Geometric Algorithms for Metric and Graph Learning

by

Neshat Mohammadi
M.Sc., from University of Illinois at Chicago, 2017

DISSERTATION

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2022

Chicago, Illinois

Defense Committee:
Prof. Lev Reyzin, Chair and Advisor
Prof. Anastasios Sidiropoulos, Advisor
Prof. Xinhua Zhang
Prof. Xiaorui Sun
Dr. Vahid Noroozi, NVIDIA

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER PAGE

LIST OF FIGURES

LIST OF ABBREVIATIONS

UIC	University of Illinois at Chicago
NP	Non-deterministic Polynomial-time
LLOC	Learning Lines with Ordinal Constraints
WLLOC	Weighted instance of Learning Lines with Ordinal Constraints
MQ	Membership Queries
EQ	Equivalence Counting Queries
EC	Edge Counting Queries
BEC	Bipartite Edge Counting Queries
LMNN	Large-Margin Nearest Neighbors

ACKNOWLEDGMENTS

I want to express my gratitude to my advisors, Lev Reyzin and Anastasios (Tasos) Sidiropoulos for their kindness and support throughout my Ph.D. I'm fortunate that I had the opportunity to work with two of the brightest mind in TCS during my time at UIC.

Ph.D. is a Marathon that tests your endurance to the core. I want to thank my family and friends for their endless support without whom I couldn't get to the finish line. I also want to thank my co-authors for being great collaborators and all the useful discussions that we had.

I'm so grateful for having my best friend Mona who supported me through highs and lows and my dearest cousin Tara who hugged me with her words from miles away to motivate me to keep going when I was about to give up.

This dissertation is dedicated to my sister Fatemeh, the joy of my life, my greatest supporter and biggest critic who helped me to see my blind sides and to my parents for their immense love and encouragement to pursue my dreams and for all the sacrifices that they made to pave my way. I cannot put my gratitude and love for them into words.

NM

SUMMARY

Graph and metric space representations are currently popular due to their multiple applications in modeling complex data sets from social networks to human genome sequences. In this dissertation, we examine various problems on metrics and graphs through the lens of geometric algorithms. The work in this dissertation can be categorized into three parts:

Metric learning: In a metric space representation, each element is considered a point, and the similarity or dissimilarity between two objects is encoded by their pairwise distance. Our goal is to find a unique mapping from the initial metric to the host metric. The problem of learning a target underlying distance function can be reformulated as an optimization problem, where the objective function captures the extent to which a solution satisfies the input constraints.

In particular, we explore the problem of learning lines with ordinal constraints and propose a solution by leveraging the geometric properties of metric spaces.

Stability of metric data: Using Bilu-Linial stability of metrics is a relatively new perspective that can expose interesting structural properties that can motivate a re-exploration of some of the famous NP-hard problems. Bilu-Linial stability introduced a new point of view on complexity, where instead of focusing on worst-case elements of a problem, we instead focus on particular classes of inputs. We study the Steiner tree problem, one of the famous NP problems, by assuming Bilu-Linial stability, we give strong geometric structural properties that need to be satisfied by stable instances. Then by strengthening and using these geometric properties we show that 1.59-stable instances of Euclidean Steiner trees are polynomial-time solvable.

SUMMARY (Continued)

Graph learning and verification: In a graph learning problem, the goal is to learn or verify a hidden graph or its properties by having query access to the graphs. We study various queries (edge detection, edge counting), for both directed and undirected graphs but we focus mainly on bipartite edge counting queries and on undirected graphs. We give a randomized algorithm for learning graphs using $O(m \log n)$ bipartite edge counting queries as well as a randomized constant-query graph verification algorithm.

CHAPTER 1

INTRODUCTION

Rapid growth of technology and social media has created massive data sets that are difficult to analyze or store efficiently. These issues shed light on the importance of using structured data sets to design efficient algorithms. The growing demand on using graph and metric spaces on modeling the big data highlighted the wide applications of graph and metric spaces in modeling big data. Geometric methods offer various tools for the analysis of complex and convoluted data sets. Some of the most famous examples of efficient algorithms on metric spaces are nearest-neighbor search, clustering, and dimensionality reduction.

My research aims to tackle computational obstacles, especially in metric learning problems, in order to burnish the theoretical foundations for designing computationally efficient algorithms designed for real-world problems. The ultimate goal is to bridge the gap between theory and applications to form a conceptual ground for less complex learning algorithms in practice, especially on metric and graph data. My work can be split into three main categories: learning metrics, data stability on metrics, and graph learning. This section provides a brief introduction to each of these areas.

My research identifies and addresses computational obstacles to establish a theoretical foundation to then design computationally efficient principled algorithms that can help to achieve better predictive models for real- life problems, especially, in the medical context. The goal is to bridge the gap between theoretical and applied computer science and use the integration of

theoretical and applied methods to form a conceptual ground for less complex learning algorithms in practice to unravel the ground truth. In most of the medical studies, data sets consist of a set of objects or patients' information which contains some information that represents the similarity or dissimilarity of certain pairs. For instance, the distribution pattern of a contagious disease and it's possible effect on society for epidemiology studies that can help to predict the distribution and effects on the global scale, DNA sequences for detecting inheritance disease, and so on. Metric spaces can be used as an authentic way to capture and demonstrate latent features in data sets. In metric space representations, each element considered a point, and the similarity or dissimilarity between two objects is encoded by their distance. To uniquely encodes the ground truth, this framework enables us to find a distance function. This metric interpretation of data sets has been promising in practice and has formed the foundation of algorithmic methods and ideas, such as clustering, dimensional reduction, nearest-neighbor search, etc.

One important aspect within metric learning is on discovering and developing methods for unraveling latent metric spaces that agree with a given set of observations. More precisely, the problem of learning the distance function can be cast as an optimization problem, where the objective function quantifies the extent to which the solution satisfies the input constraints. Almost all of the metric learning methods that are currently employed in the real world suffer from a lack of provable guarantees. So, the main goal of my research is to provide a theoretical justification for the success of current practical methods. A shortage of theoretical guarantees opens the door to heuristic solutions for solving problems. The primary issue with these heuristic methods is that we don't have any guarantee that these solutions would also work when applied

to different problems. Addressing the new problems in the era of big data, due to a limitation of time and money, requires more efficient methods that trying all the available heuristics. This sheds light on the importance of developing new algorithms for computing such a unique metric representation of data sets with a guarantee for converging to the solution.

1.1 An overview and motivation

In this dissertation, we address some on the famous problems in the area of metric and graph learning and further discuss what motivated us to approach these problems.

1.1.1 Metric learning

In a metric space representation, each element is considered a point, and the similarity or dissimilarity between two objects is encoded by their distance. This framework tries to uniquely encode the ground truth by finding a distance function. Such an interpretation of data sets has been promising in practice and has based the foundation of algorithmic methods and ideas, such as clustering, dimensionality reduction, and nearest-neighbor search.

One major focus of study within the field of metric learning is to discover and develop methods for unraveling latent metric spaces that agree with a given set of observations. More precisely, the problem of learning a target underlying distance function can be cast as an optimization problem, where the objective function quantifies the extent to which a solution satisfies the input constraints. Almost all of the metric learning methods that currently work in practice suffer from a lack of provable guarantees.

The main goal of my research is, therefore, to provide a theoretical justification for the success of the methods currently used in practice. My results on this line of research have been

published in SoCG 2019 [3] and APPROX 2020 [4] and in we preprint on arXiv [5]and "Learning Lines with Ordinal Constraints" appears in chapter 3 of this dissertation.

1.1.2 Stability

Various types of errors (such as, missing values, wrong measurements and etc.) can occur during the data collection process. These errors called noise and that noise can be seen as a "perturbation" of the metric. Most of the time, especially while collecting real data, noise encountering is unavoidable. In some sensitive data sets even a small amount of additive noise can change the optimal solution. So what if we study an instance of the problem where the noise doesn't change the optimal solution? In other words what structural properties can be provided by noise resilient instances of a problem?

This point of view was the initial motivation for a new line of research by Bilu and Linial to study the perturbation resilience in metrics. Perturbation resilience, also called Bilu-Linial stability, is one of the assumptions to help bypass the NP-hardness of optimizing various objective functions to obtain polynomial-time algorithms.

It has been shown assuming a data set satisfies some notion of resilience, it starts to display strong structural properties that admit otherwise NP-hard problems to become polynomial-time solvable. Using Bilu-Linial stability and metric properties is a relatively new perspective that can expose interesting structural properties that can motivate a re-exploration of some of the famous NP-hard problems. Bilu-Linial stability inducted a new point of view on complexity, where instead of focusing on the problem, we focus on the data. The results of my work on

stability on the metric Steiner tree has been published in CCCG 2022 [6] and can be found in chapter 4 of this dissertation.

1.1.3 Graph learning

In a graph learning problem, a hidden graph $G = (V, E)$ is known to belong to a given family $\mathcal{G}$ of labeled graphs on the vertex set $[n] := 1, 2, \ldots, n$ and we wish to identify G by some query access to the graph. The goal is to revisit some of graph-learning problems via these queries. One of the most common queries is Edge Counting (EC) queries where each query tells whether a subset of $[n]$ induces an edge of G. The main motivation of this problem is its application in DNA physical mapping. This is the main concentration of my current work and some of the initial results along with my perspective for the future of my research had been discussed in Section 6.

1.2 Our contributions

The main contribution on this dissertation is providing a new geometric insight to some of the well known problems.

We start by presenting in chapter 3 an approximation algorithm for the dense case of of finding a mapping f from a set of points into the real line problem, under ordinal triple constraints. An ordinal constraint for a triple of points (u, v, w) asserts that $|f(u) - f(v)| < |f(u) - f(w)|$. Given an instance that admits a solution that satisfies $(1 - \varepsilon)$-fraction of all constraints, our algorithm computes a solution that satisfies $(1 - O(\varepsilon^{1/8}))$-fraction of all constraints, in time $O(n^7) + (1/\varepsilon)^{O(1/\varepsilon^{1/8})}n$. [4]

Later in this dissertation we study γ-stable instances the Steiner tree problem under Bilu-Linial stability. We show strong geometric structural properties that need to be satisfied by stable instances. Then we use these geometric properties to prove that 1.59-stable instances of Euclidean Steiner trees are polynomial-time solvable. We also provide a connection between certain approximation algorithms and Bilu-Linial stability for Steiner trees. [6]

Finally, we study the problem of learning and verifying hidden graphs by having query access to the graphs to be learned, possibly from a restricted family. In this work, we focus on undirected unweighted graphs, which can also be thought of as giving the shortest path metric on the vertices in the graph (which can be considered to be the data points). This special case is often referred to as the "topological" case, as we are most concerned with the topology of the graph structure. Among our results, we show that there exists an efficient randomized algorithm that can learn any graph with $O(m \log n)$ queries. We later give another, this time inefficient, algorithm for learning any graphs that require $O(\log |\mathcal{G}|)$ queries.

CHAPTER 2

PRELIMINARIES

2.1 Preliminaries for Metric Learning

Metric representation of data has a wide range of applications because of its unique property in preserving the underlying geometry of data. In metric representation the similarity of elements is coded as the distance between them.

A metric space is a pair (X, p) , where X is a set and $\rho : X \times X \to [0, \infty)$ is a metric, satisfying symmetry, identity of indiscernibles and most importantly triangle inequality as it presented in the following axioms: [7]

1. $\rho(x, y) = \rho(y, x)$
2. $\rho(x, y) = 0 \leftrightarrow x = y$
3. $\rho(x, y) + \rho(y, z) \geqslant \rho(x, z)$.

In metric learning, the goal is to find a mapping from the initial metric to a host metric with respect to some constraints that preserve the geometry of the data. A main concern of the metric learning field, in particular the problems that we study in this dissertation, is to tune the distance function to a specific task with respect to some constraints. It has been shown to be useful when used in conjunction with models that rely on the similarity of the data for categorizing purposes such as nearest-neighbor methods and other similarity-based models.

Embedding a metric space into a geometrical space is a well known problem in computational geometry.

Definition 2.1.1 (Metric embedding of a metric space [8]). *A metric embedding of a metric space $M = (X, \rho)$ into a host space $M = (Y, \rho')$ is a mapping, such that $f : X \to Y$.*

Definition 2.1.2 (Distortion of an embedding). *[8] The distortion of an embedding f is defined the minimum c, such that there exists $r > 0$, with*

$$\rho(x, y) \leqslant r \cdot \rho'(f(x), f(y)) \leqslant c \cdot \rho(x, y).$$

The distortion is a measure that indicates the extend to which an embedding preserves the geometry of the initial metric space.

The above definition conveys the following properties,

- if the distortion $c = 1$ then the mapping is an isometry;
- the distortion of the concatenation of two embeddings is equal to the product of their respective distortions.

Now that we recall the basic definitions in metric learning literature we can move on to the following important lemmas and theorems that motivated us to explore this direction an formed the foundation of this work.

Theorem 2.1.3 (Johnson-Lindenstrauss flattening lemma). *Given, X a set of an n-points in a Euclidean space (i.e., $X \subset l_2$), and let $\epsilon \in (0, 1]$ be given. Then there exists a $(1+\epsilon)$-embedding of X into l_2^k, where $k = O(\epsilon^{-2} \log n)$. [7]*

In his book, Matousek [7] points out that the Johnson-Lindenstrauss flattening theorem lets us change any problem over a metric on n points to another problem by changing the metric from l_2 to $l_2^{O(\log n)}$, so long as we are willing to incur a small distortion in the distances. This means that we can store only $O(n \log n)$ bits (instead of $O(n^2)$) and still approximately reconstruct the n^2 pairwise distances.

Success in metric learning can be defined in various ways. For example, success can be defined by minimizing the number of violated constraints or designing a low distortion embedding. The main focus of this dissertation is on computing low-distortion metric embeddings for line metric spaces which can be formulate as finding a mapping of maximum accuracy. Therefore, we define the accuracy as follow:

Definition 2.1.4 (Accuracy in metric learning). *Given a set of constraints we define the accuracy of a mapping to be the fraction of all constraints that are satisfied.*

For example, success can be defined minimizing the number of violated constraints or designing a low distortion embedding. The main focus of this dissertation is on computing low-distortion metric embeddings for line metric spaces.

2.1.1 Well-known metric learning algorithms

Large-Margin Nearest Neighbors (LMNN) [1] is one of the most famous algorithms in the field of metric learning. The high-level intuition of the goal of LMNN is that given a new sample

of data it should share the same label as its nearest neighbors while being far from the data points with different labels. The relative distance of having the same label as nearest neighbors and different labels with far data point can be considered as a relative distance that induces margins, therefore, it is called Large-Margin Nearest Neighbors. LMNN aims to minimize the number of violated constraints with respect to the relative distance constraints. The following figure provides a better visualization on the intuition behind LMNN:

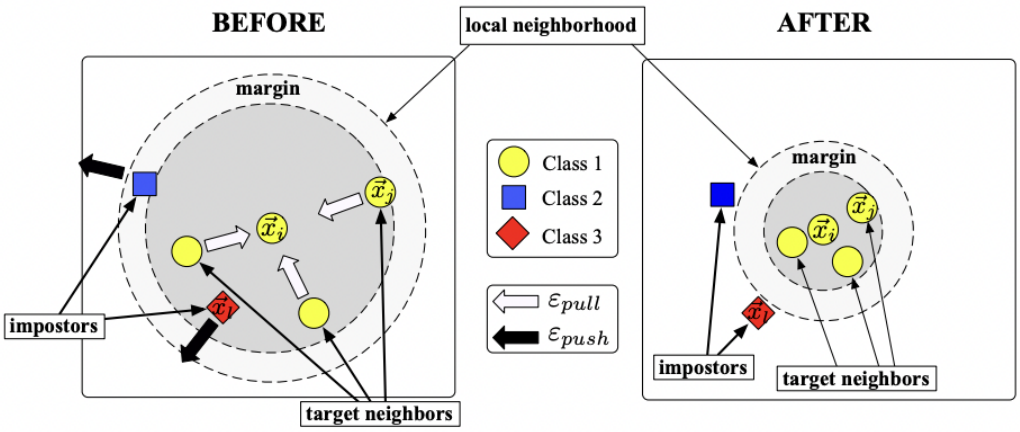


Figure 1. Visualization of the intuition behind LMNN design. [1]

2.1.2 Unsupervised vs weakly supervised metric learning

Weakly supervision is a special case where noisier sources of supervision are used to generate larger training sets faster rather than using the traditional methods to create training data (i.e. labeling samples manually).

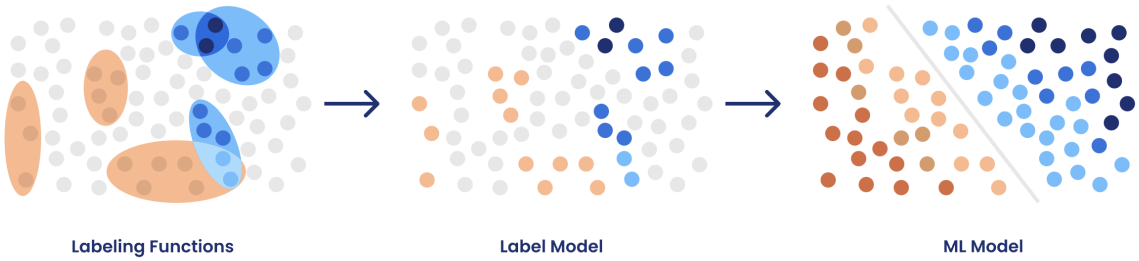


Figure 2. Weak Supervision using Label Model to train ML Model. [2]

In unsupervised metric learning we don't use any notion of supervision for creating the training data. The most famous example of these line of work is dimensionallity reduction. In this work we mainly focus on the weakly supervised instances of metric learning problems. Here we studied the weakly supervised ML with respect to comparative constraints. There are another set of famous constraints known as contrastive constraints. In [3] authors study another instance of this problem with respect to contrastive constraints.

