THE UNIVERSITY OF CHICAGO

NEW ALGORITHMIC RESULTS IN CLUSTERING AND PARTITIONING

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY

ERASMO TANI

CHICAGO, ILLINOIS
AUGUST 2024

In loving memory of

Giuseppe Merello

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This thesis would not have been possible without the help and support of many people. First, I wish to thank my advisor, Lorenzo Orecchia. Lorenzo not only spent years supporting me through my research endeavours, but he also taught me most of what I know about spectral graph theory and optimization. I benefited greatly from his unique perspective on the process of mathematical research. I will never cease to be amazed by his ability to turn a dubious scribble on a whiteboard into a bona fide mathematical proof. I also wish to thank my research collaborators Konstantinos Ameranis, Antares Chen, Olga Medrano Martín del Campo, and Max Ovsiankin, with whom I have shared hours discussing projects and solving problems at the board.

A special thanks also goes to all the people who have dedicated a large part of their scarce and precious time to mentoring me over the years, often without any promise of return on their investment. First, I would like to thank Alina Ene, who served as my PhD co-advisor during my years at Boston University. In my short time at BU, Alina provided me with invaluable guidance and career advice. Then, I wish to thank Yury Makarychev and Ali Vakilian, both of whom, in addition to patiently advising me in my research, agreed to serve on my dissertation committee. I learned a tremendous amount from working with them, and it is largely thanks to their incredible commitment to mentorship. Finally, during my later PhD years, I had the opportunity to receive guidance from Maurizio Malpede and Soheil Shayegh who helped me broaden my perspective on research and academic life.

A personal shoutout goes to my favorite coworker Adela DePavia. Adela and I ("The Dynamic Duo", as we came to be known in some circles) shared several research projects and braved the turbulent waters of graduate school together. I cherish every argument over what problems to work on, how to best write our papers, or whether "Gérard Depardieu was really in that movie".

I was also lucky to be surrounded by many other amazing colleagues both at Boston

University and at the University of Chicago. In particular, at BU, I was glad to share my PhD journey with Gavin Brown, Laura Greige, Andy Huynh, Ali Raza ($\times 2$), Parul Sohal, Konstantinos Sotiropoulos, Isidora Tourni, and Nithin Varma. At UChicago I had the pleasure of joining a PhD cohort that included Alex Hoover, Neng Huang, Jafar Jafarov, Jonathan Liu, Tushant Mittal, and Ryan *S.Y.G.* Robinett.

During my time at UChicago, I often found refuge with the Computational and Applied Math group, and I am extremely thankful to them for always making me feel welcome. In particular, I wish to thank Andrew Dennehy, Jimmy Lederman, Phillip Lo, Peter Nekrasov, Mark Olson, Sue Parkinson, Abigail Poteshman, Solomon Quinn, Nathan Waniorek, Angela Wang, and Dan Xiang for their friendship.

I also wish to thank my family at home: Alessandra, Marco, and Marzia, who have continued to support me unconditionally for almost thirty years. There is no doubt that I would not be where I am today had it not been for their help. Outside of Italy, I could count on Cora Frederick, Enrico Moiso, and Marco Piccardo to be my home away from home.

Last, but certainly not least, I wish to thank Rachel Thomas. Over the course of the past thirteen years, Rachel has gone from being a most trusted friend to my life partner, and I feel truly privileged to have met her. Through the highs and lows of life, across three continents, Rachel has been there for me throughout, as the one and only certainty in an otherwise uncertain world.

Like a lighthouse in the night.

Like a flower in the desert.

ABSTRACT

Clustering and partitioning tasks have found widespread applications across computing. In machine learning, clustering represents the quintessential unsupervised learning task: grouping similar data points to discover structure in data. In operations research and combinatorial optimization, one is often interested in finding bottlenecks in a network, to identify possible weakness and points of failure.

In this work, we discuss recent progress in better understanding computational aspects of clustering and partitioning. Our primary goal is establishing formal mathematical guarantees on the performance of clustering algorithms, as well as proving impossibility results to determine the inherent hardness of the problems we consider.

In the first part of the thesis, we discuss graph partitioning tasks, focusing on the theory behind finding small vertex separators: few vertices which, when removed, disconnect the graph into large pieces. We design approximation algorithms for this problem, based on rounding natural convex relaxations. We also outline a recently uncovered connection between this problem and the fastest mixing random walk process on a graph with a target stationary distribution.

In the second part of this work we discuss some algorithmic results in partitioning hypergraphs. We introduce a new, expressive class of hypergraph cut functions. We then design approximation algorithms for hypergraph generalizations of the minimum conductance cut problem by leveraging and extending techniques from spectral graph theory to the hypergraph regime. We prove our results for all the cut functions in our newly-defined class. In the process, we also improve on a popular primal-dual algorithmic framework for graph partitioning algorithms.

Finally, we address the problem of learning partitions in an interactive way, by querying a same-cluster oracle, which determines whether two points belong to the same cluster. In this context we develop and analyze novel error-resistant algorithms, and provide complementary

lower bounds, showing that our algorithms achieve optimal query complexity. To this end, we develop a new analytic framework based on modeling this task as a Rényi-Ulam liar game.

CHAPTER 1

INTRODUCTION AND PRELIMINARIES

1.1 Introduction

The central paradigm of machine learning is that concepts are determined through examples. In this context, a teacup is not defined in terms of some fixed specification of its material, shape, color, or function, but by its likeness to other teacups. This approach allows for flexibility and adaptability, enabling models to generalize from specific instances to broader categories. As a result, it is not surprising that clustering, the task of dividing a set into groups of similar objects, has taken a central role in modern unsupervised learning.

In this work, we look at some recent results in the algorithmic theory of clustering and partitioning. In particular, we begin by considering the task of partitioning graphs and hypergraphs. Here, a common goal of interest is to partition the vertex set into two relatively large sets which are weakly connected with each other. This idea is usually operationalized by considering ratio-cut objectives, which measure the quality of a cut as the ratio of the strength of the connections across the cut over the size of the smaller of its separated components. Perhaps the most popular such objective is the conductance of a cut. Given a cut $(S, \overline{S})$, the conductance of S is given by:

$$\phi(S) = \frac{w(E(S, \overline{S}))}{\min\{\mathrm{Vol}(S), \mathrm{Vol}(\overline{S})\}},$$

where the numerator is the sum of the weights of the edges crossing the cut, and the denominator is the sum of the degrees of the vertices in S or $\overline{S}$ (whichever is smaller). The problem of finding cuts of small conductance serves as a fundamental primitive in the study of approximation algorithms, where it is used as a subroutine in many divide-and-conquer schemes. Moreover, the minimum conductance of a graph regulates the worst-case mixing time of its natural random walk process. This combinatorial quantity is also closely linked

to the *spectral gap* of a graph, the magnitude of the second smallest eigenvalue of the normalized laplacian matrix. This connection, which is embodied by the celebrated Cheeger's inequality (Alon (1986)), allows one to obtain an $O(\sqrt{\phi(G)})$-approximation to the minimum conductance value $\phi(G)$. A fruitful perspective on this result interprets the spectral gap as the solution to a semidefinite programming relaxation of the problem of computing $\phi(G)$ (see e.g. Orecchia and Vishnoi (2011); Orecchia (2011)). The semidefinite programming approach has culminated in the seminal work of Arora et al. (2009), who, by combining spectral and metric techniques, design a polynomial time $O(\sqrt{\log n})$-approximation algorithm to the problem.

The main shortcoming of the work of Arora et al. is that it only provides polynomial-time guarantees, which are often insufficient for the large instances occurring in the analysis of big data networks. This has inspired numerous efforts to design *fast* approximation algorithms for computing minimum conductance cuts. Khandekar et al. (2009) introduced the cut-matching game: a primal-dual framework that reduces the task of approximating the minimum conductance cut to solving a sequence of s-t max-flow problems, and consequently allowed them to obtain a nearly-linear time $O(\log^2 n)$-approximation algorithm for the problem. Since their work, the framework has been refined (Orecchia et al. (2008); Louis (2010); Nanongkai and Saranurak (2017)) and found numerous applications in the design of graph approximation algorithms (Bernstein et al. (2022); Chuzhoy and Khanna (2019); Chuzhoy et al. (2020); Chuzhoy (2023)).

More recently, much of the theory developed for graph conductance has been extended to minimum vertex separator. This problem is analogous to that of minimum conductance cut, with the key difference that the size of the interface is measured by the magnitude of the vertex set connecting the two parts of the cut, as opposed to the weights of the crossing edges. In Chapter 2, we discuss some recent results on approximating vertex separators. We also highlight how the semidefinite relaxation to this problem is tightly linked to fastest

mixing random walk over the graph with a fixed degree sequence.

In Chapter 3, we consider the problem of finding the minimum ratio-cut in hypergraphs. One key issue that arises when generalizing spectral graph theory techniques to hypergraphs is that, in contrast to graph edges, hyperedges can intersect cuts in many non-trivial ways. Depending on the application, one may want to penalize these differently. This leads to the problem of defining a suitable *cut function* for every hyperedge of the graph. Following the work of Chen et al. (2023a), we introduce polymatroidal cut functions, a broad class of cut functions which encompasses most classes considered in the literature so far. We will give a polynomial time $O(\sqrt{\log n})$-approximation algorithm for the minimum ratio-cut problem for this class of objectives. This yields the most general version of the result of Arora et al. (2009) to date. After that, we describe a method to obtain fast $O(\log n)$-approximation algorithms for the problem, by extending the cut-matching game framework of Khandekar et al. (2009).

In Chapter 4, we consider a clustering problem of a different type. Here, an agent tries to learn a hidden k-partition $\mathcal{C}$ of a finite set V. The agent has access to an oracle that can answer questions of the form: "Are u and v part of the same cluster in $\mathcal{C}$?". This model captures situations in which one may obtain information by querying human agents or by running scientific experiments.

While this problem is well understood in its simplest form (Liu and Mukherjee (2022)), until recently little work had been done on the design of error-resilient algorithms for this task. Following recent progress by DePavia et al. (2023), we discuss the problem in the error-tolerant setting. In particular, we describe algorithms to learn partitions while maintaining tolerance to a bounded amount of error. We consider different error models in both the k-known and the k-unknown settings. Finally, we construct an analytical framework based on Rényi-Ulam liar games and use it to obtain lower bounds, showing that our algorithms achieve the optimal worst-case query complexity for the problem.

1.2 Technical Preliminaries

In this section, we review some fundamental mathematical preliminaries and introduce the relevant notation for the rest of this work.

Partitions Given a positive integer n we denote with $[n]$ the set $\{1, ..., n\}$. Given any finite set V we denote with $\binom{V}{2}$ the set $\{\{i, j\} \subseteq V \mid i \neq j\}$. Moreover, a k-partition of V is a collection of k pairwise disjoint non-empty subsets $\mathcal{C} = \{C_1, ..., C_k\}$ of V such that $\bigcup_{a=1}^{k} C_a = V$. We will denote by n the cardinality $|V|$ of V and we may assume without loss of generality that $V = [n]$. Given any partition $\mathcal{C}$ of V and two elements u and v of V, we write $u \sim_{\mathcal{C}} v$ (resp. $u \not\sim_{\mathcal{C}} v$) for the statement $\exists C \in \mathcal{C}, \{u, v\} \subseteq C$ (resp.$\nexists\, C \in \mathcal{C}, \{u, v\} \subseteq C$). We denote by $[u]_{\mathcal{C}}$ the equivalence class of u with respect to $\mathcal{C}$, i.e. $[u]_{\mathcal{C}}$ is the unique $C \in \mathcal{C}$ containing u. Given two partitions $\mathcal{C}_1$ and $\mathcal{C}_2$ we say $\mathcal{C}_2$ is a *refinement* of $\mathcal{C}_1$ if for every $C \in \mathcal{C}_2$ there is a $C' \in \mathcal{C}_1$ such that $C \subseteq C'$.

