adding more than one points is counter-productive.

To show i., let us assume that $H_{i-1} \cap P_\pi^{k+1-i} \neq \emptyset$ and j is the smallest integer in $1 \leq j \leq k+1-i$ such that $\pi(j)$ is contained in H_{i-1}. We show that adding one more point hurts the expectation, and the proof for adding more than one points follows the same fashion. $\mathcal{A}^0$ will wait till P_π^{j-1} and add a random point from $\{\pi(1), \pi(2), \ldots, \pi(j-1)\}$. Hence, $\mathcal{A}^0$ does not assign probabilities to any point in $\{\pi(j+1), \ldots, \pi(k+1-i)\}$. We note that adding any point in $\{\pi(j+1), \ldots, \pi(k+1-i)\}$ wouldn't help. Hence, any algorithm that assigns non-zero probabilities to $\{\pi(j+1), \ldots, \pi(k+1-i)\}$ will do worse than $\mathcal{A}^0$ in expectation.

To show ii., for simplicity, we show that adding two points upon seeing P_π^k hurts the expectation, and the proof for adding more than two points follows the same fashion. We show this by induction assuming that $\mathcal{A}^0$ is the best algorithm in expectation for all $i < k$. By symmetry, without loss of generality, we assume that $H_1 = \{1, 2\}$. We have

$$\begin{aligned}
& \mathbb{E}\left(|H_k||H_1 = \{1,2\}\right) \\
= \;& 2 + \left(\sum \mathbb{E}\left(|H_k||\pi^{-1}(1) < \pi^{-1}(2)\right)\right) \mathbb{P}\left(\pi^{-1}(1) < \pi^{-1}(2)\right) \\
& + \left(\sum \mathbb{E}\left(|H_k||\pi^{-1}(1) > \pi^{-1}(2)\right)\right) \mathbb{P}\left(\pi^{-1}(1) > \pi^{-1}(2)\right) \\
\geq \;& 2 + 2 \cdot \frac{1}{2}\left(\frac{1}{k-1}\sum_{i=1}^{k-2} E_i\right) = 2 + \frac{1}{k-1}\sum_{i=1}^{k-2} E_i = 1 + E_{k-1} > E_k,
\end{aligned}$$

where the first inequality follows from i. and the inductive hypothesis.

□

With the help of Lemma 8, we can prove the following lower bound.

Theorem 6. *There exists an* $\Omega(\log n)$ *lower bound for the competitive ratio for the* ***Online Network Construction with Ordered Constraints*** *problem against an oblivious adversary.*

Proof. The adversary divides the vertex set into two parts U and V, where $|U| = \sqrt{n}$ and $|V| = n - \sqrt{n}$, and gives the constraints as follows. First, it forces a complete graph in U by giving the constraint $\{u_i, u_j\}$ for each pair of vertices $u_i, u_j \in U$. At this stage both the algorithm and optimal solution will have a clique in U, which costs $\Theta(n)$.

Then, for each $v \in V$, first fix a random permutation π_v on U and give the ordered constraints

$$\mathcal{O}_{(v,i)} = (\pi_v(1), \pi_v(2), \ldots, \pi_v(i), v) \quad \text{for } i = 1, \ldots, \sqrt{n}.$$

First note that all these constraints can be satisfied by adding $e_v = \{\pi_v(1), v\}$ for each $v \in V$ which costs $\Theta(n)$. However, the adversary gives constraints in the following order:

$$\mathcal{O}_{(v,\sqrt{n})}, \mathcal{O}_{(v,\sqrt{n}-1)}, \ldots, \mathcal{O}_{(v,1)}.$$

To satisfy $O_{(v,i)}$, any algorithm just need to make sure that at least one point in P_π^i is chosen to connect to v, and hence the algorithm is in fact solving an instance of the Online Permutation Hitting Set problem on input π at this stage. By Proposition 8, we know that all algorithm solving the Online Permutation

Hitting Set problem will add $\Omega\left(\log \sqrt{n}\right)$ points in expectation, and this shows that any algorithm to the Online Network Construction with Ordered Constraints problem needs to add $\Omega(n + n \log n)$ edges in expectation in total. This gives us the desired result because $\mathrm{OPT} = O(n)$.

□

Now we study the online problem when it is known that an optimal graph can be a star or a path. These special cases are challenging in their own right and are often studied in the literature to develop more general techniques (Angluin et al., 2015).

3.3.2 Stars

Theorem 7. *The optimal competitive ratio for the **Online Network Construction with Ordered Constraints** problem when the algorithm knows that the optimal solution forms a **star** is asymptotically* $3/2$.

Proof. For the lower bound, we note that the adversary can simply give $\mathcal{O}_i = (v_1, v_2, v_i)$ for all $i = 3, \ldots, n$ obliviously for the first $n - 2$ rounds. Then an algorithm, besides adding $\{v_1, v_2\}$ in the first round, can only choose from adding either $\{v_1, v_i\}$ or $\{v_2, v_i\}$, or both in each round. Note that v_1 or v_2 have to be the center since the first constraint ensures that $\{v_1, v_2\}$ is an edge. After the first $n - 2$ rounds, the adversary counts the number of v_1 and v_2's neighbors, and chooses the one with fewer neighbors, say v_1, to be the center by giving the constraints (v_1, v_i) for all $i = 3, \ldots, n$ where no edge (v_1, v_i) exists. Since the algorithm has to add at least $\lceil (n-2)/2 \rceil$ edges that are unnecessary in the hindsight, we get an asymptotic lower bound $3/2$.

For the upper bound, assume that either v_1 or v_2 is the center and the first ordered constraint is $\mathcal{O}_1$ is $(v_1, v_2, \ldots)$, the algorithm works as follows:

1. It adds $\{v_1, v_2\}$ in the first round.
2. For any constraint (including the first) that starts with v_1 and v_2, the algorithm always adds edges of the form (v_1, v_k) or (v_2, v_k) where $k \neq 1, 2$ in such a way that the following two conditions hold:
   - Degree of v_1 and degree of v_2 differ by at most 1.
   - The degree of v_k is 1 for $k \neq 1, 2$
3. Upon seeing a constraint that does not start with v_1 and v_2, which reveals the center of the star, it connects the center to all vertices that are not yet connected to the center.

Since the algorithm adds, at most $n/2 - 1$ edges to the wrong center, this gives us an asymptotic upper bound $3/2$, which matches the lower bound.

□

3.3.3 Paths

In the next two theorems, we give matching lower and upper bounds (in the limit) for path graphs.

Theorem 8. *The **Online Network Construction with Ordered Constraints** problem when the algorithm knows that the optimal solution forms a **path** has an asymptotic lower bound of* 2 *for the competitive ratio.*

Proof. Let us name the vertices $v_1, v_2, v_3, \ldots, v_n$. For $3 \leq i \leq n$, define the **pre-degree** of a vertex v_i to be the number of neighbors v_i has in $\{v_1, v_2, v_3, \ldots, v_{i-1}\}$. Algorithm 3 below is a simple strategy

Give ordered constraint $\mathcal{O} = (v_1, v_2, v_3, \ldots, v_n)$ to the algorithm;

for $i = 3$ to n **do**

  if the pre-degree of v_i is at least 2 **then**

    continue;

  else

    pick a path on $\{v_1, v_2, v_3, \ldots, v_{i-1}\}$ (say P_i) that satisfies all the constraints up to this round (the existence of P_i follows by induction) and an endpoint u of the path that is not connected to v_i, and give the algorithm the constraint (v_i, u);

  end if

end for

Algorithm 3: Forcing pre-degree to be at least 2

the adversary can take to force $v_3, \ldots, v_n$ to all have pre-degree at least 2. Since any algorithm will add at least $2n - 3$ edges, this gives an asymptotic lower bound of 2.

Suppose P_i was the path picked in round i (i.e. P_i satisfies all constraints up to round i). Then, P_i along with the edge (v_i, u) is a path that satisfies all constraints up to round $i + 1$. Hence by induction, for all i, there is a path that satisfies all constraints given by the adversary up to round i.

□

Theorem 9. *The competitive ratio for the* ***Online Network Construction with Ordered Constraints*** *problem when the algorithm knows that the optimal solution forms a* ***path*** *has an asymptotic upper bound of* 2.

Proof. For our algorithm matching the upper bound, we use the PQ-trees, introduced by Booth and Lueker (Booth and Lueker, 1976), which keep track all consistent permutations of vertices given contiguous intervals of vertices, since vertices in any ordered constraint form a contiguous interval if the underlying graph is a path. Our analysis is based on ideas from Angluin et al. (Angluin et al., 2015), who also use PQ-trees for analyzing the general problem.[1]

A **PQ-tree** is a tree whose leaf nodes are the vertices and each internal node is either a **p-node** or a **q-node**.

- A **p-node** has two or more children of any type. The children of a p-node form a contiguous interval that can be in any order.
- A **q-node** has three or more children of any type. The children of a q-node form a contiguous interval, but can only be in the given order or its reverse.

Every time a new interval constraint comes, the tree updates itself by identifying any of the eleven patterns, P0, P1,. . . , P6, and Q0, Q1, Q2, Q3, of the arrangement of nodes and replacing it with each correspondent replacement. The update fails when it cannot identify any of the patterns, in which case

---

[1] Angluin et al. (Angluin et al., 2015) have a small error in their argument because their potential function fails to explicitly consider the number of p-nodes, which creates a problem for some of the PQ-tree updates. We fix this, without affecting their asymptotic bound. For the ordered constraints case, we are also able to obtain a much finer analysis.

the contiguous intervals fail to produce any consistent permutation. We refer readers to Section 2 of Booth and Lueker (Booth and Lueker, 1976) for a more detailed description of PQ-trees.

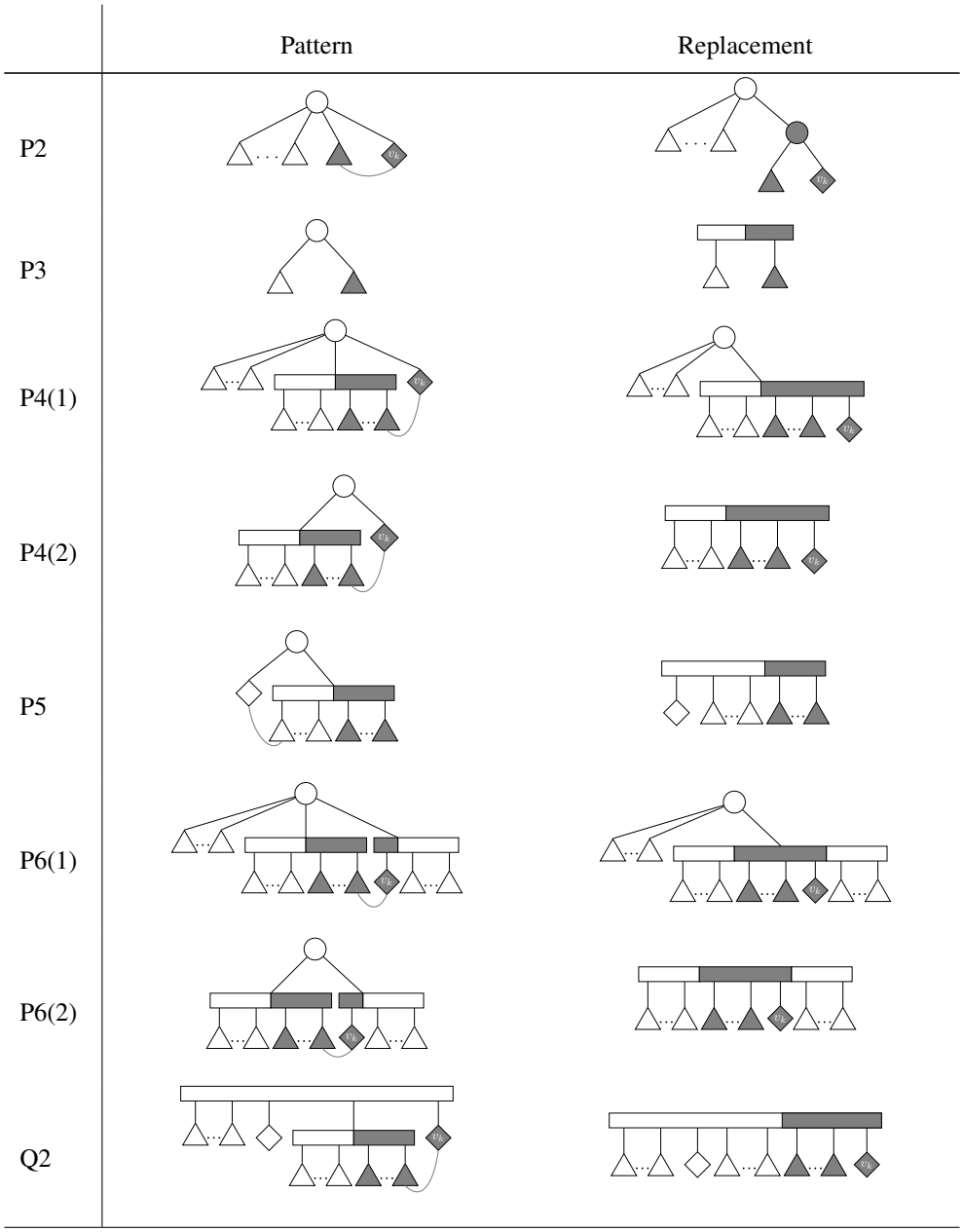


TABLE I: Specific patterns and replacements that appear through the algorithm. P4(1) denotes the case of P4 where the top p-node is retained in the replacement and P4(2) denotes the case where the top p-node is deleted. The same is true for P6. P0, P1, Q0, and Q1 are just relabelling rules, and we have omitted them because no edges need to be added. We use the same shapes to represent p-nodes, q-nodes, and subtrees as in Booth and Lueker's paper for easy reference, and we use diamonds to represent leaf nodes.

The reason we can use a PQ-tree to guide our algorithm is because of an observation made in Section 3.1 that each ordered constraint $(v_1, v_2, v_3, \ldots, v_{k-1}, v_k)$ is equivalent to $k-1$ connectivity constraints $S_2, \ldots, S_k$, where $S_i = \{v_1, \ldots, v_i\}$. Note that each connectivity constraint corresponds to an interval in the underlying graph. So upon seeing one ordered constraints, we reduce the PQ-tree with the equivalent interval constraints, *in order*. Then what our algorithm does is simply to add edge(s) to the graph every time a pattern is identified and replaced with its replacement, so that the graph satisfies all the seen constraints. Note that to reduce the PQ-tree with one interval constraint, there may be multiple patterns identified and hence multiple edges may be added.

Before running into details of how the patterns determine which edge(s) to add, we note that, without loss of generality, we can assume that the algorithm is in either one of the following two stages.

- The PQ-tree is about to be reduced with $\{v_1, v_2\}$.
- The PQ-tree is about to be reduced with $\{v_1, \ldots, v_k\}$, when the reductions with $\{v_1, v_2\} \cdots, \{v_1, \ldots, v_{k-1}\}$ have been done.

Because of the structure of constraints discussed above, we do not encounter all PQ-tree patterns in their full generality, but in the special forms demonstrated in Table I. Based on this, we make three important observations which can be verified by carefully examining how a PQ-tree evolves along with our algorithm.

1. The only p-node that can have more than two children is the root.
2. At least one of the two children of a non-root p-node is a leaf node.
3. For all q-nodes, there must be at least one leaf node in any two adjacent children. Hence, Q3 doesn't appear.

Now we describe how the edges are going to be added. Note that a PQ-tree inherently learns edges that appear in the optimum solution even when those edges are not forced by constraints. Apart from adding edges that are necessary to satisfy the constraints, our algorithm will also add any edge that the PQ-tree inherently learns. For all the patterns except Q2 such that a leaf node v_k is about to be added as a child to a different node, we can add one edge joining v_k to v_{k-1}. For all such patterns except Q2, it is obvious that this would satisfy the current constraint and all inherently learnt edges are also added. For Q2, the PQ-tree could learn two edges. The first edge is (v_k, v_{k-1}). The second one is an edge between the leftmost subtree of the daughter q-node (call T_l) and the node to its left (call v_l). Based on Observation 3, v_l is a leaf. But based on the algorithm, one of these two edges is already added. Hence, we only need to add one edge when Q2 is applied. For P5, we add the edge as shown in Table I.

	$\sum_{p\in P} c(p)$	$\lvert P\rvert$	$\lvert Q\rvert$	$-\Delta\Phi$	number of edges added
P2	1	1	0	$-a-b$	1
P3	-2	-1	1	$2a+b-c$	0
P4(1)	-1	0	0	a	1
P4(2)	-2	-1	0	$2a+b$	1
P5	-2	-1	0	$2a+b$	1
P6(1)	-1	0	-1	$a+c$	1
P6(2)	-2	-1	-1	$2a+b+c$	1
Q2	0	0	-1	c	1
Q3	0	0	-2	$2c$	1

TABLE II: How the terms in the potential function: $\sum_{p\in P} c(p)$, $|P|$, and $|Q|$ change according to the updates.

Let us denote by P and Q the sets of p-nodes and q-nodes, respectively, and by $c(p)$ the number of children node p has. And let potential function ϕ of a tree T be defined as

$$\phi(T) = a\sum_{p\in P} c(p) + b|P| + c|Q|,$$

where a, b, and c are coefficients to be determined later.