2.2 Preliminaries for stability

Definition 2.2.1 (Bilu-Linial stability [9][1]). *Let P be a problem over a metric (X, ρ) with a unique optimal solution* OPT*. Let P' be a γ-perturbation of the problem where the metric is now given by (X, ρ') under the condition that,*

$$\forall x_1, x_2 \in \rho(x_1, x_2) \leqslant \rho'(x_1, x_2) \leqslant \gamma\rho(x_1, x_2)$$

for $\gamma \geqslant 1$ with solution OPT$'$*. We say that P is γ-stable if* OPT $=$ OPT$'$ *for any γ-perturbation.*

The main concern of computational complexity theory is to study worst-case of computational problems. A problem is said to be NP-hard (non-deterministic polynomial-time) if every problem in NP reduces to it in polynomial time. We do not expect there to be polynomial-time algorithms for solving all instances of these problems. These problems are not only interesting from a theoretical perspective but also because of their application in formulating certain real-world tasks. In this notion of stability, we are not interested in all instances of these problems, but only the stable ones, which sometimes correspond to those that we care about solving in reality.

The goal of this line of research is to study instances that have complexity lower than the worst case, but are not trivial. [9]

---

[1] Later in this dissertation, we also use a different notion of stability in Definition 3.4.5. We note that these two notions are distinct.

Proposition 2.2.2 ([9]). *"Let $\gamma > 1$. A weighted graph G graph with maximum cut $(S, \bar{S})$ is γ-stable (with respect to Max-Cut) if, for every vertex set $T \neq S, \bar{S}$,"*

$$w(E(S, \bar{S}) \backslash E(T, \bar{T})) > \gamma \cdot w(E(T, \bar{T}) \backslash E(S, \bar{S}))$$

The above preposition shows how γ-stable instances of graphs can be generated which provides a better view of an interesting aspect of γ-stable instances which is its obvious scaleability. According to Bilu-Linial [9] scaleability is implied if perturbation of all weights in a graph by a constant factor should not a ect its stability.

2.3 Preliminaries for learning with queries

The query learning of graphs has received ample recognition lately since it can be applied into many different contexts. In particular, query learning can be used in biology and evolutionary tree reconstruction, where the the genetic distance between two species can be measured with the goal of placing all the species onto one tree. [10] [11]. In this section we provide a brief overview of some of the most popular queries for learning graphs.

Learning with queries can be modeled as interaction between a teacher or an oracle and a student or an active learner. In this model, the learner ask questions in order to learn a target hypothesis c^* by using the following queries that an an oracle or a teacher responds:

- Membership Queries (MQ): the student or active learner gives an example x, and the teacher provides its label $c^*(x)$;

- Equivalence Queries (EQ): the learner proposes a hypothesis h and the teacher checks if h is equal to the target concept; if not, teachers provides a counter-example x such that $c^*(x) \neq h(x)$.

Indeed there is a relationship between query learning and the standard PAC-learning model [12].

Theorem 2.3.1. *"Let $\mathcal{C}$ be a concept class that is e ciently exactly learnable with membership and equivalence queries, then $\mathcal{C}$ is* PAC*-learnable using membership queries. " [13]*

Although, the main concentration of the work that is presented in this dissertation is on learning undirected unwighted graphs via BEC queries, we give an overview on other popular queries that inspired this work and can be useful to explore the future direction of this work.

Definition 2.3.2. *(Edge detection query* (ED)*) Given a set of vertices V, the query* ED(S) *checks if there is an edge between any two vertices in $S \subseteq V$.* ED *queries are well known for their applications in genome sequencing and studied in [14–16].*

Definition 2.3.3. *(Edge counting query* (EC)*) Given a set of vertices V,* EC *outputs the count of edges in the sub-graph induced by $S \subseteq V$. This query has extensively been used in bio-informatics and was studied in [17, 18].*

Definition 2.3.4. *(Shortest path query* (SP)*) Given a graph $G = (V, E)$, the query $SP(u, v)$ outputs the length of the shortest path in G between two vertices; if no such a path exists, it outputs ∞. The main application of this query is in evolutionary tree literature. [10, 19, 20]*

We also use the master theorem for computing the complexity of `EdgeLearn` algorithm. We repeat the definition of master theorem as a reminder for our audience.

Theorem 2.3.5 (Master theorem [21]). *Let $a > 0$ be an integer, $b > 1$ be a real number and c be a positive real number and d a non-negative real number. Given a recurrence of the following form*

$$T(n) = \begin{cases} aT(\frac{n}{b}) + n^c & n > 1 \\ \\ d & n = 1 \end{cases}$$

then for n a power of b,

1. *if* $\log_b^a < c$, $T(n) = \theta(n^c)$
2. *if* $\log_b^a = c$, $T(n) = \theta(n^c \log n)$
3. *if* $\log_b^a > c$, $T(n) = \theta(n^{\log_b^a})$

CHAPTER 3

LEARNING LINES WITH ORDINAL CONSTRAINTS

This chapter is based upon the paper "Learning Lines with Ordinal Constraints" authored by B. Fan, D. Ihara, N. Mohammadi, F. Sgherzi, A. Sidiropoulos, M. Valizadeh, published in the International Conference on Approximation Algorithms for Combinatorial Optimization Problems in 2020 (APPROX 2020). [4].

Some parts of this work have also appeared in Francesco Sgherzi's M.S. thesis.

My main contribution to this work was proving the embedding is near optimal by giving an upper bound for the number of brittle triples which is presented in section 3.5.

3.1 Introduction

Geometric methods provide several tools for the analysis of complicated data sets, such as nearest-neighbor search, clustering, and dimensionality reduction. The key abstraction is to encode a set of objects by mapping each object to a point in some metric space, such that the distance between points quantifies the pairwise dissimilarity between the corresponding objects. The success of this paradigm crucially depends on the metrical representation used to encode the data. Motivated by this fact, metric learning aims at developing methods for discovering an underlying metric space from proximity information (we refer the reader to [22, 23] for a detailed exposition). There are several different formulations of the metric learning problem that have been considered in the literature. Here, we focus on the popular case of *ordinal* constraints.

In this case, the input consists of a set of points $X = [n]$, together with a set $\mathcal{T}$ of ordered triples (u, v, w) of points, representing the fact that u is *more similar* to v than to w. The goal is to find a mapping $f : X \to Y$, for some host metric space (Y, ρ), such that for all $(u, v, w) \in \mathcal{T}$, we have

$$\rho(f(u), f(v)) < \rho(f(u), f(w)). \tag{3.1}$$

In general, there might be no mapping f that satisfies all constraints of the form (Equation 3.1), so we are interested in the algorithmic problem of computing a mapping that minimizes the fraction of violated constraints. We focus on the case where the host space is the real line, so the objective can be formulated as computing a mapping $f : [n] \to \mathbb{R}$, where for each $(u, v, w) \in \mathcal{T}$ we have the constraint

$$|f(u) - f(v)| < |f(u) - f(w)|. \tag{3.2}$$

We refer to this problem as Line Learning with Ordinal Constraints (LLOC).

3.1.1 Our contribution

We present an approximation algorithm for learning a line metric space under ordinal constraints, for the case of dense instances. Here, the density condition means that all ordinal information is given, i.e. for any distinct $u, v, w \in [n]$, we have either $(u, v, w) \in \mathcal{T}$, or $(u, w, v) \in \mathcal{T}$. Our main result is summarized in the following.

Theorem 3.1.1. *There exists an algorithm that given an instance of LLOC that admits a solution satisfying $(1-\varepsilon)$-fraction of all constraints, outputs a solution that satisfies $(1-O(\varepsilon^{1/8}))$-fraction of all constraints, in time $O(n^7) + (1/\varepsilon)^{O(1/\varepsilon^{1/8})}n$.*

Brief overview of our approach. The main idea used to obtain Theorem 3.1.1 is to first compute an ordering that is close to the ordering of the points in the optimal solution. This is done by "guessing" a point p^* that lies within the few left-most points in an optimal solution, and such that p^* is not involved in many violated constraints. We show that the ordinal constraints involving p^* can be used to order the points by first solving an instance of the Minimum Feedback Arc Set problem on a tournament, and then computing a topological ordering of the remaining acyclic graph. We use this ordering to partition the points into "buckets", and we show that for almost all buckets, almost all their points must be mapped inside an interval that does not contain many other points. This property allows us to define a smaller instance of the problem by contracting each bucket into a single point. This new smaller instance can be solved exactly, and its solution can be pulled back to the original problem.

3.1.2 Related work

Metric learning. Another popular formulation of the metric learning problem uses *contrastive* constraints. In this case, the input consists of a set of points $X = [n]$, together with sets $\mathcal{S}, \mathcal{D} \subseteq \binom{X}{2}$, where $\mathcal{S}$ contains pairs labeled as *similar*, and $\mathcal{D}$ contains pairs labeled as *dissimilar*. The goal is to find a mapping $f : X \to Y$, for some host metric space (Y, ρ), such that for all $\{u, v\} \in \mathcal{S}$,

$$\rho(f(u), f(v)) \leqslant \ell,$$

and for all $\{u, v\} \in \mathcal{D}$,

$$\rho(f(u), f(v)) \geqslant h,$$

for some given threshold values $\ell, h > 0$. This problem is easily seen to be a generalization of Correlation Clustering. It has been studied for the case dense instances, when the host metric space is either Euclidean or a tree [3]. The main result of [3] is a FPTAS for the case where there exists a mapping that satisfies all constraints, that is allowed to violate the constraints by a small multiplicative factor which is referred to as *contrastive distortion*. In contrast, in the present work, we do not introduce any distortion, and we do not need to assume that there exists a mapping satisfying all the constraints.

We also note that the case of arbitrary instances (i.e., not necessarily dense) under contrastive constraints has been studied for the setting of learning Mahalanobis metric spaces (i.e., when X is a set of points in d-dimensional Euclidean space, and f is required to be linear) [24]. This version of the problem is related to the theory of LP-type problems.

Embedding into the line. The problem of computing a geometric representation of a data set into the real line has been studied extensively in various forms. This is arguably the simplest instance of dimensionality reduction, which is also a prototypical unsupervised metric learning task. Various objectives have been studied, including multiplicative [25–30], additive [31], and average [32, 33] distortion. We refer the read to [34] for a detailed exposition. A related notion is ordinal embeddings, where one seeks to obtain mappings that approximately preserve the relative ordering of pairwise distances [35, 36]. We remark that a key difference between these works and our result is that they seek to minimize the *ordinal distortion*, which is a multiplicative factor of

violation of the ordinal constraints, while we are interested in minimizing the number of violated ordinal constraints (without introducing ordinal distortion).

Betweenness. In the Betweenness problem we are given some set $X = [n]$ and a set $\mathcal{T}$ of ordered triples $(a, b, c) \in [n]^3$. The goal is to find a bijection $g : [n] \to [n]$ such that for any $(a, b, c) \in \mathcal{T}$, $g(b)$ appears between $g(a)$ and $g(c)$. This problem has been studied extensively in the literature. It is known to be MAXSNP-hard [37] (see also [38]), and remains hard to approximate even on dense instances [39]. The case of tournaments has been shown to admit a PTAS [40], while the best approximation algorithm for general instances is the 1/3-approximation obtained by taking a uniformly random ordering, assuming the Unique Games conjecture [41] (see also [42]).

The Betweenness problem is conceptually similar to the Line Learning with Ordinal Constraints problem studied here. However, as we now explain, the two problems have some important differences. A first difference is that the ordinal constraint (Equation 3.2) does not imply any ordering constraint[1]. A second difference is that the solution space to the Line Learning with Ordinal Constraints problem that we study is larger. In other words, the ordering of the points is not always enough to recover a nearly-optimal constraint. For example, consider the instance on $X = \{0, 2, 4, \ldots, 2k, 2k + 1, \ldots, 3k\}$, with all constraints $(u, v, w) \in X^3$, such that $|u - v| < |u - w|$. Clearly, setting f to be the identity results in a solution that satisfies all

[1] For example, the constraint (u, v, w) is satisfied by both solutions $f(u) = 1, f(v) = 2, f(w) = 3$, and $f(u) = 2, f(v) = 1, f(w) = 4$, however the former solution implies the ordering $f(u) < f(v) < f(w)$, while the latter implies $f(v) < f(u) < f(w)$

constraints. However, just the ordering of the points in f is not enough to obtain a good solution: setting $g(u_i) = i$, where $g(u_1) < g(u_2) < \ldots < g(u_n)$ results in a solution g that violates a constant fraction of all constraints.

3.1.3 Organization

The rest of the paper is organized as follows. Section 3.2 presents, as a warm up, an exact polynomial-time algorithm for the case where there exists a solution that satisfies all constraints. Section 3.3 presents the algorithm for the general case. Section 3.4 presents the analysis. Section 3.5 gives the proof of a technical Lemma which is used in the proof of the main result.

3.2 Warm up: An exact algorithm with no violations

We now describe an exact polynomial-time algorithm for the case where there exists an optimal solution that satisfies all constraints. This algorithm is significantly simpler than the one used to prove our main result. However, it illustrates the main idea of using the constraints involving some point p to deduce an ordering of all points, and then using this ordering to obtain an embedding into the line. The algorithm is summarized in the following.

Theorem 3.2.1. *Let ε^* be the fraction of violated constraints, then there exists a polynomial-time algorithm which given an instance $([n], \mathcal{T})$ of the LLOC problem, either computes a mapping $f : [n] \to \mathbb{R}$ that satisfies all the constraints, or correctly decides that no such mapping exists.*

Proof. Fix some optimal mapping $f^* : [n] \to \mathbb{R}$, that satisfies all constraints in $\mathcal{T}$. We guess $p = \arg\min_{x \in [n]} f^*(x)$. For all $i, j \in [n]$, let $d_{i,j} = |f^*(x_j) - f^*(x_i)|$. We first determine the ordering of all the points on the real line, and then we compute the mapping using their distance constraints and solving some LP.

Suppose that $[n] = \{x_1, \ldots, x_n\}$, such that

$$f^*(p) = f^*(x_1) < f^*(x_2) < \ldots < f^*(x_n).$$

Since $\varepsilon^* = 0$, it follows that for all $i < j \in [n]$, we have $d_{1,i} < d_{1,j}$, and $(1, i, j) \in \mathcal{T}$. Therefore, for any $q, q' \in [n]$, we can decide whether $f^*(q) < f^*(q')$ or $f^*(q') < f^*(q)$ based on whether $(p, q, q') \in \mathcal{T}$ or $(p, q', q) \in \mathcal{T}$. Therefore, we can compute the ordering $x_1, \ldots, x_n$ of $[n]$ by running a sorting algorithm using pairwise comparisons.

We now compute a mapping using an LP. For any $i < j \in \{1, \ldots, n\}$, we have $|f^*(x_i) - f^*(x_j)| = \sum_{t=i}^{j-1} d_{t,t+1}$. Therefore for each $(x_i, x_j, x_k) \in \mathcal{T}$, the constraint $|f^*(x_i) - f^*(x_j)| < |f^*(x_i) - f^*(x_k)|$ can be written as $\sum_{t=i}^{j-1} d_{t,t+1} < \sum_{t=i}^{k-1} d_{t,t+1}$. Thus computing the desired mapping f can be done by computing a feasible solution to the following LP:

$$d_{i,j} \geqslant 0 \text{ for all } i < j \in [n]$$
$$\sum_{t=i}^{j-1} d_{t,t+1} < \sum_{t=i}^{k-1} d_{t,t+1} \text{ for all } (x_i, x_j, x_k) \in \mathcal{T}$$

This concludes the proof. □

3.3 The algorithm for the general case

In this Section we present the algorithm for the general case of the problem. The algorithm uses as a subroutine an exact algorithm for a generalized weighted version of the problem. This

exact algorithm is used on small instances that are constructed via a process which we refer to as a *retraction*.

3.3.1 Retractions

We now define a weighted version of the metric learning problem, where each constraint is associated with some weight, and the goal is to maximize the total weight of all satisfied constraints. Formally, an input to the Weighted Line Learning with Ordinal Constraints (WLLOC) problem is defined by a tuple $([b], \mathcal{T}, w)$, where $b \in \mathbb{N}$, and $\mathcal{T}$ are as before, and $w : \mathcal{T} \to \mathbb{R}$ is a weight function. The goal is to find a solution $f : [b] \to [0, 1]$ that minimizes the total weight of violated constraints.

Theorem 3.3.1. *There exists an exact algorithm for the WLLOC problem with running time* $O(n^{3n})$.

Proof. We identify the space of possible solutions with $[0, 1]^n$, by mapping each solution $f : [b] \to [0, 1]$ to the vector $x_f = (f(1), \ldots, f(n)) \in [0, 1]^n$. For any $(i, j, k) \in \mathcal{T}$, we have the constraint

$$|f(i) - f(j)| < |f(i) - f(k)|.$$

The feasible region for this constraint is thus defined as a union of certain cells in an arrangement $A_{(i,j,k)}$ of a constant number of open halfspaces in $\mathbb{R}^n$. Let A be the arrangement obtained as the union of all halfspaces for all $(i, j, k) \in \mathcal{T}$. It is known that any arrangement of a halfspaces in $\mathbb{R}^b$ has complexity $O(a^b)$ (see [43] and references therein), and thus A has complexity $O(|\mathcal{T}|^n) = O(n^{3n})$. By enumerating all the cells in this arrangement, we find a solution that

satisfies a set of constraints of maximum total weight, which results in an algorithm with running time $O(n^{3n})$. □

As mentioned earlier, the exact algorithm from Theorem 3.3.1 will be used as a subroutine on smaller instances. The following Definition describes a process for mapping large unweighted instances to smaller weighted ones.