Majorization and Schur-Convexity We say a sequence $\boldsymbol{x} = \{x_i\}_{i \in [n]}$ majorizes a sequence $\boldsymbol{y} = \{y_i\}_{i \in [n]}$ (write $\boldsymbol{x} \succeq \boldsymbol{y}$) if, for every $k \in [n]$:

$$\sum_{i=1}^{k} x_{\pi(i)} \geq \sum_{i=1}^{k} y_{\tau(i)}$$

where π and τ are permutations chosen so that $x_{\pi(1)} \geq \cdots \geq x_{\pi(n)}$ and $y_{\tau(1)} \geq \cdots \geq y_{\tau(n)}$. Note that this is sometimes referred to as weak majorization, but we will not need this distinction. A function $f : \mathbb{R}^n \to \mathbb{R}$ is said to be Schur-convex (resp. concave) if, whenever $\boldsymbol{x} \succeq \boldsymbol{y}$, we have $f(\boldsymbol{x}) \geq f(\boldsymbol{y})$ (resp. $f(\boldsymbol{y}) \geq f(\boldsymbol{x})$). Every function which is symmetric and convex (resp. concave) is Schur-convex (resp. concave).

Linear Algebra We use lowercase boldface characters to denote vectors, e.g. $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, and uppercase boldface characters to denote matrices, e.g. $\mathbf{A} \in \mathbb{R}^{n \times n}$. Given a finite set V

with cardinality n, we may identify any function $x : V \to \mathbb{R}$ with a vector $\boldsymbol{x} \in \mathbb{R}^n$ by fixing an ordering $\{v_1, ..., v_n\}$ for the elements of V and letting $\boldsymbol{x}_i = x(v_i)$ for every $i \in [n]$. As such, we may use the notation $\mathbb{R}^n$ and $\mathbb{R}^V$ interchangeably. For any two matrices $\mathbf{A}$ and $\mathbf{B}$, we denote by $\mathbf{A}^\top$ the transpose of $\mathbf{A}$ and by $\mathbf{A} \bullet \mathbf{B}$ the matrix inner product of $\mathbf{A}$ and $\mathbf{B}$, given by $\mathbf{A} \bullet \mathbf{B} = \mathrm{tr}(\mathbf{A}\mathbf{B}^\top)$. We say a matrix $\mathbf{A}$ is positive-semidefinite (PSD, write $\mathbf{A} \succeq 0$) if $\boldsymbol{x}^\top \mathbf{A} \boldsymbol{x} \geq 0$ for every $\boldsymbol{x} \in \mathbb{R}^n$. We may also make use of the PSD ordering, and indicate by $\mathbf{A} \succeq \mathbf{B}$ the fact that $\mathbf{A} - \mathbf{B} \succeq 0$. A positive-definite matrix is one in which the quadratic form is strictly greater than zero at every non-zero vector, for that we write $\mathbf{A} \succ 0$ and the strict ordering $\mathbf{A} \succ \mathbf{B}$ is defined in the obvious way. Any positive matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ induces an inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathbf{A}} \stackrel{\text{def}}{=} \boldsymbol{x}^\top \mathbf{A} \boldsymbol{y}$ on $\mathbb{R}^n$. We denote by $\perp_{\mathbf{A}}$ orthogonality with respect to this inner product so that $\boldsymbol{x} \perp_{\mathbf{A}} \boldsymbol{y}$ if and only if $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathbf{A}} = 0$. Given a collection of vectors $\{\boldsymbol{v}^{(i)}\}_{i \in [n]}$ where each $\boldsymbol{v}^{(i)} \in \mathbb{R}^n$ their Gram matrix is the matrix $\mathbf{X}$ defined by $\mathbf{X}_{ij} = \left\langle \boldsymbol{v}^{(i)}, \boldsymbol{v}^{(j)} \right\rangle$. A matrix $\mathbf{X} \in \mathbb{R}^{n \times n}$ is PSD if and only if it is the Gram matrix of a collection of vectors. A line (or linear) embedding of G is a function $x : V \to \mathbb{R}$ often represented as a vector $\boldsymbol{x} \in \mathbb{R}^V$. A vector embedding of G is a collection of vectors $\{\boldsymbol{v}^{(i)}\}_{i \in V}$ where each $\boldsymbol{v}^{(i)} \in \mathbb{R}^k$.

Graphs An undirected, unweighted graph $G = (V, E)$ is a pair containing a finite vertex set V and a subset $E \subseteq \binom{V}{2}$. The elements of E are the *edges* of the graph G. Each edge $\{u, v\}$ is said to connect vertex u and vertex v and may be denoted by uv for ease of notation. An undirected weighted graph is a triple $G = (V, E, w)$ in which V and E are defined as above, and $w : E \to \mathbb{R}_{\geq 0}$ is a function assigning non-negative weights to every edge $e \in E$. Note that we may assume without loss of generality that every graph is weighted, and every unweighted graph will be thought of as having weight equal to 1 for every edge. We may sometimes denote by $\boldsymbol{w}_e$ the weight $w(e)$ of e in G, and extend w additively, writing $w(F)$ for the sum of the weights of the edges in some subset $F \subseteq E$. We will always denote by n the cardinality of V and by m the cardinality of E. The degree of a vertex $i \in V$ is the quantity

$\deg(i) = \sum_{ij \in E} \boldsymbol{w}_{ij}$. The volume of a set of vertices is the sum of the degree of the vertices it contains: $\mathrm{Vol}(S) = \sum_{i \in S} \deg(i)$. We denote by $\triangle(G)$ the maximum combinatorial degree of G, i.e.: $\triangle(G) \stackrel{\text{def}}{=} \max_{i \in v} |\{e \in E \mid i \in e\}|$. Given two (not necessarily disjoint) sets of vertices $S, T \subseteq V$, we write $E(S,T)$ for the set of edges with one endpoint in S and the other endpoint in T. A cut in a graph G is a partition of its vertex set into two sets $(S, \overline{S})$.

Graph Matrices Given a graph $G = (V, E, w)$, its adjacency matrix is the symmetric matrix $\mathbf{A}$ satisfying:

$$\mathbf{A}_{ij} = \begin{cases} \boldsymbol{w}_{ij} & \text{if } ij \in E, \\ 0 & \text{otherwise.} \end{cases}$$

Its degree matrix is the diagonal matrix $\mathbf{D}$ where $\mathbf{D}_{ii} = \deg(i)$, and its Laplacian matrix is the matrix $\mathbf{L} \stackrel{\text{def}}{=} \mathbf{D} - \mathbf{A}$. It will sometimes be convenient to assign a measure $\mu : 2^V \to \mathbb{R}$ to the vertices of G. This will always be assumed to be additive, so that $\mu(S) + \mu(T) = \mu(S \cup T)$ for every pair of disjoint subsets S and T of V. Given a measure $\mu : 2^V \to \mathbb{R}$, we denote by $\boldsymbol{\mu}$ the vector in $\mathbb{R}^n$ with $\boldsymbol{\mu}_i = \mu(i)$, and by $\mathbf{M} \in \mathbb{R}^{n \times n}$ the diagonal matrix with $\mathbf{M} \stackrel{\text{def}}{=} \mathrm{diag}(\boldsymbol{\mu})$. We will also frequently make use of the matrix $\mathbf{L}(K_\mu)$ defined as:

$$\mathbf{L}(K_\mu) \stackrel{\text{def}}{=} \mathbf{M} - \frac{\boldsymbol{\mu}\boldsymbol{\mu}^\top}{\mu(V)}.$$

This matrix is conveniently designed so that its quadratic form evaluated at a line embedding $\boldsymbol{x} \in \mathbb{R}^V$ measures the μ-weighted variance of $\boldsymbol{x}$ (up to a $\mu(V)$ factor):

$$\boldsymbol{x}^\top \mathbf{L}(K_\mu) \boldsymbol{x} = \sum_{i \in V} \boldsymbol{\mu}_i \boldsymbol{x}_i^2 - \frac{1}{\mu(V)} \left(\sum_{i \in V} \boldsymbol{\mu}_i \boldsymbol{x}_i \right)^2 = \frac{1}{2} \sum_{\substack{i,j \in V \\ i \neq j}} \frac{\boldsymbol{\mu}_i \boldsymbol{\mu}_j}{\mu(V)} (\boldsymbol{x}_i - \boldsymbol{x}_j)^2.$$

This can be generalized to use $\mathbf{L}(K_\mu)$ to measure the μ-weighted variance of a vector embedding of the vertices in V. In fact, given such an embedding $\{\boldsymbol{v}^{(i)}\}_{i \in V}$, and letting $\mathbf{X}$ be

the Gram matrix of this set of vectors, then:

$$\mathbf{L}(K_\mu) \bullet \mathbf{X} = \sum_{i \in V} \boldsymbol{\mu}_i \left\| \boldsymbol{v}^{(i)} \right\|_2^2 - \left\| \sum_{i \in V} \boldsymbol{\mu}_i \boldsymbol{v}^{(i)} \right\|^2 = \frac{1}{2} \sum_{\substack{i,j \in V \\ i \neq j}} \frac{\boldsymbol{\mu}_i \boldsymbol{\mu}_j}{\mu(V)} \left\| \boldsymbol{v}^{(i)} - \boldsymbol{v}^{(j)} \right\|_2^2.$$

Graph Conductance, Spectral Gap and Cheeger's Inequality Given a graph G and subset S, the conductance of S is the quantity:

$$\phi(S) \stackrel{\text{def}}{=} \frac{w(E(S, \overline{S}))}{\min\{\text{Vol}(S), \text{Vol}(\overline{S})\}}.$$

The conductance $\phi(G)$ of G is defined as the minimum conductance of any set $S \subseteq V$. Computing the minimum conductance is known to be NP-Hard and finding efficient, high quality approximation to $\phi(G)$ is a central problem in approximation algorithms. In practice, a popular way of estimating the value of $\phi(G)$ is to compute the spectral gap of G:

$$\lambda_2(G) = \min_{\boldsymbol{x}} \frac{\boldsymbol{x}^\top \mathbf{L} \boldsymbol{x}}{\boldsymbol{x}^\top \mathbf{L}(K_\mu) \boldsymbol{x}}.$$

This serves as a convex relaxation to $\phi(G)$ and is related to $\phi(G)$ through Cheeger's Inequality:

$$\frac{\lambda_2}{8} \leq \phi(G) \leq 2\sqrt{\lambda_2(G)}.$$

Submodular functions Given a finite set V a function $\delta : 2^V \to \mathbb{R}$ is submodular if, for all $A, B \subseteq V$ we have:

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

A submodular function is monotone if $\delta(A) \leq \delta(B)$ whenever $A \subseteq B$. The Base polyhedron of a submodular function $\delta : 2^V \to \mathbb{R}$ is the set:

$$B(\delta) \stackrel{\text{def}}{=} \{\boldsymbol{y} \in \mathbb{R}^V \mid \boldsymbol{y}(h) = \delta(V), \boldsymbol{y}(S) \leq \delta(S) \text{ for every } S \subseteq V\},$$

where, for any $S \subseteq V$, $\boldsymbol{y}(S) \stackrel{\text{def}}{=} \sum_{i \in S} \boldsymbol{y}_i$. The Lovász extension of δ is the function $\bar{\delta} : \mathbb{R}^V \to \mathbb{R}$ given by:

$$\bar{\delta}(\boldsymbol{x}) \stackrel{\text{def}}{=} \max_{\boldsymbol{y} \in B(\delta)} \langle \boldsymbol{y}, \boldsymbol{x} \rangle .$$

When δ is monotone non-decreasing and $F(\emptyset) = 0$, we define its positive submodular polyhedron as:

$$\mathcal{P}_+(\delta) \stackrel{\text{def}}{=} \left\{ \boldsymbol{x} \in \mathbb{R}^V_{\geq 0} \, : \, \forall S \subseteq V, \, \langle \boldsymbol{x}, \mathbf{1}_S \rangle \leq \delta(S) \right\}.$$

Hypergraphs A hypergraph $G = (V, E, w)$ is a generalization of a graph in which E is a set of hyperedges: a collection of arbitrary subsets of V. The rank of a hypergraph G is the cardinality of the largest hyperedge: $\text{rank(G)} \stackrel{\text{def}}{=} \max_{h \in E} |h|$. The sparsity of a hypergraph G is the quantity:

$$\text{sp}(G) \stackrel{\text{def}}{=} \sum_{h \in E} |h|.$$

This represents an intuitive notion of instance size: it is the number of non-zero entries in an incidence matrix of G.

1.3 Bibliographical Notes

A discussion of majorization and Schur-convexity can be found in Chapter 13 of Steele (2004). For NP-Hardness of the minimum conductance cut problem see, e.g. Leighton and Rao (1999); Šíma and Schaeffer (2006). The graph Cheeger's Inequality is due to Alon and Milman (1985) (See also Alon (1986); Chung (1997)). The notation $\mathbf{L}(K_\mu)$ is due to Lorenzo Orecchia, see e.g. Orecchia and Vishnoi (2011). For background on submodular functions we refer the reader to the excellent text of Bach (2013).