We want to upper bound the number of edges added for each pattern by the drop of potential function. We collect the change in the three terms in the potential function that each replacement causes in Table II,

and we can solve a simple linear system to get that choosing $a = 2$, $b = -3$, and $c = 1$ is sufficient. For ease of analysis, we add a dummy vertex v_{n+1} that does not appear in any constraint. Now, the potential function starts at $2n - 1$ (a single p-node with $n + 1$ children) and decreases to 2 when a path is uniquely determined. Hence, the number of edges added by the algorithm is $2n - 3$, which gives the desired asymptotic upper bound.

□

CHAPTER 4

DETECTING NETWORK ANOMALIES VIA NON-LOCAL CURVATURES

This chapter is based on the preprint How did the shape of your network change? (On detecting anomalies in static and dynamic networks via change of non-local curvatures) by DasGupta, Bhaskar and Janardhanan, Mano Vikash and Yahyanejad, Farzaneh (DasGupta et al., 2018).

4.1 Introduction

Useful insights for many complex systems are often obtained by representing them as networks and analyzing them using *graph-theoretic* and *combinatorial* algorithmic tools (DasGupta and Liang, 2016; Newman, 2010; Albert and l. Barabási, 2002). In principle, we can classify these networks into *two* major classes:

- ▷ *Static* networks that model the corresponding system by *one* fixed network. Examples of such networks include biological signal transduction networks *without* node dynamics, and most social networks.
- ▷ *Dynamic* networks where *elementary components* of the network (such as nodes or edges) are added and/or removed as the network *evolves* over time. Examples of such networks include biological signal transduction networks *with* node dynamics, causal networks reconstructed from DNA microarray time-series data, biochemical reaction networks and *dynamic* social networks.

Typically, such networks may have so-called *critical* (*elementary*) *components* whose presence or absence alters some significant *non-trivial non-local* property of these networks. For example:

- ▷ For a *static network*, there is a rich history in finding various types of critical components dating back to quantifications of *fault-tolerance* or *redundancy* in electronic circuits or routing networks. Recent examples of practical application of determining critical and non-critical components in the context of systems biology include quantifying *redundancies* in biological networks (Kolb and Whishaw, 1996; Tononi et al., 1999; Albert et al., 2011) and confirming the existence of *central influential* neighborhoods in biological networks (Albert et al., 2014).
- ▷ For a *dynamic network*, critical components may correspond to a set of nodes or edges whose addition and/or removal *between two time steps* alters a significant topological property of the network. Popularly also known as the *anomaly detection* or *change-point detection* (Aminikhanghahi and Cook, 2017; Kawahara and Sugiyama, 2009) problem, these types of problems have been studied over the last several decades in data mining, statistics and computer science *mostly in the context of time series data* with applications to areas such as medical condition monitoring (Yang et al., 2006; Bosc et al., 2003), weather change detection (Ducre-Robitaille et al., 2003; Reeves et al., 2007) and speech recognition (Chowdhury et al., 2011; Rybach et al., 2009).

In this paper we seek to address research questions of the following *generic* nature:

> *"Given a static or dynamic network, identify the critical components of the network that "encode" significant non-trivial global properties of the network"*.

To identify critical components, one first needs to provide details for following four specific items:

(*i*) *network model selection*,

(*ii*) *network evolution rule for dynamic networks*,

(*iii*) *definition of elementary critical components*, and

(*iv*) *network property selection* (*i.e.*, the global properties of the network to be investigated).

The specific details for these items for this paper are as follows:

(*i*) Network model selection: Our network model will be undirected graphs.

(*ii*) Network evolution rule for dynamic networks: Our dynamic networks follow the time series model and are given as a sequence of networks over *discrete* time steps, where each network is obtained from the previous one in the sequence by adding and/or deleting some nodes and/or edges.

(*iii*) Critical component definition: Individual edges are elementary members of critical components.

(*iv*) Network property selection: The network measure for this paper will be appropriate notions of "network curvature". More specifically, we will use **(*a*) Gromov-hyperbolic combinatorial curvature** based on the properties of *exact and approximate* geodesics distributions and higher-order connectivities and **(*b*) geometric curvatures** *based on identifying network motifs with geometric complexes* ("geometric motifs" in systems biology jargon) *and then using Forman's combinatorializations*.

4.1.1 Some basic definitions and notations

For an undirected unweighted graph $G = (V, E)$ of n nodes $v_1, \ldots, v_n$, the following notations related to G are used throughout:

- $v_{i_1} \leftrightarrow v_{i_2} \leftrightarrow v_{i_3} \leftrightarrow \cdots \leftrightarrow v_{i_{k-1}} \leftrightarrow v_{i_k}$ denotes a path of *length* $k-1$ consisting of the edges $\{v_{i_1}, v_{i_2}\}, \{v_{i_2}, v_{i_3}\}, \ldots, \{v_{i_{k-1}}, v_{i_k}\}$.

- $\overline{u, v}$ and $\mathrm{dist}_G(u, v)$ denote a *shortest path* and the distance (*i.e.*, number of edges in $\overline{u, v}$) between nodes u and v, respectively.
- $\mathsf{diam}(G) = \max_{v_i, v_j} \{\mathrm{dist}_G(v_i, v_j)\}$ denotes the *diameter* of G.
- $G \setminus E'$ denotes the graph obtained from G by removing the edges in E' from E.

A ε-approximate solution (or simply an ε-approximation) of a minimization (*resp*., maximization) problem is a solution with an objective value no larger than (*resp*., no smaller than) ε times (*resp*., $1/\varepsilon$ times) the value of the optimum; an algorithm of *performance* or *approximation ratio* ε produces an ε-approximate solution. A problem is ε-*inapproximable* under a certain complexity-theoretic assumption means that the problem does not admit a polynomial-time ε-approximation algorithm assuming that the complexity-theoretic assumption is true. We will also use other *standard* definitions from structural complexity theory as readily available in any graduate level textbook on algorithms such as (Vazirani, 2001).

4.1.2 Why use network curvature measures?

Prior researchers have proposed and evaluated a number of established network measures such as *degree-based measures* (*e*.g., degree distribution), *connectivity-based measures* (*e*.g., clustering coefficient), *geodesic-based measures* (*e*.g., betweenness centrality) and other more novel network measures (Colizza et al., 2006; Latora and Marchior, 2007; Albert et al., 2011; Bassett et al., 2011) for analyzing networks. The network measures considered in this paper are "appropriate notions" of *network curvatures*. As provably demonstrated in published research works such as (Albert et al., 2014; Weber et al., 2016a; Weber et al., 2016b; Samal et al., 2018), these network curvature measures saliently encode *non-trivial higher-order correlation* among nodes and edges that *cannot* be obtained by other

popular network measures. Some important characteristics of these curvature measures that we consider are (Albert et al., 2014, Section (III))(Jonckheerea et al., 2011):

- These curvature measures depend on *non-trivial global* network properties, as opposed to measures such as *degree distributions* or *clustering coefficients* that are *local* in nature or *dense subgraphs* that use *only* pairwise correlations.
- These curvature measures can mostly be computed efficiently in polynomial time, as opposed to measures such as *community decompositions*, *cliques* or *densest-k-subgraphs*.
- When applied to real-world biological and social networks, these curvature measures can explain *many phenomena one frequently encounters in real network applications that are not easily explained by other measures* such as:
  - paths mediating up- or down-regulation of a target node starting from the same regulator node in *biological regulatory networks* often have many small crosstalk paths, and
  - existence of congestions in a node that is not a hub in *traffic networks*.

  Further details about the suitability of our curvature measures for real biological or social networks are provided in Section 4.2.1.1 for Gromov-hyperbolic curvature and Section 4.2.2.3 for geometric curvatures.

Curvatures are very natural measures of anomaly of higher dimensional objects in mainstream physics and mathematics (Bridson and Haefliger, 1999; Berger, 2012). However, networks are *discrete objects* that *do not necessarily have an associated natural geometric embedding*. Our paper seeks to adapt the definition of curvature from the non-network domains in a suitable way for detecting network anomalies.

For example, in networks with sufficiently small Gromov-hyperbolicity and sufficiently large diameter a suitably small subset of nodes or edges can be removed to stretch the geodesics between two distinct parts of the network by an exponential amount leading to extreme implications on the expansion properties of such networks (Benjamini, 1998; DasGupta et al., 2018), which is akin to the characterization of singularities (an extreme anomaly) by geodesic incompleteness (*i.e.*, stretching all geodesics passing through the region infinitely) (Hawking and Penrose, 1996). *It is our hope that research works in this paper will stimulate further research concerning the exciting interplay between curvatures from network and non-network domains, a much desired goal in our opinion*.

4.1.2.1 Scalar vs. vector curvature

In this paper, we consider a scalar curvature measure $\mathfrak{C} \stackrel{\text{def}}{=} \mathfrak{C}(G) : \mathfrak{G} \rightarrow \mathbb{R}$. The standard Gromov-hyperbolic curvature measure is always a scalar value. Geometric curvatures however could also be defined by a vector by looking at local curvatures at all *elementary components* (*e*.g., nodes or edges) of a network, and defining the overall curvature as a *vector* of these values. We leave algorithmic analysis of such geometric vector curvatures, which seems to require considerably different combinatorial and optimization tools, for future research. Both scalar and vector versions of curvatures are used in physics and mathematics to study higher-dimensional objects with their own pros and cons. For example, for a two-dimensional curve, the standard curvature as defined by Cauchy is a scalar curvature whereas the normal vector used in the study of differential geometry of curves is a vector curvature. Even though a casual glance may seem to suggest that the scalar curvature is a weak concept with inadequate influence on the global geometry of the higher-dimensional object that is being studied, there exists non-trivial

results (*e*.g., the positive mass theorem of Schoen, Yau and Witten) that suggest that this may *not* be the case.

4.1.3 Why only the edge-deletion model?

In this paper we add or delete edges from a network while keeping the node set the same. This scenario captures a wide variety of applications such as inducing desired outcomes in disease-related biological networks via gene knockout (Saadatpour et al., 2011; Zanudo and Albert, 2015), inference of minimal biological networks from indirect experimental evidences or gene perturbation data (Albert et al., 2007; Albert et al., 2008; Wagner, 2002), and finding influential nodes in social and biological networks (Albert et al., 2011), to name a few. However, the node addition/deletion model or a mixture of node/edge addition/deletion model is also significant in many other applications. Although some of our complexity results can be easily extended for bounded-degree graphs to the node deletion model, we do not outline these generalizations here but leave it as a separate future research topic.

4.1.4 Two examples in which curvature measures detect anomaly where other simpler measures do not

It is obviously practically impossible to compare our curvatures measures for anomaly detection with respect to *every possible* other network measure that has been used in prior research works. However, we do still provide two illustrative examples of comparing our curvature measures to the well-known *densest subgraph measure* which is defined as follows. Given a graph $G = (V, E)$, the densest subgraph measure find a subgraph (S, E_S) induced by a subset of nodes $\emptyset \subset S \subseteq V$ that maximizes the ratio (density) $\rho(S) \stackrel{\text{def}}{=} \frac{|E_S|}{|S|}$. Let $\rho(G) \stackrel{\text{def}}{=} \max_{\emptyset \subset S \subseteq V}\{\rho(S)\}$ denote the density of a densest subgraph of G. An efficient polynomial time algorithm to compute $\rho(G)$ using a max-flow technique was first provided

by Goldberg (Goldberg, 1984). *We urge the readers to review the definitions of the relevant curvature measures (in Section 4.2) and the anomaly detection problems (in Section 4.3) in case of any confusion regarding the examples we provide.*

4.1.4.1 Extremal anomaly detection for a static network

Consider the extremal anomaly detection problem (Problem Eadp in Section 4.3.1) for a network $G = (V, E)$ of 10 nodes and 20 edges as shown in Fig. 4 using the geometric curvature $\mathfrak{C}_3^2$ as defined by Equation (Equation 4.1). It can be easily verified that $\mathfrak{C}_3^2(G) = 6$ and $\rho(G) = 9/4$. Let $\widetilde{E} = E$ and suppose that we set our targeted decrease of the curvature or density value to be 75% of the original value, *i.e.*, we set $\gamma = 3/4 \times \mathfrak{C}_3^2(G) = 9/2$ for the geometric curvature measure and $\gamma = 3/4 \times \rho(G) = 27/16$ for the densest subgraph measure. It is easily verified that $\mathfrak{C}_3^2(G \setminus \{e_1\}) = 0$, thus showing $\mathsf{OPT}_{\text{Eadp}_{\mathfrak{C}_3^2}}(G, \widetilde{E}, \gamma) = 1$. However, many more than just one edge will need to be deleted from G to bring down the value of $\rho(G)$ to $27/16$.

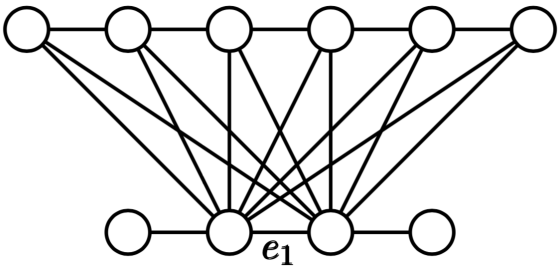


Figure 4: Toy example of extremal anomaly detection discussed in Section 4.1.4.1.

4.1.4.2 Targeted anomaly detection for a dynamic biological network

$$x_1(0) = x_2(0) = x_3(0) = x_4(0) = 0$$

$$\forall t = 0, 1, 2, \ldots\ldots$$

$$x_2(t+1) = x_2(t) + 0.8x_1(t)$$

$$x_3(t+1) = x_3(t) + 0.8x_2(t)$$

$$x_4(t+1) = x_4(t) + 0.8x_3(t) + \frac{0.4\,x_1(t)}{1 + \mathbf{e}^{-3.66\,t+11}}$$

t	...	2	3	...
$x_1(t)$	...	1.00	1.00	...
$x_2(t)$	...	0.80	0.80	...
$x_3(t)$	...	0.64	0.64	...
$x_4(t)$	...	0.52	0.90	...

microarray output

$> 0.60 \Rightarrow$ **ON (1)**

$< 0.55 \Rightarrow$ **OFF (0)**

$\delta =$ **value of Gromov curvature**

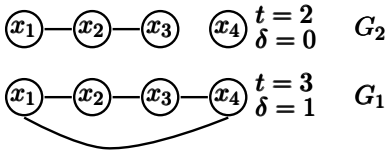


Figure 5: Toy example of targeted anomaly detection discussed in Section 4.1.4.2.

Consider the targeted anomaly detection problem (Problem Tadp in Section 4.3.2) for a dynamic biological network of 4 variables x_1, x_2, x_3, x_4 as shown in Fig. 5, where x_1 affects x_4 with a delay, using the Gromov-hyperbolic curvature (Definition 12). Suppose that the network inference from microarray data is done by incorporating a time delay of two in the hitting-set approach of Krupa (Jarrah et al., 2007). It can be easily verified that $\mathfrak{C}_{\mathrm{Gromov}}(G_1) = \rho(G_1) = 1$, $\mathfrak{C}_{\mathrm{Gromov}}(G_2) = 0$, and $\rho(G_2) = 1/2$. Since $\mathfrak{C}_{\mathrm{Gromov}}(G_1 \setminus \{x_1, x_4\}) = 0$ it follows that $\mathsf{OPT}_{\mathsf{Tadp}_{\mathfrak{C}_{\mathrm{Gromov}}}}(G_1, G_2) = 1$; however, 2 edges will need to be deleted from G_1 to bring down the value of $\rho(G_1)$ to $\rho(G_2)$.

4.1.5 Algebraic approaches for anomaly detection

In contrast to the combinatorial/geometric graph-property based approach investigated in this paper and elsewhere, an alternate approach for anomaly detection is the *algebraic tensor-decomposition based approach* studied in the contexts of dynamic social networks (Sun et al., 2006) and pathway reconstructions in cellular systems and microarray data integration from several sources (Alter and Golub, 2005; Omberg et al., 2007). This approach is quite different from the ones studied in this paper with its own pros and cons.

4.1.6 Remarks on the organization of our proofs

Many of our proofs in Sections 4.4–4.5 are long, complicated and/or involve tedious calculations. For easier understanding and to make the paper more readable, when appropriate we have included a subsection generically titled "*Proof techniques and relevant comments regarding Theorem . . . . . .*" before providing the actual detailed proofs. *The reader is cautioned however that these brief subsections are meant to provide some general idea and subtle points behind the proofs and should not be considered as a substitution for more formal proofs.*

4.2 Two notions of graph curvature

For this paper, a *curvature* for a graph G is a *scalar-valued* function $\mathfrak{C} \stackrel{\text{def}}{=} \mathfrak{C}(G) : \mathfrak{G} \to \mathbb{R}$. There are several ways in which network curvature can be defined depending on the type of global properties the measure is desired to affect; in this paper we consider two such definitions as described subsequently.

4.2.1 Gromov-hyperbolic curvature

This measure for a metric space was first suggested by Gromov in a group theoretic context (Gromov, 1987). The measure was first defined for *infinite* continuous metric space (Bridson and Haefliger, 1999),

but was later also adopted for *finite* graphs. Usually the measure is defined via *geodesic triangles* as stated in Definition 12. For this definition, it would be useful to consider the given graph G as a metric graph, *i.e.*, we identify (by an isometry) any edge $\{u, v\} \in E$ with the real interval $[0, 1]$ and thus any point in the interior of the edge $\{u, v\}$ can also be thought as a (virtual) node of G. Define a geodesic triangle $\Delta_{u,v,w}$ to be an ordered triple of three shortest paths ($\overline{u, v}$, $\overline{u, w}$ and $\overline{v, w}$) for the three nodes u, v, w in G.