Definition 3.3.2 (Retraction). *Given an instance $\phi = ([n], \mathcal{T})$ of the LLOC problem, and some partition $\mathcal{B} = \{B_1, \ldots, B_b\}$ of $[n]$, we define the $\mathcal{B}$-retraction of ϕ to be the instance $\phi' = ([b], \mathcal{T}', w)$ of the WLLOC problem where for any $(i, j, k) \in \mathcal{T}'$, we have*

$$w((i, j, k)) = |\mathcal{T} \cap (B_i \times B_k \times B_j)| .$$

3.3.2 The algorithm

The last ingredient we need is an approximation algorithm for the Minimum Feedback Arc Set problem on tournaments, which is summarized in the following.

Theorem 3.3.3 (Kenyon-Mathieu & Schudy [44]). *There exists a randomized algorithm for the Minimum Feedback Arc Set problem on weighted tournaments. Given $\epsilon > 0$, it outputs a solution with expected cost at most $(1 + \epsilon)$OPT. The expected running time is $O(1/\varepsilon)n^6 + 2^{\tilde{O}(1/\varepsilon)}n^2 + 2^{2^{\tilde{O}(1/\varepsilon)}}n$.*

We are now ready to describe the general algorithm. Let $\mathcal{T}_n$ denote the set of all ordered triples of distinct elements in $[n]$. Recall that the input consists of a set $\mathcal{T} \subseteq \mathcal{T}_n$, such that

for any set of distinct $i, j, k \in [n]$, we have that exactly one of the triples (i, j, k) and (i, k, j) is contained in $\mathcal{T}$.

The algorithm proceeds in the following steps:

Step 1: Exhaustively computing a left-most point. Iterate Steps 2–5 for all values $p \in [n]$.

Step 2: Cycle removal. Construct a tournament $G^{(p)} = ([n], A^{(p)})$, where

$$A^{(p)} = \{(i, j) : (p, i, j) \in \mathcal{T}\}.$$

Compute an $O(1)$-approximate minimum feedback arc set, $F^{(p)} \subset A^{(p)}$, in $G^{(p)}$, using the algorithm in Theorem 3.3.3.

Step 3: Ordering. Compute a topological ordering $z_1^{(p)}, \ldots, z_n^{(p)}$ of $G^{(p)} \backslash F^{(p)}$.

Step 4: Retraction. Let $b = O(\varepsilon^{-1/8})$. For any $i \in [b]$, let

$$\mathcal{B}_i^{(p)} = \bigcup_{j=(i-1)n/b+1}^{in/b} \{z_j^{(p)}\}.$$

Let $\psi^{(p)}$ be the $\mathcal{B}^{(p)}$-retraction of $\phi^{(p)}$.

Step 5: Extension. Using the algorithm from Theorem 3.3.1, we compute an optimal solution $g^{(p)} : [b] \to [0, 1]$ for the instance $\psi^{(p)}$ of WLLOC. We define $f^{(p)} : [n] \to [0, 1]$ by setting for any $i \in [n]$, $f^{(p)}(i) = g^{(p)}(j)$, where $j \in [b]$ such that $i \in B_j^{(p)}$. The algorithm outputs the solution $f^{(p)}$.

Step 6: Return the best. Return the best solution found among $f^{(1)}, \ldots, f^{(n)}$.

This completes the description of the algorithm.

3.4 Analysis of the algorithm

This Section presents the analysis of the algorithm, which is the proof of Theorem 3.1.1.

For the remainder of the analysis, let us fix some optimal solution $f_{\mathsf{OPT}} : [n] \to [0,1]$ for the instance $([n], \mathcal{T})$ of the LLOC problem. Fix a numbering $\{x_1, \ldots, x_n\} = [n]$, such that

$$f_{\mathsf{OPT}}(x_1) \leqslant f_{\mathsf{OPT}}(x_2) \leqslant \ldots \leqslant f_{\mathsf{OPT}}(x_n).$$

For any $f : [n] \to [0,1]$, for any $i \in [n]$, and for any $\alpha \in [0,1]$, we say that i is *α-good in f*, if at least α-fraction of the constraints of the form $(i,j,k) \in \mathcal{T}$ are satisfied; i.e.:

$$|\{(i,j,k) \in \mathcal{T} : |f(i) - f(j)| < |f(i) - f(k)|\}| \geqslant \alpha \binom{n-1}{2}.$$

We first argue that there exists a $(1 - \varepsilon^{1/2})$-good point that is close to the left-most point in the optimal solution:

Lemma 3.4.1. *There exists $i^* \in [2\varepsilon^{1/2} n]$, such that x_{i^*} is $(1 - \varepsilon^{1/2})$-good in f_{OPT}.*

Proof. Let ξ be the total number of constraints violated by f_{OPT}. We have $\xi \leqslant \varepsilon \cdot |\mathcal{T}| = \varepsilon n \binom{n-1}{2}$. Suppose that there exists no $i \in [2\varepsilon^{1/2} n]$ such that x_i is $(1 - \varepsilon^{1/2})$-good. Therefore every $i \in [2\varepsilon^{1/2} n]$ participates in at least $\varepsilon^{1/2} \binom{n-1}{2}$ violated constraints of the form (i,j,k), for some

$j, k \in [n]$. Thus the total number of violated constraints is at least $\xi \geqslant 2n\varepsilon\binom{n-1}{2}$, which is a contradiction, concluding the proof. □

For the remainder of this section, fix some $i^* \in [2\varepsilon^{1/2}n]$, such that x_{i^*} is $(1-\varepsilon^{1/2})$-good, as in Lemma 3.4.1. Let f' be the embedding obtained from f_{OPT} by exchanging the images of x_1 and x_{i^*}, that is for all $i \in [n]$,

$$f'(x_i) = \begin{cases} f_{\mathsf{OPT}}(x_{i^*}) & \text{if } i = 1 \\ f_{\mathsf{OPT}}(x_1) & \text{if } i = i^* \\ f_{\mathsf{OPT}}(x_i) & \text{otherwise} \end{cases}$$

We next show that f' is near-optimal.

Lemma 3.4.2. *The total number of violated constraints in f' is at most $(\varepsilon + O(1/n))n\binom{n-1}{2}$.*

Proof. Let $\mathcal{T}_1 \subseteq \mathcal{T}$ be the set of constraints that are violated in f' and in f_{OPT}. Let $\mathcal{T}_2 \subseteq \mathcal{T}$ be the set of constraints that are violated in f' but not in f_{OPT}. We have $|\mathcal{T}_1| \leqslant \varepsilon n\binom{n-1}{2}$. Since f_{OPT} and f' differ only on x_1 and x_{i^*}, it follows that every constraint $(i, j, k) \in \mathcal{T}_2$ must contain at least one of 1 and i^*. There are at most $6n^2$ such constraints. Thus $|\mathcal{T}_2| \leqslant 6n^2$. We conclude that the total number of constraints violated in f' is at most $|\mathcal{T}_1| + |\mathcal{T}_2| \leqslant (\varepsilon + O(1/n))n\binom{n-1}{2}$, which concludes the proof. □

The next Lemma shows that x_{i^*} remains $(1 - O(\varepsilon^{1/2}))$-good in f'.

Lemma 3.4.3. *We have that x_{i^*} is $(1 - O(\varepsilon^{1/2}))$-good in f'.*

Proof. Let $\gamma = (x_{i*}, j, k) \in \mathcal{T}$, and suppose that γ is satisfied in f_{OPT}. If

$$f_{\mathsf{OPT}}(x_{i*}) \leqslant f_{\mathsf{OPT}}(j) \leqslant f_{\mathsf{OPT}}(k),$$

then, since $f'(j) = f_{\mathsf{OPT}}(j)$, and $f'(k) = f_{\mathsf{OPT}}(k)$, it follows that

$$f'(x_{i*}) \leqslant f'(j) \leqslant f'(k),$$

and thus γ is also satisfied in f'.

Thus, the only possible constraints of the form $(x_{i*}, j, k) \in \mathcal{T}$, that are not violated in f_{OPT}, but are violated in f', must satisfy either $f_{\mathsf{OPT}}(j) < f_{\mathsf{OPT}}(x_{i*})$, or $f_{\mathsf{OPT}}(k) < f_{\mathsf{OPT}}(x_{i*})$. In other words, we must have $\{j, k\} \cap \{x_1, \ldots, x_{i*-1}\} \neq \varnothing$. Therefore, there are at most $2\varepsilon^{1/2}n^2$ such constraints. Since x_{i*} is $(1 - \varepsilon^{1/2})$-good in f_{OPT}, it follows that x_{i*} is $(1 - O(\varepsilon^{1/2}))$-good in f', which concludes the proof. □

Let $F' = \{(j, k) \in A^{(i^*)} : (x_{i*}, j, k) \in \mathcal{T}$ and f' violates $(x_{i*}, j, k)\}$. The next Lemma shows F' is a valid feedback arc set for $G^{(i^*)}$.

Lemma 3.4.4. *F' is a feedback arc set for $G^{(i^*)}$, with $|F'| \leqslant (O(\varepsilon^{1/2}))\binom{n-1}{2}$.*

Proof. By Lemma 3.4.3, x_{i*} is $(1 - O(\varepsilon^{1/2}))$-good, and thus $|F'| \leqslant (O(\varepsilon^{1/2}))\binom{n-1}{2}$. Thus, it suffices to show that F' is a feedback vertex set. For any $(j, k) \in A^{(i^*)} \backslash F'$, we have that (x_{i*}, j, k)

is satisfied in f'. Since x_{i*} is mapped to the left-most point in f', it follows that $f'(j) < f'(k)$. Thus,

$$x_{i*}, x_2, x_3, \ldots, x_{i*-1}, x_1, x_{i*+1}, x_{i*+2}, \ldots, x_n$$

is a topological ordering of $G^{(i*)} \backslash F'$, and thus F' is a feedback arc set, which concludes the proof. □

If the instance admits a solution with no violations, then it can be shown that the bucketing $\mathcal{B}^{(i*)}$ computed by the algorithm agrees with a partition of the optimal solution to contiguous disjoint intervals. In the following, we show that, in the general case, the bucketing is "close" to such a partition. First, we introduce a notion of "stability" which formalizes what it means for a bucket to be close to an optimal interval.

Definition 3.4.5 (Stability). *Let $i \in [b]$. We say that i is* stable *if there exists some interval $I \subset \mathbb{R}$, such that*

$$\left| I \cap f' \left(B_i^{(i*)} \right) \right| \geqslant (1 - \varepsilon^{1/8}) \cdot n/b,$$

and

$$\left| I \cap f' \left([n] \backslash B_i^{(i*)} \right) \right| \leqslant \varepsilon^{1/8} \cdot n/b,$$

We also say that i is I-stable. We say that i is unstable *(I-unstable) if it is not stable (I-stable).*

The following Lemma gives a characterization of unstable buckets.

Lemma 3.4.6. *Suppose that $i \in [b]$ is unstable. Then there exist pairwise disjoint intervals $I_1, I_2, I_3 \subset \mathbb{R}$, that appear in this order from left to right in the line, such that*

$$\left|I_1 \cap f'\left(B_i^{(i^*)}\right)\right| \geqslant n\varepsilon^{1/8}/(2b),$$

$$\left|I_3 \cap f'\left(B_i^{(i^*)}\right)\right| \geqslant n\varepsilon^{1/8}/(2b),$$

and

$$\left|I_2 \cap f'\left([n]\backslash B_i^{(i^*)}\right)\right| > \varepsilon^{1/8} \cdot n/b,$$

Proof. Let $I_1 \subset \mathbb{R}$ be the minimal interval that contains the $n\varepsilon^{1/8}/(2b)$ left-most points in $f'(B_i^{(i^*)})$, and let $I_3 \subset \mathbb{R}$ be the minimal interval that contains the $n\varepsilon^{1/8}/(2b)$ right-most points in $f'(B_i^{(i^*)})$. Let $I_2 \subset \mathbb{R}$ be the maximal interval that is contained between I_1 and I_3. Since $\varepsilon^{1/8} < 1$, we have that $I_1 \cap I_3 = \varnothing$, and therefore, all intervals I_1, I_2, I_3 are well-defined and pairwise disjoint. By construction, I_1 and I_3 each contains exactly $n\varepsilon^{1/8}/(2b)$ points in $f'(B_i^{(i^*)})$. Therefore, it remains to show that I_2 contains more than $\varepsilon^{1/8}n/b$ points in $f'([n]\backslash B_i^{(i^*)})$. Suppose, for the sake of contradiction, that I_2 contains at most $\varepsilon^{1/8}n/b$ in $f'([n]\backslash B_i^{(i^*)})$. Then, I_2 contains exactly $(1-\varepsilon^{1/8})n/b$ points in $f'(B_i^{(i^*)})$, and at most $\varepsilon^{1/8}n/b$ points in $f'([n]\backslash B_i^{(i^*)})$, implying that B_i is stable, which is a contradiction. This concludes the proof. □

We next show that for each unstable bucket, the feedback arc set must contain many edges incident to vertices in the bucket.

Lemma 3.4.7. *Let $i \in [b]$ be unstable. Then, $F^{(i^*)} \cup F'$ contains at least $\varepsilon^{1/4}n^2/(2b^2)$ arcs having exactly one endpoint in $B_i^{(i^*)}$.*

Proof. Let $I_1, I_2, I_3 \subset \mathbb{R}$ be the intervals given by Lemma 3.4.6. Let $v \in [n] \backslash B_i^{(i^*)}$, such that $f'(v) \in I_2$. Pick $j \in [b]$, such that $v \in B_j^{(i^*)}$. We consider two cases:

Case 1: Suppose that $j < i$. Let $u \in B_i^{(i^*)}$, such that $f'(u) \in I_1$. If $(v, u) \in A^{(i^*)}$, then it follows that f' violates (x_{i^*}, v, u), and thus $(v, u) \in F'$. Otherwise, we have $(u, v) \in A^{(i^*)}$. Since u appears after v in the topological sort of $G^{(i^*)} \backslash F^{(i^*)}$, it follows that $(u, v) \in F^{(i^*)}$. Thus, in either case, $F^{(i^*)} \cup F'$ contains either (u, v) or (v, u). Therefore, $F^{(i^*)} \cup F'$ contains at least $n\varepsilon^{1/8}/(2b)$ arcs having u as an endpoint.

Case 2: Suppose that $j > i$. This case is similar to Case 1, and is included for completeness. Let $u \in B_i^{(i^*)}$, such that $f'(u) \in I_3$. If $(u, v) \in A^{(i^*)}$, then it follows that f' violates (x_{i^*}, u, v), and thus $(u, v) \in F'$. Otherwise, we have $(v, u) \in A^{(i^*)}$. Since u appears before v in the topological sort of $G^{(i^*)} \backslash F^{(i^*)}$, it follows that $(v, u) \in F^{(i^*)}$. Thus, in either case, $F^{(i^*)} \cup F'$ contains either (u, v) or (v, u). Therefore, $F^{(i^*)} \cup F'$ contains at least $n\varepsilon^{1/8}/(2b)$ arcs having u as an endpoint.

We conclude that, in either case, for any $u \in B_i^{(i^*)}$, $F^{(i^*)} \cup F'$ contains at least $n\varepsilon^{1/8}/(2b)$ arcs having u as an endpoint. Summing over all $u \in B_i^{(i^*)}$, we obtain that $F^{(i^*)} \cup F'$ contains at least $\varepsilon^{1/4}n^2/(2b^2)$ arcs having an endpoint in $B_i^{(i^*)}$. This concludes the proof. □

Next, we bound the number of unstable buckets.

Lemma 3.4.8. *Let $J = \{i \in [b] : i \text{ is unstable}\}$, we have $|J| \leqslant O(\varepsilon^{1/4})2b^2$.*

Proof. By Lemma 3.4.3 we have that x_{i*} is $(1 - O(\varepsilon^{1/2}))$-good in f', and by Lemma 3.4.4 we have that $G^{(i*)}$ admits a feedback arc set of size at most $(O(\varepsilon^{1/2}))\binom{n-1}{2}$. Thus, by Theorem 3.3.3, the algorithm computes some feedback arc set $F^{(i*)} \subset A^{(i*)}$, with $|F^{(i*)}| = O(\varepsilon^{1/2}n^2)$. We note that here we only use Theorem 3.3.3 to obtain a $O(1)$-approximation. By Lemma 3.4.7,

$$\begin{aligned} |J| &\leqslant |F^{(i*)} \cup F'|/(\varepsilon^{1/4}n^2/(2b^2)) \\ &\leqslant O(\varepsilon^{1/4})2b^2, \end{aligned}$$

which concludes the proof. □

For any stable $i \in [b]$, let $\mathcal{I}_i \subset \mathbb{R}$ be the interval that contains at least $(1 - \varepsilon^{1/8})n/b$ points in $f'(B_i^{(i*)})$, and at most $\varepsilon^{1/8}n/b$ other points. Let also $\mathcal{J}_i \subset \mathcal{I}_i$ be an open interval that contains all but the $\varepsilon^{1/8}n/b$ leftmost points in $f'(B_i^{(i*)}) \cap \mathcal{I}_i$, and the $\varepsilon^{1/8}n/b$ rightmost points in $f'(B_i^{(i*)}) \cap \mathcal{I}_i$. Thus, $|\mathcal{J}_i \cap f'(B_i^{(i*)})| \geqslant (1 - 3\varepsilon^{1/8})n/b$. It follows that for any $i \neq j \in [b]$, such that both i and j are stable, we have $\mathcal{J}_i \cap \mathcal{J}_j = \varnothing$.