CHAPTER 2

RECENT RESULTS ON APPROXIMATING VERTEX CONDUCTANCE

Given a graph $G = (V, E)$, the minimum vertex conductance problem asks one to find a small subset $C \subseteq V$ which, when removed, leaves G disconnected into two large pieces. This can be thought of as a vertex analogue of the minimum conductance cut problem. Just like the latter, the former is known to be NP-Hard.

Understanding the structure of vertex separators in a graph has many applications. In fact, vertex separators can model bottlenecks in communication networks, and thus approximation algorithms for this task can be used to certify the robustness of the network in question. Moreover, finding small separators is a key subroutine in many graph approximation algorithms, which leverage recursive divide-and-conquer approaches to solve more complex problems.

In this chapter we consider recent advances in approximation algorithms for this problem, as well as its connection with the fastest mixing diffusion problem with target stationary distribution: i.e. the problem of assigning weights to the edges of a graph so as to maximize the mixing of a diffusion process which tends to a target distribution.

2.1 Problem Statement and Related Variants

For the rest of this section, we assume that we have fixed a connected, unweighted graph $G = (V, E)$ and an additive measure $\mu : V \to \mathbb{R}$ on the vertices of G.

Definition 1 (Vertex Separator)**.** A vertex separator is a tripartition (L, C, R) of V such that there are no edges in E between L and R.

We will be concerned with finding vertex separators for which the central part C is small,

while the left and the right parts L and R are relatively large. To formalize this objective, we introduce the notion of vertex conductance.

Definition 2 (Vertex Conductance)**.** The *vertex-conductance* of a vertex separator (L, C, R) with respect to μ is the quantity:

$$q_\mu(L, C, R) := \frac{\mu(C)}{\min\{\mu(L \cup C), \mu(C \cup R)\}}.$$

The vertex-conductance of the graph G is then given by the minimization problem:

$$q_\mu(G) := \min_{\substack{(L,C,R) \\ \text{vertex separator}}} q_\mu(L, C, R). \tag{2.1}$$

Note that the above is only defined for $C \neq \emptyset$. Furthermore, if either $\mu(L) = 0$ or $\mu(R) = 0$ we have $q(L, C, R) = 1$.

As in the standard arguments for edge conductance, it will often be easier to work with a symmetrized version of vertex conductance, given by the following definition.

Definition 3 (Symmetrized Vertex Conductance)**.** The *symmetrized vertex conductance* of a vertex separator (L, C, R) is the quantity:

$$\overline{q}_\mu(L, C, R) := \frac{\mu(C)}{\mu(L \cup C) \cdot \mu(C \cup R)} \cdot \mu(V)^1.$$

One should note the following properties of q_μ and $\overline{q}_\mu$, which follow easily from their respective definitions:

Proposition 1. *We have:*

1. *For any vertex separator,* $\overline{q}_\mu(L, C, R) \in [0, 1]$,

1. Note that this is just a re-normalized version of what in Feige et al. (2008) is called the *vertex sparsity* α_π.

2. *For any trivial vertex separator, i.e.* $\mu(L) = 0$ *or* $\mu(R) = 0$, *we have* $\overline{q}_\mu(L, C, R) = 1$,

3. *For any non-trivial vertex separator,* $\overline{q}_\mu(L, C, R) < 1$.

Proof. The expression is trivially non-negative whenever μ is non-negative. Furthermore:

$$\overline{q}_\mu(L, C, R) \leq 1 \Leftrightarrow \mu(C)\mu(V) \leq \mu(L \cup C)\mu(C \cup R) \Leftrightarrow \frac{\mu(L)\mu(R)}{\mu(C)} \geq 0$$

□

It is easy to see that $\overline{q}_\mu$ is within a factor of 2 of q_μ.

Proposition 2. *For any separator* (L, C, R) *we get:*

$$q_\mu(L, C, R) \leq \overline{q}_\mu(L, C, R) \leq 2 \cdot q_\mu(L, C, R).$$

For this reason, in the rest of this chapter we will then be primarily concerned with solving:

$$\overline{q}_\mu(G) = \min_{\substack{(L,C,R) \\ \text{vertex separator}}} \overline{q}_\mu(L, C, R).$$

As pointed out by Feige et al. (2008), whenever an algorithm returns a trivial cut (Say $L = \varnothing$) one can instead return a minimum weight vertex cut. These cuts are legitimate vertex separators that are computable in polynomial time, and give a non-trivial value of $\overline{q}_\mu(L, C, R)$. Section 3 of the paper by Feige et al. describes a way of computing these vertex cuts in polynomial time. Moreover, should the optimal vertex separator satisfy $\mu(C) > \max\{\mu(L), \mu(R)\}$, one can return a minimum weight vertex cut and obtain a constant factor approximation to $\overline{q}_\mu(G)$. We will therefore be concerned with the case in which $\mu(C) \leq \max\{\mu(L), \mu(R)\}$.

We will also consider the following alternative objective:

$$\tilde{\alpha}_\mu(L, C, R) := \mu(V) \cdot \frac{\mu(C)}{\max\{\mu(L)\mu(C \cup R), \mu(R)\mu(C \cup L)\}},$$

and define the corresponding optimal quantity for the graph as:

$$\tilde{\alpha}_\mu(G) := \min_{\substack{(L,C,R) \\ \text{vertex separator}}} \tilde{\alpha}_\mu(L, C, R).$$

It is not hard to show that whenever $\mu(C) \leq \max\{\mu(L), \mu(R)\}$, the value of $\tilde{\alpha}(L, C, R)$ is within a factor of 2 of $\overline{q}_\mu(G)$. In particular we have the following:

Lemma 3. *Whenever the vertex separator (L, C, R) minimizing $\overline{q}_\mu(L, C, R)$ satisfies $\mu(C) \leq \max\{\mu(L), \mu(R)\}$, we have:*

$$\overline{q}_\mu(G) \leq \tilde{\alpha}_\mu(G) \leq 2\overline{q}_\mu(G).$$

2.2 A (Non-Convex) Quadratic Relaxation

In this section, we introduce a non-convex quadratic relaxation (QP) for the $\tilde{\alpha}_\mu$ objective. This is a fundamental stepping stone in designing algorithms to approximate $\tilde{\alpha}_\mu$, and connecting vertex separators to the fastest mixing diffusion process. After that, we state and prove the central result of this section: that we can round any optimal solution to this quadratic program to an integral solution to the vertex separator problem of integral objective value $O\left(\sqrt{\overline{q}_\mu(G)}\right)$ (Theorem 6). This result can be thought of as a vertex analogue of Cheeger's Inequality.

Consider the following quadratic program:

$$\begin{aligned} \min \quad & \sum_{i \in V} \mu(i) \boldsymbol{s}_i^2 \\ \text{s.t.} \quad & \forall \{i, j\} \in E \quad (\boldsymbol{x}_i - \boldsymbol{x}_j)^2 \leq (\boldsymbol{s}_i + \boldsymbol{s}_j)^2 \end{aligned} \tag{QP}$$

$$\boldsymbol{x}^\top L(K_\mu)\boldsymbol{x} \geq 1$$

$$\boldsymbol{x} \in \mathbb{R}^n$$

$$\boldsymbol{s} \in \mathbb{R}^n_{\geq 0}$$

This program is a relaxation of the $\tilde{\alpha}_\mu$ objective, as we now show.

Proposition 4. QP *is a relaxation of* $\tilde{\alpha}_\mu(L, C, R)$.

Proof. Given a vertex separator (L, C, R), satisfying $\tilde{\alpha}_\mu(L, C, R) = \kappa$, we can construct two solutions to the quadratic program: $Sol_1 := (\boldsymbol{x}^{(1)}, \boldsymbol{s}^{(1)})$ and $Sol_2 := (\boldsymbol{x}^{(2)}, \boldsymbol{s}^{(2)})$ to (QP) where:

$$\boldsymbol{x}_i^{(1)} = \begin{cases} \sqrt{\frac{\mu(V)}{\mu(L)\mu(C\cup R)}} & \text{if } i \in L \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \boldsymbol{s}_i^{(1)} = \begin{cases} \sqrt{\frac{\mu(V)}{\mu(L)\mu(C\cup R)}} & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases}$$

and:

$$\boldsymbol{x}_i^{(2)} = \begin{cases} \sqrt{\frac{\mu(V)}{\mu(R)\mu(C\cup L)}} & \text{if } i \in R \\ 0 & \text{otherwise} \end{cases} \quad \text{and} \quad \boldsymbol{s}_i^{(2)} = \begin{cases} \sqrt{\frac{\mu(V)}{\mu(R)\mu(C\cup L)}} & \text{if } i \in C \\ 0 & \text{otherwise} \end{cases}$$

Note that both of the above solutions are feasible. Consider, for instance, Sol_1, then:

$$\boldsymbol{x}^{(1)^\top} L(K\mu)\boldsymbol{x}^{(1)} = \sum_{(i,j)\in V^2} \frac{\mu(i)\mu(j)}{2\mu(V)}(\boldsymbol{x}_i - \boldsymbol{x}_j)^2 = \sum_{(i,j)\in L\times C\cup R} \frac{\mu(i)\mu(j)}{\mu(L)\mu(C\cup R)} = 1$$

and for each edge $e = \{i, j\}$, either $\boldsymbol{x}_i^{(1)} = \boldsymbol{x}_j^{(1)}$, in which case $\left(\boldsymbol{x}_i^{(1)} - \boldsymbol{x}_j^{(1)}\right)^2 = 0 \leq \left(\boldsymbol{s}_i^{(1)} + \boldsymbol{s}_i^{(1)}\right)^2$, or $i \in L$ and $j \in C$ in which case:

$$\left(\boldsymbol{x}_i^{(1)} - \boldsymbol{x}_j^{(1)}\right)^2 = \frac{\mu(V)}{\mu(L)\mu(C\cup R)} = \left(\boldsymbol{s}_i^{(1)} + \boldsymbol{s}_j^{(1)}\right)^2.$$

One can show feasibility of Sol_2 in the same way. Sol_1 satisfies:

$$QP(Sol_1) = \sum_{i \in V} \mu(i) \boldsymbol{s}_i^{(1)^2} = \frac{\mu(C)}{\mu(L)\mu(C \cup R)},$$

and similarly:

$$QP(Sol_2) = \sum_{i \in V} \mu(i) \boldsymbol{s}_i^{(2)^2} = \frac{\mu(C)}{\mu(R)\mu(C \cup L)}$$

giving $QP^* \leq \kappa$. □

Recall that in the regime of interest $\tilde{\alpha}_\mu$ is within a constant factor of $\bar{q}_\mu$.

Corollary 5. *Whenever the minimum vertex conductance cut satisfies $\mu(C) \leq \max\{\mu(L), \mu(R)\}$:*

$$QP^* \leq 2\overline{q}_\mu(L, C, R).$$

We conclude this section by presenting the following key result, which shows that we can round the solutions of the quadratic program to their integral counterparts with only a square root loss.

Theorem 6 (Hard side of Vertex Cheeger's Inequality)**.** *Given a solution to (QP) above with objective value λ we can construct a vertex separator (L, C, R) satisfying:*

$$q_\mu(L, C, R) \leq 2\sqrt{\lambda}.$$

Proof. We will use a rounding argument that mimics a popular technique for the minimum edge conductance problem.