Definition 12 (Gromov-hyperbolic curvature measure via geodesic triangles)**.** *For a geodesic triangle $\Delta_{u,v,w}$, let $\mathfrak{C}_{\text{Gromov}}(\Delta_{u,v,w})$ be the minimum number such that $\overline{u, v}$ lies in a $\mathfrak{C}_{\text{Gromov}}(\Delta_{u,v,w})$-neighborhood of $\overline{u, w} \cup \overline{v, w}$, i.e., for every node x on $\overline{u, v}$, there exists a node y on $\overline{u, w}$ or $\overline{v, w}$ such that $\text{dist}_G(x, y) \leq \mathfrak{C}_{\text{Gromov}}(\Delta_{u,v,w})$. Then the graph G has a Gromov-hyperbolic curvature (or Gromov hyperbolicity) of $\mathfrak{C}_{\text{Gromov}} \stackrel{\text{def}}{=} \mathfrak{C}_{\text{Gromov}}(G)$ where $\mathfrak{C}_{\text{Gromov}}(G) = \min_{u,v,w \in V} \{\mathfrak{C}_{\text{Gromov}}(\Delta_{u,v,w})\}$.*

An infinite collection $\mathcal{G}$ of graphs belongs to the class of $\mathfrak{C}_{\text{Gromov}}$-Gromov-hyperbolic graphs if and only if any graph $G \in \mathcal{G}$ has a Gromov-hyperbolic curvature of $\mathfrak{C}_{\text{Gromov}}$. Informally, any infinite metric space has a finite value of $\mathfrak{C}_{\text{Gromov}}$ if it behaves metrically in the large scale as a *negatively curved* Riemannian manifold, and thus the value of $\mathfrak{C}_{\text{Gromov}}$ can be related to the standard scalar curvature of a hyperbolic manifold. For example, a simply connected complete Riemannian manifold whose sectional curvature is below $\alpha < 0$ has a value of $\mathfrak{C}_{\text{Gromov}} = O\big(\sqrt{-\alpha}\big)$ (see (Roe, 1996)). This is a major justification of using $\mathfrak{C}_{\text{Gromov}}$ as a notion of curvature of any metric space.

For an n-node graph G, $\mathfrak{C}_{\text{Gromov}}(G)$ and a 2-approximation of $\mathfrak{C}_{\text{Gromov}}(G)$ can be computed in $O\left(n^{3.69}\right)$ and in $O\left(n^{2.69}\right)$ time, respectively (Fournier et al., 2015). It is easy to see that if G is a tree then $\mathfrak{C}_{\text{Gromov}}(G) = 0$. Other examples of graph classes for which $\mathfrak{C}_{\text{Gromov}}(G)$ is a small constant

include *chordal graphs*, *cactus of cliques*, *AT-free* graphs, *link graphs of simple polygons*, and *any* class of graphs with a *fixed* diameter. A small value of Gromov-hyperbolicity is often crucial for algorithmic designs; for example, several routing-related problems or the diameter estimation problem become easier for networks with small $\mathfrak{C}_{\mathrm{Gromov}}$ values (Chepoi and Estellon, 2007; Chepoi et al., 2008; Chepoi et al., 2012; Gavoille and Ly., 2005). There are many well-known measures of curvature of a continuous surface or other similar spaces (*e*.g., curvature of a manifold) that are widely used in many branches of physics and mathematics. It is possible to relate Gromov-hyperbolic curvature to such other curvature notions indirectly via its scaled version, *e*.g., see (Jonckheere et al., 2007; Narayan and Saniee, 2011; Jonckheere et al., 2011).

4.2.1.1 Is Gromov-hyperbolic curvature a suitable statistically significant measure for real-world networks ?

Recently, there has been a surge of empirical works measuring and analyzing the Gromov curvature $\mathfrak{C}_{\mathrm{Gromov}}$ of networks, and many real-world networks (*e*.g., preferential attachment networks, networks of high power transceivers in a wireless sensor network, communication networks at the IP layer and at other levels) were observed to have a small constant value of $\mathfrak{C}_{\mathrm{Gromov}}$ (Narayan and Saniee, 2011; Papadopoulos et al., 2010; Jonckheere and Lohsoonthorn, 2004; Jonckheere et al., 2007; Ariaei et al., 2008). The authors in (Albert et al., 2014) analyzed 11 well-known biological networks and 9 well-known social networks for their $\mathfrak{C}_{\mathrm{Gromov}}$ values and found all but one network had a *statistically significant small* value of $\mathfrak{C}_{\mathrm{Gromov}}$. These references also describe implications of range of $\mathfrak{C}_{\mathrm{Gromov}}$ on the actual real-world applications of these networks. As mentioned in the following subsection,

the Gromov-hyperbolicity measure is **fundamentally different from the commonly used topological properties for a graph**.

4.2.1.2 Some clarifying remarks regarding Gromov-hyperbolicity measure

As pointed out in details by the authors in (DasGupta et al., 2018, Section 1.2.1), the Gromov-hyperbolicity measure $\mathfrak{C}_{\text{Gromov}}$ enjoys many non-trivial topological characteristics. In particular, the authors in (DasGupta et al., 2018, Section 1.2.1) point out the following:

- ▷ $\mathfrak{C}_{\text{Gromov}}$ is *not* a hereditary or monotone property.
- ▷ $\mathfrak{C}_{\text{Gromov}}$ is *not* necessarily the same as tree-width measure (see also (de Montgolfier et al., 2011; Albert et al., 2014)), or other standard combinatorial properties (*e*.g., betweenness centrality, clustering coefficient, dense sub-graphs) that are commonly used in the computer science literature.
- ▷ "Close to hyperbolic topology" is *not* necessarily the same as "close to tree topology".

4.2.2 Geometric curvatures

In this section, we describe generic geometric curvatures of graphs by using correspondence with topological objects in higher dimension.

4.2.2.1 Some basic topological concepts

We first review some basic concepts from topology; see introductory textbooks such as (Henle, 1994; Gamelin and Greene, 1999) for further information. Although not necessary, the reader may find it useful to think of the underlying metric space as the r-dimensional real space $\mathbb{R}^r$ be for some integer $r > 1$.

- A subset $S \subseteq \mathbb{R}^r$ is *convex* if and only if for any $x, y \in S$, the *convex combination* of x and y is also in S.
- A set of $k+1$ points $x_0, \ldots, x_k \in \mathbb{R}^r$ are called *affinely independent* if and only if for all $\alpha_0, \ldots, \alpha_k \in \mathbb{R}$ $\sum_{j=0}^{k} \alpha_j x_j = 0$ and $\sum_{j=0}^{k} \alpha_j = 0$ implies $\alpha_0 = \cdots = \alpha_k = 0$.
- The *k-simplex* generated by a set of $k+1$ affinely independent points $x_0, \ldots, x_k \in \mathbb{R}^r$ is the subset $\mathcal{S}\big(x_0, \ldots, x_k\big)$ of $\mathbb{R}^r$ generated by *all* convex combinations of $x_0, \ldots, x_k$.
  - Each $(\ell+1)$-subset $\big\{x_{i_0}, \ldots, x_{i_\ell}\big\} \subseteq \big\{x_0, \ldots, x_k\big\}$ defines the ℓ-simplex $\mathcal{S}\big(x_{i_0}, \ldots, x_{i_\ell}\big)$ that is called a *face* of dimension ℓ (or a *ℓ-face*) of $\mathcal{S}\big(x_0, \ldots, x_k\big)$. A $(k-1)$-face, 1-face and 0-face is called a *facet*, an *edge* and a *node*, respectively.
- A (closed) *halfspace* is a set of points satisfying $\sum_{j=1}^{r} a_j x_j \leq b$ for some $a_1, \ldots, a_r, b \in \mathbb{R}$. The convex set obtained by a bounded non-empty intersection of a finite number of halfspaces is called a *convex polytope* (*convex polygon* in two dimensions).
  - If the intersection of a halfspace and a convex polytope is a subset of the halfspace then it is called a *face* of the polytope. Of particular interests are faces of dimensions $r-1$, 1 and 0, which are called *facets*, *edges* and *nodes* of the polytope, respectively.
- A *simplicial complex* (or just a complex) is a topological space constructed by the union of simplexes via topological associations.

4.2.2.2 Geometric curvature definitions

Informally, a *complex* is "glued" from nodes, edges and polygons via topological identification. We first define k-complex-based *Forman's combinatorial Ricci curvature* for elementary components

(such as nodes, edges, triangles and higher-order cliques) as described in (Forman, 2003; Weber et al., 2016a; Weber et al., 2016b), and then obtain a scalar curvature that takes an appropriate linear combination of these values (via Gauβ-Bonnet type theorems (Bloch, 2014)) that correspond to the so-called *Euler characteristic* of the complex that is topologically associated with the given graph. In this paper, we consider such Euler characteristics of a graph to define geometric curvature.

To begin the topological association, we (topologically) associate a q-simplex with a $(q+1)$-clique $\mathcal{K}_{q+1}$; for example, 0-simplexes, 1-simplexes, 2-simplexes and 3-simplexes are associated with nodes, edges, 3-cycles (triangles) and 4-cliques, respectively. Next, we would also need the concept of an "order" of a simplex for more non-trivial topological association. Consider a p-face f^p of a q-simplex. An order d association of such a face, which we will denote by the notation f^p_d with the additional subscript d, is associated with a sub-graph of *at most* d nodes that is obtained by starting with $\mathcal{K}_{p+1}$ and then *optionally* replacing each edge by a path between the two nodes. For example,

- f^0_d is a node of G for all $d \geq 1$.
- f^1_2 is an edge, and f^1_d for $d > 2$ is a path having at most d nodes between two nodes adjacent in G.
- f^2_3 is a triangle (cycle of 3 nodes or a 3-cycle), and f^2_d for $d > 3$ is obtained from 3 nodes by connecting every pair of nodes by a path such that the total number of nodes in the sub-graph is at most d.

Naturally, the higher the values of p and q are, the more complex are the topological associations. Let $\mathcal{F}_d^k$ be the set of all f_d^k's in G that are topologically associated. With such associations via p-faces of order d, the Euler characteristics of the graph $G = (V, E)$ and consequently the curvature can defined as

$$\mathfrak{C}_d^p(G) \stackrel{\text{def}}{=} \sum_{k=0}^{p} (-1)^k \left| \mathcal{F}_d^k \right| \tag{4.1}$$

It is easy to see that both $\mathfrak{C}_d^0(G)$ and $\mathfrak{C}_d^1(G)$ are too simplistic to be of use in practice. Thus, we consider the next higher value of p in this paper, namely when $p = 2$. Letting $\mathcal{C}(G)$ denote the number of cycles of at most $d + 1$ nodes in G, we get the measure

$$\mathfrak{C}_d^2(G) = |V| - |E| + |\mathcal{C}(G)|$$

4.2.2.3 Are geometric curvatures a suitable measure for real-world networks ?

The usefulness of geometric curvatures for real-world networks was demonstrated in publications such as (Weber et al., 2016a; Weber et al., 2016b; Samal et al., 2018).

4.3 Formalizations of two anomaly detection problems on networks

In this section, we formalize two versions of the anomaly detection problem on networks. An underlying assumption on the behind these formulations is that the graph adds/deletes *edges* only while keeping the same set of nodes.

4.3.1 Extremal anomaly detection for static networks

The problems in this subsection are motivated by a desire to quantify the extremal sensitivity of static networks. The basic decision question is: "*is there a subset among a set of prescribed edges*

whose deletion may change the network curvature significantly?". This directly leads us to the following *decision* problem:

Problem name: Extremal Anomaly Detection Problem (Eadp$_{\mathfrak{C}}(G, \widetilde{E}, \gamma)$)

Input: • A curvature measure $\mathfrak{C} : \mathfrak{G} \to \mathbb{R}$

• A connected graph $G = (V, E)$, an edge subset $\widetilde{E} \subseteq E$ such that $G \setminus \widetilde{E}$ is connected and a real number $\gamma < \mathfrak{C}(G)$ (*resp.*, $\gamma > \mathfrak{C}(G)$)

Decision question: is there an edge subset $\widehat{E} \subseteq \widetilde{E}$ such that $\mathfrak{C}(G \setminus \widehat{E}) \leq \gamma$ (*resp.*, $\mathfrak{C}(G \setminus \widehat{E}) \geq \gamma$) ?

Optimization question: if the answer to the decision question is "yes" then minimize $|\widehat{E}|$

Notation: if the answer to the decision question is "yes" then the minimum possible value of $|\widehat{E}|$ is denoted by $\mathsf{OPT}_{\text{Eadp}_{\mathfrak{C}}}(G, \widetilde{E}, \gamma)$

The following comments regarding the above formulation should be noted:

▷ For the case $\gamma < \mathfrak{C}(G)$ (*resp.*, $\gamma > \mathfrak{C}(G)$) we allow $\mathfrak{C}(G \setminus \widetilde{E}) > \gamma$ (*resp.*, $\mathfrak{C}(G \setminus \widetilde{E}) < \gamma$), thus $\widehat{E} = \widetilde{E}$ need *not* be a feasible solution at all.

▷ The curvature function is only defined for connected graphs, thus we require $G \setminus \widetilde{E}$ to be connected.

▷ The edges in $E \setminus \widetilde{E}$ can be thought of as "critical" edges needed for the functionality of the network. For example, in the context of inference of minimal biological networks from indirect experimental evidences (Albert et al., 2007; Albert et al., 2008), the set of critical edges represent direct biochemical interactions with concrete evidence.

4.3.2 Targeted anomaly detection for dynamic networks

These problems are primarily motivated by change-point detections between two successive discrete time steps in dynamic networks (Aminikhanghahi and Cook, 2017; Kawahara and Sugiyama, 2009), but they can also be applied to static networks when a subset of the final desired network *is* known. Fig. 5 illustrates targeted anomaly detection for a dynamic biological network.

Problem name: Targeted Anomaly Detection Problem ($\text{Tadp}_{\mathfrak{C}}(G_1, G_2)$)

Input: • Two connected graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with $E_2 \subset E_1$

• A curvature measure $\mathfrak{C} : \mathfrak{G} \to \mathbb{R}$

Valid solution: a subset of edges $E_3 \subseteq E_1 \setminus E_2$ such that $\mathfrak{C}(G_1 \setminus E_3) = \mathfrak{C}(G_2)$.

Objective: *minimize* $|E_3|$.

Notation: the minimum value of $|E_3|$ is denoted by $\mathsf{OPT}_{\text{Tadp}_{\mathfrak{C}}}(G_1, G_2)$

4.4 Computational complexity of extremal anomaly detection problems

4.4.1 Geometric curvatures: exact and approximation algorithms for $\text{Eadp}_{\mathfrak{C}^2_d}$

The following theorem is stated without proof as it also appears in (Yahyanejad, 2019).

Theorem 10.

(*a*) *The following statements hold for* $\text{E}adp_{\mathfrak{C}^2_d}(G, \widetilde{E}, \gamma)$ *when* $\gamma > \mathfrak{C}^2_d(G)$*:*

(*a*1) *We can decide in polynomial time the answer to the decision question (i.e., if there exists any feasible solution* $\widehat{E}$ *or not).*

(*a*2) *If a feasible solution exists then the following results hold:*

(*a*2-1) *Computing* $\mathsf{OPT}_{\text{E}adp_{\mathfrak{C}^2_d}}(G, \widetilde{E}, \gamma)$ *is* NP*-hard for all* d *that are multiple of* 3*.*

(a2-2) *If γ is sufficient larger than $\mathfrak{C}_d^2(G)$ then we can design an approximation algorithm that approximates both the cardinality of the minimal set of edges for deletion and the absolute difference between the two curvature values. More precisely, if $\gamma \geq \mathfrak{C}_d^2(G) + \left(\frac{1}{2} + \varepsilon\right)(2|\widetilde{E}| - |E|)$ for some $\varepsilon > 0$, then we can find in polynomial time a subset of edges $E_1 \subseteq \widetilde{E}$ such that*

$$|E_1| \leq 2\,\mathsf{OPT}_{\mathrm{E}adp_{\mathfrak{C}_d^2}}(G, \widetilde{E}, \gamma) \text{ and } \frac{\mathfrak{C}_d^2(G \setminus E_1) - \mathfrak{C}_d^2(G)}{\gamma - \mathfrak{C}_d^2(G)} \geq \frac{4\varepsilon}{1 + 2\varepsilon}$$

(b) *The following statements hold for $\mathrm{E}adp_{\mathfrak{C}_d^2}(G, \widetilde{E}, \gamma)$ when $\gamma < \mathfrak{C}_d^2(G)$:*

(b1) *We can decide in polynomial time the answer to the decision question (i.e., if there exists any feasible solution $\widehat{E}$ or not).*

(b2) *If a feasible solution exists and γ is not too far below $\mathfrak{C}_d^2(G)$ then we can design an approximation algorithm that approximates both the cardinality of the minimal set of edges for deletion and the absolute difference between the two curvature values. More precisely, letting Δ denote the number of cycles of G of at most $d+1$ nodes that contain at least one edge from $\widetilde{E}$, if $\gamma \geq \mathfrak{C}_d^2(G) - \frac{\Delta}{1+\varepsilon}$ for some $\varepsilon > 0$ then we can find in polynomial time a subset of edges $E_1 \subseteq \widetilde{E}$ such that*

$$|E_1| \leq 2\,\mathsf{OPT}_{\mathrm{E}adp_{\mathfrak{C}_d^2}}(G, \widetilde{E}, \gamma) \text{ and } \frac{\mathfrak{C}_d^2(G \setminus E_1) - \mathfrak{C}_d^2(G)}{\gamma - \mathfrak{C}_d^2(G)} \leq 1 - \varepsilon$$

(b3) *If $\gamma < \mathfrak{C}_d^2(G)$ then, even if $\gamma = \mathfrak{C}_d^2(G \setminus \widetilde{E})$ (i.e., a trivial feasible solution exists), computing $\mathsf{OPT}_{\mathrm{E}adp_{\mathfrak{C}_d^2}}(G, \widetilde{E}, \gamma)$ is at least as hard as computing $\mathrm{T}adp_{\mathfrak{C}_d^2}(G_1, G_2)$ and therefore **all** the hardness results for $\mathrm{T}adp_{\mathfrak{C}_d^2}(G_1, G_2)$ in Theorem 12 also apply to $\mathsf{OPT}_{\mathrm{E}adp_{\mathfrak{C}_d^2}}(G, \widetilde{E}, \gamma)$.*

4.4.2 Gromov-hyperbolic curvature: computational complexity of Eadp$_{\mathfrak{C}_{\text{Gromov}}}$

Theorem 11. *The following statements hold for* E*adp*$_{\mathfrak{C}_{\text{Gromov}}}(G, \widetilde{E}, \gamma)$ *when* $\gamma > \mathfrak{C}_{\text{Gromov}}(G)$*:*

(a) *Deciding if there exists a feasible solution is* NP*-hard.*

(b) *Even if a trivial feasible solution exists, it is* NP*-hard to design a polynomial-time algorithm to approximate* $\mathsf{OPT}_{\text{E}adp_{\mathfrak{C}_{\text{Gromov}}}}(G, \widetilde{E}, \gamma)$ *within a factor of* $c\,n$ *for some constants* $c > 0$*, where* n *is the number of nodes in* G *and* m *is the number of edges in* G*.*

4.4.2.1 Proof techniques and relevant comments regarding Theorem 11

From a high level point of view, Theorem 11 is proved by suitably modifying the reductions used in the proof of Theorem 13.