Intuitively, we intend to find a solution that satisfies a nearly-optimal fraction of constraints, while ignoring all constraints that involve points that are mapped outside the intervals $\mathcal{J}_i$, where $i \in [b]$ is stable. To that end, we define a small set of points that the analysis can safely "ignore":

$$X_{Noise} = \bigcup_{i \in [b] : i \text{ stable}} \left\{ v \in B_i^{(i*)} : f'(v) \notin \mathcal{J}_i \right\}.$$

Since $|\mathcal{J}_i \cap f'(B_i^{(i*)})| \geqslant (1 - 3\varepsilon^{1/8})n/b$, it follows that

$$|X_{Noise}| \leqslant 3\varepsilon^{1/8} n \tag{3.3}$$

Let also, for any $i \in [b]$,

$$\bar{B}_i^{(i^*)} = B_i^{(i^*)} \backslash X_{Noise}.$$

We identify a set of triples $(i, j, k) \in [b]^3$ for which, intuitively, it is difficult to satisfy at least some significant fraction of all constraints with one point from each of the clusters $B_i^{(i^*)}$, $B_j^{(i^*)}$, and $B_k^{(i^*)}$. Formally, we say that some $(i, j, k) \in [b]^3$ is *brittle* if there exist $u, u' \in \mathcal{J}_i$, $v, v' \in \mathcal{J}_j$, and $w, w' \in \mathcal{J}_k$, such that

$$|u - v| < |u - w|,$$

and

$$|u' - v'| > |u' - w'|.$$

Intuitively, the above property implies that if for all $t \in [b]$, all points in $\bar{B}_t^{(i^*)}$ get mapped to the same point $p_t \in \mathcal{J}_t$, then there exist choices for the points $\{p_t\}_t$, such that some constraint in $\bar{B}_i^{(i^*)} \times \bar{B}_j^{(i^*)} \times \bar{B}_k^{(i^*)}$ is violated; in other words, if a triple (i, j, k) is not brittle, then the choice of the points p_t does not affect the satisfiability of the constraints in $\bar{B}_i^{(i^*)} \times \bar{B}_j^{(i^*)} \times \bar{B}_k^{(i^*)}$.

We are now ready to show that the retraction computed by the algorithm admits a solution of low total cost.

Lemma 3.4.9. *The instance $\psi^{(i^*)}$ of $WLLOC$ constructed in Step 4 admits a solution that satisfies constraints of total weight at least $|\mathcal{T}| \cdot (1 - O(\varepsilon^{1/8}))$.*

Proof. We define a mapping $g : [b] \to [0,1]$, and $g' : [b] \to [0,1]$, as follows. For each $i \in [b]$, pick $v_i \in \mathcal{B}_i^{(i^*)}$, arbitrarily, and set

$$g'(i) = f'(v_i).$$

For any $j \in [b]$, we set

$$g(j) = g'(i),$$

where $i \in [b]$ is the unique integer such that $i \in B_i^{(i^*)}$. By the definition of the WLLOC instance $\psi^{(i^*)}$, the total weight of the constraints violated by g' equals the total number of constraints violated by g. It therefore suffices to upper bound the number of constraints in $\mathcal{T}$ that are violated by g.

We define a partition $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3 \cup \mathcal{T}_4 \cup \mathcal{T}_5$, where

$$\mathcal{T}_0 = \{(u,v,w) \in \mathcal{T} : f' \text{ violates } (u,v,w)\},$$

$$\mathcal{T}_1 = \{(u,v,w) \in \mathcal{T} : \text{ at least two of } u,v,w \text{ are in the same cluster in } \mathcal{B}^{(i^*)}\},$$

$$\mathcal{T}_2 = \{(u,v,w) \in \mathcal{T} : u \in B_i^{(i^*)}, v \in B_j^{(i^*)}, w \in B_k^{(i^*)}, \text{ at least one of } i,j,k \text{ is unstable}\},$$

$$\mathcal{T}_3 = \{(u,v,w) \in \mathcal{T} : u \in B_i^{(i^*)}, v \in B_j^{(i^*)}, w \in B_k^{(i^*)}, \text{ and } (i,j,k) \text{ is brittle}\},$$

$$\mathcal{T}_4 = \{(u,v,w) \in \mathcal{T} : \{u,v,w\} \cap X_{Noise} \neq \varnothing\},$$

$$\mathcal{T}_5 = \mathcal{T} \backslash (\mathcal{T}_0 \cup \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3 \cup \mathcal{T}_4).$$

By Lemma 3.4.2 we have

$$|\mathcal{T}_1| \leqslant (\varepsilon + O(1/n))n\binom{n-1}{2}.$$

Since every cluster in $\mathcal{B}^{(i^*)}$ has n/b points, we have

$$|\mathcal{T}_1| \leqslant 3n^3/b^2. \tag{3.4}$$

In order to bound $|\mathcal{T}_3|$ we need a bound on the number of brittle triples. This is done in Lemma 3.4.8, which appears in Section 3.5. We thus have

$$|\mathcal{T}_2| \leqslant O(\varepsilon^{1/4})2n^3b. \tag{3.5}$$

By Lemma 3.5.5 we have

$$|\mathcal{T}_3| \leqslant n^3/b. \tag{3.6}$$

By (Equation 3.3) we have

$$|\mathcal{T}_4| \leqslant O(\varepsilon^{1/8})n^3 \tag{3.7}$$

Let $(u, v, w) \in \mathcal{T}_5$. By the definition of $\mathcal{T}_5$, we have that $u \in \bar{B}_i^{(i^*)}$, $v \in \bar{B}_j^{(i^*)}$, and $w \in \bar{B}_k^{(i^*)}$, for some distinct $i, j, k \in [b]$, such that (i, j, k) is not brittle, and f' satisfies (u, v, w), that is

$$|f'(u) - f'(v)| < |f'(u) - f'(w)|.$$

By the definition of a brittle triple we get

$$|g(u) - g(v)| < |g(u) - g(w)|,$$

and thus g satisfies (u, v, w). We obtain that g satisfies all constraints in $\mathcal{T}_5$. Thus, by (Equation 3.4)–(Equation 3.7), the number of constraints violated by g is at most $|\mathcal{T}_0|+\ldots+|\mathcal{T}_4| \leqslant n^3 O(\varepsilon^{1/8})$, which concludes the proof. □

We are now ready to prove our main result.

Proof of Theorem 3.1.1. By Lemma 3.4.9 we have that WLLOC instance $\psi^{(i^*)} = ([b], \mathcal{T}', w)$ constructed at Step 4 of the algorithm, admits a mapping $g' : [b] \to \mathbb{R}$, such that the total weight of the constraints in $\mathcal{T}'$ violated by g' is at most $O(\varepsilon^{1/8} n^3)$. Therefore, in Step 5, using the exact algorithm from Theorem 3.3.1, we compute a mapping $g : [b] \to \mathbb{R}$, violating the same total weight as g'. By the definition of retraction, it follows that the mapping $f^{(i^*)}$ computed in Step 5 violates at most $O(\varepsilon^{1/8} n^3)$ constraints in $\mathcal{T}$, as required.

It remains to bound the running time. Step 2 uses the algorithm from Theorem 3.3.3 to obtain a $O(1)$-approximate minimum feedback arc set, and thus takes time $O(n^6)$. Step

3 takes time $O(n)$ and Step 4 takes time $O(n^2)$. Step 5 runs the algorithm from Theorem 3.3.1 on an input of size b, and thus takes time $O(b^{3b}) + O(n)$. Step 6 requires computing the number of violated constraints in each of the n solutions, and thus takes total time $O(n^4)$. Due to Step 1, the Steps 2–5 are repeated n times, and thus the total running time is at most $O(n^7 + b^{3b}n) = O(n^7) + (1/\varepsilon)^{O(1/\varepsilon^{1/8})}n$, which concludes the proof. □

3.5 Bounding the number of brittle triples

This Section is devoted to proving an upper bound on the number of brittle triples. We begin by deriving a simple condition that is a consequence of brittleness.

Lemma 3.5.1. *Let $j < i < k \in [b]$. We have that if (i, j, k) is brittle, then there exist $p_i \in \mathcal{J}_i$, $p_j \in \mathcal{J}_j$, $p_k \in \mathcal{J}_k$, such that*

$$p_i - p_j = p_k - p_i.$$

Proof. If (i, j, k) is brittle, it is easy to see that $\mathcal{J}_i$ must be located between $\mathcal{J}_j$ and $\mathcal{J}_k$; otherwise, any representative point chosen in $\mathcal{J}_i$ must be closer to all the points in $\mathcal{J}_j$ than those in $\mathcal{J}_k$, or vice versa. By definition, there exist $p_i \in \mathcal{J}_i$, $p'_j \in \mathcal{J}_j$, $p'_k \in \mathcal{J}_k$, such that

$$p_i - p'_j \geqslant p'_k - p_i,$$

and $p''_j \in \mathcal{J}_j$, $p''_k \in \mathcal{J}_k$, such that

$$p_i - p''_j < p''_k - p_i.$$

Without loss of generality, assume $p'_j < p''_j$ and $p'_k < p''_k$, and define $\delta \in [0,1]$. Comparing $d_{ij}(\delta) = p_i - (p'_j + \delta(p''_j - p'_j))$ and $d_{ik}(\delta) = (p'_k + \delta(p''_k - p'_k)) - p_i$, we have $d_{ij}(0) - d_{ik}(0) \geqslant 0$ and $d_{ij}(1) - d_{ik}(1) < 0$. There exist $\delta' \in [0,1]$, s.t. $d_{ij}(\delta') - d_{ik}(\delta') = 0$.

Define $p_j = (p'_j + \delta'(p''_j - p'_j)) \in \mathcal{J}_i$ and $p_k = (p'_k + \delta'(p''_k - p'_k)) \in \mathcal{J}_j$, we have

$$p_i - p_j = p_k - p_i,$$

which concludes the proof. □

Lemma 3.5.2. *Let $i_1, i_2, i_3, j_1, j_2, j_3, k_1, k_2, k_3 \in \mathbb{R}$, with $i_1 < i_2 < i_3$, $j_1 < j_2 < j_3$, $k_1 < k_2 < k_3$. For any $\alpha, \beta, \gamma \in \{1,2\}$, let $H_{\alpha,\beta,\gamma}$ be the axis-parallel parallelepiped defined by*

$$H_{\alpha,\beta,\gamma} := \mathcal{CH}(\{(i_{\alpha+\alpha'}, j_{\beta+\beta'}, k_{\gamma+\gamma'}) : \alpha', \beta', \gamma' \in \{0,1\}\}).$$

Let h be any plane in $\mathbb{R}^3$. Then, there exist $\alpha^*, \beta^*, \gamma^* \in \{0,1\}$, such that h does not intersect the interior of $H_{\alpha^*,\beta^*,\gamma^*}$.

Proof. For any $d \geqslant 2$, any d-dimensional halfspace containing the origin must also contain at least one d-orthant. The assertion follows immediately from the case $d = 3$. □

Lemma 3.5.3. *Let $i, j, k \in [b]$, with $j + 1 < i$, and $i + 1 < k$. Then, there exist $i', j', k' \in \{0,1\}$ such that $(i + i', j + j', k + k')$ is not brittle.*

Proof. Define the plane

$$h = \{(x_I, x_J, x_K) \in \mathbb{R}^3 : x_I - x_J = x_K - x_I\}.$$

By Lemma 3.5.1, we have that if $(i + i', j + j', k + k')$ is brittle, then h must intersect the hyperrectangle $\mathcal{J}_{i+i'} \times \mathcal{J}_{j+j'} \times \mathcal{J}_{k+k'}$. However, by Lemma 3.5.2, it follows that there exist $i', j', k' \in \{0, 1\}$, such that h does not intersect $\mathcal{J}_{i+i'} \times \mathcal{J}_{j+j'} \times \mathcal{J}_{k+k'}$, and thus $(i+i', j+j', k+k')$ is not brittle, which concludes the proof. □

Lemma 3.5.4 (Brittle convexity). *Let $\{e_1, e_2, e_3\}$ be the standard orthonormal basis in $\mathbb{R}^3$. Let $v \in [b-2]^3$, and let $w \in \{e_1, e_2, e_3\}$, such that v and $v + 2w$ are both brittle. Then, $v + w$ is also brittle.*

Proof. By Lemma 3.5.1, there exist $p_i \in \mathcal{J}_i$, $p_j \in \mathcal{J}_j$, $p_k \in \mathcal{J}_k$, such that

$$p_i - p_j = p_k - p_i. \tag{3.8}$$

Let $w = (i', j', k')$. Similarly, there exist $q_i \in \mathcal{J}_{i+2i'}$, $q_j \in \mathcal{J}_{j+2j'}$, $q_k \in \mathcal{J}_{k+2k'}$, such that

$$q_i - q_j = q_k - q_i. \tag{3.9}$$

For any $\alpha \in [0, 1]$, let

$$z_i^{(\alpha)} = (1 - \alpha)p_i + \alpha q_i,$$

$$z_j^{(\alpha)} = (1-\alpha)p_j + \alpha q_j,$$

$$z_k^{(\alpha)} = (1-\alpha)p_k + \alpha q_k.$$

Let us assume that $w = e_1$. The cases $w = e_2$ and $w = e_3$ can be handled in a similar manner. We have that for all $\alpha \in [0,1]$, $z_j^{(\alpha)} \in \mathcal{J}_j$, and $z_k^{(\alpha)} \in \mathcal{J}_k$. Moreover, $z_i^{(0)} \in \mathcal{J}_i$, and $z_i^{(1)} \in \mathcal{J}_{i+2}$, which implies that there exists some $\alpha^* \in [0,1]$, such that $z_i^{(\alpha^*)} \in \mathcal{J}_{i+1}$. We have

$$\begin{aligned} z_i^{(\alpha^*)} - z_j^{(\alpha^*)} &= (1-\alpha^*)p_i + \alpha^* q_i - (1-\alpha^*)p_j - \alpha^* q_j \\ &= (1-\alpha^*)(p_i - p_j) + \alpha^*(q_i - q_j) \\ &= (1-\alpha^*)(p_k - p_i) + \alpha^*(q_k - q_i) \\ &= (1-\alpha^*)p_k + \alpha^* q_k - (1-\alpha^*)p_i - \alpha^* q_i \\ &= z_k^{(\alpha^*)} - z_i^{(\alpha^*)}, \end{aligned}$$

which by Lemma 3.5.1 implies that $v + w$ is brittle, and concludes the proof. □

We are now ready to bound the number of brittle triples, which is the main result of this Section.

Lemma 3.5.5. *The number of brittle triples is at most $O(b^2)$.*

Proof. Let $B \subseteq [b]^3$ be the set of all brittle triples, and let $B' = [b]^3 \backslash B$. For any $s \in \{0,1\}^3$, let

$$U_s = s \cdot b/2 + [b/2]^3,$$

and $B_s = B \cap U_s$. Since $B = \bigcup_s B_s$, and there are only 8 different values for s, it suffices to show that for any $s \in \{0,1\}^3$, $|B_s| = O(b^2)$. We shall prove this for the case $s = (0,0,0)$. All remaining cases can be handled in a similar manner.

For the remainder for the proof, let $s = (0,0,0)$. By Lemma 3.5.3, it follows that for any $v \in B_s^3$, there exists $v' \in B'$, with $v' - v \in \{0,1\}^3$. This implies that there exists $u \in B$, and $u' \in B'$, with $u - v \in \{0,1\}^3$, $u' - v \in \{0,1\}^3$, and $u' - u \in \{e_1, e_2, e_3\}$, where $\{e_1, e_2, e_3\}$ is the standard orthonormal basis in $\mathbb{R}^3$. Let $t = u' - u$. By Lemma 3.5.4, it follows by induction that for any $i \in \{1, \ldots, b/2\}$, the triple $u + i \cdot t$ is brittle. Let

$$R_v = \bigcup_{i=1}^{b/2} \{u + c \cdot i\}.$$

Thus $R_v \subseteq B'$. Note that, since $s = (0,0,0)$, we have

$$|R_v| \geqslant b/2. \tag{3.10}$$

For any $j \in \{1,2,3\}$, we say that v is *type-j*, if $t = e_j$.

Let

$$B_{s,j} = \{v \in B_s : v \text{ is type-}j\}.$$

Let $j^* \in \{1,2,3\}$, such that $|B_{s,j^*}| \geqslant |B_s|/3$.

By the above construction, it follows that for any $v, w \in B_{s,j^*}$, with $\|v - w\|_\infty \geqslant 2$, we have $R_v \cap R_w = \varnothing$. We greedily construct some $C \subseteq B_{s,j^*}$ as follows. We start with $C := \varnothing$, and

$D := B_{s,j*}$. While $D \neq \varnothing$, we pick any $v \in D$, and we set $C := C \cup \{v\}$, and $D := D \backslash \mathsf{ball}_\infty(v, 1)$, where $\mathsf{ball}_\infty(v, r)$ denotes the ℓ_∞-ball of radius r centered at v. For every v added to C, we delete at most 9 elements from D, and thus

$$|C| \geqslant |B_{s,j*}|/9 \geqslant |B_s|/27.$$

Since for any $v, w \in C$, we have $\|v - w\|_\infty$, it follows that $R_v \cap R_w = \varnothing$. Combining with (Equation 3.10), we get

$$b^3 \geqslant |B'| \geqslant \left| \bigcup_{v \in C} R_v \right| = \sum_{v \in C} |R_v| \geqslant |C| \cdot b/2 \geqslant |B_s| \cdot b/54,$$

and thus $|B_s| \leqslant 54b^2$, which concludes the proof. □

CHAPTER 4

SOLVING STABLE STEINER TREE INSTANCES

This chapter is based upon the paper "On Geometry of Stable Steiner Tree Instances" authored by J. Freitag, N. Mohammadi, A. Potukuchi, L. Reyzin, published in the Canadian Conference in Computational Geometry in 2022 (CCCG 2022). [6]

My main contribution to this work was to the development of structural properties that appear in Sections 4.3 and 4.4.