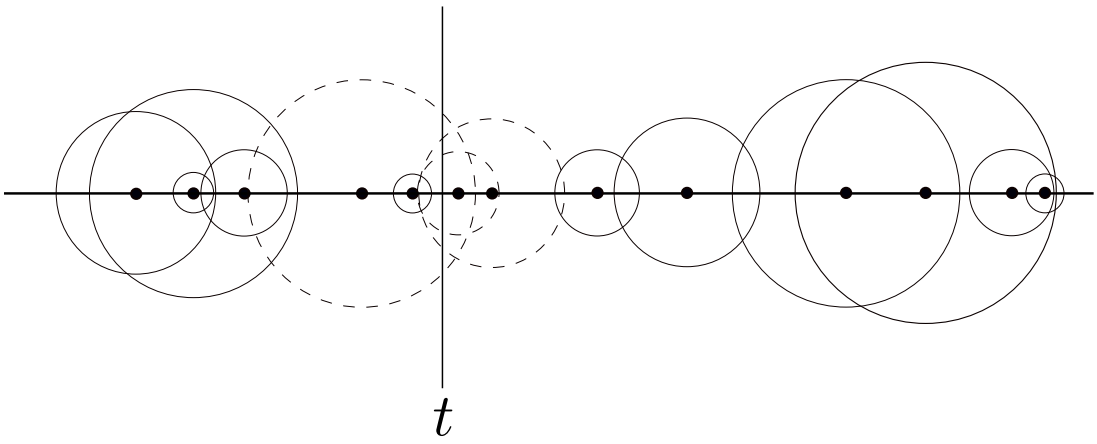

Figure 2.1: Given a solution (x, s) to QP, this provides an embedding of the vertices of G into $\mathbb{R}$, where the image x_i of each vertex i is surrounded by a circle of radius $\boldsymbol{s}_i$. We round the QP solution by searching for the optimal t. In the picture, the three vertices at the centers of the dashed spheres belong to the cut C_t.

We begin by noting that the problem of solving QP is equivalent to solving the following program instead:

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{s}} \quad & \frac{\sum_{i\in V}\mu(i)\boldsymbol{s}_i^2}{\boldsymbol{x}^\top \mathbf{L}(K_\mu)\boldsymbol{x}} \\ \forall\{i,j\}\in E \quad & (\boldsymbol{x}_i-\boldsymbol{x}_j)^2 \le (\boldsymbol{s}_i+\boldsymbol{s}_j)^2 \\ & \boldsymbol{x}\in\mathbb{R}^n \\ & \boldsymbol{s}\in\mathbb{R}^n_{\ge 0}, \end{aligned} \tag{QPR}$$

since solutions to QPR can be converted to solutions to QP of equal objective value by simple homogeneous rescaling. As such, we will exclusively be concerned with rounding arbitrary solutions to QPR. Given any $i \in V$, we define:

$$\boldsymbol{x}_i^- := \boldsymbol{x}_i - \boldsymbol{s}_i \qquad \text{and} \qquad \boldsymbol{x}_i^+ := \boldsymbol{x}_i + \boldsymbol{s}_i.$$

For any real number t, we let:

$$L_t := \{i \in V \mid \boldsymbol{x}_i^+ < t\}, \quad C_t := \{i \in V \mid t \in [\boldsymbol{x}_i^-, \boldsymbol{x}_i^+]\}, \quad R_t := \{i \in V \mid t < \boldsymbol{x}_i^-\}$$

(See Figure 2.1). Note that rescaling any solution to QPR by a constant yields another feasible solution to QPR of equal objective. Similarly, adding or subtracting a constant value from each $\boldsymbol{x}_i$ does not violate any constraints, nor does it alter the value of the objective of QPR.

Therefore, we shall assume that the given solution to QPR satisfies the following two assumptions.

Median Centering: For any $\varepsilon > 0$:

$$\mu(L_{-\varepsilon} \cup C_{-\varepsilon}) \leq \mu(C_{-\varepsilon} \cup R_{-\varepsilon}) \text{ and } \mu(L_{\varepsilon} \cup C_{\varepsilon}) \geq \mu(C_{\varepsilon} \cup R_{\varepsilon}).$$

Unit Scaling Let $\ell := \min_{i \in V} \boldsymbol{x}_i^-$ and $u := \max_{i \in V} \boldsymbol{x}_i^+$, then $\ell^2 + u^2 = 1$.

Consider the (random) vertex separator (L_t, C_t, R_t) obtained by sampling t from $[\ell, u]$ with pdf $f(t) = 2|t|$. Note that, by the median centering and unit scaling assumptions, $\ell < 0$ and:

$$\int_{\ell}^{u} 2|t| = u^2 + \ell^2 = 1.$$

By construction, if $t > 0$, $\mu(C_t \cup R_t) = \min\{\mu(L_t \cup C_t), \mu(C_t \cup R_t)\}$, while if $t < 0$, we have $\mu(L_t \cup C_t) = \min\{\mu(L_t \cup C_t), \mu(C_t \cup R_t)\}$. For any $i \in V$, define the event $\mathcal{E}_i$ to be:

$$\mathcal{E}_i := \begin{cases} (t \in (0, \boldsymbol{x}_i^+)) & \text{if } \boldsymbol{x}_i > 0, \\ (t \in (\boldsymbol{x}_i^-, 0)) & \text{if } \boldsymbol{x}_i < 0. \end{cases}$$

i.e. $\mathcal{E}_i$ is the event that t lies between zero, and the furthest point from zero in the circle of

radius $\boldsymbol{s}_i$ centered around $\boldsymbol{x}_i$. We have that:

$$\Pr[\mathcal{E}_i] = (|\boldsymbol{x}_i| + \boldsymbol{s}_i)^2.$$

Moreover, by the median centering property, if $\mathcal{E}_i$ occurs then $i \in \operatorname{argmin}\{\mu(L_t \cup C_t), \mu(C_t \cup R_t)\}$. We then have:

$$\begin{aligned}
\mathop{\mathbb{E}}_t [\min\{\mu(L_t \cup C_t), \mu(C_t \cup R_t)\}] &= \sum_i \mu(i) \Pr[i \in \operatorname{argmin}\{\mu(L_t \cup C_t), \mu(C_t \cup R_t)\}] \\
&\geq \sum_i \mu(i) \Pr[\mathcal{E}_i] = \sum_{i \in V} \mu(i)(|\boldsymbol{x}_i| + \boldsymbol{s}_i)^2 \\
&\geq \sum_{i \in V} \mu(i)(\boldsymbol{x}_i^2 + \boldsymbol{s}_i^2)
\end{aligned}$$

and:

$$\begin{aligned}
\mathop{\mathbb{E}}_t [\mu(C_t)] &= \sum_{i \in V} \mu(i) \Pr[i \in C_t] = \sum_{i \in V} \mu(i) \Pr[t \in [\boldsymbol{x}_i^-, \boldsymbol{x}_i^+]] \\
&= \sum_{i \in V} \mu(i) |\operatorname{sign}(\boldsymbol{x}_i^+)(\boldsymbol{x}_i^+)^2 - \operatorname{sign}(\boldsymbol{x}_i^-)(\boldsymbol{x}_i^-)^2| \leq \sum_{i \in V} \mu(i) |\boldsymbol{x}_i^+ - \boldsymbol{x}_i^-| (|\boldsymbol{x}_i^+| + |\boldsymbol{x}_i^-|) \\
&\leq \sqrt{\sum_{i \in V} \mu(i)(\boldsymbol{x}_i^+ - \boldsymbol{x}_i^-)^2} \sqrt{\sum_{i \in V} \mu(i)(|\boldsymbol{x}_i^+| + |\boldsymbol{x}_i^-|)^2} \\
&\leq \sqrt{\sum_{i \in V} 2\mu(i)\boldsymbol{s}_i^2} \sqrt{\sum_{i \in V} 2\mu(i)((\boldsymbol{x}_i^+)^2 + (\boldsymbol{x}_i^-)^2)} = 2\sqrt{\sum_{i \in V} \mu(i)\boldsymbol{s}_i^2} \sqrt{\sum_{i \in V} 2\mu(i)(\boldsymbol{x}_i^2 + \boldsymbol{s}_i^2)} \\
&= 2\sqrt{\lambda\big(\boldsymbol{x}^\top L(K_\mu)\boldsymbol{x}\big)} \sqrt{\sum_{i \in V} 2\mu(i)(\boldsymbol{x}_i^2 + \boldsymbol{s}_i^2)} \\
&\leq 2\sqrt{\lambda} \sqrt{\sum_i \mu(i)(\boldsymbol{x}_i^2 + \boldsymbol{s}_i^2)} \sqrt{\sum_{i \in V} 2\mu(i)(\boldsymbol{x}_i^2 + \boldsymbol{s}_i^2)} = 2\sqrt{2\lambda} \left(\sum_i \mu(i)(\boldsymbol{x}_i^2 + \boldsymbol{s}_i^2) \right),
\end{aligned}$$

so that:

$$\frac{\mathop{\mathbb{E}}_{t}\left[\mu(C_t)\right]}{\mathop{\mathbb{E}}_{t}\left[\min\{\mu(L_t \cup R_t), \mu(C_t \cup R_t)\}\right]} \leq 2 \cdot \frac{\sqrt{\lambda}\left(\sum_i \mu(i)(\boldsymbol{x}_i^2 + \boldsymbol{s}_i^2)\right)}{\sum_{i \in V} \mu(i)(\boldsymbol{x}_i^2 + \boldsymbol{s}_i^2)} = 2\sqrt{2\lambda}.$$

In particular, there must be a choice of t such that:

$$\frac{\mu(C_t)}{\mu(L_t \cup C_t)\mu(C_t \cup R_t)} \leq 2\sqrt{2\lambda}.$$

Hence, it is sufficient to try all ($O(n)$-many) cuts arising this way, and return the one with the lowest objective to obtain the desired guarantees. The search for the best sweep cut can be carried out with elementary methods in time $O(m \log n)$.

□

2.3 Integrality Gaps

In this section, we show that the loss incurred in rounding solutions to the QP is necessary by providing a matching integrality gap for the problem. We will also give a separate $\Omega(\sqrt{\log n})$ integrality gap.

Integrality Gap from Path Graph We begin by considering a path graph $P_n = ([n], E(P_n))$ for some odd n, where:

$$E(P_n) := \{\{i, i+1\} \mid i = 1, ..., n-1\},$$

equipped with the uniform measure over the vertices $\mu \equiv 1$. The optimal solution to the minimum conductance cut problem for P_n is given by letting:

$$L_P = \left\{i \mid i < \frac{n}{2}\right\} \qquad C_P = \left\{\left\lceil \frac{n}{2} \right\rceil\right\} \qquad R_P = \left\{i \mid i > \frac{n}{2} + 1\right\}.$$

This cut has symmetrized vertex conductance:

$$\overline{q}_\mu(L_P, C_P, R_P) = \frac{n}{\lfloor n/2 \rfloor^2} = \Omega\left(\frac{1}{n}\right),$$

and:

$$\tilde{\alpha}_\mu(L_P, C_p, R_p) = \Omega\left(\frac{1}{n}\right).$$

On the other hand we can construct a QP solution $(\boldsymbol{x}^{(P)}, \boldsymbol{s}^{(P)})$ as follows:

$$\boldsymbol{x}_i^P := i \qquad\qquad \boldsymbol{s}_i^P = 1,$$

which clearly respects edge covering constraints. We obtain:

$$\boldsymbol{x}^\top L(K_\mu)\boldsymbol{x} = \sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{x}_{\lceil n/2 \rceil})^2 = \sum_{i=1}^{n}(i - \lceil n/2 \rceil)^2 \geq \sum_{i=1}^{\lceil n/4 \rceil}\left(\left\lceil \frac{n}{4} \right\rceil - \left\lceil \frac{n}{2} \right\rceil\right)^2 = \Omega\left(n^3\right),$$

while:

$$\sum_{i \in V} \mu(i)\boldsymbol{s}_i^2 = \sum_{i \in V} \boldsymbol{s}_i^2 = n,$$

so that, by rescaling the solution dividing by $\boldsymbol{x}^\top L(K_\mu)\boldsymbol{x}$ we obtain another QP solution of value:

$$QP(\boldsymbol{x}^{(P)}, \boldsymbol{s}^{(P)}) = O\left(\frac{1}{n^2}\right).$$

This immediately implies that one cannot uniformly improve on the rounding given in the previous section.

Integrality Gaps from the Hypercube We can check that the problem (QP) has an $\Omega(\sqrt{\log n})$ integrality gap by considering the hypercube graph on n vertices and the uniform measure $\mu \equiv 1$. Let $V = \{-1, 1\}^d$ for some even number $d = \log n$. For any $u, v \in V$, let $d_H(u, v)$ be the hamming distance between u and v. Consider the hypercube graph on V:

$G = (V, E)$, where $E = \{\{u, v\} \in \binom{V}{2} \mid d_H(v, u) = 1\}$.