4.4.2.2 Proof of Theorem 11

To prove ***(a)*** we will use a simpler version of the proof of Theorem 13 reusing the same notations. Our graph G will be the same as the graph G_1 in that proof, except that we do *not* add the complete graph $K_{|V''|}$ on the nodes $w_0, w_1, \ldots, w_{|V''|-1}$ and consequently we also do *not* have the edge $\{u, w_0\}$. We set $\widetilde{E} = E'$ and $\gamma = \frac{n}{2} + 1$. The proof of Theorem 13 shows that $\mathfrak{C}_{\text{Gromov}}(G) < \gamma$, $\mathfrak{C}_{\text{Gromov}}(G \setminus E') \leq \gamma$ for any subset of edges $E' \subseteq \widetilde{E}$, and $\mathfrak{C}_{\text{Gromov}}(G \setminus E') = \gamma$ for a subset of edges $\emptyset \subset E' \subset \widetilde{E}$ if and only if the given cubic graph has a Hamiltonian path between the two specified nodes, thereby showing NP-hardness of the feasibility problem.

To prove ***(b)*** the same construction in the proof of Theorem 13 works: G is the same as the graph G_1 in that proof, $\gamma = \frac{n}{2} + 1$, $\widetilde{E}$ is the set of edges whose deletion produced G_2 from G_1, and the trivial

feasible solution is G_2. Note that the proof of Theorem 13 shows $\mathfrak{C}_{\mathrm{Gromov}}(G) < \gamma$, $\mathfrak{C}_{\mathrm{Gromov}}(G \backslash E') \leq \gamma$ for any subset of edges $\emptyset \subset E' \subseteq \widetilde{E}$ and $\mathfrak{C}_{\mathrm{Gromov}}(G_2) = \gamma$.

4.5 Computational complexity of targeted anomaly detection problems

4.5.1 Geometric curvatures: computational hardness of $\mathbf{Tadp}_{\mathfrak{C}^2_d}(G_1, G_2)$

For two functions $f(n)$ and $g(n)$ of n, we say $f(n) = O^*(g(n))$ if $f(n) = O(g(n)\, n^c)$ for some positive constant c. In the sequel we will use the following two complexity-theoretic assumptions: the *unique games conjecture* (Ugc) (Khot, 2002; Trevisan, 2012), and the *exponential time hypothesis* (Eth) (Impagliazzo and Paturi, 2001; Impagliazzo et al., 2001; Woeginger, 2003).

Theorem 12.

(a) *Computing* $\mathsf{OPT}_{\mathrm{T}adp_{\mathfrak{C}^2_3}}(G_1, G_2)$ *is* NP*-hard.*

(b) *There are no algorithms of the following type for* $\mathrm{T}adp_{\mathfrak{C}^2_d}(G_1, G_2)$ *for* $4 \leq d \leq o(n)$ *when* G_1 *and* G_2 *are* n*-node graphs:*

- ***(b1)*** *a polynomial time* $(2-\varepsilon)$*-approximation algorithm for any constant* $\varepsilon > 0$ *assuming* Ugc *is true,*
- ***(b2)*** *a polynomial time* $(10\sqrt{5} - 21 - \varepsilon) \approx 1.36$*-approximation algorithm for any constant* $\varepsilon > 0$ *assuming* $P \neq$ NP*,*
- ***(b3)*** *a* $O^*\left(2^{o(n)}\right)$*-time exact computation algorithm assuming* Eth *is true, and*
- ***(b4)*** *a* $O^*\left(n^{o(\kappa)}\right)$*-time exact computation algorithm if* $\mathsf{OPT}_{\mathrm{T}adp_{\mathfrak{C}^2_3}}(G_1, G_2) \leq \kappa$ *assuming* Eth *is true.*

4.5.1.1 Proof techniques and relevant comments regarding Theorem 12

- To prove **(*a*)**, we prove the results by reducing the triangle deletion problem (Tdp) to that of solving $\text{Tadp}_{\mathfrak{C}_3^2}$. Tdp was shown to be NP-hard by Yannakakis in (Yannakakis, 1978).
- To prove **(*b*)**, we provide suitable approximation-preserving reductions from Mnc.
- **(on proofs in (*b*3) and (*b*4))** For these proofs, the idea is to start with an instance of 3-Sat, use "sparsification lemma" in (Impagliazzo et al., 2001) to generate a family of Boolean formulae, reduce each of these formula to Mnc, and finally reduce each such instance of Mnc to a corresponding instance of $\text{Tadp}_{\mathfrak{C}_d^2}$.

4.5.1.2 Proof of Theorem 12

The goal of the *minimum node cover* problem (Mnc) for a graph G is to select a subset of nodes of *minimum* cardinality such that at least one end-point of *every* edge has been selected; let $\mathsf{OPT}_{\text{Mnc}}(G)$ denote the cardinality of the subset of nodes that is an optimal solution of Mnc. The (standard) Boolean satisfiability problem is denoted by Sat, and its restricted case when every clause has exactly k literals will be denoted by k-Sat (Garey and Johnson, 1979). Consider Sat or k-Sat and let Φ be an input instance (*i.e.*, a Boolean formula in conjunctive normal form) of it. The following inapproximability results are known for Mnc:

($\star_{\textbf{Mnc}}$) There exists a polynomial time algorithm that transforms a given instance Φ of Sat to an input instance graph $G = (V, E)$ of Mnc such that the following holds for any constant $0 < \varepsilon < \frac{1}{4}$, assuming Ugc to be true (Khot and Regev, 2008a):

$$\text{if } \Phi \text{ is satisfiable then} \quad \mathsf{OPT}_{\text{Mnc}}(G) \leq \left(\tfrac{1}{2} + \varepsilon\right)|V|$$

$$\text{if } \Phi \text{ is not satisfiable then} \quad \mathsf{OPT}_{\text{Mnc}}(G) \geq (1 - \varepsilon)\,|V|$$

($\star\star_{\textbf{Mnc}}$) There exists a polynomial time algorithm that transforms a given instance Φ of Sat to an input instance graph $G = (V, E)$ of Mnc such that the following holds for any constant $0 < \varepsilon < 16 - 8\sqrt{5}$ and for some $0 < \alpha < 2|V|$, assuming P$\neq$NP (Dinur and Safra, 2005a):

$$\text{if } \Phi \text{ is satisfiable then} \quad \mathsf{OPT}_{\text{Mnc}}(G) \leq \left(\tfrac{\sqrt{5}-1}{2} + \varepsilon\right)\alpha$$

$$\text{if } \Phi \text{ is not satisfiable then} \quad \mathsf{OPT}_{\text{Mnc}}(G) \geq \left(\tfrac{71-31\sqrt{5}}{2} - \varepsilon\right)\alpha$$

(note that $\left(\frac{71-31\sqrt{5}}{2}\right) \,/\, \left(\frac{\sqrt{5}-1}{2}\right) = 10\sqrt{5} - 21 \approx 1.36$).

($\star\star\star_{\textbf{Mnc}}$) There exists a polynomial time algorithm (*e*.g., see (Garey and Johnson, 1979, page 54)) that transforms a given instance Φ of 3-Sat of n variable and m clauses to to an input instance graph $G = (V, E)$ of Mnc with $|V| = 3n + 2m$ nodes and $|E| = n + m$ edges such that such that Φ is satisfiable if and only if $\mathsf{OPT}_{\text{Mnc}}(G) = n + 2m$.

Proofs of (*a*) We will prove the results by reducing the triangle deletion problem to that of computing $\text{Tadp}_{\mathfrak{C}_3^2}$. The *triangle deletion problem* (Tdp) can be stated as follows: *Given G find the minimum number*

of edges (which we will denote by $\mathsf{OPT}_{\mathsf{T}dp}(G)$*) to be deleted from* G *to make it triangle-free*. Tdp was shown to be NP-hard by Yannakakis in (Yannakakis, 1978).

Consider an instance $G = (V, E)$ of Tdp where $V = \{u_1, \ldots, u_n\}$ and $E = \{e_1, \ldots, e_m\}$. We create an instance $G_1 = (V', E_1)$ and $G_2 = (V', E_2)$ (with $\emptyset \subset E_2 \subset E_1$) of $\text{Tadp}_{\mathfrak{C}_3^2}$ in the following manner:

- ▷ For each $u_i \in V$, we create a node $v_i \in V'$. There are n such nodes in V'.
- ▷ If $\{u_i, u_j\} \in E$, then we add the edge $\{v_i, v_j\}$ to E_1. We call these edges as "original" edges. Let E_d be the set of all original edges; note that $|E_d| = m$.
- ▷ To ensure that G_2 is a connected graph, we add two new nodes w_i^1, w_i^2 in V' corresponding to each node $v_i \in V'$ for $i = 1, 2, \ldots, n-1$, and add three new edges $\{v_i, w_i^1\}$, $\{w_i^1, w_i^2\}$ and $\{w_i^2, v_{i+1}\}$ in E_1. This step adds $2n - 2$ new nodes and $3n - 3$ new edges to V_1 and E_1, respectively. We call the new edges added in this step as "connectivity" edges.
- ▷ For each $\{u_i, u_j\}) \in E$, we create a new node $v_{i,j}$ in V' and add two new edges $\{u_i, v_{ij}\}$ and $\{v_{ij}, u_j\}$ to E_1. This step creates a new triangle corresponding to each original edge. We call the new edges added in this step as "triangle-creation" edges. This step adds m new nodes and $2m$ new edges to V_1 and E_1, respectively, and exactly m new triangles.

Define $E_2 = E_1 \setminus E_d$. Thus, we have $|V'| = 3n + m - 2$, $|E_1| = 3n + 3m - 3$, $|E_2| = 3n + 2m - 3$, and G_2 contains *no* triangles. Let Δ is the number of triangles in G_1 created using only original edges (the "original triangles"); note that Δ is also equal to the number of triangles in G. Then,

$\mathfrak{C}_3^2(G_1) = |V'| - (3m + 3n - 3) + (\Delta + m)$ and $\mathfrak{C}_3^2(G_2) = |V'| - (2m + 3n - 3)$. The following lemma completes our NP-hardness proof.

Lemma 9. $\mathsf{OPT}_{\mathrm{T}dp}(G) = \mathsf{OPT}_{\mathrm{T}adp_{\mathfrak{C}_3^2}}(G_1, G_2)$.

Proof. Proof of $\mathsf{OPT}_{\mathrm{Tdp}}(G) \geq \mathsf{OPT}_{\mathrm{Tadp}_{\mathfrak{C}_3^2}}(G_1, G_2)$.

Let $E_{\mathrm{opt}} \subset E$ be an optimum solution of Tdp on G, $E'_{\mathrm{opt}} = \{\{v_i, v_j\} \,|\, \{u_i, u_j\} \in E\} \subseteq E_d$, and consider the graph $G_3 = (V', E_1 \setminus E'_{\mathrm{opt}})$. Note that G_3 has *no* original triangles and has exactly $m - |E'_{\mathrm{opt}}|$ triangles involving triangle-creation edges, and thus

$$\mathfrak{C}_3^2(G_3) = |V'| - (3n + 3m - 3 - |E'_{\mathrm{opt}}|) + (m - |E'_{\mathrm{opt}}|) = \mathfrak{C}_3^2(G_2)$$

and therefore $\mathsf{OPT}_{\mathrm{Tadp}_{\mathfrak{C}_3^2}}(G_1, G_2) \leq |E'_{\mathrm{opt}}| = |E_{\mathrm{opt}}| = \mathsf{OPT}_{\mathrm{Tdp}}(G)$.

Proof of $\mathsf{OPT}_{\mathrm{Tdp}}(G) \leq \mathsf{OPT}_{\mathrm{Tadp}_{\mathfrak{C}_3^2}}(G_1, G_2)$.

Suppose that $E'_{\mathrm{opt}} \subset E_d$ is an optimum solution of q edges of $\mathrm{Tadp}_{\mathfrak{C}_3^2}$ on G_1 and G_2, let $G_3 = (V', E_1 \setminus E'_{\mathrm{opt}})$ be the graph obtained from G_1 by removing the edges in E'_{opt}, and let $E' = \big\{\{u_i, u_j\} \,|\, \{v_i, v_j\} \in E'_{\mathrm{opt}}\big\} \subseteq E$. Let $q = |E'_{\mathrm{opt}}|$, $e'_1, e'_2, \ldots, e'_q$ be an arbitrary ordering of the edges in E'_{opt} and δ'_i (for $i = 1, 2, \ldots, q$) is the number of triangles in G_1 that contains the edge e'_i but *none* of the edges $e'_1, \ldots, e'_{i-1}$. Note that, for each i, exactly $\delta'_i - 1$ triangles out of the δ'_i triangles are

original triangles. Let $\Delta' \leq \Delta$ be the number of original triangles removed by removing the edges in E'_{opt}; thus, $\Delta' = \sum_{i=1}^{q} (\delta'_i - 1)$. Simple calculations now show that

$$\begin{aligned}\mathfrak{C}_3^2(G_3) &= |V'| - (3n + 3m - 3 - |E'_{\text{opt}}|) + \left(\Delta + m - \sum_{i=1}^{q} \delta_i\right) \\ &= |V'| - (3n + 3m - 3 - |E'_{\text{opt}}|) + \left(\Delta + m - q - \sum_{i=1}^{q} (\delta_i - 1)\right) \\ &= |V'| - (3n + 3m - 3 - |E'_{\text{opt}}|) + \left(\Delta + m - |E'_{\text{opt}}| - \Delta'\right) = |V'| - (3n + 2m - 3) + (\Delta - \Delta')\end{aligned}$$

Consequently, $\mathfrak{C}_3^2(G_3) = \mathfrak{C}_3^2(G_2)$ implies $\Delta' = \Delta$ and E' is a valid solution of Tdp on G. This implies $\mathsf{OPT}_{\text{Tdp}}(G) \leq |E'| = |E'_{\text{opt}}| = \mathsf{OPT}_{\text{Tadp}_{\mathfrak{C}_3^2}}(G_1, G_2)$. □

Proofs of (*b*1) and (*b*2)

Consider an instance graph $G = (V, E)$ of Mnc with n nodes and m edges where $V = \{v_1, v_2, \ldots, v_n\}$ and $E = \{e_1, e_2, \ldots, e_m\}$. Let $\emptyset \subset V_{\text{Mnc}} \subset V$ be an optimal solution of $\mathsf{OPT}_{\text{Mnc}}(G) = |V_{\text{Mnc}}|$ nodes for this instance of Mnc. We then create an instance $G_1 = (V', E_1)$ and $G_2 = (V', E_2)$ (with $\emptyset \subset E_2 \subset E_1$) of $\text{Tadp}_{\mathfrak{C}_d^2}$ for a given $d \geq 4$ in the following manner:

- For each $v_i \in V$, we create d new nodes $\{v_i^1, v_i^2, \ldots, v_i^d\}$ in V', and a d-cycle containing the edges $\{v_i^1, v_i^2\}$,$\{v_i^2, v_i^3\}$,$\ldots$,$\left\{v_i^{d-1}, v_i^d\right\}$, $\{v_i^d, v_i^1\}$ in E_1. We call the cycles generated in this step as the "node cycles". This creates a total of dn nodes in V' and dn edges in E_1.
- For each edge $\{v_i, v_j\} \in E$, we do the following:
  - Create $d - 4$ new nodes $u_{i,j,1}^1$,$u_{i,j,2}^1$,$\ldots$,$u_{i,j,\lceil \frac{d-4}{2} \rceil}^1$ and $u_{i,j,1}^2$,$u_{i,j,2}^2$,$\ldots$,$u_{i,j,\lfloor \frac{d-4}{2} \rfloor}^2$ in V'.