4.1 Introduction and previous work

In this work, we initiate the study of Steiner tree instances that are stable to multiplicative perturbations to the distances in the underlying metric. Our analysis lies in the Bilu-Linial stability [45] setting, which provides a way to study tractable instances of NP-hard problems.

Instances that are γ-stable in the Bilu-Linial model have the property that the structure of the optimal solution is not only unique, but also does not change even when the underlying distances among the input points are perturbed by a multiplicative factor $\gamma > 1$. In their original paper, Bilu and Linial analyzed MAX-CUT clustering, and since their seminal work, other problems have been analyzed including center-based clustering [46–48], multi-way cut problems [49], and metric TSP [50].[1]

[1] Bilu-Linial stability is one among other notions of data stability studied in the literature [51, 52]. This is in contrast to notions of algorithmic stability, which focus on properties algorithms as opposed to data, see e.g. [53–57].

Here, we look at the metric Steiner tree problemand also the more restricted Euclidean version. For general metrics, the Steiner tree problem is known to be APX-hard in the worst case [58]. For the Euclidean metric, a PTAS is known [59].

In this paper we begin by providing strong geometric structural properties that need to be satisfied by stable instances. These point to the existence of algorithms for non-trivial families. We then make use of, and strengthen, these geometric properties to show that 1.59-stable instances of Euclidean Steiner trees are polynomial-time solvable. Finally, we discuss the connections between certain approximation algorithms and Bilu-Linial stability for Steiner trees.

4.2 Model and definitions

In this section, we recall the relevant definitions. Fist we define the Steiner tree problem, which is among Karp's 21 original NP-hard problems [60]. It has various applications including in network design, circuit layouts, and phylogenetic tree reconstruction.

Definition 4.2.1 (the Steiner tree problem). *Consider an undirected graph $G = (V, E)$ with edge weights $w_e \in \mathbb{R}_0^+$ for every edge $e \in E$, and a set $T \subseteq V$ of terminals. A Steiner tree S is a tree in the graph G that spans all terminal vertices T and may contain some of the non-terminals (also called Steiner points). The goal is to find such a tree of lowest weight, which we call* OPT,

$$\mathrm{OPT} = \arg\min_{S} \sum_{e \in S} w_e.$$

We can assume without loss of generality[1] that the vertices are points in a metric space and the weights of the edges are given by the distance function – when the input is in the form of a metric, we call this the **metric Steiner tree problem**. Our results use properties of metric spaces, but move freely between the metric space and graph representations of the problem. When the metric is Euclidean, this is called the **Euclidean Steiner tree problem**.

Now we move on to defining Bilu-Linial stability for the Steiner tree problem on metrics.

Definition 4.2.2 (**Bilu-Linial γ-stabile instances**). *Let $I = (G, w)$ be an instance of a metric Steiner tree problem and $\gamma > 1$. I is γ-stable if for any function $w' : V \times V \to \mathbb{R}_0^+$ such that $\forall u, v \in V$,*

$$w_{uv} \leqslant w'_{uv} \leqslant \gamma w_{uv},$$

the optimal Steiner tree OPT' under w' is equal to the optimal Steiner tree OPT under w.

We note that the perturbations can be such that instances originally satisfying the metric or Euclidean properties no longer have to satisfy these properties after perturbation. We also note that due to the triangle inequality, no instances have stability 2 or greater in the metric setting.

Notation: For a graph G, w_{ab}^G is the weight of edge ab in G. We abbreviate $w_{ab} = w_{ab}^G$ and $w'_{ab} = w_{ab}^{G'}$. Let $\mathrm{OPT} \subseteq E(G)$ denote the minimum weight Steiner tree of G, let $w(\mathrm{OPT}) = w^G(\mathrm{OPT})$ denote the weight of the Steiner tree.

---

[1] For any graph with distances specified on edges, a metric can be formed by taking the vertices to be points and considering the shortest path distances in the graph between pairs of vertices. Solving (or approximating) the Steiner tree problem on a metric formed in this matter solves (or approximates) the problem on the original graph. See Vazirani [61] for further discussion of this issue.

4.3 Structural properties in general metrics

In this section, we work in the context of a general metric space, and we develop interesting restrictions on the types of problems with γ-stable solutions, for various values of γ.

The techniques of this section *do not* give, in complete generality, an efficient algorithm for finding the optimal Steiner tree for any value of γ less than 2, a problem we leave open. However, when more information about the metric space is available, one can use the structural results here to give restrictions on the arrangements of Steiner points which does yield a definitive solution. In particular,

1. In Section 4.4, we use Lemma 4.3.1 to give an algorithm for the Euclidean metric when $\gamma > 2^{2/3}$.
2. More generally, in the case that no two Steiner points are adjacent in the optimal solution, Lemma 4.3.8 together with the other results of the section can be used to give an efficient and very simple algorithm to find the minimal weight Steiner tree. Other more general situations can be efficiently handled via only slightly more elaborate arguments - e.g. if one has a bound on the length of the longest path of Steiner points in the optimal solution.

Lemma 4.3.1. *The degree of any Steiner point in the optimal solution is greater than* $\frac{2}{2-\gamma}$.

Proof. Consider a Steiner node s in the optimal solution, that is connected to $(m \neq n)$ other points, $a_1, ..., a_m$. Let $\overline{w} = \sum_{i=1}^{m} \frac{w_{sa_i}}{m}$, and let w_{sa_1} and w_{sa_m} be such that $w_{sa_1} + w_{sa_m} \geqslant 2\overline{w}$. Let G' be obtained by perturbing each edge sa_i by a factor of γ. Let

$$\mathrm{T}' := (\mathrm{OPT} \backslash \{sa_1, \ldots, sa_m\}) \cup \{a_1a_2, \ldots, a_{m-1}a_m\}.$$

Clearly, T′ is also a Steiner tree. Triangle inequality gives us $w_{a_i a_{i+1}} \leqslant w_{sa_i} + w_{s_{a_{i+1}}}$. So, we have

$$\begin{aligned} w'(\mathrm{T}') &\leqslant w'(\mathrm{OPT}) - \sum_{i=1}^{m} \gamma w_{sa_i} \\ &\quad + \sum_{i=1}^{m-1} \left(w_{sa_i} + w_{sa_{i+1}}\right) \\ &= w'(\mathrm{OPT}) - \sum_{i=1}^{m} \gamma w_{sa_i} + \sum_{i=2}^{m-1} 2w_{sa_i} \\ &\quad + w_{sa_1} + w_{sa_m}. \end{aligned}$$

Using the fact that $w'(\mathrm{T}') > w'(\mathrm{OPT})$, we have

$$\sum_{i=1}^{m} \gamma w_{sa_i} < \sum_{i=2}^{m-1} 2w_{sa_i} + w_{sa_1} + w_{sa_m}$$

or

$$\gamma \cdot \overline{w} m < (2m-2)\overline{w}.$$

Rearranging, we have

$$\frac{2}{2-\gamma} < m.$$

□

Now we state some additional structural properties of optimal Steiner trees in γ-stable instances. These are not used in Section 4.4. Nevertheless, we hope that they are of independent interest.

Proposition 4.3.2. *If $a, b \in V(\mathrm{OPT})$ are nearest neighbors in the graph, then the edge ab is in the optimal solution.*

Lemma 4.3.3. *Suppose $ab, bc \in \mathrm{OPT}$, then*

1. $w_{ac} > \gamma \cdot \max\{w_{ab}, w_{bc}\}$.
2. $\frac{2}{\gamma} \cdot w_{ac} > w_{ab} + w_{bc}$.
3. $(\gamma - 1) \cdot w_{ab} < w_{bc}$, $(\gamma - 1) \cdot w_{bc} < w_{ab}$.

Proof. We handle the three parts in turn:

1. Assume w.l.o.g. $w_{ab} \geqslant w_{bc}$. Suppose that $w_{ac} \leqslant \gamma \cdot \max\{w_{ab}, w_{bc}\}$, let G' be obtained by perturbing ab by a factor of γ. Then $(\mathrm{OPT}\backslash\{ab\}) \cup \{ac\}$ is also a Steiner tree in G' of weight $w'(\mathrm{OPT})$ contradicting stability. This completes the proof of 1.
2. The proof of 2. follows from 1. and the fact that $\max\{w_{ab}, w_{bc}\} \geqslant \frac{w_{ab}+w_{bc}}{2}$.
3. Let G' be obtained by perturbing bc by a factor of γ. Then $\mathrm{T}' := \mathrm{OPT}\backslash\{bc\} \cup \{ac\}$ is also a Steiner tree of weight

$$w'(\mathrm{T}') = w(\mathrm{OPT}) - w_{bc} + w_{ac} \leqslant w(\mathrm{OPT}) + w_{ab}. \tag{4.1}$$

On the other hand, stability gives us that

$$w'(\mathrm{T}') > w'(\mathrm{OPT}) = w(\mathrm{OPT}) + (\gamma - 1)w_{bc}. \tag{4.2}$$

Putting (Equation 4.1) and (Equation 4.2) together gives us that $(\gamma - 1) \cdot w_{bc} \leqslant w_{ab}$.

Repeating the same argument but swapping bc for ab gives us $(\gamma - 1) \cdot w_{ab} \leqslant w_{bc}$.

□

Lemma 4.3.4. *Let H be a subgraph of* OPT *with at least one edge. Let $ab \in H$. Fix any vertex $c \in V(\mathrm{OPT}) \backslash V(H)$ satisfying $w_{ca} \leqslant \gamma(\gamma - 1) \cdot w_{ab}$; then we have $ca \in$* OPT.

Proof. If $ca \notin \mathrm{OPT}$, then adding the edge ac to OPT produces a cycle which includes edge ac. Suppose that the cycle also includes ab. Let G' be obtained by perturbing ab by a factor of γ. Then $(\mathrm{OPT} \backslash \{ab\}) \cup \{ac\}$ is a Steiner tree of weight at most $w'(\mathrm{OPT})$, contradicting stability.

If the cycle does not include ab, it includes some edge other than ac which has an endpoint at a. This edge, call it ad, is in OPT. By Lemma 4.3.3, $w_{ad} > (\gamma - 1)w_{ba}$. Let G' be obtained by perturbing ad by a factor of γ. We have $w'_{ad} > \gamma(\gamma - 1)w_{ba} \geqslant w_{ac}$. Then $(\mathrm{OPT} \backslash \{ad\}) \cup \{ac\}$ is a Steiner tree of weight less than $w'(\mathrm{OPT})$, again contradicting stability. □

Lemma 4.3.5. *Let $\gamma > \frac{1+\sqrt{5}}{2}$. Let $ab \in H$, a subgraph of* OPT. *Suppose that c is a vertex with $w_{ca} \geqslant \gamma \cdot w_{ab}$, then $ca \notin$* OPT.

Proof. Let $\gamma' = \frac{w_{ca}}{w_{ab}}$. Note that $\gamma' \geqslant \gamma$ is some real number larger than $\frac{1+\sqrt{5}}{2}$. If $ac \in \mathrm{OPT}$, then by part 1. of Lemma 4.3.3, we must have

$$\frac{w_{bc}}{w_{ac}} > \gamma.$$

On the other hand,

$$\begin{aligned} \frac{w_{bc}}{w_{ac}} &\leqslant \frac{w_{ab} + w_{ac}}{w_{ac}} \\ &\leqslant \frac{w_{ab} + \gamma' w_{ab}}{\gamma' w_{ab}} \\ &\leqslant \frac{1+\gamma'}{\gamma'}. \end{aligned}$$

We now have a contradiction as long as $\frac{1+\gamma'}{\gamma'} < \gamma$. The function $f(x) = \frac{1+x}{x}$ is decreasing for $x > 0$ and $f(x) < x$ for any $x \geqslant \frac{1+\sqrt{5}}{2}$. So, we have that

$$\frac{1+\gamma'}{\gamma'} < \frac{1+\gamma}{\gamma} < \gamma$$

as desired. □

Proposition 4.3.6. *Let H be a subgraph of* OPT *with at least one edge. Suppose that $ab \in H$ and suppose that $c \in V(\mathrm{OPT}) \backslash V(H)$ with $w_{bc} < \gamma(\gamma-1)w_{ab}$. Then we must have $w_{bc} < \frac{w_{ab}}{\gamma-1}$ and $w_{ab} < \frac{w_{bc}}{\gamma-1}$.*

Proof. By Lemma 4.3.4, we must have that $bc \in \mathrm{OPT}$. Therefore, property 3. of Lemma 4.3.3 gives us the desired inequalities. □

When $\gamma(\gamma-1)^2 > 1$ Proposition 4.3.6 strengthens the bounds of Lemma 4.3.4. This holds, for instance, when $\gamma > 1.755$. In this case, we obtain:

Proposition 4.3.7. *Assume that $\gamma(\gamma-1)^2 > 1$. Assume that H is a subgraph of* OPT *with at least one edge. Let $ab \in H$. Fix any vertex $c \in V(\mathrm{OPT}) \backslash V(H)$. Then we have $w_{ca} < \frac{1}{\gamma-1} \cdot w_{ab}$ if and only if $ca \in$* OPT*.*

Proof. By Lemma 4.3.4 and the assumption that $\gamma(\gamma-1) > \frac{1}{\gamma-1}$, we must have that $ac \in \mathrm{OPT}$. If $w_{ca} \geqslant \frac{1}{\gamma-1} \cdot w_{ab}$, we can not have edge ac in OPT by Lemma 4.3.3 part 3.

□

Let $B = \{b_1, \ldots, b_m\}$ be vertices (either terminal or Steiner points). For a vertex a, we denote by $T(a, B)$ the tree on vertex set a, B in which a is connected to each element of B. Let the *average weight of* $T(a, B)$ be

$$\frac{\sum_{i=1}^{m} w_{ab_i}}{m}.$$

Suppose that H is a subgraph of OPT. We call $T(a, B)$ a *terminal component fan* relative to H if a is a Steiner point and B are all terminals or vertices in distinct connected components of H each with at least two vertices. We call the collection of components of H together with the terminals not in H the *terminal components of H.*

Lemma 4.3.8. *Let $\gamma > 1.755$ and suppose that H is a subgraph of* OPT *and in the optimal solution, no two Steiner points are adjacent. Suppose that $T(a, B)$ with $B = \{b_1, \ldots, b_m\}$ is a terminal component fan such that:*

- *The average weight of $T(a, B)$ is less than all edges not in H which connect two terminal components of H,*
- *the average weight of $T(a, B)$ is minimal among all terminal component fans, and*
- *the weights of the edges of $T(a, B)$ are all within a factor of $\frac{1}{\gamma-1}$ of each other.*

Then $T(a, B)$ is a subgraph of OPT.

Proof. Suppose that the fan $T(a, B)$ is not in OPT. Specifically, if there are $k < m$ edges of $T(a, B)$ which are not in OPT, then there are at least k edges of OPT$\backslash H$ such that in OPT $\cup\, T(a, B)$ we may remove these k edges and still have a Steiner tree.[1] Moreover, since no two Steiner points are adjacent, these edges are either

- terminal to terminal edges, or
- part of a terminal component fan.

In the first case, the terminal to terminal edges have weight at least $\frac{\sum_{i=1}^{m} w_{ab_i}}{m}$. In this case perturb this edge by a factor of γ, and swap it with one edge of the terminal component fan $T(a, B)$. Since the weights of edges of T are within a factor of $\frac{1}{\gamma-1}$ of each other and their

---

[1] In the case that $k = m$, there may be only $m - 1$ such edges, as a may not be in OPT, but the argument works identically in that case.

average weight is $\frac{\sum_{i=1}^{m} w_{ab_i}}{m}$, this swap decreases the weight of the resulting Steiner tree after the perturbation.

Similarly in the case that one of the k edges is in another terminal component fan, T_1, the average weight of edges in that fan is at least $\frac{\sum_{i=1}^{m} w_{ab_i}}{m}$, and applying part 3. of Lemma 4.3.3, the minimal weight edge in T_1 is at least $(\gamma-1)\cdot\frac{\sum_{i=1}^{m} w_{ab_i}}{m}$. Now, perturb such an edge by a factor of γ to make the weight at least $\gamma \cdot (\gamma - 1) \cdot \frac{\sum_{i=1}^{m} w_{ab_i}}{m}$, which is larger than the weight of the largest weight edge of $T(a, B)$, which is a most $\frac{1}{\gamma-1} \cdot \frac{\sum_{i=1}^{m} w_{ab_i}}{m}$ because $\gamma > 1.755$.

Performing any of these k swaps yields a lower weight Steiner tree than OPT under the above perturbations, contradicting γ-stability. □

4.4 Euclidean Steiner trees

In this section, we consider the restriction of the Steiner tree problem to the Euclidean metric.

Definition 4.4.1 (angle). *Let a_1, a_2, b be points on a Euclidean metric. Then we call $\angle a_1 b a_2$ the* angle *between a_1, a_2 at b.*

Under the assumption of γ-stability the minimum angle between two terminal points at their common Steiner neighbor can be bounded from below as a function of γ.

Lemma 4.4.2. *For a γ-stable instance of a Euclidean Steiner tree, the angle between two terminal points at their common Steiner neighbor in the tree should be greater than $2\sin^{-1}(\gamma/2)$.*

Proof. Let's assume, for a γ-stable instance of Steiner tree, the angle between two terminal points a_1, and a_2 at a Steiner point b is θ. Without loss of generality, let $w_{a_1 b} =: w \geqslant w_{a_2 b}$.