The vertex separator minimizing vertex conductance with respect to the uniform measure in this graph is given by:

$$L := \left\{ u \mid \sum_i u_i < 0 \right\}, \qquad C := \left\{ u \mid \sum_i u_i = 0 \right\}, \qquad R := \left\{ u \mid \sum_i u_i > 0 \right\}.$$

This corresponds to the vertex cut cutting all balanced vectors, that is vectors containing an equal number of 1 and -1 entries. Let $t := \binom{d}{d/2} = |C|$ then:

$$\tilde{\alpha}_\mu(L, C, R) = \frac{|V| \cdot |C|}{|C \cup R| \cdot |L|} = \frac{n \cdot t}{\frac{n+t}{2} \cdot \frac{n-t}{2}} = \Omega\left(\frac{1}{\sqrt{\log n}}\right)$$

where we used the fact that for a general $k \in \mathbb{N}$: $\binom{2k}{k} \sim \frac{4^k}{\sqrt{\pi k}}$. On the other hand, consider the QP solution where:

$$\boldsymbol{x}_{v_j} = \sum_i (v_j)_i \qquad \text{and} \qquad \boldsymbol{s}_{v_j} = 2$$

for all $v_j \in V$. (Here $(v_j)_i$ is -1 or 1 depending on the value of the i^{th} entry of v_j). We then have $\sum_{v \in V} \boldsymbol{x}_v = 0$, giving:

$$\boldsymbol{x}^T L(K_{\vec{\mathbf{1}}}) \boldsymbol{x} = \sum_{v \in V} \boldsymbol{x}_v^2 - \frac{1}{n} \left(\sum_{v \in V} \boldsymbol{x}_v \right)^2 = \sum_{v \in V} \boldsymbol{x}_v^2$$

and hence:

$$QP(\boldsymbol{x}, \boldsymbol{s}) = \frac{\sum_j \boldsymbol{s}_{v_j}^2}{\boldsymbol{x}^T L(K_{\vec{\mathbf{1}}}) \boldsymbol{x}} = \frac{O(n)}{O\left(n \log n\right)} = O\left(\frac{1}{\log n}\right).$$

Hence $QP^* = O\left(\frac{\tilde{\alpha}_\mu(G)}{\sqrt{\log n}}\right)$, giving the desired integrality gap.

2.4 Approximation Algorithms via Semidefinite Relaxation

In this section, we describe a semidefinite programming relaxation of the quadratic program. We begin by relaxing the quadratic program to the following vector program:

$$\begin{aligned}
\min \quad & \sum_{i \in V} \mu(i) \left\| \mathbf{z}_i \right\|_2^2 \\
\text{s.t.} \forall ij \in E \quad & \left\| \mathbf{v}^{(i)} - \mathbf{v}^{(j)} \right\|_2^2 \leq \left\| \mathbf{z}^{(i)} + \mathbf{z}^{(j)} \right\|_2^2 \\
& \sum_{i \in V} \mu(i) \left\| \mathbf{v}^{(i)} \right\|_2^2 - \frac{1}{\mu(V)} \left(\sum_{i \in V} \mu(i) \left\| \mathbf{v}^{(i)} \right\|_2 \right)^2 \geq 1 \\
& \{\mathbf{v}^{(i)}\}_{i \in V}, \{\mathbf{z}^{(i)}\}_{i \in V} \in \mathbb{R}^n,
\end{aligned}$$

which is equivalent to the following SDP:

$$\begin{aligned}
\min \quad & \mathbf{M} \bullet \mathbf{S} \qquad \text{(VSep-SDP)} \\
\text{s.t.} \forall ij \in E \quad & \mathbf{L}_{ij} \bullet \mathbf{X} \leq \mathbf{L}_{ij}^{pos} \bullet S \\
& \mathbf{L}(K_\mu) \bullet \mathbf{X} \geq 1 \\
& \mathbf{X} \succeq 0, \mathbf{S} \succeq 0, \mathbf{S} \geq 0.
\end{aligned}$$

Here, $\mathbf{L}_{ij}^{pos}$ is the positive edge Laplacian for the $\{i, j\}$ edge, i.e. the matrix $(\mathbf{e_i}+\mathbf{e_j})(\mathbf{e_i}+\mathbf{e_j})^\top$.

We can then show that $\mathbf{S}$ above can be chosen to be rank 1 without loss of generality.

Lemma 7. *Given any feasible solution $(\mathbf{S}, \mathbf{X})$ to the SDP, one can find another solution $(\mathbf{S}', \mathbf{X})$ of equal objective value such that $rank(\mathbf{S}') = 1$, in polynomial time. In particular, there exists a solution $(\mathbf{S}, \mathbf{X})$ to the above SDP, such that* $\text{rank}(\mathbf{S}) = 1$.

Proof. Let $(\mathbf{S}^*, \mathbf{X}^*)$ be an arbitrary solution to the above SDP. Let $\boldsymbol{r} \in \mathbb{R}^n$ be the vector:

$$\boldsymbol{r} := (\sqrt{\mathbf{S}_{11}}, ..., \sqrt{\mathbf{S}_{nn}}).$$

One can easily check that $(\boldsymbol{r}\boldsymbol{r}^\top, \mathbf{X}^*)$ has the same objective value as $(\mathbf{S}^*, \mathbf{X}^*)$. Moreover, the second constraint as well as the positive semi-definiteness and non-negativity constraints are trivially satisfied. One only needs to verify that the first constraint is satisfied. Since $S^* \succeq 0$, we have, by Sylvester's criterion that:

$$\mathbf{S}_{ii}\mathbf{S}_{jj} \geq \mathbf{S}_{ij}^2, \tag{2.5}$$

and hence:

$$\begin{aligned}\mathbf{L}_{ij} \bullet X \leq \mathbf{L}_{ij}^{pos} \bullet \mathbf{S} = \mathbf{S}_{ii} + \mathbf{S}_{jj} + 2\mathbf{S}_{ij} &\overset{(2.5)}{\leq} \mathbf{S}_{ii} + \mathbf{S}_{jj} + 2\sqrt{\mathbf{S}_{ii}}\sqrt{\mathbf{S}_{jj}} \\ &= (\boldsymbol{r}_i + \boldsymbol{r}_j)^2 = \mathbf{L}_{ij}^{pos} \bullet \boldsymbol{r}\boldsymbol{r}^\top,\end{aligned}$$

so that $\boldsymbol{r}\boldsymbol{r}^\top$ satisfies the first constraint as needed. □

In light of the above lemma, one can consider the following Rank-1 version of the above semidefinite program:

$$\begin{aligned} \min \quad & \sum_{i \in V} \mu(i)\, \mathbf{s}_i^2 \\ \text{s.t.} \forall ij \in E \quad & \mathbf{L}_{ij} \bullet \mathbf{X} \leq (\mathbf{s}_i + \mathbf{s}_j)^2 \\ & \mathbf{L}(K_\mu) \bullet \mathbf{X} \geq 1 \\ & \mathbf{X} \succeq 0, \mathbf{s} \in \mathbb{R}^n_{\geq 0}. \end{aligned} \tag{R1SDP}$$

and note that R1SDP* = VSep-SDP*. One should also note the following.

Lemma 8. *VSep-SDP* = R1SDP* $\leq$ QP*.*

Proof. Given any solution $(\boldsymbol{x}, \boldsymbol{s})$ to the quadratic program with objective value ρ, $(xx^\top, S)$ is a solution to the rank-1 SDP with objective value ρ. □

For the sake of convenience we will actually solve the following SDP:

$$\begin{aligned} \min \quad & \sum_{i \in V} \mu(i) \boldsymbol{s}_i \\ \text{s.t.} \forall ij \in E \quad & \mathbf{L}_{ij} \bullet \mathbf{X} \leq \boldsymbol{s}_i + \boldsymbol{s}_j \\ & \mathbf{L}(K_\mu) \bullet \mathbf{X} \geq 1 \\ & \mathbf{X} \succeq 0, \boldsymbol{s} \in \mathbb{R}^n_+. \end{aligned} \tag{R1SDP2}$$

The SDP (R1SDP2) is obtained from $R1SDP$ by renaming the variables ($\boldsymbol{s}_i^2 \to \boldsymbol{s}_i$) and tightening the first constraint. We then have the following:

Lemma 9. $R1SDP^* \leq R1SDP2^* \leq 2 \cdot R1SDP^*$.

Proof. Given any solution $(\mathbf{X}, \boldsymbol{s})$ to R1SDP2 with corresponding objective value ρ, we have that $(\mathbf{X}, \boldsymbol{t})$, where, for $i = 1, ..., n$: $\boldsymbol{t}_i := \sqrt{\boldsymbol{s}_i}$ is a feasible solution to R1SDP with the same objective value, yielding the first inequality. On the other hand, given a solution $(\mathbf{X}, \boldsymbol{t})$ to R1SDP with objective value ρ', the solution $(\mathbf{X}, \boldsymbol{s})$ where $\boldsymbol{s}_i := 2 \cdot \boldsymbol{t}_i^2$ is a feasible point in R1SDP2 with corresponding objective value $2 \cdot \rho'$. □

We then have:

Corollary 10. $R1SDP2^* \leq 2 \cdot QP^*$.

2.5 Rounding the SDP

In this section, we describe two rounding procedures for the SDP given above with distinct approximation guarantees.

Rounding SDP Solutions with Loss $O(\log \Delta(G))$

We first show that we can round an optimal solution of (R1SDP2) of value ρ to a solution of (QP) of value $O(\rho \log \Delta(G))$. To do so, we mimic the technique used for Section 9 of Louis et al. (2013). We shall assume the R1SDP2 solution satisfies $\mathbf{L}(K_\mu) \bullet \mathbf{X} = 1$, which is true at optimality. We may also assume that $\mathbf{X}$ is given as the Gram matrix of a collection of vectors $\{\boldsymbol{v}^{(i)}\}_{i=1}^n$. These can be computed using the Cholesky decomposition of $\mathbf{X}$. We round the SDP solution using Algorithm 1.

Algorithm 1 SDP rounding

Input: $(\{\boldsymbol{v}^{(i)}\}_{i=1}^n \subseteq \mathbb{R}^D, \boldsymbol{s} \in \mathbb{R}^n)$, solution to the vector program corresponding to R1SDP2 of value ρ.

1. For each edge $\{i,j\} \in E$ let $\boldsymbol{v}^{(ij)} = \boldsymbol{v}^{(i)} + \frac{\sqrt{\boldsymbol{s}_i}}{\sqrt{\boldsymbol{s}_i}+\sqrt{\boldsymbol{s}_j}}(\boldsymbol{v}^{(j)} - \boldsymbol{v}^{(i)})$
2. Let $\boldsymbol{u} \sim \mathcal{N}(0,1)^n$ be a random gaussian vector, with mean zero.
3. For $i \in V$, let $p_i := \left\langle \boldsymbol{u}, \boldsymbol{v}^{(i)} \right\rangle$, for $\{i,j\} \in E$, let $p_{ij} := \left\langle \boldsymbol{u}, \boldsymbol{v}^{(ij)} \right\rangle$
4. Let $r_i := \max_{j:j\sim i}(p_i - p_{ij})^2$
5. Let $\boldsymbol{s}^{(Q)} \in \mathbb{R}^n$ be the vector given by: $\boldsymbol{s}_i^{(Q)} := \sqrt{2r_i}$, and $\boldsymbol{p} \in \mathbb{R}^n$ be the vector with $\boldsymbol{p}_i = p_i$
6. Return $(\boldsymbol{p}, \boldsymbol{s}^{(Q)})$ as a QP solution.

We will begin by showing that the output to Algorithm 1 satisfies the edge constraints of the quadratic program.

Lemma 11. *The output $(\boldsymbol{p}, \boldsymbol{s}^{(Q)})$ of Algorithm 1 satisfies:*

$$(\boldsymbol{p}_i - \boldsymbol{p}_j)^2 \leq (\boldsymbol{s}_i^{(Q)} + \boldsymbol{s}_j^{(Q)})^2.$$

Proof.

$$(\boldsymbol{p}_i - \boldsymbol{p}_j)^2 = (p_i - p_{ij} + p_{ij} - p_j)^2 \leq 2(p_i - p_{ij})^2 + 2(p_{ij} - p_j)^2 \leq 2r_i + 2r_j$$
$$\leq (\sqrt{2r_i} + \sqrt{2r_j})^2 = (\boldsymbol{s}_i^{(Q)} + \boldsymbol{s}_j^{(Q)})^2.$$

□

We can now argue about the objective of the rounded solution. We will use the following:

Lemma 12. *Let $Y_1, ..., Y_d$ be d normal random variables with mean 0 and variance at most σ^2. Let Y be defined by $Y := \max_{i \in [d]} Y_i^2$. Then:*

$$\mathbb{E}[Y] \leq 4\sigma^2 \log d.$$

Proof (following the proof of Fact B.3 in Louis et al. (2013)). For any $p \in \mathbb{N}$, we have:

$$\mathbb{E}\left[\max_i Y_i^2\right] \leq \mathbb{E}\left[\left(\sum_i Y_i^{2p}\right)^{1/p}\right] \leq \left(\mathbb{E}\left[\sum_i Y_i^{2p}\right]\right)^{1/p} \leq \left(d\sigma^{2p}\frac{(2p)!}{p!2^p}\right)^{1/p}$$
$$\leq d^{1/p}\sigma^2 p = 2\sigma^2 \log d,$$

where we choose $p = \log_2 d$.