– Add $\lceil \frac{d-2}{2} \rceil$ new edges $\left\{v_i^1, u_{i,j,1}^1\right\}, \left\{u_{i,j,1}^1, u_{i,j,2}^1\right\}, \ldots, \left\{u_{i,j,\lceil \frac{d-4}{2} \rceil - 1}^1, u_{i,j,\lceil \frac{d-4}{2} \rceil}^1\right\}, \left\{u_{i,j,\lceil \frac{d-4}{2} \rceil}^1, v_j^1\right\}$ and $\lfloor \frac{d-2}{2} \rfloor$ new edges $\left\{v_i^2, u_{i,j,1}^2\right\}, \left\{u_{i,j,1}^2, u_{i,j,2}^2\right\}, \ldots, \left\{u_{i,j,\lfloor \frac{d-4}{2} \rfloor - 1}^2, u_{i,j,\lfloor \frac{d-4}{2} \rfloor}^2\right\}, \left\{u_{i,j,\lfloor \frac{d-4}{2} \rfloor}^2, v_j^2\right\}$ in E_1. Note that these edges create a d-cycle involving the two edges $\left\{v_i^1, v_i^2\right\}$ and $\left\{v_j^1, v_j^2\right\}$; we refer to this cycle as an "edge cycle".

These steps create a total of $(d-4)m$ additional nodes in V' and $(d-2)m$ additional edges in E_1.

- Let $E_2 = E_1 \setminus \left\{\left\{v_i^1, v_i^2\right\} \mid 1 \leq i \leq n\right\}$.

Thus, $|V'| = dn + (d-4)m$, $|E_1| = dn + (d-2)m$ and $|E_2| = (d-1)n + (d-2)m$. To verify that the reduction is possible for any d in the range of values as claimed in the theorem, note that

$$d \leq o(|V'|) \equiv d/|V'| \leq o(1) \Leftarrow n^{-1} \leq o(1)$$

and the last inequality is trivially true. By **($\star_{\textbf{Mnc}}$)** and **($\star\star_{\textbf{Mnc}}$)**, the proof is complete once we prove the following lemma.

Lemma 10. $\mathsf{OPT}_{\text{Mnc}}(G) = \mathsf{OPT}_{\text{T}adp_{\mathfrak{C}_d^2}}(G_1, G_2)$.

Proof. Let $E_d = E_1 \setminus E_2$. Let f be the total number of cycles of at most d edges in G_1; thus

$$\mathfrak{C}_d^2(G_1) = |V'| - |E_1| + f = -2m + f$$

Note that any cycle of at most d edges containing an edge from E_d *must* be either a node cycle or an edge cycle since a cycle containing an edge from E_d that is *neither* a node cycle *nor* an edge cycle has a

number of edges that is at least $2 + 2 \times \left\lfloor \frac{d-2}{2} \right\rfloor + \left\lceil \frac{d-2}{2} \right\rceil = d + \left\lfloor \frac{d-2}{2} \right\rfloor > d$ since $d \geq 4$. Since removing all the edges in E_d removes *every* node and *every* edge cycle,

$$\mathfrak{C}_d^2(G_2) = |V'| - |E_2| + (f - n - m) = \big(|V'| - |E_1| + f\big) - m = \mathfrak{C}_d^2(G_1) - m$$

Given an optimal solution $V_{\text{Mnc}} \subset V$ of Mnc on G of $\mathsf{OPT}_{\text{Mnc}}(G)$ nodes, consider the graph $G_3 = (V', E_3)$ where $E_3 = E_1 \setminus E'_d$ and $E'_d = \left\{ \left\{ v_i^1, v_i^2 \right\} \mid v_i \in V_{\text{Mnc}} \right\} \subseteq E_d$. Since every edge of G is incident on one or more nodes in V_{Mnc}, *every* edge cycle and *exactly* $|E'_d| = \mathsf{OPT}_{\text{Mnc}}(G)$ node cycles of G_1 are removed in G_3, and thus

$$\mathfrak{C}_d^2(G_3) = |V'| - (|E_1| - \mathsf{OPT}_{\text{Mnc}}(G)) + (f - \mathsf{OPT}_{\text{Mnc}}(G) - m) = \mathfrak{C}_d^2(G_1) - m = \mathfrak{C}_d^2(G_2)$$

This shows that $\mathsf{OPT}_{\text{Tadp}_{\mathfrak{C}_d^2}}(G_1, G_2) \leq \mathsf{OPT}_{\text{Mnc}}(G)$. Conversely, consider an optimal solution $E'_d \subseteq E_d$ of $\text{Tadp}_{\mathfrak{C}_d^2}$ for G_1 and G_2, and let $G_3 = (V', E_3)$ where $E_3 = E_1 \setminus E'_d$. Note that *exactly* $|E'_d| = \mathsf{OPT}_{\text{Tadp}_{\mathfrak{C}_d^2}}(G_1, G_2)$ node cycles of G_1 are removed in G_3. Let m' be the number of edge cycles of G_1 removed in G_3. Then,

$$\mathfrak{C}_d^2(G_3) = |V'| - (|E_1| - |E'_d|) + (f - |E'_d| - m') = \mathfrak{C}_d^2(G_1) - m'$$

and consequently m' must be equal to m to satisfy the constraint $\mathfrak{C}_d^2(G_3) = \mathfrak{C}_d^2(G_1) - m$, which implies that G_3 contains *no* edge cycles. This implies that, for every edge cycle involving the two edges $\left\{ v_i^1, v_i^2 \right\}$ and $\left\{ v_j^1, v_j^2 \right\}$ in G_1, at least one of these two edges must be in E''_d, which in turn implies that the

set of nodes $V'' = \left\{v_i \mid \left\{v_i^1, v_i^2\right\} \in E'_d\right\}$ in G contains at least one of the nodes v_i or v_j for every edge $\{v_i, v_j\} \in E$. Thus, V'' is a valid solution of Mnc on G and $\mathsf{OPT}_{\text{Mnc}}(G) \leq |V''| = |E''_d| = \mathsf{OPT}_{\text{Tadp}_{\mathfrak{C}_d^2}}(G_1, G_2)$. □

Proof of (*b*3)

We describe the proof for $d = 4$ only; the proof for $d > 4$ is very similar. Suppose, for the sake of contradiction, that one *can* in fact compute $\mathsf{OPT}_{\text{Tadp}_{\mathfrak{C}_4^2}}(G_1, G_2)$ in $O^*\left(2^{o(n)}\right)$ time where each of G_1 and G_2 has n nodes. We start with an instance Φ of 3-Sat having n variables and m clauses. The "sparsification lemma" in (Impagliazzo et al., 2001) proves the following result:

> *for every constant $\varepsilon > 0$, there is a constant $c > 0$ such that there exists a $O\left(2^{\varepsilon n}\right)$-time algorithm that produces from Φ a set of t instances $\Phi_1, \ldots, \Phi_t$ of* 3-S*at on these n variables with the following properties:*
>
> - $t \leq 2^{\varepsilon n}$,
> - *each Φ_j is an instance of* 3-S*at with $n_j \leq n$ variables and $m_j \leq cn$ clauses, and*
> - *Φ is satisfiable if and only if at least one of $\Phi_1, \ldots, \Phi_t$ is satisfiable.*

For each such above-produced 3-Sat instance Φ_j, we now use the reduction mentioned in ($\star\star\star_{\text{Mnc}}$) to produce an instance $G_j = (V_j, E_j)$ of Mnc of $|V_j| = 3n_j + 2m_j \leq (3 + 2c)\, n$ nodes and $|E_j| = n_j + m_j \leq (1+c)\, n$ edges such that Φ_j is satisfiable if and only if $\mathsf{OPT}_{\text{Mnc}}(G_j) = n_j + 2m_j$. Now, using the reduction as described in the proof of parts **(*b*1)** and **(*b*2)** of this theorem and Lemma 10 thereof, we obtain an instance $G_{1,j} = (V'_j, E_{1,j})$ and $G_{2,j} = (V'_j, E_{2,j})$ of $\text{Tadp}_{\mathfrak{C}_3^2}$ such that $|V'_j| = 4|V_j| < (12 + 8c)n$. By

assumption, we can compute $\mathsf{OPT}_{\mathrm{Tadp}_{\mathfrak{C}_3^2}}(G_{1,j}, G_{2,j})$ in $O^*\big(2^{o(n)}\big)$, and consequently $\mathsf{OPT}_{\mathrm{Mnc}}(G_j)$ in $O^*\big(2^{o(n)}\big)$ time, which in turn leads us to decide in $O^*\big(2^{o(n)}\big)$ time if Φ_j is satisfiable for every j. Since $t \leq 2^{\varepsilon n}$ for every constant $\varepsilon > 0$, this provides a $O^*\big(2^{o(n)}\big)$-time algorithm for 3-Sat, contradicting Eth.

Proof of (*b*4)

The proof is very similar to that in **(b3)** except that now we start with the following lower bound result on parameterized complexity (*e*.g., see (Cygan et al., 2015, Theorem 14.21)):

> *assuming* Eth *to be true, if* $\mathsf{OPT}_{\mathrm{Mnc}}(G) \leq k$ *then there is* no $O^*\big(n^{o(k)}\big)$*-time algorithm for exactly computing* $\mathsf{OPT}_{\mathrm{Mnc}}(G)$.

4.5.2 Gromov-hyperbolic curvature: computational hardness of $\mathbf{Tadp}_{\mathfrak{C}_{\mathrm{Gromov}}}$

Theorem 13. *It is* NP*-hard to design a polynomial-time algorithm to approximate* $\mathrm{T}adp_{\mathfrak{C}_{\mathrm{Gromov}}}(G_1, G_2)$ *within a factor of* $c\,n$ *for some constant* $c > 0$*, where* n *is the number of nodes in* G_1 *or* G_2 *and* m *is the number of edges in* G_1.

4.5.2.1 Proof techniques and relevant comments regarding Theorem 13

- The reduction is from the *Hamiltonian path problem for cubic graphs* (Cubic-Hp), and shown schematically in Fig. 6. On a high level, the idea is to amplify the different between Hamiltonian and non-Hamiltonian paths to a large size difference of "geodesic" triangles (*cf*. Definition 12) such that application of known results such as (Rodríguez and Tourís, 2004, Lemma 2.1) can lead to a large difference of the corresponding Gromov-hyperbolicity values. To get the maximum possible amplification (maximum gap in lower bound) we need to make very careful and precise arguments

regarding the Gromov-hyperbolicities of classes of graphs. The reader should note that Gromov-hyperbolicity value is not necessarily related to the circumference of a graph, and thus the reduction cannot rely simply on presence or absence of long paths or long cycles in the constructed graph.

- The inapproximability reduction necessarily requires some nodes with large (close to linear) degrees even though with start with Cubic-Hp in which every node has degree exactly 3. We conjecture that our large inapproximability bounds do *not* hold when the given graphs have nodes of bounded degree, but have been unable to prove it so far.

4.5.2.2 Proof of Theorem 13

We will prove our inapproximability result via a reduction from the *Hamiltonian path* problem for cubic graphs (Cubic-Hp) which is defined as follows: "given a cubic (*i.e.*, a 3-regular) graph $G = (V, E)$ and two specified nodes $u, v \in V$, does G contain a Hamiltonian path between u and v, *i.e.*, a path between u and v that visits every node of G exactly once"? Cubic-Hp is known to be NP-complete (Garey et al., 1976).

Consider an instance $G = (V, E)$ and $v_1, v_n \in V$ of Cubic-Hp of n nodes and $m = 3n/2$ edges where $V = \{v_1, v_2, \ldots, v_n\}$, $E = \{e_1, e_2, \ldots, e_m\}$ and the goal is to determine if there is a Hamiltonian path between v_1 and v_n (see Fig. 6 (*a*)). We first introduce three new nodes v_0, v_{n+1} and v_{n+2}, and connect them to the nodes in G by adding three new edges $\{v_0, v_1\}$, $\{v_n, v_{n+1}\}$ and $\{v_{n+1}, v_{n+2}\}$, resulting in the graph $G' = (V', E')$ (see Fig. 6 (*b*)). It is then trivial to observe the following:

- G has a Hamiltonian path between v_1 and v_n if and only if G' has a Hamiltonian path between v_0 and v_{n+2}.

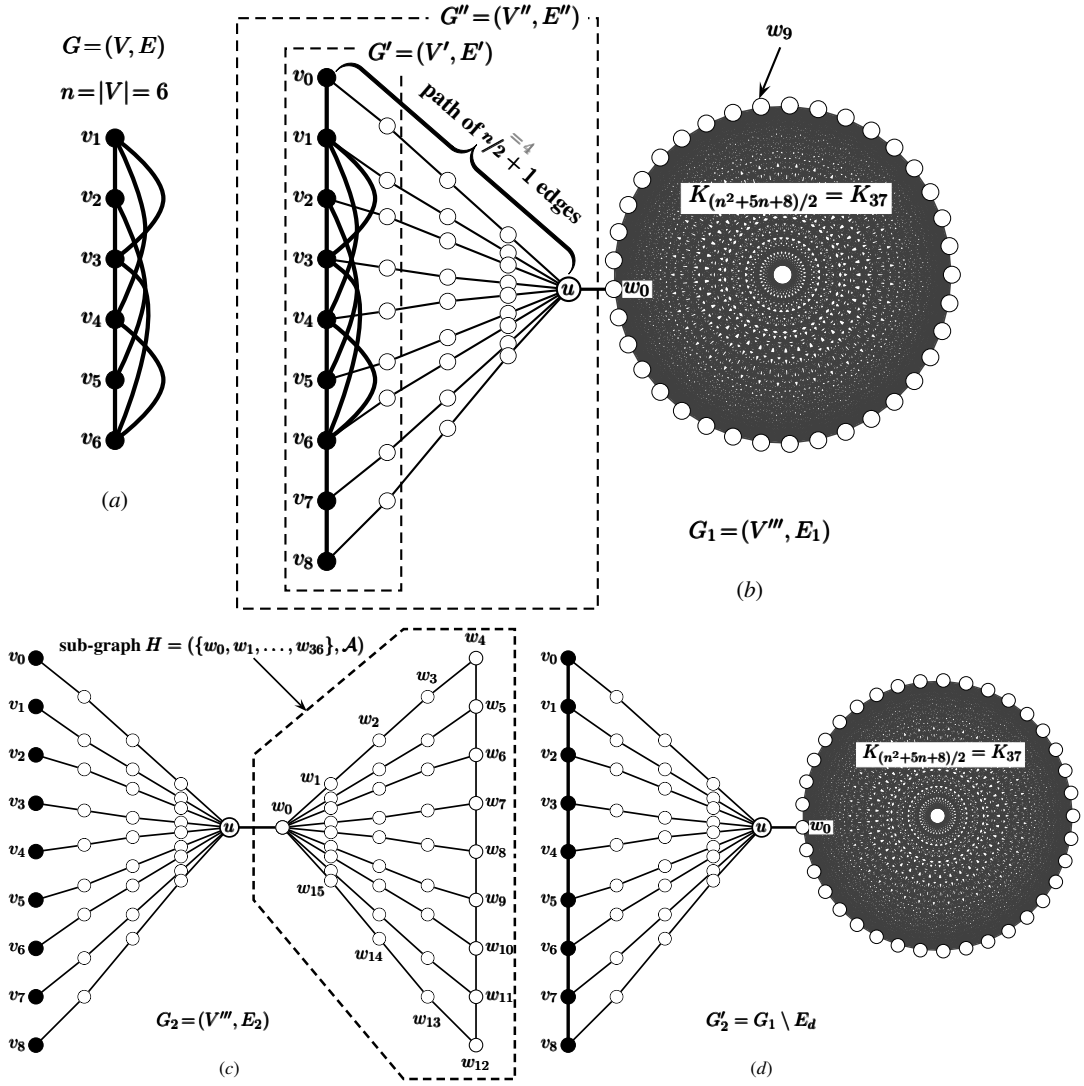


Figure 6: Illustration of the reduction in Theorem 13. (a) The input graph $G = (V, E)$ for the Hamiltonian path problem for cubic graphs (Cubic-Hp). (b) and (c) The graphs $G_1 = (V'', E_1)$ and $G_2 = (V'', E_2)$ for the generated instance of $\text{Tadp}_{\mathfrak{C}_{\text{Gromov}}}(G_1, G_2)$. The graph $G' = (V', E')$ obtained from the given graph G' by adding three extra nodes and three extra edges. (d) An optimal solution G'_2 for $\text{Tadp}_{\mathfrak{C}_{\text{Gromov}}}(G_1, G_2)$ if G contains a Hamiltonian path between v_1 and v_n.

- If G' does have a Hamiltonian path then such a path must be between the two nodes v_0 and v_{n+2}.