Clearly $w_{a_1a_2} > \gamma w$, since otherwise, perturbing edge a_1b by a factor of γ allows one to replace a_1b by a_1a_2 in a minimal Steiner tree, contradicting stability. Let us use α to denote the angle $\angle a_1a_2b$. Clearly, $\alpha \geqslant \pi/2 - \theta/2$. Thus by the sine rule, we have

$$\frac{\gamma w}{\sin\theta} < \frac{w_{a_1a_2}}{\sin\theta} = \frac{w}{\sin\alpha} \leqslant \frac{w}{\sin(\pi/2 - \theta/2)}.$$

Rearranging, we have

$$\begin{aligned} \gamma &< \frac{\sin\theta}{\sin(\pi/2 - \theta/2)} \\ &= \frac{2\sin(\theta/2)\cos(\theta/2)}{\cos(\theta/2)} \\ &= 2\sin(\theta/2) \end{aligned}$$

as desired. □

Thus we immediately get the following Corollary.

Corollary 4.4.3. *For a γ-stable instances of Steiner tree where $\gamma > \sqrt{2}$, the angle between two terminal points at their common neighbor in the optimal Steiner tree is greater than $\pi/2$.*

We say that a matrix $M \in \mathbb{R}^{d\times d}$ is *positive semi-definite* if for every $v \in \mathbb{R}^d$, it holds that $v^T Rv \geqslant 0$.

Lemma 4.4.4. *If there are N points in $\mathbb{R}^d$ such that the angle between every pair with respect to a point u is at least $\theta > (\pi/2)$, then $N \leqslant 1 - \frac{1}{\cos\theta}$.*

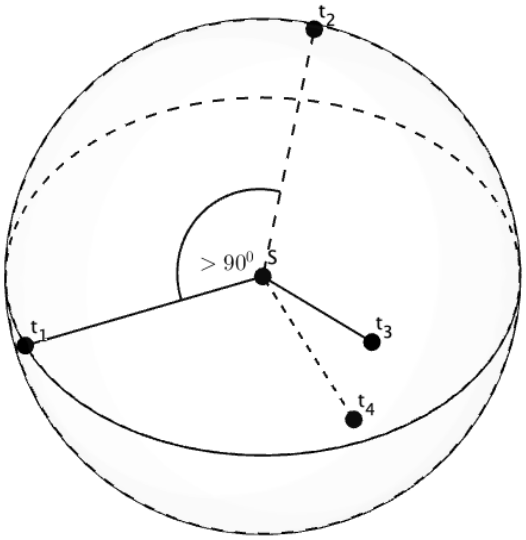


Figure 3. An example of points t_1, t_2, t_3, and t_4 surrounding Steiner point s at angles over $\theta > 90$ degrees. No more than $1 - \frac{1}{\cos\theta}$ can fit, independent of the dimension.

Proof. Let $\theta > \pi/2$ and let $v_1, \ldots, v_N \in \mathbb{R}^d$ be unit vectors in $\mathbb{R}^d$ such that $\langle v_i, v_j \rangle \leqslant \cos\theta$. Consider the matrix V whose columns are the v_i's. By construction $V^T V$ is positive semi-definite. Indeed, for any $u \in \mathbb{R}^d$, we have $u^T(V^T V)u = \langle Vu, Vu \rangle \geqslant 0$.

If $N - 1 > \frac{-1}{\cos\theta}$, then the sum of every row is negative. This is because each diagonal entry of $V^T V$ is 1, and every non-diagonal entry is at most $\cos\theta$. So we have that $\mathbf{1}^T(V^T V)\mathbf{1} < 0$ where $\mathbf{1} = (1, 1, \ldots, 1)$. This contradicts the positive semidefiniteness of $V^T V$. So it must be the case that $N \leqslant 1 - \frac{1}{\cos\theta}$. □

Corollary 4.4.5. *For $\gamma > \sqrt{2}$ the degree of a Steiner node in the optimal solution is at most $\frac{\gamma^2}{\gamma^2-2}$.*

Proof. Consider any two neighbors u, w of a given vertex v, and assume that $\angle uvw = \theta$. From Lemma 4.4.2 we have

$$\gamma < 2\sin(\theta/2).$$

So

$$\gamma^2 < 4\sin^2(\theta/2)$$

and so $\gamma^2/2 < 2\sin^2(\theta/2)$ or $1 - \gamma^2/2 > 1 - 2\sin^2(\theta/2)$. Since $\cos(\theta) = 1 - 2\sin^2(\theta/2)$, we have

$$\cos(\theta) < 1 - \gamma^2/2$$

or

$$\theta > \cos^{-1}(1 - \gamma^2/2).$$

Since the angle between any two neighbors of v is at least $\cos^{-1}(1 - \gamma^2/2)$, Lemma 4.4.4 gives us that there are at most $1 - \frac{2}{2-\gamma^2} = \frac{\gamma^2}{\gamma^2-2}$ of them. □

Corollary 4.4.6. *When $\gamma > 1.59$, the optimal Steiner tree for a γ-stable instance does not have Steiner nodes.*

Proof. This happens when the min degree imposed by stability is larger than the max degree imposed by the packing bound. By Lemma 4.3.1 and Corollary 4.4.5, this happens when we

have the following:

$$\frac{\gamma^2}{\gamma^2 - 2} \leqslant \frac{2}{2 - \gamma}$$

By solving the above equation for γ we get $\gamma \geqslant 2^{2/3}$, which is bounded from above by 1.59. □

This geometric property implies that for 1.59-stable instances, Steiner points will not be used in the optimal solution. Hence, an MST algorithm on just the terminal points will give the answer in polynomial time.

Finally, we point to the existence of Gilbert and Pollak's the Steiner ratio conjecture [62], which states that in the Euclidean plane, there always exists an MST within a cost of $2/\sqrt{3}$ of the minimum Steiner tree, and the behavior of this ratio for higher dimensions is yet unknown. Assuming this conjecture, in certain cases it may imply some limitations on the stability of Euclidean instances, especially in low dimensions, using the idea that even if the Steiner tree distances are "blown up" by more than the Steiner ratio, one could instead use the MST instead and get a cheaper solution. Unfortunately, because the MST may overlap with the Steiner tree, we cannot give a concrete statement.

4.5 Using approximation algorithms to solve stable instances

In this section we give a general argument about how strong approximation algorithms for Steiner tree problems give stability guarantees. We note that it is known that an FPTAS for the Steiner tree would imply P=NP [58], so there is no hope to use the result below in the general metric case. But if at some future point an FPTAS for the Euclidean variant of the Steiner tree problem is developed (currently, only a PTAS is known to exist [59]), then this

would immediately imply the existence of polynomial-time algorithms for stable instances for any constant $\gamma > 1$.

Theorem 4.5.1. *An FPTAS for the Steiner tree problem gives a polynomial time algorithm for optimally solving any γ-stable Steiner tree problem in time poly$(n, (\gamma-1)^{-1})$. In particular, this gives a polynomial-time algorithm for any constant $\gamma > 1$.*

Proof. Assume we are given an FPTAS for the Steiner tree problem. This means that we have an algorithm that runs in time poly$(n, 1/\epsilon)$ on instances of size n to give $(1+\epsilon)$-approximations to the optimum Steiner tree. Now consider a γ-stable instance for constant $\gamma > 1$. We run our FPTAS on that instance with $\epsilon = \frac{\gamma-1}{2n}$ to get a Steiner tree S' with weight within OPT$(1 + (\gamma-1)/2n)$. We now claim that every edge in the optimal solution whose weight is at least $\frac{\text{OPT}}{n}$ must be in S'. Suppose it isn't – then we could perturb such an edge by γ and increase the weight of the optimal solution to OPT$(1 + (\gamma-1)/n)$ without increasing the weight of S', and S' would become cheaper than OPT, thereby violating γ-stability.

By the fractional pigeonhole principle, the most expensive edge of the FPTAS satisfies the desired property above and is therefore in OPT. Hence, we can contract this edge into a new vertex and get a new instance with $n-1$ vertices at γ-stability. We can continue this process, getting one new edge of the optimal in each iteration, until we have a constant-size problem that we can brute-force. □

We note that the above technique could be used to convert even slightly weaker (than FPTAS) approximation algorithms to nontrivial stability guarantees. Lemma 4.4.4.

CHAPTER 5

LEARNING GRAPHS WITH BIPARTITE EDGE COUNTING QUERIES

5.1 Introduction

Graph learning has been extensively used in the last few decades in many scientific fields, especially, in bio-informatics [17] and has received much attention lately due to its wide application in modeling complex data sets as graph problems and solving them efficiently via graph theory techniques [63, 64]. For example, human genome sequencing [14–16, 65] or protein synthesis can efficiently be modeled by the graph and various problems such as protein mutations or genetics disease can be formulated as a graph problem. In this chapter, first, we consider the problem of learning a hidden graph $G = (V, E)$ with BEC queries. Then we study the same learning problem by leveraging a graph verification result that we provide.

The goal in a verification problem is to verify the hidden graph G via query access to G. It is a lesser-known problem in compare to graph learning. One example of the application of graph verification can be performing a sanity check on the result of graph learning. Imagine we had some errors in learning expensive social networks or brain connectome network, before moving to the next step of the project, we must make become certain that we correctly learned the targeted graph. Therefore, we need to ask, can we verify we learned the targeted graph efficiently or do we need to repeat the learning step before moving forward? This question naturally arises in the real world setting where the nature of the real work data sets is often noisy.

The main focus of this work is on introducing and using BEC queries on undirected unweighted graphs. In addition, we show how BEC queries can be stimulated with other queries such as EC and OC queries.

5.2 Previous work

For the graph verification problem, Beerliova et al. [66] study the problem of verifying and finding networks via distance queries and show, unless P = NP, there is no $O(\log n)$-competitive algorithm.

Our work is mostly inspired by Reyzin and Srivastava's work on learning and verifying graphs by focusing on EC queries [67]. They give an algorithm for learning graph partitions using $O(n \log n)$ EC queries. They also introduce the problem of verifying graphs properties by using EC queries and provide a randomized algorithm with error ϵ for graph verification problem using $O\left(\log \frac{1}{\epsilon}\right)$ EC queries.

Angluin and Chen [16] give an algorithm that can learn any arbitrary graph with $O(\log n)$ adaptive ED queries by repeatedly dividing the graph into independent subgraphs, and eliminating new connections between subgraphs from previously discivered edges via ED queries, and uses a version of binary search to find new edges within each subgraph. Angluin and Chen later generalize their results for learning arbitrary graphs via ED queries to hypergraphs in [15] by using fundamentally different techniques. One of their main results is giving an algorithm that finds an arbitrary edge in a hypergraph of dimension r using only $O(r \log n)$ edge-detecting queries and then improve the round complexity to $O(\log m + r)$ using only $O(\log m \log n)$ more

queries. Angluin and Chen's work was carried on in [14,65,68] for learning geometrically restricted families of graphs, such as stars, cliques, and matchings.

As it has been mentioned in the previous section, we mainly focus on learning undirected graphs but there are other versions of graph learning problem on directed graphs. Wang and Honorio [69] study the problem of learning bounded-degree directed trees by using path queries and give a randomized algorithm for learning directed trees of n nodes with node degree at most d, by asking at most $O(dn \log^2 n)$ path queries. Their result motivated Janardhanan and Reyzin [70] to study the problem of learning a directed graph by using path queries and give bounds for learning graphs with n vertices and k strongly connected components and bounded degree directed trees where they give an algorithms for learning "almost-trees" – directed trees. Although we didn't study the directed version of this problem here but the mentioned paper can be considered as an inspiration for the future exploration of this work.

In addition to learning different graph families, researchers often explored using different queries. Bipartite Independent Set (BIS) queries were introduced by Beame et al. [71] and inspired Addanki et al. [72] to consider the problem of learning the count of edges in a graph $G = (V, E)$ where $|V| = n$ by BIS query access to G. Beame et al.'s [71] introduction of BIS queries motivated us to consider BEC queries that can be described an EC-type generalization of BIS queries.

In this work we extend the results of Reyzin and Srivastava in [67] for learning and verifying graphs via EC to BEC queries that we consider here.

Both the learning and verification tasks can be related to the field of Property Testing, where the goal is to test small parts of the adjacency matrix of a graph to determine a global property of the graph. Goldrich and Goldwassar did a thorough survey on Property Testing in [73].

5.3 Model and definitions

The problem of graph learning involves the learner being given a set of vertices V in a hidden graph $G = (V, E)$. The learner's goal is to discover the set of edges.

Definition 5.3.1 (Edge Counting Queries (EC)). *Given a subset A of vertices $A \subseteq V$,* $\mathrm{EC}(A)$ *returns the number of edges both of whose endpoints lie in A.*

One of many known results is Reyzin and Srivavasta's [67] general results for learning graphs. It employs binary search and motivates some of our algorithms.

Lemma 5.3.2 (EC for Learning Graphs). *It is su cient to use $O(m \log n)$* EC *queries to learn a hidden graph on n vertices. [67]*

Now we are ready to define the query types we used in this work.

Definition 5.3.3 (Bipartite Independent Set Queries $\mathrm{BIS}(A, B)$). *Given two disjoint subsets $A, B \subseteq V$,* $\mathrm{BIC}(A, B)$ *returns 1 if there is at least an edge with one endpoint in A and one in B or 0 if there is no edge between A and B.*

Definition 5.3.4 (Bipartite Edge Counting Queries ($\mathrm{BEC}(A, B)$)). *Given two disjoint subsets $A, B \subseteq V$,* $\mathrm{BEC}(A, B)$ *returns the number of edges that have one endpoint in A and one in B.*

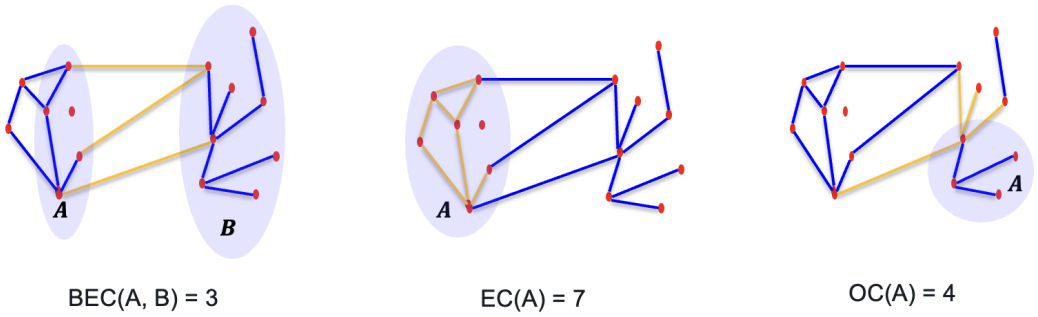


Figure 4. A visual example of how OC, EC and BEC queries work.

Definition 5.3.5 (Outgoing Counting Queries (OC)). *Given a subset A of vertices $A \subseteq V$, $\mathrm{OC}(A)$ returns the number of edges with one endpoint in A and the other end point in $\bar{A}$.*

In the next section we show how to simulate these queries via the other queries.

5.4 Preliminary results

Lemma 5.4.1 (EC and OC). *Let $A \subseteq V$ be chosen such that each $v_i \in V$ will be included in A with probability $1/2$, independently of each other v_i. Then,*

$$\mathbb{E}[\mathrm{OC}(A)] = \frac{1}{2}\mathrm{EC}(V), \tag{5.1}$$

where the expectation is taken over the randomness in the choice of the query.

Proof. Let $e_{i,j} \in E$. The probability that $v_i \in A$ and $v_j \in \bar{A}$ is $1/4$ because each vertex ends up in A or $\bar{A}$ with probability $1/2$ independently. Similarly the probability that $v_j \in A$ and $v_i \in \bar{A}$

is also 1/4. Since these are mutually exclusive events we have the probability of that edge being detected as $1/4 + 1/4 = 1/2$. Let $\mathbb{1}_{(e,A)}$ be the indicator variable that takes value 1 when e has one endpoint in A and one in $\bar{A}$, and hence for all $e \in E$, $\mathbb{E}[\mathbb{1}_{(e,A)}] = \frac{1}{2}$.

$$\begin{aligned}\mathbb{E}[\mathrm{OC}(A)] =& \mathbb{E}[\sum_{e\in E} \mathbb{1}_{(e,A)}] \\ =& \sum_{e\in E} \mathbb{E}[\mathbb{1}_{(e,A)}] \\ =& \sum_{e\in E} 1/2 = \frac{1}{2}|E| \\ =& \frac{1}{2}\mathrm{EC}(V).\end{aligned}$$

□

Lemma 5.4.2 (Simulation of BEC with OC). *A* BEC *query can be simulated by* 3 OC *queries.*

Proof. We will perform the following queries: $\mathrm{OC}(A)$, $\mathrm{OC}(B)$, $\mathrm{OC}(A \cup B)$, We can write the results as a function of the sets defined above.

$$\begin{aligned}\mathrm{OC}(A) &= |A_{\mathrm{out}}| + |AB| \\ \mathrm{OC}(B) &= |B_{\mathrm{out}}| + |AB| \\ \mathrm{OC}(A \cup B) &= |A_{\mathrm{out}}| + |B_{\mathrm{out}}|\end{aligned} \tag{5.2}$$

Where $|AB|$ is the number of edges with one endpoint in A and the other in B, and $|A_{\text{out}}|$, $|B_{\text{out}}|$ are, respectively, the numbers of edges with one end point in A (B, resp.) and the other end point in $\bar{A}\backslash B$ ($\bar{B}\backslash A$, resp.).

We now examine the quantity,

$$\begin{aligned}\frac{\text{OC}(A)+\text{OC}(B))-(\text{OC}(A\cup B))}{2} =& \frac{|A_{\text{out}}|+|AB|+|B_{\text{out}}|+|AB|-|A_{\text{out}}|-|B_{\text{out}}|}{2}\\ =&\text{BEC}(A,B)\\ =&|AB|.\end{aligned}$$

□

Lemma 5.4.3 (Simulation of EC with BEC). *We can simulate one* EC *query with* $O(n)$ BEC *queries.*

We proceed with the proof by using the divide and conquer.