□

Lemma 13. *For every $i \in V$:*

$$\mathbb{E}[r_i] \leq 4\boldsymbol{s}_i \log \Delta(G).$$

Proof.

$$\mathop{\mathbb{E}}_u [r_i] = \mathop{\mathbb{E}}_u \left[\max_{j \sim i}(\left\langle \boldsymbol{v}^{(i)} - \boldsymbol{v}^{(ij)}, \boldsymbol{u} \right\rangle)^2\right] \leq 4 \max_{j \sim i} \left\| \boldsymbol{v}^{(i)} - \boldsymbol{v}^{(ij)} \right\|^2 \log \Delta(G)$$
$$\leq 4 \max_{j \sim i} \frac{\boldsymbol{s}_i}{(\sqrt{\boldsymbol{s}_i} + \sqrt{\boldsymbol{s}_j})^2} \left\| \boldsymbol{v}^{(i)} - \boldsymbol{v}^{(j)} \right\|^2 \log \Delta(G)$$

$$\leq 4 \max_{j \sim i} \frac{\boldsymbol{s}_i}{(\sqrt{\boldsymbol{s}_i} + \sqrt{\boldsymbol{s}_j})^2} (\boldsymbol{s}_i + \boldsymbol{s}_j) \log \Delta(G) \leq 4 \boldsymbol{s}_i \log \Delta(G).$$

□

This then gives:

$$\mathop{\mathbb{E}}_{u} \left[\mathrm{QP}(\boldsymbol{p}, \boldsymbol{s}^{(Q)}) \right] = \mathop{\mathbb{E}}_{u} \left[\sum_{i \in V} \mu(i) (\boldsymbol{s}_i^{(Q)})^2 \right] = 2 \sum_{i \in V} \mu(i) \mathop{\mathbb{E}}_{u} [r_i] \leq 8 \log \Delta(G) \sum_{i \in V} \mu(i) \boldsymbol{s}_i = 8 \log \Delta(G) \rho.$$

Then by Markov's inequality we get:

$$\Pr \left[\sum_{i \in V} \mu(i) (\boldsymbol{s}_i^{(Q)})^2 \geq 192 \log \Delta(G) \rho \right] \leq \frac{1}{24}. \tag{2.8}$$

Lemma 14. *The spread of the projection satisfies:*

$$\Pr \left[\boldsymbol{p}^\top \mathbf{L}(K_\mu) \boldsymbol{p} \geq \frac{1}{2} \right] \geq \frac{1}{12}.$$

Proof. We have:

$$\mathop{\mathbb{E}}_{u} \left[\boldsymbol{p}^\top \mathbf{L}(K_\mu) \boldsymbol{p} \right] = \sum_i \mu(i) \left\| \boldsymbol{v}^{(i)} \right\|^2 - \frac{1}{\mu(V)} \left\| \sum_i \mu(i) \boldsymbol{v}^{(i)} \right\|^2 = L(K_\mu) \bullet X = 1,$$

and also:

$$\boldsymbol{p}^\top \mathbf{L}(K_\mu) \boldsymbol{p} = \sum_{i,j \in V} \frac{\mu(i)\mu(j)}{\mu(V)} (\boldsymbol{p}_i - \boldsymbol{p}_j)^2$$

which is a sum of gaussian random variables. We can then apply the following lemma proved in Louis et al. (2013):

Lemma 15 (Lemma 9.8 from Louis et al. (2013))**.** *Suppose $Z_1, ..., Z_m$ are gaussian random*

variables (not necessarily independent) such that $\mathbb{E}\left[\sum_i Z_i^2\right] = 1$ *then:*

$$\Pr\left[\sum_i Z_i^2 \geq \frac{1}{2}\right] \geq \frac{1}{12}.$$

When applied to the rounding, this lemma gives:

$$\Pr\left[\boldsymbol{p}^\top \mathbf{L}(K_\mu)\boldsymbol{p} \geq \frac{1}{2}\right] \geq \frac{1}{12}$$

as needed. □

By Lemma 14, Equation (2.8), and the union bound, we have:

$$\Pr\left[\frac{\sum_{i\in V} \mu(i)(\boldsymbol{s}_i^{(Q)})^2}{\boldsymbol{p}^\top \mathbf{L}(K_\mu)\boldsymbol{p}} \leq 384 \log \Delta(G)\rho\right] \geq \frac{1}{24}$$

giving that with constant probability we can round to a QP solution of value $O(\rho \log \Delta(G))$.

Rounding SDP Solution with Loss O(rank($\mathbf{X}$))

We now show how to round a solution $(\mathbf{X}, \boldsymbol{s})$ to (R1SDP2) with value ρ, where $\text{rank}(\mathbf{X}) = k$, to a solution to (QP) with value $O(\rho k)$. More generally, this technique allows us to round any solution to (R1SDP2) where a ξ-fraction of the variance of the embedding is contained within k dimensions, to a solution to (QP) with an objective value of $k\rho/\xi$ (see Proposition 17).

We shall again assume that $\mathbf{L}(K_\mu) \bullet \mathbf{X} = 1$. We round the solution using the following algorithm:

We can verify that the returned solution satisfies the edge constraints, since for every

Algorithm 2 SDP rounding (rank version)

Input: $(\{\boldsymbol{v}^{(i)}\}_{i=1}^{n} \subseteq \mathbb{R}^D, \boldsymbol{s} \in \mathbb{R}^n)$, solution to the vector program corresponding to R1SDP2 of value ρ.
Let $\mathbf{V} \in \mathbb{R}^{n \times k}$ be the matrix where the i^{th} row is $\boldsymbol{v}^{(i)}$.

1. Let $\boldsymbol{u}$ be a unit eigenvector of $\mathbf{V}^\top \mathbf{M} \mathbf{V}$ corresponding to its largest eigenvalue,
2. For $i \in V$, let $p_i := \sqrt{k} \cdot \left\langle \boldsymbol{u}, \boldsymbol{v}^{(i)} \right\rangle$,
3. Let $\boldsymbol{s}$ be given by: $\boldsymbol{s}_i^{(Q)} := \sqrt{k \boldsymbol{s}_i}$, and $\boldsymbol{p}$ be given by: $\boldsymbol{p}_i = p_i$,
4. Return $(\boldsymbol{p}, \boldsymbol{s}^{(Q)})$ as a QP solution.

edge ij in E we have:

$$\begin{aligned}(\boldsymbol{p}_i - \boldsymbol{p}_j)^2 &= (\sqrt{k} \left\langle \boldsymbol{u}^{(i)}, \boldsymbol{v}^{(i)} - \boldsymbol{v}^{(j)} \right\rangle)^2 \leq k \left\| \boldsymbol{v}^{(i)} - \boldsymbol{v}^{(j)} \right\|^2 \leq k(\boldsymbol{s}_i + \boldsymbol{s}_j) \\ &= (\boldsymbol{s}_i^{(Q)})^2 + (\boldsymbol{s}_j^{(Q)})^2 \leq (\boldsymbol{s}_i^{(Q)} + \boldsymbol{s}_j^{(Q)})^2.\end{aligned}$$

Then, we can observe that the variance constraint is satisfied:

$$\begin{aligned}\boldsymbol{p}^\top \mathbf{L}(K_\mu) \boldsymbol{p} &= k \cdot \boldsymbol{u}^\top \mathbf{V}^\top \mathbf{L}(K_\mu) \mathbf{V} \boldsymbol{u} = k \lambda_{\max}(\mathbf{V}^\top \mathbf{L}(K_\mu) \mathbf{V}) \geq \sum_{i=1}^{k} \lambda_i(\mathbf{V}^\top \mathbf{L}(K_\mu) \mathbf{V}) \\ &= \mathbf{L}(K_\mu) \bullet \mathbf{X} = 1.\end{aligned}$$

Finally, the objective value of the QP solution found equals:

$$\mathrm{QP}(\boldsymbol{p}, \boldsymbol{s}^{(Q)}) = \sum_{i \in V} \mu(i) (\boldsymbol{s}_i^{(Q)})^2 = \sum_{i \in V} k \boldsymbol{s}_i \mu(i) = \rho \cdot k,$$

giving the following result:

Proposition 16. *Any solution $(\mathbf{X}, \boldsymbol{s})$ to (R1SDP2) of value ρ can be rounded to a solution to (QP) of value $O(\rho \operatorname{rank}(\mathbf{X}))$.*

It is worth noting that the same technique shows the following more general result:

Proposition 17. *Given a solution* $(\mathbf{X}, \boldsymbol{s})$ *to (R1SDP2), of value* ρ*, where* $\mathbf{X} = \mathbf{V}\mathbf{V}^\top$ *satisfying:*

$$\sum_{i=n-\ell+1}^{n} \lambda_i(\mathbf{V}^\top \mathbf{L}(K_\mu)\mathbf{V}) \geq \xi \sum_{i=1}^{n} \lambda_i(\mathbf{V}^\top \mathbf{L}(K_\mu)\mathbf{V}),$$

where $\lambda_1, ..., \lambda_n$ *denote the eigenvalue in ascending order, this can be rounded to a solution to (QP) with objective value* $k\rho/\xi$*.*

2.6 Constant Factor Approximation for Planar Graphs

In this section, we show that the results given in the previous sections imply that one can obtain a constant factor approximation to $\tilde{\alpha}_\mu$ in planar graphs. This follows trivially from the following statement.

Proposition 18. *Let* G *be a planar graph, then for any* μ *given an optimal solution* $(\mathbf{X}^*, \boldsymbol{s}^*)$ *to* $R1SDP2$ *one can efficiently find an optimal solution* $(\mathbf{X}', \boldsymbol{s}^*)$ *satisfying:*

$$\operatorname{rank}(\mathbf{X}') \leq 3.$$

In order to prove this, we make use of the following notion:

Definition 4 (Colin de Verdière Graph Parameter)**.** Let $G = (V, E)$ be an undirected graph. Let $\mathcal{M}_G$ be the set of symmetric matrices $M \in \mathbb{S}^n$ satisfying:

(M1) $\mathbf{M}_{ij} < 0$ whenever $i \neq j$ and $ij \in E$, and $\mathbf{M}_{ij} = 0$ whenever $i \neq j$ and $ij \notin E$,

(M2) $\mathbf{M}$ has exactly one negative eigenvalue (of multiplicity 1),

(M3) There is no non-zero $\mathbf{X} \in \mathbb{S}^n$ satisfying $\mathbf{M}\mathbf{X} = 0$ and $\mathbf{X}_{ij} = 0$ whenever $i = j$ or $\mathbf{M}_{ij} \neq 0$,

then, the Colin de Verdière parameter of G is the maximum corank of any matrix $\mathbf{M}$ in $\mathcal{M}_G$:

$$\mathrm{cdv}(G) := \max_{\mathbf{M} \in \mathcal{M}_G} \mathrm{nullity}(\mathbf{M}).$$

In 1990, de Verdiere (1990) showed that $\mathrm{cdv}(G) \leq 3$ if G is planar, and later van der Holst gave a simple proof of the following stronger fact.

Theorem 19 (van der Holst (1995))**.** *If G is a planar graph, then any matrix satisfying* (M1) *and* (M2) *has corank at most 3.*

From this, letting:

$$\mathcal{E}_{\mathbf{A},\mathbf{B}}(\lambda) \stackrel{\text{def}}{=} \dim\{\mathbf{v} \mid \mathbf{A}\mathbf{v} = \lambda \mathbf{B}\mathbf{v}\},$$

we can derive the following fact that will be key to the structure of the solution of our SDP.