Note that $|V'| = n + 3$ and $|E'| = (3n/2) + 3$. We next create the graph $G'' = (V'', E'')$ from G' in the following manner (see Fig. 6 (*b*)):

- We add a set of $1+(n^2+3n)/2$ new nodes $u, v_{0,1}, \ldots, v_{0,n/2}, v_{1,1}, \ldots, v_{1,n/2}, \ldots, v_{n+2,1}, \ldots, v_{n+2,n/2}$. For notational convenience, we set $u \stackrel{\text{def}}{=} v_{i,0}$ for all $i \in \{0, 1, \ldots, n+2\}$ and $v_j \stackrel{\text{def}}{=} v_{j,(n/2)+1}$ for all $j \in \{0, 1, \ldots, n+2\}$.
- We add a set of $n + 3$ disjoint paths (each of length $\frac{n}{2} + 1$) $\mathcal{P}_0, \mathcal{P}_1, \ldots, \mathcal{P}_{n+2}$ where $\mathcal{P}_j \stackrel{\text{def}}{=} v_{j,0} \leftrightarrow v_{j,1} \leftrightarrow v_{j,2} \leftrightarrow \cdots \leftrightarrow v_{j,\frac{n}{2}+1}$.

Note that $|V''| = n+4+\frac{n^2+3n}{2} = \frac{n^2+5n}{2}+4$ and $|E''| = \frac{3n}{2}+3+(n+3)\left(\frac{n}{2}+1\right) = \frac{n^2}{2}+4n+6$. We now create an instance $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ (with $\emptyset \subset E_2 \subset E_1$) of $\text{Tadp}_{\mathfrak{C}_{\text{Gromov}}}(G_1, G_2)$ from G'' in the following manner (see Fig. 6 (*b*)–(*c*)):

- The graph $G_1 = (V''', E_1)$ is obtained by modifying G'' as follows:
  - Add a complete graph $K_{|V''|}$ on $|V''| = \frac{n^2+5n}{2} + 4$ nodes $w_0, w_1, \ldots, w_{|V''|-1}$ and the edge $\{u, w_0\}$. This step adds $|V''|$ new nodes and $\binom{|V''|}{2} + 1$ new edges.

  Thus, we have $|V'''| = 2\,|V''| = n^2 + 5n + 8$, and

$$|E_1| = |E''| + \binom{|V''|}{2} + 1 = \frac{n^4 + 10n^3 + 43n^2 + 102n + 104}{8}$$

- The graph $G_2 = (V''', E_2)$ is obtained from G_1 as follows. Let $\mathcal{A}$ be the set of edges of a sub-graph of the graph $K_{|V''|}$ (added in the previous step) that is isomorphic to the graph $(V'', \widehat{E})$ where

$$\widehat{E} = \big(E'' \setminus E\big) \bigcup \big\{ \{v_j, v_{j+1}\} \mid j \in \{0, 1, \ldots, n+1\} \big\}$$

and **the node w_0 is mapped to the node u in the isomorphism**. Such a sub-graph can be trivially found in polynomial time. For notational convenience we number the nodes in this sub-graph such that the order of the nodes in the largest cycle (having $2n + 4$ edges) of this sub-graph is $w_0, w_1, \ldots, w_{2n+3}$ (see Fig. 6 (c)). We then set $E_2 = E'' \cup A \cup \{u, w_0\}$. Thus,

$$|E_2| = |E''| + |\mathcal{A}| + 1 = |E''| + |\widehat{E}| + 1 = |E''| + \big(|E''| - |E| + n + 2\big) + 1 = n^2 + \frac{15n}{2} + 12$$

We first need to prove some bounds on the hyperbolicities of various graphs and sub-graphs that appear in our reduction. It is trivial to see that $\mathfrak{C}_{\text{Gromov}}(K_{|V''|}) = 0$. Define $\widetilde{\Delta_{u,v,w}}(G)$ be a geodesic triangle which contributes to the *minimality* of the value of $\mathfrak{C}_{\text{Gromov}}(G)$, *i.e.*, one of the shortest paths, say $\overline{u,v}$, lies in a $\mathfrak{C}_{\text{Gromov}}(\widetilde{\Delta_{u,v,w}}(G))$-neighborhood of the union $\overline{u,w} \cup \overline{v,w}$ of the other two shortest paths, but $\overline{u,v}$ does *not* lie in a δ-neighborhood of $\overline{u,w} \cup \overline{v,w}$ for any $\delta < \mathfrak{C}_{\text{Gromov}}(\widetilde{\Delta_{u,v,w}}(G))$. The following two facts are well-known.

Fact 1. *For any geodesic triangle $\Delta_{u,v,w}$, from the definition of $\mathfrak{C}_{\text{Gromov}}(\Delta_{u,v,w})$ (cf.* Definition 12) *it follows that* $\mathfrak{C}_{\text{Gromov}}(\Delta_{u,v,w}) \leq \max\left\{ \left\lfloor \text{dist}_G(u,v)/2 \right\rfloor, \left\lfloor \text{dist}_G(v,w)/2 \right\rfloor, \left\lfloor \text{dist}_G(u,w)/2 \right\rfloor \right\}$.

Fact 2 ((Rodríguez and Tourís, 2004, Lemma 2.1)). *We may assume that* $\widetilde{\Delta_{u,v,w}}(G)$ *is a* simple *geodesic triangle,* i.e.*, the three shortest paths* $\overline{u,v}$*,* $\overline{u,w}$ *and* $\overline{v,w}$ *do not share any nodes other than* u*,* v *or* w*.*

Let H denote the (node-induced) sub-graph $(\{w_0, w_1, \ldots, w_{|V''|-1}\}, \mathcal{A})$ of G_2.

Lemma 11. $\mathfrak{C}_{\mathrm{Gromov}}(G_2) = \mathfrak{C}_{\mathrm{Gromov}}(H) = \frac{n}{2} + 1$.

Proof. By Fact 2 $\widetilde{\Delta_{p,q,r}}(G_2)$ must be a simple geodesic triangle and therefore can only include edges in $\mathcal{A}$. Since the diameter of the sub-graph H is $n+2$, for any geodesic triangle $\Delta_{p,q,r}$ of H we have $\max\left\{\left\lfloor \mathrm{dist}_G(p,q)/2 \right\rfloor, \left\lfloor \mathrm{dist}_G(q,r)/2 \right\rfloor, \left\lfloor \mathrm{dist}_G(p,r)/2 \right\rfloor\right\} \leq n+2$ and thus by Fact 1 we have $\mathfrak{C}_{\mathrm{Gromov}}(G_2) = \mathfrak{C}_{\mathrm{Gromov}}(H) = \mathfrak{C}_{\mathrm{Gromov}}(\widetilde{\Delta_{p,q,r}}(G_2)) \leq \frac{n}{2} + 1$. Thus, it suffices we provide a simple geodesic triangle $\Delta_{p,q,r}$ of H for some three nodes p, q, r of H such that $\mathfrak{C}_{\mathrm{Gromov}}(\Delta_{p,q,r}(H)) = \frac{n}{2} + 1$. Consider the simple geodesic triangle $\Delta_{w_0, w_{\frac{n}{2}+1}, w_{\frac{3n}{2}+3}}$ of H consisting of the three shortest paths $\mathcal{Q}_1 \stackrel{\mathrm{def}}{=} w_0 \leftrightarrow w_1 \leftrightarrow w_2 \leftrightarrow \cdots \leftrightarrow w_{\frac{n}{2}} \leftrightarrow w_{\frac{n}{2}+1}$, $\mathcal{Q}_2 \stackrel{\mathrm{def}}{=} w_{\frac{n}{2}+1} \leftrightarrow w_{\frac{n}{2}+2} \leftrightarrow w_{\frac{n}{2}+3} \leftrightarrow \cdots \leftrightarrow w_{\frac{3n}{2}+2} \leftrightarrow w_{\frac{3n}{2}+3}$ and $\mathcal{Q}_3 \stackrel{\mathrm{def}}{=} w_{\frac{3n}{2}+3} \leftrightarrow w_{\frac{3n}{2}+4} \leftrightarrow w_{\frac{3n}{2}+5} \leftrightarrow \cdots \leftrightarrow w_{2n+3}, w_0$, and consider the node w_{n+2} that is the mid-point of the shortest path $\mathcal{Q}_2$ (see Fig. 6 (c)). It is easy to verify that the distance of the node w_{n+2} from the union of the two shortest paths $\mathcal{Q}_1$ and $\mathcal{Q}_3$ is $\frac{n}{2} + 1$. □

Now, suppose that we can prove the following two claims:

(completeness) if G has a Hamiltonian path between v_1 and v_n then $\mathsf{OPT}_{\mathrm{Tadp}_{\mathfrak{C}_{\mathrm{Gromov}}}}(G_1, G_2) \leq \frac{n}{2} + 1$

(soundness) if G has no Hamiltonian paths between v_1 and v_n then

$$\mathsf{OPT}_{\mathrm{Tadp}_{\mathfrak{C}_{\mathrm{Gromov}}}}(G_1, G_2) \geq \frac{n^3+3n^2+2n}{2}$$

Note that this proves the theorem since $\frac{\frac{n^3+3n^2+2n}{2}}{\frac{n}{2}+1} > \frac{n^2}{5} = \Omega\left(|V'''|\right)$.

Proof of completeness

Suppose that G has a Hamiltonian path between v_1 and v_n, say $v_1 \leftrightarrow v_2 \leftrightarrow v_3 \leftrightarrow \cdots \leftrightarrow v_{n-1} \leftrightarrow v_n$. Thus, G'' has a Hamiltonian path $v_0 \leftrightarrow v_1 \leftrightarrow v_2 \leftrightarrow v_3 \leftrightarrow \cdots \leftrightarrow v_{n-1} \leftrightarrow v_n \leftrightarrow v_{n+1} \leftrightarrow v_{n+2}$ between v_0 and v_{n+2}. We remove the $\frac{n}{2}+1$ edges in $E_d = E'' \setminus \left\{ \{v_j, v_{j+1}\} \,|\, j = 0, 1, \ldots, n+1 \right\}$ that are not in this Hamiltonian path resulting in the graph $G_2' = G_1 \setminus E_d$ (see Fig. 6 (*d*)). To show that $\mathfrak{C}_{\text{Gromov}}(G_2') = \mathfrak{C}_{\text{Gromov}}(G_2)$, note that by Fact 2 $\widetilde{\Delta_{p,q,r}}(G_2')$ must be a simple geodesic triangle and therefore

$$\begin{aligned}\mathfrak{C}_{\text{Gromov}}(G_2') &= \max\left\{ \mathfrak{C}_{\text{Gromov}}(G'' \setminus E_d),\, \mathfrak{C}_{\text{Gromov}}(K_{|V''|}) \right\} \\ &= \max\left\{\mathfrak{C}_{\text{Gromov}}(G'' \setminus E_d),\, 0\right\} = \mathfrak{C}_{\text{Gromov}}(G'' \setminus E_d)\end{aligned}$$

Since $G'' \setminus E_d$ is isomorphic to H, by Lemma 11 $\mathfrak{C}_{\text{Gromov}}(G'' \setminus E_d) = \mathfrak{C}_{\text{Gromov}}(H) = \mathfrak{C}_{\text{Gromov}}(G_2)$.

Proof of soundness

Assume that G has no Hamiltonian paths between v_1 and v_n, and let $E_d \subseteq E_1 \setminus E_2$ be the optimal set of edges that need to be deleted to obtain the graph $G_2' = (V''', E_1 \setminus E_d)$ such that $\mathfrak{C}_{\text{Gromov}}(G_2') = \mathfrak{C}_{\text{Gromov}}(G)$. By Fact 2, $\widetilde{\Delta_{p,q,r}}(G_2')$ must be a simple geodesic triangle and therefore

$$\mathfrak{C}_{\text{Gromov}}(G_2') = \max\left\{ \mathfrak{C}_{\text{Gromov}}(G'' \setminus E_d),\, \mathfrak{C}_{\text{Gromov}}(K_{|V''|} \setminus E_d) \right\} = \mathfrak{C}_{\text{Gromov}}(G_2) = \frac{n}{2} + 1 \quad (4.2)$$

Lemma 12. $\mathfrak{C}_{\text{Gromov}}(G'' \setminus E_d) \leq \frac{n}{2}$.

Proof. Since G has no Hamiltonian paths between v_1 and v_n, $\mathsf{diam}(G' \setminus E_d) \leq n+1$. Assume, for the sake of contradiction, that $\mathfrak{C}_{\text{Gromov}}(G'' \setminus E_d) \geq \frac{n}{2}+1$. By Fact 1, $\mathfrak{C}_{\text{Gromov}}(G'' \setminus E_d) = \mathfrak{C}_{\text{Gromov}}(\widetilde{\Delta_{p,q,r}}(G'' \setminus E_d)) \leq \max\left\{\left\lfloor \text{dist}_{G''\setminus E_d}(p,q)/2 \right\rfloor, \left\lfloor \text{dist}_{G''\setminus E_d}(q,r)/2 \right\rfloor, \left\lfloor \text{dist}_{G''\setminus E_d}(p,r)/2 \right\rfloor\right\}$, and thus at least one of the three distances in the left-hand-side of the above inequality, say $\text{dist}_{G''\setminus E_d}(p,q)$, must be at least $n+2$. Let $\mathcal{L}(\mathcal{C}(H))$ and $\mathcal{L}(H)$ denote the length (number of edges) of a (simple) cycle $\mathcal{C}$ and the length of the *longest (simple) cycle* of a graph H. Since $\mathfrak{C}_{\text{Gromov}}(\widetilde{\Delta_{p,q,r}}(G'' \setminus E_d)) > 0$ and $\widetilde{\Delta_{p,q,r}}(G'' \setminus E_d)$ must be a simple geodesic triangle, there must be at least one cycle, say $\mathcal{C}$, in $G'' \setminus E_d$ containing p, q and r. Now, note that

$$\mathcal{L}(\mathcal{C}(G'' \setminus E_d)) \leq \mathcal{L}(G'' \setminus E_d) \leq 2\left(\frac{n}{2}+1\right) + \mathsf{diam}(G' \setminus E_d) \leq 2n+3$$

and therefore $\text{dist}_{G''\setminus E_d}(p,q) \leq \left\lfloor \frac{2n+3}{2} \right\rfloor = n+1$, which provides the desired contradiction. □

By Lemma 12 and Equation (Equation 4.2) it follows that $\mathfrak{C}_{\text{Gromov}}(K_{|V''|} \setminus E_d) = \frac{n}{2}+1$.

Lemma 13. *If* $\mathfrak{C}_{\text{Gromov}}(K_{|V''|} \setminus E_d) \geq \frac{n}{2}+1$ *then* $|E_d| \geq \frac{n^3+3n^2+2n}{2}$.

Proof. Since $\mathfrak{C}_{\text{Gromov}}(K_{|V''|} \setminus E_d) = \mathfrak{C}_{\text{Gromov}}(\widetilde{\Delta_{p,q,r}}(K_{|V''|} \setminus E_d)) = \frac{n}{2}+1$, by Fact 1 at least one of the three distances $\text{dist}_{K_{|V''|}\setminus E_d}(p,q)$, $\text{dist}_{K_{|V''|}\setminus E_d}(q,r)$ or $\text{dist}_{K_{|V''|}\setminus E_d}(p,r)$, say $\text{dist}_{K_{|V''|}\setminus E_d}(p,q)$, must be at least $n+2$. This implies that $K_{|V''|} \setminus E_d$ must contain a shortest path of length $n+2$, say $\mathcal{Q} \stackrel{\text{def}}{=} w_0 \leftrightarrow w_1 \leftrightarrow w_2 \leftrightarrow \cdots \leftrightarrow w_{n+1} \leftrightarrow w_{n+2}$. We now claim that *no* node from the set $W_1 = \left\{w_{n+3}, w_{n+4}, \ldots, w_{|V''|-1}\right\}$ is connected to *more than* 3 nodes from the set $W_2 = \{w_0, w_1, \ldots, w_{n+2}\}$ in $K_{|V''|} \setminus E_d$. To show this by contradiction, suppose that some node $w_i \in W_1$ is connected to four nodes

$w_j, w_k, w_\ell, w_r \in W_2$ with $j < k < \ell < r$. Then $r \geq j + 3$ which implies $\mathrm{dist}_{K_{|V''|} \setminus E_d}(w_j, w_r) \leq 2$, contradicting the fact that $\mathcal{Q}$ is a shortest path. It thus follows that

$$|E_d| \geq \big((n+3) - 3\big)|W_1| = n(|V''| - (n+3)) = n\left(\frac{n^2+3n}{2} + 1\right) = \frac{n^3+3n^2+2n}{2} \square$$

The above lemma obviously completes the proof of soundness of our reduction.

4.6 Conclusion and future research

Notions of curvatures of higher-dimensional geometric shapes and topological spaces play a *fundamental* role in physics and mathematics in characterizing anomalous behaviours of these higher dimensional entities. However, using curvature measures to detect anomalies in networks is *not* yet very common due to several reasons such as lack of preferred geometric interpretation of networks and lack of experimental evidences that may lead to specific desired curvature properties. In this paper we have attempted to formulate and analyze curvature analysis methods to provide the foundations of systematic approaches to find critical components and anomaly detection in networks by using two measures of network curvatures, namely the Gromov-hyperbolic combinatorial curvature and the geometric curvature measure. This paper must *not* be viewed as uttering the final word on appropriateness and suitability of specific curvature measures, but rather should be viewed as a stimulator and motivator of further theoretical or empirical research on the exciting interplay between notions of curvatures from network and non-network domains.

There is a *plethora* of interesting future research questions and directions raised by the topical discussions and results in this paper. Some of these are stated below.