Proof. We start by dividing the vertex set V of graph G into two equal size partition V_1 and V_2 where $|V_1| = n/2$ and $|V_2| = n/2$. Now we can claim the count of the edges can be computed as bellow:

$$|E| = |E(V_1)| + |E(B_2)| + |E_{\text{crossing}}| \tag{5.3}$$

We can find the number of crossing edges by running a BEC query on V_1 and V_2. To find the count of edges on each partition we continue the recursion on each side until there is a single vertex left in each partition and the only thing that left to count is crossing.

$$T(n) = 2T(n/2) + 2 \tag{5.4}$$

By master theorem we have $T(n) = \theta(n^{\log_2^2}) = \theta(cn)$. □

Based on [67], we know that we can learn any arbitrary graph with $|E| \log n$ EC queries. From lemma 5.4.1 we know that we can simulate an EC query with $O(n)$ BEC queries so $n\,|E| \log n$ BEC queries are sufficient to learn any graph. If the graph is sparse this result is better than n^2.

Corollary 5.4.4 (of 5.4.2 and 5.4.1). *We can simulate one* EC *query with* $O(n)$ OC *queries.*

5.5 A divide and conquer approach for learning graphs

In this section we will introduce an algorithm that learns any hidden graph $G = (V, E)$ with $O(m \log n)$ BEC queries efficiently in the size of an input.

5.5.1 Algorithm

As you can see in the above algorithm, the bottleneck is to find and learn the crossing edges. Therefore, we provide `CrossLearn` where we learn the crossing by binary searching.

5.5.2 Proof of correctness

Theorem 5.5.1. *Given a graph* $G = (V, E)$*, with* $|V| = n$ *and* $|E| = m$*,* `EdgeLearn` *learns its edges using* $O(m \log n)$ BEC *queries in expectation and in time polynomial in* m *and* n*.*

Algorithm 1 `EdgeLearn`(S), An algorithm for learning the edges in $S \subseteq V$ using BEC queries.

Require: $S \subseteq V$

if $|S| = 1$ *or* $|S| = 0$ **then**
  return $(S, \varnothing)$

end

Divide S uniformly at random into S_l and S_r with $|S_l| = \lfloor n/2 \rfloor$ and $|S_r| = \lceil n/2 \rceil$

$E_l =$ `EdgeLearn`(S_l)

$E_r =$ `EdgeLearn`(S_r)

$E_{l,r} =$ `CrossLearn`(S_l, S_r)

$E_S = E_l \cup E_r \cup E_{l,r}$ output (S, E_S)

Proof. We start by partitioning the n nodes u.a.r into two sets. `EdgeLearn` recursively learns the edges by partitioning the vertices and calling `CrossLearn` to learn the crossing edges. `CrossLearn` itself is a recursive algorithm that uses binary search to learn the cross edges, and the probability of each edge ending up within a given partition is $\binom{\frac{n}{2}}{2}/n \approx \frac{1}{4}$ and being the probability of an edge being crossing partitions is $(\frac{n}{2})(\frac{n}{2})/\binom{n}{2} \approx \frac{1}{2}$. In expectation, half of the edges crossing the partition and a quarter end up in each partition. Also the size of each partition is $n/2$ in expectation.

Let $T(n, m)$ be the expected number of queries used by `EdgeLearn`. We can write the recursion as follows:

$$T(n, m) = 2T\left(\frac{n}{2}, \frac{m}{4}\right) + f(m, n), \tag{5.5}$$

Algorithm 2 `CrossLearn`(S_1, S_2) Learns all cross edges between disjoint sets S_1 and S_2 using BEC queries.

Require: $S_1 \cup S_2 = V$ **if** $|S_1| = |S_2| = 1$, $v_1 \in S_1$ ***and*** $v_2 \in S_2$ **then**

  if $\mathrm{BEC}(v_1, v_2) = 1$ **then**
    `return` $E = \{e_{1,2}\}$
  end
  else
    `return` $\varnothing$
  end

end

Split $S_1 \rightarrow \left|S_1^{'}\right| = \left\lfloor \frac{S_1}{2} \right\rfloor, |S_1^{"}| = \left\lceil \frac{S_1}{2} \right\rceil$

Split $S_2 \rightarrow \left|S_2^{'}\right| = \left\lfloor \frac{S_2}{2} \right\rfloor, |S_2^{"}| = \left\lceil \frac{S_2}{2} \right\rceil$

if $\mathrm{BEC}(S_1^{'}, S_2^{'}) \neq 0$ **then**
  $E_1 = \mathrm{CrossLearn}(S_1^{'}, S_2^{'})$
end

if $\mathrm{BEC}(S_1^{'}, S_2^{''}) \neq 0$ **then**
  $E_2 = \mathrm{CrossLearn}(S_1^{'}, S_2^{''})$
end

if $\mathrm{BEC}(S_1^{''}, S_2^{'}) \neq 0$ **then**
  $E_3 = \mathrm{CrossLearn}(S_1^{"}, S_2^{'})$
end

if $\mathrm{BEC}(S_1^{''}, S_2^{''}) \neq 0$ **then**
  $E_4 = \mathrm{CrossLearn}(S_1^{''}, S_2^{''})$
end

`return` $E = E_1 \cup E_2 \cup E_3 \cup E_4$

where $f(m,n)$ is the query complexity of `CrossLearn`. To analyze the quantity $f(m,n)$ we observe that `CrossLearn` takes $O(m \log n)$ queries to learn the crossing edges. therefore we have

$$T(n,m) = 2T\left(\frac{n}{2}, \frac{m}{4}\right) + O\left(\frac{m}{2}\log n\right) \tag{5.6}$$

By substitution for $2T\left(\frac{n}{2}, \frac{m}{4}\right)$ we can get:

$$\begin{aligned}
\sum_{i=0}^{\log n} 2^i \frac{m}{2^{2i}} \log \frac{n}{2^i} &= \sum_{i=0}^{\log n} \frac{m}{2^i} \log \frac{n}{2^i} \\
&\leqslant \sum_{i=0}^{\log n} \frac{m}{2^i} \log n \\
&\leqslant m \log n \sum_{i=0}^{\infty} \frac{1}{2^i} \\
&\leqslant m \log n.
\end{aligned} \tag{5.7}$$

□

Corollary 5.5.2. *Any tree can be learned using* $O(n \log n)$ BEC *queries in expectation and in polynomial time.*

5.6 Graph verification and learning

In this section, we give a lemma that says that a random BEC query is able to distinguish two graphs with probability at least 1/4. This can be viewed as a graph verification result [74] in the following sense: imagine a graph G is presented to a verifier that needs to decide whether this graph is equal to the target graph known to a BEC oracle. A random query can be given to the oracle and then compared to the same query simulated on the graph to be verified. If (and

only if) the graphs are different, this difference will be exposed with probability $\geqslant 1/4$, and this detection probability can be boosted to any success probability $1-\epsilon$ by repeating a random query $O(\log(1/\epsilon))$ times. [1] We use this verification idea to come up with a learning algorithm as explained below.

Lemma 5.6.1. *Let $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ be graphs such that $E_1 \neq E_2$. Let $U \subset V$ be a uniformly random subset of vertices in V. Then,*

$$\Pr[\mathrm{BEC}_{G_1}(U, V-U) \neq \mathrm{BEC}_{G_2}(U, V-U)] \geqslant 1/4$$

Proof. Let $G_\triangle$ be the symmetric difference of G_1 and G_2, in particular $G_\triangle = (V, E_1 \triangle V_2)$.

The graph G either has an odd degree vertex v or all the vertices have even degrees. We proceed with this proof by bounding the probability of failure both in the presence of the odd degree vertex v and in the absence of it.

The failure event in both cases is defined when BEC cannot distinguish between G_1 and G_2 which happens when $\mathrm{BEC}_{G_1}(U, V-U) = \mathrm{BEC}_{G_2}(U, V-U)$.

Let $N(.)$ be the neighborhood of a vertex. Let a be $|N_{G_1}(v)|$ and b be $|N_{G_2}(v)|$. Therefore, the degree of v is defined $|N_{G_\triangle}(v)| = a + b$.

Case 1: An odd degree vertex v exists.

---

[1] Reyzin and Srivastava [67] gave a similar graph verification result to the one we present in Lemma 5.6.1 for EC queries.

We start by partitioning vertex set V uniformly at random into two sets U and $V-U$ and we fix an ordering in this assignment such v is assigned to its partition last. Our analysis examines what happens right before v is assigned to U or not.

We first define some quantities before v is assigned. Let us call $x = \text{BEC}_{G_1 \setminus G_2}(U-v, V-U-v)$ and $y = \text{BEC}_{G_2 \setminus G_1}(U-v, V-U-v)$.

We have:

- Event 1: $v \in U$. Then call $\text{BEC}_{G_1 \setminus G_2}(U, V-U) = a' + x$ and $\text{BEC}_{G_2 \setminus G_1}(U, V-U) = b' + y$.
- Event 2: $v \notin U$. Then call $\text{BEC}_{G_1}(U, V-U) = a'' + x = a - a' + x$ and $\text{BEC}_{G_2}(U, V-U) = b'' + y = b - b' + y$.

Then, one of the following two things can happen:

1. $x = y$

   The failure events are 1) $x + a' = y + b'$, which is equivalent to $a' = b'$ and 2) $x + a'' = y + b''$, which is equivalent to $a'' = b''$, but they cannot happen at the same time. If $a' = b'$ and $b'' = a'' \rightarrow a - a' = b - b'$ happen at the same time, by adding up these two equations we will have $a = b$ that contradicts our initial assumption that the vertex has an odd degree.

2. $x \neq y$

   Same line of reasoning as the previous case can be applied here. The failure events are $x + a' = y + b'$ or $x + a'' = y + b''$ but these two cases cannot happen at the same time. If they happen at the same time then by adding these two equations we will have

$2x + a' + a'' = 2y + b' + b'' \rightarrow 2x + a = 2x' + b \rightarrow a - b = 2(x + y)$. This contradicts v having odd degree, i.e. $a + b$ being odd.

Therefore, in the presence of the odd degree vertex, with probability at least 1/2 (which is $\geqslant 1/4$) we can distinguish G_1 and G_2.

Case 2: All vertices in the symmetric difference graph have even degrees.

Consider two adjacent vertices u, v. We again consider the analysis when all but u and v are assigned to their respective partitions.

$\text{BEC}_{G_1/G_2}(U - \{u, v\}, V - U - \{u, v\}) = x$ and $\text{BEC}_{G_2/G_1}(U - \{u, v\}, V - U - \{u, v\}) = y$ and call $K = x + y$.

Before moving to the proof we need to define some notation:

Definition 5.6.2. *Let a cut be a partition (A, B) of the vertex set V and let $E(A, B)$ be the set of edges with one endpoint in A and the other in B.*

After partitioning u and v they either end up in a similar partition or a different partition.

For u and v are in the same partition we have:

- If $v \in U$ and $u \in U$ then we define a'_s [1] and b'_s to be the number of additional edges added to the cut such that:

$$|E_{G_1/G_2}(U + \{u, v\}, V - U)| = x + a'_s$$

---

[1] Note that s stands for u and v ending up in the same partition.

and

$$|E_{G_2/G_1}(U, V - U + \{u, v\})| = y + b'_s$$

.

- If $v \notin U$ and $u \notin U$ then we define a''_s and b''_s to be the number of additional edges added to the cut such that:

$$|E_{G_1/G_2}(U, V - U + \{u, v\})| = x + a''_s$$

and

$$|E_{G_2/G_1}(U, V - U + \{u, v\})| = y + b''_s$$

Vertices u and v end up in different partitions:

- If $v \in U$ and $u \notin U$ then we define a'_d [1] and b'_d to be the number of additional edges added to the cut such that:

$$|E_{G_1/G_2}(U + \{v\}, V - U + \{u\})| = x + a'_d$$

and

$$|E_{G_2/G_1}(U + \{v\}, V - U + \{u\})| = y + b'_d$$

.

---

[1] Note that d stands for u and v ending up in the different partition.

- If $v \notin U$ and $u \in U$ then we define a''_d and b''_d to be the number of additional edges added to the cut such that:

$$|E_{G_1/G_2}(U + \{u\}, V - U + \{v\})| = x + a''_d$$

and

$$|E_{G_2/G_1}(U + \{u\}, V - U + \{v\})| = y + b''_d$$

.

Regardless of the values of x and y we can define the failure events, i.e. that the answers to the BEC query end up the same for both graphs, for the 4 cases, each of which happen with probability 1/4, where u and v can be assigned, as follows:

$$\begin{aligned} x + a'_s &= y + b'_s \\ x + a''_s &= y + b''_s \end{aligned} \tag{5.8}$$

$$\begin{aligned} x + a'_d &= y + b'_d \\ x + a''_d &= y + b''_d \end{aligned} \tag{5.9}$$

Before adding the last two vertices, either BEC detected similar counts for both G_1 and G_2 or BEC detected different counts for G_1 and G_2:

1. $x = y$

   Now we can rewrite the equation for similar and dissimilar case and we get $a = b$ for similar and $a = b + 1$ for dissimilar case. We reach to contradiction since these two equation cannot hold at the same time. Hence, with probability $\geqslant 1/4$ we can detect G_1 from G_2.

2. $x \neq y$

   We can rewrite Equation 5.8 and Equation 5.9 by adding the formulas that we have for similar case and the dissimilar case. Then, we get $2(x - y) = b - a$ for similar case and $2(x - y) = b - a + 1$ for dissimilar case. This implies if $b - a$ is even then $b - a + 1$ cannot be even so these two events cannot happen at the same time. Therefore, we conclude with probability $1/4$ the random BEC query produces different answer for G_1 and G_2.

□

Theorem 5.6.3. *Let $\mathcal{G}$ be a set of graphs. We can learn over $\mathcal{G}$ using $O(\log(|\mathcal{G}|))$ BEC queries in expectation.*

Proof. Let $\mathcal{G}_t$ be the set of uneliminated graphs at step t, and let $\mathcal{G}_0 = \mathcal{G}$. At each time step t we ask a random BEC query. By linearity of expectation, we can eliminate at least $\geqslant 1/4$ of the incorrect solutions (i.e. $|\mathcal{G}_t| - 1$ of the possible graphs) in expectation each iteration t using Lemma 5.6.1 by iterating over all un-eliminated graphs and removing those that are inconsistent

with the current query. Therefore, we can learn the target using $\log_{4/3}(|\mathcal{G}|) = O(\log|\mathcal{G}|)$ queries. □

Note this has improved query complexity over 5.5.1 at a (severe) cost to running time. In particular, the running time is $\Theta(|\mathcal{G}|)$, which is inefficient for exponential families.

Corollary 5.6.4. *We can learn any tree on n nodes using $O(n\log n)$ BEC queries.*

Proof. By Cayley's theorem we know that for every integer $n > 0$, the number of trees on n labeled vertices is n^{n-2}. Getting the logarithm of $\log(n^{n-2}) = O(n\log n)$. □

The above bound matches the query bound that we obtain for `EdgeLearn` in the case of trees, but it is not efficient because it would have to enumerate through all of the n^{n-2} trees.

Corollary 5.6.5. *We can learn any star on n nodes using $O(\log n)$ BEC queries.*

Proof. Only n different stars can exist on n nodes since each node can be center once and then the rest should be leaves. Therefore, we $O(\log n)$ BEC queries are sufficient for learning stars. □

From this corollary we can conclude that there exists an efficient algorithm for learning any given star with $O(\log n)$ queries. This algorithm runs in polynomial time because the class of stars contains only n elements and can be efficiently enumerated.

Corollary 5.6.6. *We can learn any graph on n nodes using $O(n^2)$ BEC queries.*

Proof. There are 2^{n^2} graphs on n nodes so getting the logarithm of that we get n^2. □

This result provides an improved query complexity in comparison to the query complexity of $n^2 \log n$ that we would obtain by using `EdgeLearn` for dense graphs, but it matches query complexity of brute force of querying once per edge, and it's significantly worse in terms of running time.

Our method, however, does sometimes get an advantage in query usage (though not running time) over `EdgeLearn` and brute-force search, as illustrated by the following corollary.

Corollary 5.6.7. *Let $\mathcal{G}$ be a family of cliques minus a path. We can learn over this family by using $O(n \log n)$ BEC queries.*

Proof. We have $\frac{n!}{2}$ undirected path on n vertices. If we have a clique that doesn't have one of these $\frac{n!}{2}$ we can learn it in $O(\log(n!)) = O(n \log n)$ □

As we mentioned before, although this algorithm, unlike `EdgeLearn`, is not efficient in terms of running time, it is better in terms of query complexity. The main application of this result could be when queries are very expensive (e.g. running human phase of pharmaceutical experiments) but computational resources are plenty (e.g. access to parallel computing).

5.6.1 A visual example on learning a graph with BEC queries

In this section we provide a visual example on learning a target path on $n = 4$ vertices with BEC query. There are $\frac{4!}{2} = 12$ possible solutions for 4 nodes.