Corollary 20. *Let G be a planar graph, then $\mathcal{E}_{\mathbf{L},\mathbf{M}}(\lambda_2) \leq 3$.*

Proof. One simply needs to verify that $\mathcal{L} - \lambda_2 \mathbf{I}$ satisfies (M1) and (M2) above and then apply Theorem 19 above. □

Proof of Proposition 18. Let $(\mathbf{X}^*, \boldsymbol{s}^*)$ be an optimal solution to R1SDP2 then $\mathbf{X}^*$ must be a solution to $P_{\boldsymbol{s}^*}$, where, for any choice of $\boldsymbol{s}$, $P_{\boldsymbol{s}}$ is the following feasibility problem:

$$\begin{aligned} P_s: \quad \min \quad & 0 \\ \text{s.t.} \quad & \mathbf{L}_{ij} \bullet \mathbf{X} \leq \boldsymbol{s}_i + \boldsymbol{s}_j \quad \forall ij \in E \\ & \mathbf{L}(K_\mu) \bullet \mathbf{X} \geq 1 \\ & \mathbf{X} \succeq 0. \end{aligned}$$

The dual of $P_{\boldsymbol{s}}$ is given by:

$$D_{\boldsymbol{s}}: \quad \max \quad \alpha - \sum_{ij \in E} \boldsymbol{w}_{ij}(\boldsymbol{s}_i + \boldsymbol{s}_j)$$

$$\text{s.t.} \quad \sum_{ij \in E} \boldsymbol{w}_{ij} \mathbf{L}_{ij} \succeq \alpha \mathbf{L}(K_\mu)$$

$$\boldsymbol{w} \in \mathbb{R}^m_{\geq 0},\ \alpha \in \mathbb{R}_{\geq 0}.$$

Note that for any feasible solution $\mathbf{X}$, the matrix:

$$\mathbf{X}' = \mathbf{X} - \frac{\mathbf{M}\mathbf{1}^\top\mathbf{1}\mathbf{M}}{(\mathbf{1}^\top\mathbf{M}^2\mathbf{1})^2}(\mathbf{X} \bullet \mathbf{M}\mathbf{1}\mathbf{1}^\top\mathbf{M})$$

satisfying $\text{span}\{\mathbf{M}\vec{\mathbf{1}}\} \subseteq \ker(\mathbf{X}')$, also gives $\mathbf{L}(K_\mu) \bullet \mathbf{X}' = \mathbf{L}(K_\mu) \bullet \mathbf{X}$ and $\mathbf{L}_{ij} \bullet \mathbf{X}' = \mathbf{L}_{ij} \bullet \mathbf{X}$. So for any feasible solution $\mathbf{X}$ to $P_{\boldsymbol{s}}$ there exists a corresponding feasible solution $\mathbf{X}'$ satisfying $\text{span}\{\mathbf{M}\vec{\mathbf{1}}\} \subseteq \ker(\mathbf{X}')$, and we shall assume that $\mathbf{X}^*$ is chosen to have this property.

By strong duality, we know that there exists some matching assignment of the dual variables $(\alpha^*, \boldsymbol{w}^*)$ such that:

$$\alpha^* = \sum_{ij \in E} \boldsymbol{w}^*_{ij}(\boldsymbol{s}^*_i + \boldsymbol{s}^*_j) = \lambda_2(G, \mu, \boldsymbol{w}^*),$$

where:

$$\lambda_2(G, \mu, \boldsymbol{w}) \stackrel{\text{def}}{=} \min_{\boldsymbol{x} \perp_{\mathbf{M}} \vec{\mathbf{1}}} \frac{\boldsymbol{x}^\top \mathbf{L}_{\boldsymbol{w}} \boldsymbol{x}}{\boldsymbol{x}^\top \mathbf{M} \boldsymbol{x}}$$

with $\mathbf{L}_{\boldsymbol{w}} = \sum_{ij \in E} \boldsymbol{w}_{ij} \mathbf{L}_{ij}$.

Furthermore $\mathbf{X}^*$,$\boldsymbol{s}^*$, α^* $\boldsymbol{w}^*$ satisfy the dual complementary slackness condition:

$$\mathbf{X}^* \bullet \left(\sum_{ij \in E} \boldsymbol{w}^*_{ij} \mathbf{L}_{ij} - \alpha^* \mathbf{L}(K_\mu) \right) = 0.$$

We also have:

$$\operatorname{corank}\left(\sum_{ij\in E} \boldsymbol{w}^*_{ij}\mathbf{L}_{ij} - \alpha^*\mathbf{L}(K_\mu)\right) = \mathcal{E}_{\mathbf{L}_{\boldsymbol{w}},\mathbf{M}}(\lambda_2(G,\mu,\boldsymbol{w})) + 1$$

and $\operatorname{span}\{\mathbf{M}\vec{\mathbf{1}}\} \subseteq \ker(\sum_{ij\in E} \boldsymbol{w}^*_{ij}\mathbf{L}_{ij} - \alpha^*\mathbf{L}(K_\mu))$. As discussed before, we may choose $\mathbf{X}^*$ such that $\operatorname{span}\{\mathbf{M}\vec{\mathbf{1}}\} \subseteq \ker(\mathbf{X}^*)$. From the above, and using Corollary 20:

$$\operatorname{rank}(\mathbf{X}^*) \leq \operatorname{corank}\left(\sum_{ij\in E} \boldsymbol{w}^*_{ij}\mathbf{L}_{ij} - \alpha^*\mathbf{L}(K_\mu)\right) - 1 = \mathcal{E}_{\mathbf{L}_{\boldsymbol{w}},\mathbf{M}}(\lambda_2(G,\mu,\boldsymbol{w})) \leq 3.$$

□

2.7 Connection to Fastest Mixing Random Walk with Target Degree Sequence

In this section, we discuss how solving the dual of our SDP relaxation can naturally be interpreted as finding the fastest mixing diffusion process with a given stationary distribution.

The dual of R1SDP2 is given by:

$$\begin{aligned} \max \quad & \alpha \\ \text{s.t.} \quad & \sum_{ij\in E} \boldsymbol{w}_{ij}\mathbf{L}_{ij} \succeq \alpha\mathbf{L}(K_\mu) \\ \forall i \in V \quad & \sum_{j:j\sim i} \boldsymbol{w}_{ij} \leq \mu(i) \\ & \boldsymbol{w} \in \mathbb{R}^m_{\geq 0},\ \alpha \in \mathbb{R}_{\geq 0}. \end{aligned} \tag{D}$$

We interpret this SDP dual as minimizing the mixing time of a certain random walk. Consider the continuous analogue of the natural random walk on G:

$$\dot{\boldsymbol{p}}_i = \sum_{j \sim i} \frac{\boldsymbol{w}_{ij}}{\deg(j)} \boldsymbol{p}_j - \boldsymbol{p}_i$$

where $\dot{\boldsymbol{p}}$ is the time derivative of some function $\boldsymbol{p} : [0, \infty) \to \mathbb{R}^V$. This process satisfies:

$$\|\boldsymbol{p}(t) - \boldsymbol{\pi}\|_{D^{-1}}^2 \leq e^{-2\lambda_2(G) \cdot t} \cdot \|\boldsymbol{p}(0) - \boldsymbol{\pi}\|_{D^{-1}}^2 ,$$

where $\boldsymbol{\pi}$ is the probability distribution proportional to the degree, so that the value of $\lambda_2(G)$ regulates the mixing time of the process. In particular, by rewriting the above as a linear systems of ODEs, we find that the dual SDP (D) is the solution to the following problem:

Fastest Mixing Random Walk with Target Distribution.

Given an undirected graph $G = (V, E)$ and a target distribution $\mu : V \to \mathbb{R}_{\geq 0}$, find the assignment of weights $\boldsymbol{w} \in \mathbb{R}_{\geq 0}^E$ minimizing the mixing time of the continuous-time random walk process:

$$\begin{cases} \frac{d\boldsymbol{p}}{dt} = -\mathbf{L}_{\boldsymbol{w}} \mathbf{D}_{\boldsymbol{w}}^{-1} \boldsymbol{p} \\ \boldsymbol{p}(0) = \boldsymbol{p}_0 \end{cases}$$

subject to $\deg_{\boldsymbol{w}}(i) = \mu(i)$.

In the above, we allow the solution to make use of self-loops, so as to guarantee that the target distribution is indeed achievable. For any vertex $i \in V$, the weight of the corresponding self-loop is taken to be the slack: $\mu(i) - \sum_{j \sim i} \boldsymbol{w}_{ij}$. Note that the constraint imposed guarantees that $\lim_{t \to \infty} \boldsymbol{p}(t) = \mu$ for any choice of initial distribution $\boldsymbol{p}_0$.

2.8 Bibliographical Notes

While this presentation of the material is new, several recent works present analogous results. NP-Hardness results for vertex separators were given by Bui and Jones (1992)

Feige et al. (2008) give a $O(\sqrt{\log n})$-approximation for min-ratio vertex cuts, using SDP techniques to construct metrics of negative types on the vertices of G, generalizing the celebrated algorithm of Arora et al. (2009). They also give a constant factor approximation for excluded minor families, which implies the result for planar graphs given in this chapter.

Bobkov et al. (2000) introduce a parameter λ_∞ that serves as a vertex analogue of the spectral gap. This enjoys Cheeger-type guarantees with respect to the symmetric vertex expansion. Louis et al. (2013) design an SDP relaxation of λ_∞ that allows one to efficiently approximate vertex expansion and obtain a solution of objective value $O(\sqrt{\mathrm{opt} \log \triangle(G)})$. Moreover, they show that their algorithm is optimal under the Small Set Expansion hypothesis. Following their work, Louis and Makarychev (2016) design approximation algorithms for small-set vertex expansion by reducing the problem to a hypergraph version of it.

The Fastest Mixing Markov Chain (FMMC) problem was first introduced by Boyd et al. (2006). Olesker-Taylor and Zanetti (2024) design similar approximation algorithms to those described in this chapter, achieving a dependence of $O(\log n)$ instead of $O(\log \triangle(G))$, and they highlight the connection to the FMMC problem. At around the same time, Kwok et al. (2022) improved the dependency from $\log n$ to $\log \Delta$ obtaining an equivalent bound to that presented here.

.

CHAPTER 3

PARTITIONING HYPERGRAPHS

In this chapter, we discuss recent progress on algorithms for partitioning hypergraphs.

3.1 Polymatroidal Cut Functions and Submodular Hypergraph Ratio Cut

In the preliminaries, we introduced the minimum conductance cut problem in graphs. Recall that this problem asks one to find a cut $(S, \overline{S})$ in G, which minimizes:

$$\phi(S) = \frac{w(E(S, \overline{S}))}{\min\{\text{Vol}(S), \text{Vol}(\overline{S})\}}.$$

The problem naturally generalizes to hypergraphs, where one could look for a set S cutting few of the hyperedges while disconnecting the vertex set into large components. However, while there is a single way to cut an edge uv (by placing u on one side of the cut and v on the other) there may be multiple ways of cutting a hyperedge, depending on how many vertices in the hyperedge are separated by the cut, as well as which vertices get separated. This brings about the need to define a cut function $\delta_h : 2^h \to \mathbb{R}$ for every hyperedge $h \in E$, which, on input a subset $S \subseteq V$, returns some cut value $\delta_h(S \cap h)$.

Many different classes of cut functions have been defined and studied. Perhaps the simplest, most natural example is the standard hypergraph cut function, which is simply the indicator of the event that the cut intersects non-trivially with the hyperedge:

$$\delta_h^{cut}(S) = \begin{cases} 1 & \text{if } S \cap h \neq \emptyset \text{ and } S \cap h \neq h, \\ 0 & \text{otherwise.} \end{cases}$$

This is sometimes referred to as the all-or-nothing cut function (Veldt et al. (2022)). Sim-

ilarly, one may consider a directed version of it, in which every hyperedge h contains two subsets h_{head} and h_{tail}, and the cut function is given by:

$$\delta_h^{dir}(S) = \begin{cases} 1 & \text{if } S \cap h_{\text{head}} \neq \emptyset \text{ and } \overline{S} \cap h_{\text{tail}} \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Note that both these cut functions return values in the $[0, 1]$ range. We will assume every cut function has this property. This can be done without loss of generality since the hypergraphs are taken to be weighted, and one can simply normalize a cut function by reweighting the corresponding hyperedge. Li and Milenkovic (2018) consider cut functions that are symmetric and submodular. Veldt et al. (2022) consider cut functions that are computed exclusively as functions of the cardinality of the intersection of S with h.