▷ For geometric curvatures, we considered the first-order non-trivial measure $\mathfrak{C}_d^2$. It would be of interest to investigate computational complexity issues of anomaly detection problems using $\mathfrak{C}_d^p$ for $p > 2$. We conjecture that our algorithmic results for extremal anomaly detection using $\mathfrak{C}_d^2$ (Theorem 10(a2-2)&(b2)) can be extended to $\mathfrak{C}_d^3$.

▷ There are at least two more aspects of geometric curvatures that need further careful investigation. Firstly, the topological association of elementary components to higher-dimensional objects as described in this paper is by *no* means the only reasonable topological association possible. But, more importantly, other suitable notions of geometric curvatures are quite possible. As a very simple illustration, assuming that smaller dimensional simplexes edges in the discrete network setting correspond to vectors or directions in the smooth context, an analogue of the Bochner-Weitzenböck formula developed by Forman for the curvature for a simplex s can be given by the formula (Forman, 2003; Samal et al., 2018):

$$\mathfrak{F}(s) = w_s \left(\left(\sum_{s \prec s'} \frac{w_s}{w_{s'}} + \sum_{s' \prec s} \frac{w_{s'}}{w_s} \right) - \sum_{s' \| s} \Big| \sum_{s, s' \prec g} \frac{\sqrt{w_s w_{s'}}}{w_g} + \sum_{g \prec s, s'} \frac{w_g}{\sqrt{w_s w_{s'}}} \Big| \right)$$

where $a \prec b$ means a is a face of b, $a \parallel b$ means a and b have either a common higher-dimensional face or a common lower-dimensional face but *not* both, and w is a function that assigns weights to simplexes. One can then either modify the Euler characteristics as $\sum_{k=0}^{p} (-1)^k \mathfrak{F}(f_d^k)$ or by combining the individual $\mathfrak{F}(f_d^k)$ values using curvature functions defined by Bloch (Bloch, 2014).

▷ Our inapproximability results for the Gromov-hyperbolic curvature require a high average node degree. Thus, for real-world networks such as scale-free networks the inapproximability bounds may not apply.

We hypothesize that the anomaly detection problems using Gromov-hyperbolic curvatures is much more computationally tractable than what our results depict for networks with bounded average degree.

Acknowledgements

We thank Anastasios Sidiropoulos and Nasim Mobasheri for very useful discussions. This research work was partially supported by NSF grants IIS-1160995 and IIS-1814931.

CITED LITERATURE

[Abrahamsen et al. , 2016]Abrahamsen, M., Bodwin, G., Rotenberg, E., and Stöckel, M.: Graph reconstruction with a betweenness oracle. 47, 2016.

[Albert et al. , 2007]Albert, R., DasGupta, B., Dondi, R., Kachalo, S., Sontag, E., Zelikovsky, A., and Westbrooks, K.: A novel method for signal transduction network inference from indirect experimental evidence. Journal of Computational Biology, 14(7):927–949, 2007.

[Albert et al. , 2008]Albert, R., DasGupta, B., Dondi, R., and Sontag, E.: Inferring (biological) signal transduction networks via transitive reductions of directed graphs. Algorithmica, 51(2):129–159, 2008.

[Albert et al. , 2011]Albert, R., DasGupta, B., Gitter, A., Gürsoy, G., Hegde, R., Pal, P., Sivanathan, G. S., and Sontag, E. D.: A new computationally efficient measure of topological redundancy of biological and social networks. Physical Review E, 84(3):036117, 2011.

[Albert et al. , 2014]Albert, R., DasGupta, B., and Mobasheri, N.: Topological implications of negative curvature for biological and social networks. Physical Review E, 89(3):032811, 2014.

[Albert and l. Barabási, 2002]Albert, R. and l. Barabási, A.: Statistical mechanics of complex networks. Reviews of Modern Physics, 74(1):47–97, 2002.

[Alon and Asodi, 2005a]Alon, N. and Asodi, V.: Learning a hidden subgraph. SIAM Journal on Discrete Mathematics, 18(4):697–712, 2005.

[Alon and Asodi, 2005b]Alon, N. and Asodi, V.: Learning a hidden subgraph. SIAM Journal on Discrete Mathematics, 18(4):697–712, 2005.

[Alon et al. , 2009]Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J.: The online set cover problem. SIAM J. Comput., 39(2):361–370, 2009.

[Alon et al. , 2006]Alon, N., Awerbuch, B., Azar, Y., Buchbinder, N., and Naor, J. S.: A general approach to online network optimization problems. ACM Transactions on Algorithms (TALG), 2(4):640–660, 2006.

[Alon et al. , 2004a]Alon, N., Beigel, R., Kasif, S., Rudich, S., and Sudakov, B.: Learning a hidden matching. SIAM Journal on Computing, 33(2):487–501, 2004.

[Alon et al. , 2004b]Alon, N., Beigel, R., Kasif, S., Rudich, S., and Sudakov, B.: Learning a hidden matching. SIAM Journal on Computing, 33(2):487–501, 2004.

[Alter and Golub, 2005]Alter, O. and Golub, G. H.: Reconstructing the pathways of a cellular system from genome-scale signals by using matrix and tensor computations. PNAS, 102(49):17559–17564, 2005.

[Aminikhanghahi and Cook, 2017]Aminikhanghahi, S. and Cook, D. J.: A survey of methods for time series change point detection. Knowledge and Information Systems, 51(2):339–367, 2017.

[Angluin, 1988]Angluin, D.: Queries and concept learning. Machine Learning, 2(4):319–342, Apr 1988.

[Angluin et al. , 2010a]Angluin, D., Aspnes, J., and Reyzin, L.: Inferring social networks from outbreaks. In International Conference on Algorithmic Learning Theory, pages 104–118. Springer, 2010.

[Angluin et al. , 2010b]Angluin, D., Aspnes, J., and Reyzin, L.: Inferring social networks from outbreaks. In ALT, pages 104–118, 2010.

[Angluin et al. , 2010c]Angluin, D., Aspnes, J., and Reyzin, L.: Optimally learning social networks with activations and suppressions. Theor. Comput. Sci., 411(29-30):2729–2740, 2010.

[Angluin et al. , 2015]Angluin, D., Aspnes, J., and Reyzin, L.: Network construction with subgraph connectivity constraints. Journal of Combinatorial Optimization, 29(2):418–432, 2015.

[Angluin and Chen, 2008a]Angluin, D. and Chen, J.: Learning a hidden graph using $o(\log n)$ queries per edge. Journal of Computer and System Sciences, 74(4):546–556, 2008.

[Angluin and Chen, 2008b]Angluin, D. and Chen, J.: Learning a hidden graph using o (logn) queries per edge. Journal of Computer and System Sciences, 74(4):546–556, 2008.

[Ariaei et al. , 2008]Ariaei, F., Lou, M., Jonckeere, E., Krishnamachari, B., and Zuniga, M.: Curvature of sensor network: clustering coefficient. EURASIP Journal on Wireless Communications and Networking, 213185, 2008.

[Arora and Barak, 2009]Arora, S. and Barak, B.: Computational Complexity: A Modern Approach. New York, NY, USA, Cambridge University Press, 1st edition, 2009.

[Bassett et al. , 2011]Bassett, D. S., Wymbs, N. F., Porter, M. A., Mucha, P. J., Carlson, J. M., and Grafton, S. T.: Dynamic reconfiguration of human brain networks during learning. PNAS, 108(18):7641–7646, 2011.

[Beerliova et al. , 2006]Beerliova, Z., Eberhard, F., Erlebach, T., Hall, A., Hoffmann, M., Mihal'ak, M., and Ram, L. S.: Network discovery and verification. IEEE Journal on selected areas in communications, 24(12):2168–2181, 2006.

[Beigel et al. , 2001]Beigel, R., Alon, N., Kasif, S., Apaydin, M. S., and Fortnow, L.: An optimal procedure for gap closing in whole genome shotgun sequencing. In Proceedings of the fifth annual international conference on Computational biology, pages 22–30. ACM, 2001.

[Benjamini, 1998]Benjamini, I.: Expanders are not hyperbolic. Israel Journal of Mathematics, 108:33–36, 1998.

[Berger, 2012]Berger, M.: A Panoramic View of Riemannian Geometry. Springer, 2012.

[Bloch, 2014]Bloch, E.: Combinatorial ricci curvature for polyhedral surfaces and posets. preprint, [math.CO], 2014.

[Booth and Lueker, 1976]Booth, K. S. and Lueker, G. S.: Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. Journal of Computer and System Sciences, 13(3):335–379, 1976.

[Borodin and El-Yaniv, 1998]Borodin, A. and El-Yaniv, R.: Online Computation and Competitive Analysis. New York, NY, USA, Cambridge University Press, 1998.

[Bosc et al. , 2003]Bosc, M., Heitz, F., Armspach, J. P., Namer, I., Gounot, D., and Rumbach, L.: Automatic change detection in multimodal serial mri: application to multiple sclerosis lesion evolution. Neuroimage, 20(2):643–656, 2003.

[Bridson and Haefliger, 1999]Bridson, M. R. and Haefliger, A.: Metric Spaces of Non-Positive Curvature. Springer, 1999.

[Buchbinder and Naor, 2009]Buchbinder, N. and Naor, J.: The design of competitive online algorithms via a primal: dual approach. Foundations and Trends® in Theoretical Computer Science, 3(2–3):93–263, 2009.

[Chepoi et al. , 2008]Chepoi, V., Dragan, F. F., Estellon, B., Habib, M., and Vaxès, Y.: Diameters, centers, and approximating trees of δ-hyperbolic geodesic spaces and graphs. In proceedings of the 24th Annual Symposium on Computational geometry, pages 59–68, 2008.

[Chepoi et al. , 2012]Chepoi, V., Dragan, F. F., Estellon, B., Habib, M., Vaxès, Y., and Xiang, Y.: Additive spanners and distance and routing labeling schemes for δ-hyperbolic graphs. Algorithmica, 62(3-4):713–732, 2012.

[Chepoi and Estellon, 2007]Chepoi, V. and Estellon, B.: Packing and covering δ-hyperbolic spaces by balls. In Lecture Notes in Computer Science 4627, eds. M. Charikar, K. Jansen, O. Reingold, and J. D. P. Rolim, pages 59–73. Springer, 2007.

[Chockler et al. , 2007]Chockler, G., Melamed, R., Tock, Y., and Vitenberg, R.: Constructing scalable overlays for pub-sub with many topics. In Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing, pages 109–118. ACM, 2007.

[Chowdhury et al. , 2011]Chowdhury, M. F. R., Selouani, S. A., and O'Shaughnessy, D.: Bayesian on-line spectral change point detection: a soft computing approach for on-line asr. International Journal of Speech Technology, 15(1):5–23, 2011.

[Colizza et al. , 2006]Colizza, V., Flammini, A., Serrano, M. A., and Vespignani, A.: Detecting rich-club ordering in complex networks. Nature Physics, 2:110–115, 2006.

[Cygan et al. , 2015]Cygan, M., Fomin, F., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S.: Parameterized Algorithms. Springer, 2015.

[Dall'Asta et al. , 2006]Dall'Asta, L., Alvarez-Hamelin, I., Barrat, A., Vázquez, A., and Vespignani, A.: Exploring networks with traceroute-like probes: Theory and simulations. Theoretical Computer Science, 355(1):6–24, 2006.

[DasGupta et al. , 2018]DasGupta, B., Karpinski, M., Mobasheri, N., and Yahyanejad, F.: Effect of gromov-hyperbolicity parameter on cuts and expansions in graphs and some algorithmic implications. Algorithmica, 80(2):772–800, 2018.

[DasGupta and Liang, 2016]DasGupta, B. and Liang, J.: Models and Algorithms for Biomolecules and Molecular Networks. Inc, John Wiley & Sons, 2016.

[DasGupta et al. , 2018]DasGupta, B., Janardhanan, M. V., and Yahyanejad, F.: How did the shape of your network change? (on detecting anomalies in static and dynamic networks via change of non-local curvatures). Submitted, arXiv:1808.05676, 2018.

[de Montgolfier et al. , 2011]de Montgolfier, F., Soto, M., and Viennot, L.: Treewidth and hyperbolicity of the internet. In proceedings of the 10th IEEE International Symposium on Networking Computing and Applications, pages 25–32, 2011.

[Dinur and Safra, 2005a]Dinur, I. and Safra, S.: On the hardness of approximating minimum vertex cover. Annals of Mathematics, 162(1):439–485, 2005.

[Dinur and Safra, 2005b]Dinur, I. and Safra, S.: On the hardness of approximating minimum vertex cover. Annals of Mathematics, 162(1):439–485, 2005.

[Ducre-Robitaille et al. , 2003]Ducre-Robitaille, J. F., Vincent, L. A., and Boulet, G.: Comparison of techniques for detection of discontinuities in temperature series. International Journal of Climatology, 23(9):1087–1101, 2003.

[Erlebach et al. , 2006]Erlebach, T., Hall, A., Hoffmann, M., and Mihal'ák, M.: Network discovery and verification with distance queries. In Italian Conference on Algorithms and Complexity, pages 69–80. Springer, 2006.

[Feige, 1998]Feige, U.: A threshold of ln n for approximating set cover. Journal of the ACM (JACM), 45(4):634–652, 1998.

[Forman, 2003]Forman, R.: Bochner's method for cell complexes and combinatorial ricci curvature. Discrete and Computational Geometry, 29(3):323–374, 2003.

[Fournier et al. , 2015]Fournier, H., Ismail, A., and Vigneron, A.: Computing the gromov hyperbolicity of a discrete metric space. Information Processing Letters, 115(6-8):576–579, 2015.

[Gamelin and Greene, 1999]Gamelin, T. W. and Greene, R. E.: Introduction to Topology. Dover publications, 1999.

[Garey and Johnson, 1979]Garey, M. R. and Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman, 1979.

[Garey et al. , 1976]Garey, M. R., Johnson, D. S., and Tarjan, R. E.: The planar hamiltonian circuit problem is np-complete. SIAM Journal of Computing, 5:704–714, 1976.

[Garey and Johnson, 1990]Garey, M. R. and Johnson, D. S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. New York, NY, USA, W. H. Freeman & Co., 1990.

[Gavoille and Ly., 2005]Gavoille, C. and Ly., O.: Distance labeling in hyperbolic graphs. In Lecture Notes in Computer Science 3827, eds. X. Deng and D. z. Du, pages 1071–1079. Springer, 2005.

[Goldberg, 1984]Goldberg, A. V.: Finding a maximum density subgraph. Technical report, 1984.

[Gomez-Rodriguez et al. , 2012]Gomez-Rodriguez, M., Leskovec, J., and Krause, A.: Inferring networks of diffusion and influence. ACM Transactions on Knowledge Discovery from Data (TKDD), 5(4):21, 2012.

[Grebinski and Kucherov, 1998]Grebinski, V. and Kucherov, G.: Reconstructing a hamiltonian cycle by querying the graph: Application to dna physical mapping. Discrete Applied Mathematics, 88(1):147–165, 1998.

[Gromov, 1987]Gromov, M.: Hyperbolic groups. Essays in group theory, 8:75–263, 1987.

[Gupta et al. , 2012]Gupta, A., Krishnaswamy, R., and Ravi, R.: Online and stochastic survivable network design. SIAM Journal on Computing, 41(6):1649–1672, 2012.

[Hawking and Penrose, 1996]Hawking, S. and Penrose, R.: The Nature of Space and Time. Princeton University Press, 1996.

[Hein, 1989]Hein, J. J.: An optimal algorithm to reconstruct trees from additive distance data. Bulletin of mathematical biology, 51(5):597–603, 1989.

[Henle, 1994]Henle, M.: A Combinatorial Introduction to Topology. Dover publications, 1994.

[Huang et al. , 2017]Huang, Y., Janardhanan, M. V., and Reyzin, L.: Network construction with ordered constraints. FSTTCS, 2017.

[Huang, 2017]Huang, Y.: Problems in Learning under Limited Resources and Information. Doctoral dissertation, University of Illinois at Chicago, 2017.

[Impagliazzo and Paturi, 2001]Impagliazzo, R. and Paturi, R.: On the complexity of ksat. Journal of Computer and System Sciences, 62:367–375, 2001.

[Impagliazzo et al. , 2001]Impagliazzo, R., Paturi, R., and Zane, F.: Which problems have strongly exponential complexity? Journal of Computer and System Sciences, 63(4):512–530, 2001.

[Janardhanan, 2017]Janardhanan, M. V.: Graph verification using a betweenness oracle. Algorithmic Learning Theory, 2017.

[Jarrah et al. , 2007]Jarrah, A. S., Laubenbacher, R., Stigler, B., and Stillman, M.: Reverse-engineering polynomial dynamical systems. Advances in Applied Mathematics, 39(4):477–489, 2007.

[Jonckheere et al. , 2011]Jonckheere, E., Lohsoonthorn, P., and Ariaei, F.: Scaled gromov four-point condition for network graph curvature computation. Internet Mathematics, 7(3):137–177, 2011.

[Jonckheere et al. , 2007]Jonckheere, E., Lohsoonthorn, P., and Bonahon, F.: Scaled gromov hyperbolic graphs. Journal of Graph Theory, 57(2):157–180, 2007.

[Jonckheere and Lohsoonthorn, 2004]Jonckheere, E. A. and Lohsoonthorn, P.: Geometry of network security. American Control Conference, 2:976–981, 2004.