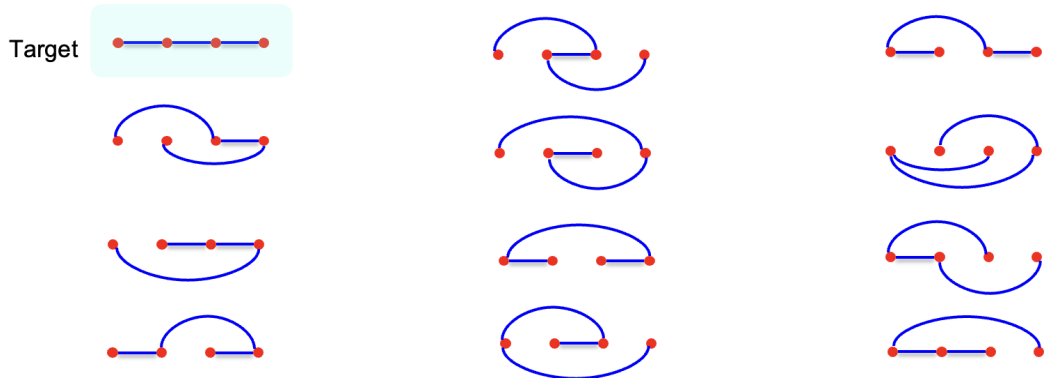


Figure 5. A visualization of all possible paths on $n = 4$ vertices

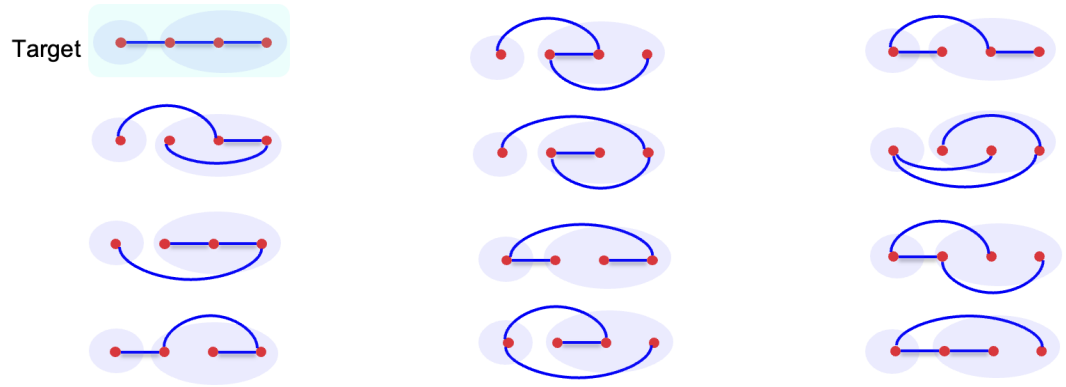


Figure 6. Randomly portioning vertices into two sets A and B.

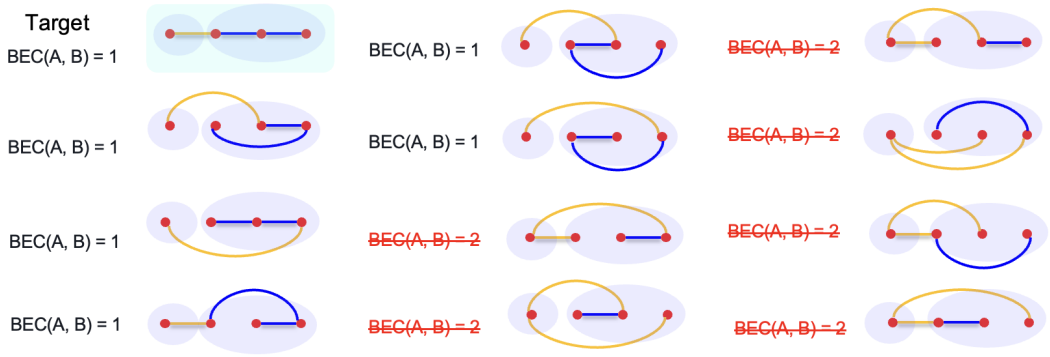


Figure 7. Running the BEC (A,B) and eliminating the solution with different response than oracle.

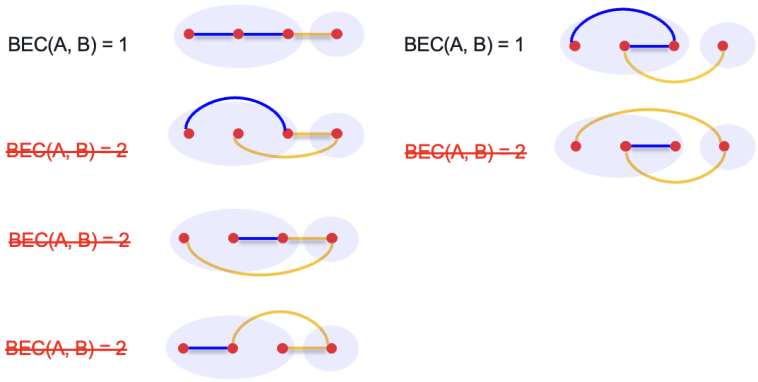
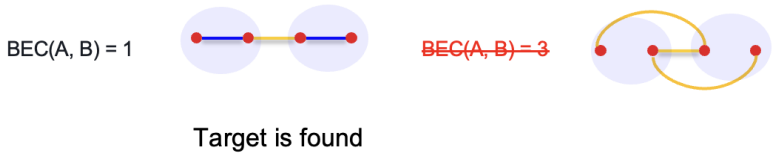


Figure 8. Repeating the random partitioning, running BEC(A, B) and elimination steps.



Target is found

Figure 9. Target path is found by running 3 BEC queries.

CITED LITERATURE

1. Weinberger, K. Q. and Saul, L. K.: Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research* , 10(2), 2009.

2. Weakly supervised. `https://snorkel.ai/weak-supervision`. Accessed: 2022-7-5.

3. Ihara, D., Mohammadi, N., and Sidiropoulos, A.: Algorithms for metric learning via contrastive embeddings. In *35th International Symposium on Computational Geometry (SoCG 2019)* . Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

4. Fan, B., Centurion, D. I., Mohammadi, N., Sgherzi, F., Sidiropoulos, A., and Valizadeh, M.: Learning lines with ordinal constraints. *arXiv preprint arXiv:2004.13202* , 2020.

5. Ihara, D., Mohammadi, N., and Sidiropoulos, A.: Learning mahalanobis metric spaces via geometric approximation algorithms. 2019.

6. Freitag, J., Mohammadi, N., Potukuchi, A., and Reyzin, L.: On the geometry of stable steiner tree instances. *arXiv preprint arXiv:2109.13457* , 2021.

7. Matousek, J.: *Lectures on discrete geometry* , volume 212. Springer Science & Business Media, 2013.

8. Sidiropoulos, A.: Computational metric embeddings. Doctoral dissertation, Massachusetts Institute of Technology, 2008.

9. Bilu, Y. and Linial, N.: Are stable instances easy? *Combinatorics, Probability and Computing* , 21(5):643–660, 2012.

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

11. Barton, N. H.: The role of hybridization in evolution. *Molecular ecology* , 10(3):551–568, 2001.

12. Valiant, L. G.: A theory of the learnable. *Communications of the ACM* , 27(11):1134–1142, 1984.

13. Mohri, M., Rostamizadeh, A., and Talwalkar, A.: *Foundations of machine learning* . MIT press, 2018.

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

15. Angluin, D., Chen, J., and Warmuth, M.: Learning a hidden hypergraph. *Journal of Machine Learning Research* , 7(10), 2006.

16. 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.

17. Bouvel, M., Grebinski, V., and Kucherov, G.: Combinatorial search on graphs motivated by bioinformatics applications: A brief survey. In *International Workshop on Graph-Theoretic Concepts in Computer Science* , pages 16–27. Springer, 2005.

18. Grebinski, V. and Kucherov, G.: Optimal reconstruction of graphs under the additive model. *Algorithmica* , 28(1):104–124, 2000.

19. King, V., Zhang, L., and Zhou, Y.: On the complexity of distance-based evolutionary tree. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* , page 444. SIAM, 2003.

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

21. Aho, A. V. and Hopcroft, J. E.: *The design and analysis of computer algorithms* . Pearson Education India, 1974.

22. Shakhnarovich, G.: Learning task-specific similarity. Doctoral dissertation, Massachusetts Institute of Technology, 2005.

23. Kulis, B. et al.: Metric learning: A survey. *Foundations and Trends® in Machine Learning* , 5(4):287–364, 2013.

24. Ihara, D., Mohammadi, N., Sgherzi, F., and Sidiropoulos, A.: Robust mahalanobis metric learning via geometric approximation algorithms. *CoRR* , abs/1905.09989, 2019.

25. Nayyeri, A. and Raichel, B.: Reality distortion: Exact and approximate algorithms for embedding into the line. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on* , pages 729–747. IEEE, 2015.

26. Nayyeri, A. and Raichel, B.: A treehouse with custom windows: Minimum distortion embeddings into bounded treewidth graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19* , ed. P. N. Klein, pages 724–736. SIAM, 2017.

27. Bădoiu, M., Chuzhoy, J., Indyk, P., and Sidiropoulos, A.: Low-distortion embeddings of general metrics into the line. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing* , pages 225–233. ACM, 2005.

28. Badoiu, M., Dhamdhere, K., Gupta, A., Rabinovich, Y., Räcke, H., Ravi, R., and Sidiropoulos, A.: Approximation algorithms for low-distortion embeddings into low-dimensional spaces. In *SODA* , volume 5, pages 119–128. Citeseer, 2005.

29. Carpenter, T., Fomin, F. V., Lokshtanov, D., Saurabh, S., and Sidiropoulos, A.: Algorithms for low-distortion embeddings into arbitrary 1-dimensional spaces. In *34th International Symposium on Computational Geometry (SoCG 2018)* . Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

30. Fellows, M. R., Fomin, F. V., Lokshtanov, D., Losievskaja, E., Rosamond, F. A., and Saurabh, S.: Distortion is fixed parameter tractable. In *International Colloquium on Automata, Languages, and Programming* , pages 463–474. Springer, 2009.

31. Badoiu, M.: Approximation algorithm for embedding metrics into a two-dimensional space. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* , pages 434–443. Society for Industrial and Applied Mathematics, 2003.

32. Dhamdhere, K., Gupta, A., and Ravi, R.: Approximation algorithms for minimizing average distortion. *Theory Comput. Syst.* , 39(1):93–111, 2006.

33. Rabinovich, Y.: On average distortion of embedding metrics into the line and into l1. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing* , pages 456–462, 2003.

34. Indyk, P., Matoušek, J., and Sidiropoulos, A.: Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry, Second Edition.* , eds. J. E. Goodman, J. O'Rourke, and C. D. Toth. Chapman and Hall/CRC, 2017.

35. Alon, N., Bădoiu, M., Demaine, E. D., Farach-Colton, M., Hajiaghayi, M., and Sidiropoulos, A.: Ordinal embeddings of minimum relaxation: general properties, trees, and ultrametrics. *ACM Transactions on Algorithms (TALG)* , 4(4):1–21, 2008.

36. Bădoiu, M., Demaine, E. D., Hajiaghayi, M., Sidiropoulos, A., and Zadimoghaddam, M.: Ordinal embedding: Approximation algorithms and dimensionality reduction. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques* , pages 21–34. Springer, 2008.

37. Chor, B. and Sudan, M.: A geometric approach to betweenness. *SIAM Journal on Discrete Mathematics* , 11(4):511–523, 1998.

38. Opatrny, J.: Total ordering problem. *SIAM Journal on Computing* , 8(1):111–114, 1979.

39. Ailon, N. and Alon, N.: Hardness of fully dense problems. *Information and Computation* , 205(8):1117–1129, 2007.

40. Karpinski, M. and Schudy, W.: Approximation schemes for the betweenness problem in tournaments and related ranking problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques* , pages 277–288. Springer, 2011.

41. Charikar, M., Guruswami, V., and Manokaran, R.: Every permutation csp of arity 3 is approximation resistant. In *2009 24th Annual IEEE Conference on Computational Complexity* , pages 62–73. IEEE, 2009.

42. Makarychev, Y.: Simple linear time approximation algorithm for betweenness. *Operations research letters* , 40(6):450–452, 2012.

43. Toth, C. D., O'Rourke, J., and Goodman, J. E.: *Handbook of discrete and computational geometry* . Chapman and Hall/CRC, 2017.

44. Kenyon-Mathieu, C. and Schudy, W.: How to rank with few errors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing* , pages 95–103, 2007.

45. Bilu, Y. and Linial, N.: Are stable instances easy? *Comb. Probab. Comput.* , 21(5):643–660, 2012.

46. Awasthi, P., Blum, A., and Sheffet, O.: Center-based clustering under perturbation stability. *Inf. Process. Lett.* , 112(1-2):49–54, 2012.

47. Balcan, M. and Liang, Y.: Clustering under perturbation resilience. *SIAM J. Comput.* , 45(1):102–155, 2016.

48. Ben-David, S. and Reyzin, L.: Data stability in clustering: A closer look. *Theor. Comput. Sci.* , 558:51–61, 2014.

49. Makarychev, K., Makarychev, Y., and Vijayaraghavan, A.: Bilu-linial stable instances of max cut and minimum multiway cut. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014* , ed. C. Chekuri, pages 890–906. SIAM, 2014.

50. Mihalák, M., Schöngens, M., Srámek, R., and Widmayer, P.: On the complexity of the metric TSP under stability considerations. In *SOFSEM 2011: Theory and Practice of Computer Science - 37th Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 22-28, 2011. Proceedings* , eds. I. Cerná, T. Gyimóthy, J. Hromkovic, K. G. Jeffery, R. Královic, M. Vukolic, and S. Wolf, volume 6543 of *Lecture Notes in Computer Science* , pages 382–393. Springer, 2011.

51. Ackerman, M. and Ben-David, S.: Clusterability: A theoretical study. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, AISTATS 2009, Clearwater Beach, Florida, USA, April 16-18, 2009* , eds. D. A. V. Dyk and M. Welling, volume 5 of *JMLR Proceedings* , pages 1–8. JMLR.org, 2009.

52. Balcan, M., Blum, A., and Gupta, A.: Approximate clustering without the approximation. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009* , ed. C. Mathieu, pages 1068–1077. SIAM, 2009.

53. Alabdulmohsin, I. M.: Algorithmic stability and uniform generalization. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada* , eds. C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, pages 19–27, 2015.

54. Ben-David, S., von Luxburg, U., and Pál, D.: A sober look at clustering stability. In *Learning Theory, 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings* , eds. G. Lugosi and H. U. Simon, volume 4005 of *Lecture Notes in Computer Science* , pages 5–19. Springer, 2006.

55. Fan, B., Ihara, D., Mohammadi, N., Sgherzi, F., Sidiropoulos, A., and Valizadeh, M.: Learning lines with ordinal constraints. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020)* . Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

56. Liu, T., Lugosi, G., Neu, G., and Tao, D.: Algorithmic stability and hypothesis complexity. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017* , eds. D. Precup and Y. W. Teh, volume 70 of *Proceedings of Machine Learning Research* , pages 2159–2167. PMLR, 2017.

57. Meulemans, W., Speckmann, B., Verbeek, K., and Wulms, J.: A framework for algorithm stability and its application to kinetic euclidean msts. In *LATIN 2018: Theoretical Informatics - 13th Latin American Symposium, Buenos Aires, Argentina, April 16-19, 2018, Proceedings* , eds. M. A. Bender, M. Farach-Colton, and M. A. Mosteiro, volume 10807 of *Lecture Notes in Computer Science* , pages 805–819. Springer, 2018.

58. Chlebík, M. and Chlebíková, J.: The steiner tree problem on graphs: Inapproximability results. *Theor. Comput. Sci.* , 406(3):207–214, 2008.

59. Arora, S.: Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. *J. ACM* , 45(5):753–782, 1998.

60. Karp, R. M.: *Reducibility among Combinatorial Problems* , pages 85–103. Boston, MA, Springer US, 1972.

61. Vazirani, V. V.: *Approximation Algorithms* . Berlin, Heidelberg, Springer-Verlag, 2001.

62. Gilbert, E. N. and Pollak, H. O.: Steiner minimal trees. *SIAM Journal on Applied Mathematics* , 16(1):1–29, 1968.

63. Regier, T., Khetarpal, N., and Majid, A.: Inferring semantic maps. *Linguistic Typology* , 17(1):89–105, 2013.

64. Malt, B. C. and Majid, A.: How thought is mapped into words. *Wiley Interdisciplinary Reviews: Cognitive Science* , 4(6):583–597, 2013.

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

66. 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.

67. 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.

68. Grebinski, V. and Kucherov, G.: Reconstructing a hamiltonian cycle by querying the graph: Application to dna physical mapping. *Discrete Applied Mathematics* , 88(1-3):147–165, 1998.

69. Wang, Z. and Honorio, J.: Reconstructing a bounded-degree directed tree using path queries. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)* , pages 506–513. IEEE, 2019.

70. Janardhanan, M. V. and Reyzin, L.: On learning a hidden directed graph with path queries, 2020.

71. Beame, P., Har-Peled, S., Ramamoorthy, S. N., Rashtchian, C., and Sinha, M.: Edge estimation with independent set oracles. *ACM Transactions on Algorithms (TALG)* , 16(4):1–27, 2020.

72. Addanki, R., McGregor, A., and Musco, C.: Non-adaptive edge counting and sampling via bipartite independent set queries. *arXiv preprint arXiv:2207.02817* , 2022.

73. Goldreich, O., Goldwasser, S., and Ron, D.: Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)* , 45(4):653–750, 1998.

74. Reyzin, L.: *Active learning of interaction networks* . Yale University, 2009.

VITA

NAME: Neshat Mohammadi

EDUCATION: Ph.D., Computer Science, University of Illinois at Chicago, Chicago, Illinois, 2022.

M.Sc., Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, Illinois, 2017.

B.Sc., Electrical Engineering, Electronics, Shahid Beheshti University, Tehran, Iran, 2009.

ACADEMIC EXPERIENCE: Research Assistant, Department of Computer Science, University of Illinois at Chicago.

Teaching Assistant, Department of Computer Science, University of Illinois at Chicago:

- CS 401: Computer Algorithm I, Spring: 2020, 2022 and Fall: 2020, 2021, 2022.
- CS 107: Introduction to Computer Programming, Fall 2017.