[Jonckheerea et al. , 2011]Jonckheerea, E., Loua, M., Bonahona, F., and Baryshnikova, Y.: Euclidean versus hyperbolic congestion in idealized versus experimental networks. Internet Mathematics, 7(1):1–27, 2011.

[Kannan et al. , 2015]Kannan, S., Mathieu, C., and Zhou, H.: Near-linear query complexity for graph inference. pages 773–784, 2015.

[Kawahara and Sugiyama, 2009]Kawahara, Y. and Sugiyama, M.: Sequential change-point detection based on direct density-ratio estimation. In SIAM International Conference on Data Mining, pages 389–400, 2009.

[Khot, 2002]Khot, S.: On the power of unique 2-prover 1-round games. In 34th ACM Symposium on Theory of Computing, pages 767–775, 2002.

[Khot and Regev, 2008a]Khot, S. and Regev, O.: Vertex cover might be hard to approximate to within 2-ε. Journal of Computer and System Sciences, 74(3):335–349, 2008.

[Khot and Regev, 2008b]Khot, S. and Regev, O.: Vertex cover might be hard to approximate to within 2. Journal of Computer and System Sciences, 74(3):335 – 349, 2008. Computational Complexity 2003.

[King et al. , 2003]King, V., Zhang, L., and Zhou, Y.: On the complexity of distance-based evolutionary tree reconstruction. In Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms, pages 444–453. Society for Industrial and Applied Mathematics, 2003.

[Kolb and Whishaw, 1996]Kolb, B. and Whishaw, I. Q.: Fundamentals of Human Neuropsychology. New York, Freeman, 1996.

[Korach and Stern, 2003]Korach, E. and Stern, M.: The clustering matroid and the optimal clustering tree. Mathematical Programming, 98(1-3):385–414, 2003.

[Korach and Stern, 2008]Korach, E. and Stern, M.: The complete optimal stars-clustering-tree problem. Discrete Applied Mathematics, 156(4):444–450, 2008.

[Latora and Marchior, 2007]Latora, V. and Marchior, M.: A measure of centrality based on network efficiency. New Journal of Physics, 9:188, 2007.

[Mathieu and Zhou, 2013]Mathieu, C. and Zhou, H.: Graph Reconstruction via Distance Oracles, pages 733–744. Berlin, Heidelberg, Springer Berlin Heidelberg, 2013.

[McKay and Wormald, 1990]McKay, B. D. and Wormald, N. C.: Asymptotic enumeration by degree sequence of graphs of high degree. European Journal of Combinatorics, 11(6):565 – 580, 1990.

[Mohri et al. , 2012]Mohri, M., Rostamizadeh, A., and Talwalkar, A.: Foundations of Machine Learning. The MIT Press, 2012.

[Moulin and Laigret, 2011]Moulin, H. and Laigret, F.: Equal-need sharing of a network under connectivity constraints. Games and Economic Behavior, 72(1):314–320, 2011.

[Narayan and Saniee, 2011]Narayan, D. and Saniee, I.: Large-scale curvature of networks. Physical Review E, 84:066108, 2011.

[Newman, 2010]Newman, M. E. J.: Networks: An Introduction. Oxford University Press, 2010.

[Omberg et al. , 2007]Omberg, L., Golub, G. H., and Alter, O.: A tensor higher-order singular value decomposition for integrative analysis of dna microarray data from different studies. PNAS, 104(47):18371–18376, 2007.

[Papadimitriou and Steiglitz, 1982]Papadimitriou, C. and Steiglitz, K.: Combinatorial Optimization: Algorithm and Complexity. Prentice Hall, 1982.

[Papadopoulos et al. , 2010]Papadopoulos, F., Krioukov, D., Boguna, M., and Vahdat, A.: Greedy forwarding in dynamic scale-free networks embedded in hyperbolic metric spaces. In IEEE Conference on Computer Communications, pages 1–9, 2010.

[Raz and Safra, 1997]Raz, R. and Safra, S.: A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, pages 475–484. ACM, 1997.

[Reeves et al. , 2007]Reeves, J., Chen, J., Wang, X. L., Lund, R., and Lu., Q. Q.: A review and comparison of changepoint detection techniques for climate data. Journal of Applied Meteorological Climatology, 46(6):900–915, 2007.

[Reyzin, 2009]Reyzin, L.: Active Learning of Interaction Networks. Doctoral dissertation, Yale University, New Haven, Connecticut, 2009.

[Reyzin and Srivastava, 2007a]Reyzin, L. and Srivastava, N.: Learning and verifying graphs using queries with a focus on edge counting. In International Conference on Algorithmic Learning Theory, pages 285–297. Springer, 2007.

[Reyzin and Srivastava, 2007b]Reyzin, L. and Srivastava, N.: Learning and verifying graphs using queries with a focus on edge counting. In International Conference on Algorithmic Learning Theory, pages 285–297. Springer, 2007.

[Reyzin and Srivastava, 2007c]Reyzin, L. and Srivastava, N.: On the longest path algorithm for reconstructing trees from distance matrices. Information processing letters, 101(3):98–100, 2007.

[Rodríguez and Tourís, 2004]Rodríguez, J. M. and Tourís, E.: Gromov hyperbolicity through decomposition of metric spaces. Acta Mathematica Hungarica, 103:53–84, 2004.

[Roe, 1996]Roe, J.: Index theory, coarse geometry, and topology of manifolds. Conference Board of the Mathematical Sciences Regional Conference, Series, 90, 1996.

[Rybach et al. , 2009]Rybach, D., Gollan, C., Schluter, R., and Ney, H.: Audio segmentation for speech recognition using segment features. In Speech and, ed. I. I. C. on Acoustics, pages 4197–4200. Signal Processing, 2009.

[Saadatpour et al. , 2011]Saadatpour, A., Wang, R. S., Liao, A., Liu, X., Loughran, T. P., Albert, I., and Albert, R.: Dynamical and structural analysis of a t cell survival network identifies novel candidate therapeutic targets for large granular lymphocyte leukemia. PLoS Computational Biology, 7, 2011.

[Saito et al. , 2008]Saito, K., Nakano, R., and Kimura, M.: Prediction of information diffusion probabilities for independent cascade model. In International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, pages 67–75. Springer, 2008.

[Samal et al. , 2018]Samal, A., Sreejith, R. P., Gu, J., Liu, S., Saucan, E., and Jost, J.: Comparative analysis of two discretizations of ricci curvature for complex networks. Scientific Reports, 8, 2018.

[Stoer and Wagner, 1997]Stoer, M. and Wagner, F.: A simple min-cut algorithm. Journal of the ACM (JACM), 44(4):585–591, 1997.

[Sun et al. , 2006]Sun, J., Tao, D., and Faloutsos, C.: Beyond streams and graphs: Dynamic tensor analysis. In 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 374–383, 2006.

[Thorup and Zwick, 2001]Thorup, M. and Zwick, U.: Compact routing schemes. In Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures, pages 1–10. ACM, 2001.

[Tononi et al. , 1999]Tononi, G., Sporns, O., and Edelman, G. M.: Measures of degeneracy and redundancy in biological networks. PNAS, 96:3257–3262, 1999.

[Trevisan, 2012]Trevisan, L.: On khot's unique games conjecture. Bulletin of the American Mathematical Society, 49(1):91–111, 2012.

[Valiant, 1984]Valiant, L. G.: A theory of the learnable. Commun. ACM, 27(11):1134–1142, November 1984.

[van Lint et al. , 2001]van Lint, J., Wilson, R., and Wilson, R.: A Course in Combinatorics. A Course in Combinatorics. Cambridge University Press, 2001.

[Vazirani, 2001]Vazirani, V.: Approximation Algorithms. Springer, 2001.

[Wagner, 2002]Wagner, A.: Estimating coarse gene network structure from large-scale gene perturbation data. Genome Research, 12:309–315, 2002.

[Waterman et al. , 1977]Waterman, M. S., Smith, T. F., Singh, M., and Beyer, W.: Additive evolutionary trees. Journal of theoretical Biology, 64(2):199–213, 1977.

[Weber et al. , 2016a]Weber, M., Jost, J., and Saucan, E.: Forman-ricci flow for change detection in large dynamic data sets. In International Conference on Information and Computational Science, 2016.

[Weber et al. , 2016b]Weber, M., Saucan, E., and Jost, J.: Can one see the shape of a network?. [math.CO], 2016.

[Woeginger, 2003]Woeginger, G.: Exact algorithms for NP-hard problems: A survey. You Shrink!, 2570, 185-207, Springer-Verlag, in Combinatorial Optimization – Eureka, 2003.

[Yahyanejad, 2019]Yahyanejad, F.: Curvature Analysis in Complex Networks: Theory and Application. Doctoral dissertation, University of Illinois at Chicago, 2019.

[Yang et al. , 2006]Yang, P., Dumont, G., and Ansermino, J. M.: Adaptive change detection in heart rate trend monitoring in anesthetized children. IEEE Transactions on Biomedical Engineering, 53(11):2211–2219, 2006.

[Yannakakis, 1978]Yannakakis, M.: Node- and edge-deletion np-complete problems. In 10th Annual ACM Symposium on Theory of Computing, pages 253–264, 1978.

[Zanudo and Albert, 2015]Zanudo, J. G. T. and Albert, R.: Cell fate reprogramming by control of intracellular network dynamics. PLOS Computational Biology, 11, 2015.

APPENDIX

This appendix contains publication agreements signed by the authors and reproductions of statements from the publishers websites detailing the use policies that allow the original publications and preprints to be reproduced in this thesis.

Publication Agreement

This is a publication agreement[1] ("this agreement") regarding a written manuscript currently entitled

Graph Verification with a Betweenness Oracle

("the article") to be published in PMLR ("the proceedings"). The parties to this Agreement are:

Mano Vikash Janardhanan

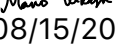
(name of corresponding author who signs on behalf of any other authors, collectively "you") and PMLR, ("the publisher").

1. By signing this form, you warrant that you are signing on behalf of all authors of the article, and that you have the authority to act as their agent for the purpose of entering into this agreement.
2. You hereby grant a Creative Commons copyright license in the article to the general public, in particular a Creative Commons Attribution 4.0 International License, which is incorporated herein by reference and is further specified at http://creativecommons.org/licenses/by/4.0/legalcode (human readable summary at http://creativecommons.org/licenses/by/4.0).
3. You agree to require that a citation to the original publication of the article in the proceedings as well as a hyperlink to the PMLR web site linking to the original paper be included in any attribution statement satisfying the attribution requirement of the Creative Commons license of paragraph 2.
4. You retain ownership of all rights under copyright in all versions of the article, and all rights not expressly granted in this agreement.
5. To the extent that any edits made by the publisher to make the article suitable for publication in the proceedings amount to copyrightable works of authorship, the publisher hereby assigns all right, title, and interest in such edits to you. The publisher agrees to verify with you any such edits that are substantive. You agree that the license of paragraph 2 covers such edits.

[1] The language of this publication agreement is based on Stuart Shieber's model open-access journal publication agreement, version 1.2, available at http://bit.ly/1m9UsNt.

6. You further warrant that:
    1. The article is original, has not been formally published in any other peerreviewed journal or in a book or edited collection, and is not under consideration for any such publication.
    2. You are the sole author(s) of the article, and that you have a complete and unencumbered right to make the grants you make.
    3. The article does not libel anyone, invade anyone's copyright or otherwise violate any statutory or common law right of anyone, and that you have made all reasonable efforts to ensure the accuracy of any factual information contained in the article. You agree to indemnify the publisher against any claim or action alleging facts which, if true, constitute a breach of any of the foregoing warranties or other provisions of this agreement, as well as against any related damages, losses, liabilities, and expenses incurred by the publisher.
7. This is the entire agreement between you and the publisher, and it may be modified only in writing. It will be governed by the laws of the Commonwealth of Massachusetts. It will bind and benefit our respective assigns and successors in interest, including your heirs. It will terminate if the publisher does not publish, in any medium, the article within one year of the date of your signature.

I HAVE READ AND AGREE FULLY WITH THE TERMS OF THIS AGREEMENT.

- Corresponding Author: Mano Vikash Janardhanan
  - Signed: 
  - Date: 08/15/2017

LIPIcs – Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik
Dagstuhl Publishing
http://www.dagstuhl.de/lipics

Event: **37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2017)**
Editors: **Satya Lokam and R. Ramanujam**

Title of contribution (named 'contribution' in the following)

Network construction with ordered constraints

Authors (named 'author' in the following)

Yi Huang, Mano Vikash Janardhanan and Lev Reyzin

Corresponding author's name, address, affiliation and email

Name: Mano Vikash Janardhanan
Address: 851 S Morgan Street, Chicago, IL 60608
Affiliation: University of Illinois at Chicago
E-Mail: mjanar2@uic.edu

18/10/2017, Kyoto

Date, City

Mano Vikash

Signature of the corresponding author

We gratefully acknowledge support from
the Simons Foundation and member institutions.

arXiv License Information

arXiv is a repository for scholarly material, and perpetual access is necessary to maintain the scholarly record. As such, arXiv keeps a permanent record of every submission and replacement announced.

arXiv does not ask that copyright be transferred. However, we require sufficient rights to allow us to distribute submitted articles in perpetuity. In order to submit an article to arXiv, the submitter must either:

- grant arXiv.org a non-exclusive and irrevocable license to distribute the article, and certify that he/she has the right to grant this license;
- certify that the work is available under one of the following Creative Commons licenses and that he/she has the right to assign this license:
  - Creative Commons Attribution license (CC BY 4.0)
  - Creative Commons Attribution-ShareAlike license (CC BY-SA 4.0)
  - Creative Commons Attribution-Noncommercial-ShareAlike license (CC BY-NC-SA 4.0);
- or dedicate the work to the public domain by associating the Creative Commons Public Domain Dedication (CC0 1.0) with the submission.

In the most common case, authors have the right to grant these licenses because they hold copyright in their own work.

We currently support three of the Creative Commons licenses. If you wish to use a different CC license, then select arXiv's non-exclusive license to distribute in the arXiv submission process and indicate the desired Creative Commons license in the actual article.

Note: if you intend to submit, or have submitted, your article to a journal then you should verify that the license you select during arXiv submission does not conflict with the journal's license or copyright transfer agreement. Many journal agreements permit submission to arXiv using the non-exclusive license to distribute, which arXiv has used since 2004. Yet the CC BY and CC BY-SA licenses permit commercial reuse and may therefore conflict with some journal agreements.

Authority to submit, publisher PDFs, etc.

arXiv cannot accept papers that contain material for which the depositor does not have the authority to submit or to agree to the license terms. This would include comments by referees (which may have separate copyright protection) and, of course, plagiarized material. Note that any single co-author normally has all needed authority to submit a paper to arXiv. Once publicly available, articles cannot be entirely removed, either at the request of the submitting author or of any co-author.

It is usually the case that PDFs found on publisher websites or supplied as proofs are the property of the publisher, which often owns the copyright and/or licenses their use. Even if the author retains copyright or permissions in the article, arXiv cannot accept PDFs that have been downloaded from a publisher's website unless arXiv has a blanket agreement with the publisher (it would be too costly in administrative time to track individual permissions).

Licenses granted are irrevocable

Authors should take care to upload an article *only if they are certain* that they will not later wish to publish it in a journal that prohibits prior distribution on an e-print server. *arXiv will not remove an announced article to comply with such a journal policy -- the license granted on submission is irrevocable*. However, granting rights for arXiv to distribute an article does not preclude later copyright assignment. Authors are thus free to publish submissions that already appear on arXiv. Authors may wish to inform the journal publisher that a prior non-exclusive license exists before transferring copyright or granting a publication license. Please check the policies of any potential publication venue before uploading to arXiv. (For the policy information of many publishers, see the SHERPA/RoMEO site.)

Copyright notices

If you have permission from a publisher to upload content to arXiv provided that you include a special copyright statement with the paper, the correct place for that statement is the first page of the text of the submission. Copyright notices should not be included in the separate metadata and will be removed.

If you have any additional questions about arXiv's copyright and licensing policies, please contact the arXiv administrators directly.

Contact

VITA

NAME Mano Vikash Janardhanan

EDUCATION

PhD, Mathematics, University of Illinois at Chicago,
Chicago, Illinois, 2019
MS, Mathematics, Indian Institute of Science Education and Research,
Thiruvananthapuram, 2014.
BS, Mathematics, Indian Institute of Science Education and Research,
Thiruvananthapuram, 2014.

PUBLICATIONS

Mohsen Aliabadi and Mano Vikash Janardhanan. "On matchable subsets in abelian groups and their linear analogues". arXiv:1808.01376

Bhaskar DasGupta, Mano Vikash Janardhanan and Farzaneh Yahyanejad. "Why did the shape of your network change? (On detecting network anomalies via non-local curvatures)". arXiv:1808.05676

Mohsen Aliabadi and Mano Vikash Janardhanan. "On local matching property in groups and vector spaces". In: Australasian Journal of Combinatorics 70 (Jan. 2017).

Mano Vikash Janardhanan. "Graph Verification using a Betweenness Oracle". In: Algorithmic Learning Theory (2017).

Yi Huang, Mano Vikash Janardhanan and Lev Reyzin. "Network Construction with Connectivity Constraints". In: FSTTCS (2017)

Mano Vikash Janardhanan and Sujith Vijay. "Ramsey Functions for Generalized Progressions". In: Integers 15 (2015).

Mano Vikash Janardhanan. "Topics in Ramsey Theory". In: arXiv:1404.7348 (MS Thesis)