We now introduce the class of polymatroidal cut functions.

Definition 5 (Polymatroidal Cut Function)**.** A cut function $\delta_h : 2^h \to \mathbb{R}_{\geq 0}$ is *polymatroidal* if, for all $S \subseteq h$, it can be expressed as

$$\delta_h(S) = \min\left\{F_h^-(S),\ F_h^+(h \setminus S)\right\},$$

where the associated functions $F_h^-, F_h^+ : 2^h \to \mathbb{R}$ are non-decreasing submodular functions such that $F_h^-(\varnothing) = F_h^+(\varnothing) = 0$. When the associated functions F_h^- and F_h^+ are identical, we refer to the cut function δ_h as *symmetric polymatroidal*.

This class is extremely versatile, as it encompasses all classes of cut functions described above, both directed and undirected.

Once a collection of cut functions $\{\delta_h\}_{h \in E}$ is defined, one can then look for a cut $(S, \overline{S})$ minimizing:

$$\phi(S) = \frac{\sum_{h \in E} w_h \delta_h(S \cap h)}{\min\{\text{Vol}(S), \text{Vol}(\overline{S})\}}.$$

In the previous chapter, we also saw that one can often generalize the problem by considering an arbitrary measure μ on the vertex set, instead of restricting themselves to the volume. This yields the following problem.

Definition 6 (Submodular Hypergraph Ratio-Cut)**.** Given a hypergraph $G = (V, E)$ with positive hyperedge weights $\boldsymbol{w} \in \mathbb{Z}^E_{>0}$, non-negative vertex weights $\boldsymbol{\mu} \in \mathbb{Z}^V_{\geq 0}$, and a collection of submodular cut functions $\{\delta_h\}_{h \in E}$, the *ratio-cut objective* on G is defined for all $S \subseteq V$ as:

$$\Psi_G(S) \stackrel{\text{def}}{=} \frac{\delta_G(S)}{\min\{\mu(S), \mu(\bar{S})\}} = \frac{\sum_{h \in E} w_h \cdot \delta_h(S \cap h)}{\min\{\mu(S), \mu(\bar{S})\}}.$$

The *ratio-cut* of G is $\Psi^*_G \stackrel{\text{def}}{=} \min_{S \subset V} \Psi_G(S)$.

We note that this problem generalizes a number of optimization problems on graphs, including graph conductance, graph expansion and their directed counterparts.

We now show that this problem is equivalent to a (non-convex) continuous optimization problem, obtained by replacing the submodular cut functions with their Lovász extensions.

Lemma 21. *Given a weighted hypergraph $G^* = (V, E, \boldsymbol{\mu}, \boldsymbol{w})$, let:*

$$\bar{\Psi}_G(\boldsymbol{x}) \stackrel{\text{def}}{=} \frac{\sum_{h \in E} w_h \bar{\delta}_h(\boldsymbol{x})}{\min_u \|\boldsymbol{x} - u\mathbf{1}\|_{\boldsymbol{\mu},1}},$$

and let:

$$\bar{\Psi}^*_G \stackrel{\text{def}}{=} \min_{\boldsymbol{x} \in \mathbb{R}^V} \bar{\Psi}_G(\boldsymbol{x}). \qquad \texttt{(RC-NonCvx)}$$

We then have $\Psi^*_G = \bar{\Psi}^*_G$. Moreover, there exists an algorithm that, given any $\boldsymbol{x} \in \mathbb{R}^V$, recovers a cut $S \subseteq V$ satisfying

$$\Psi_G(S) \leq \bar{\Psi}_G(\boldsymbol{x})$$

in time $O\big(|V| \log |V| + \mathsf{sp}(G)\big)$.

Proof. Observe that the denominator of the ratio-cut objective is a monotone submodular

cut function applied over the entire vertex set:

$$\delta_{\boldsymbol{\mu}}(S) = \min\{\boldsymbol{\mu}(S), \boldsymbol{\mu}(\bar{S})\}.$$

Consequently, for every $S \subseteq V$:

$$\Psi_G(S) = \frac{\sum_{h \in E} w_h \delta_h(S \cap h)}{\min\{\boldsymbol{\mu}(S), \boldsymbol{\mu}(\bar{S})\}} = \frac{\delta_G(S)}{\delta_{\boldsymbol{\mu}}(S)} = \frac{\overline{\delta}_G(\mathbf{1}^S)}{\overline{\delta}_{\boldsymbol{\mu}}(\mathbf{1}^S)},$$

which gives:

$$\Psi_G^* \geq \min_{\boldsymbol{x} \in \mathbb{R}^n} \frac{\overline{\delta}_G(\boldsymbol{x})}{\overline{\delta}_{\boldsymbol{\mu}}(\boldsymbol{x})}.$$

We now prove the opposite inequality by showing the second part of the lemma. Notice that the value of the ratio in the right-hand side of (`RC-NonCvx`) is invariant under the shifting of all the coordinates by a fixed constant. Furthermore, both the numerator and the denominator of the right-hand side of (`RC-NonCvx`) are homogeneous and hence their ratio is invariant under scaling $\boldsymbol{x}$ by a constant, i.e., for any $u \in \mathbb{R}$:

$$\frac{\overline{\delta}_G(u \cdot \boldsymbol{x})}{\overline{\delta}_{\boldsymbol{\mu}}(u \cdot \boldsymbol{x})} = \frac{u \cdot \overline{\delta}_G(\boldsymbol{x})}{u \cdot \overline{\delta}_{\boldsymbol{\mu}}(\boldsymbol{x})} = \frac{\overline{\delta}_G(\boldsymbol{x})}{\overline{\delta}_{\boldsymbol{\mu}}(\boldsymbol{x})}.$$

Hence, for any $\boldsymbol{x}$, by letting $\hat{\boldsymbol{x}}$ be defined as:

$$\hat{x_i} = \frac{x_i - \min_j x_j}{\max_{j,k}(x_j - x_k)},$$

we have $\max_i \hat{x}_i = 1$, $\min_i \hat{x}_i = 0$, and:

$$\frac{\overline{\delta}_G(\boldsymbol{x})}{\overline{\delta}_{\boldsymbol{\mu}}(\boldsymbol{x})} = \frac{\overline{\delta}_G(\hat{\boldsymbol{x}})}{\overline{\delta}_{\boldsymbol{\mu}}(\hat{\boldsymbol{x}})}.$$

We then have, as per Propositions 3.1 and 3.1 in Bach (2013):

$$\frac{\overline{\delta}_G(\boldsymbol{x})}{\overline{\delta}_{\boldsymbol{\mu}}(\boldsymbol{x})} = \frac{\overline{\delta}_G(\hat{\boldsymbol{x}})}{\overline{\delta}_{\boldsymbol{\mu}}(\hat{\boldsymbol{x}})} = \frac{\mathop{\mathbb{E}}_{t\sim[0,1]}\left[\delta_G(S_t(\hat{\boldsymbol{x}}))\right]}{\mathop{\mathbb{E}}_{t\sim[0,1]}\left[\delta_{\boldsymbol{\mu}}(S_t(\hat{\boldsymbol{x}}))\right]}.$$

Hence, there must exist some $t \in [0,1]$ such that:

$$\frac{\overline{\delta}_G(\boldsymbol{x})}{\overline{\delta}_{\boldsymbol{\mu}}(\boldsymbol{x})} \geq \frac{\delta_G(S_t(\hat{\boldsymbol{x}}))}{\delta_{\boldsymbol{\mu}}(S_t(\hat{\boldsymbol{x}}))}.$$

This gives:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \frac{\overline{\delta}_G(\boldsymbol{x})}{\overline{\delta}_{\boldsymbol{\mu}}(\boldsymbol{x})} = \Psi^*_G,$$

as needed for the first part of the lemma. Moreover, computing and checking all of the sets $\{S_t(\boldsymbol{x})\}_{t\in[0,1]}$ only requires time $O(n \log n)$ for sorting the entries of $\boldsymbol{x}$ and time $O(\mathsf{sp}(G) + |V|)$ for evaluating the numerator and denominator. □

3.2 Some Properties of Polymatroidal Cut Functions

In this section, we establish some technical properties of polymatroidal cut functions which will be useful in later sections. We begin by showing that polymatroidal cut functions, are in fact submodular functions.

Lemma 22. *Let V be an arbitrary finite set and $F, G : 2^V \to \mathbb{R}$ be monotone non-decreasing submodular functions. If $\delta : 2^V \to \mathbb{R}$ is the function defined by:*

$$\delta(S) \stackrel{\text{def}}{=} \min\{F(S), G(V \setminus S)\},$$

for all $S \subseteq V$, then δ is submodular.

Proof. Let $S, T \subseteq V$ and denote with $\overline{S}$ and $\overline{T}$ their complements in V. We consider four

possible cases, corresponding to four ways of determining $\delta(S)$ and $\delta(T)$. In the first two cases, we exploit the submodularity of F and G.

Case 1: $F(S) \leq G(\overline{S})$ and $F(T) \leq G(\overline{T})$, which implies $\delta(S) = F(S)$ and $\delta(T) = F(T)$. We have:

$$\begin{aligned}\delta(S \cup T) + \delta(S \cap T) &= \min\{F(S \cup T),\ G(\overline{S} \cap \overline{T})\} + \min\{F(S \cap T), G(\overline{S} \cup \overline{T})\} \\ &\leq F(S \cup T) + F(S \cap T) \\ &\leq F(S) + F(T) = \delta(S) + \delta(T).\end{aligned}$$

Case 2: $G(\overline{S}) \leq F(S)$ and $G(\overline{T}) \leq F(T)$, which implies $\delta(S) = G(\overline{S})$ and $\delta(T) = G(\overline{T})$. We have:

$$\begin{aligned}\delta(S \cup T) + \delta(S \cap T) &= \min\{F(S \cup T),\ G(\overline{S} \cap \overline{T})\} + \min\{F(S \cap T), G(\overline{S} \cup \overline{T})\} \\ &\leq G(\overline{S} \cap \overline{T}) + G(\overline{S} \cup \overline{T}) \\ &\leq G(\overline{S}) + G(\overline{T}) = \delta(S) + \delta(T).\end{aligned}$$

In the last two cases, we rely on the monotonocity of F and G.

Case 3: $F(S) \leq G(\overline{S})$ and $G(\overline{T}) \leq F(T)$, which implies $\delta(S) = F(S)$ and $\delta(T) = G(\overline{T})$. We have:

$$\begin{aligned}\delta(S \cup T) + \delta(S \cap T) &= \min\{F(S \cup T),\ G(\overline{S} \cap \overline{T})\} + \min\{F(S \cap T), G(\overline{S} \cup \overline{T})\} \\ &\leq G(\overline{S} \cap \overline{T}) + F(S \cap T) \\ &\leq G(\overline{T}) + F(S) = \delta(S) + \delta(T).\end{aligned}$$

Case 4: $G(\overline{S}) \leq F(S)$ and $F(T) \leq G(\overline{T})$, which implies $\delta(S) = G(\overline{S})$ and $\delta(T) = F(T)$.

We have:

$$\begin{aligned}\delta(S \cup T) + \delta(S \cap T) &= \min\{F(S \cup T),\ G(\overline{S} \cap \overline{T})\} + \min\{F(S \cap T), G(\overline{S} \cup \overline{T})\} \\ &\leq G(\overline{S} \cap \overline{T}) + F(S \cap T) \\ &\leq G(\overline{S}) + F(T) = \delta(S) + \delta(T).\end{aligned}$$

It follows that δ is submodular, as we have shown that for every $S, T \subseteq V$:

$$\delta(S \cup T) + \delta(S \cap T) \leq \delta(S) + \delta(T).$$

□

We also recall the following fact, which plays a crucial role in the analysis of our approximation algorithms based on metric embeddings. This is a consequence of Proposition 4.8 in Bach (2013).

Lemma 23. *Let $F : 2^V \to \mathbb{R}_{\geq 0}$ be a monotone non-decreasing submodular function with $F(\emptyset) = 0$. Its Lovász extension $\bar{F}$ is monotone and non-decreasing over $\mathbb{R}^V$.*

We now show that we can express the Lovász extension of any polymatroidal cut function δ_h in terms of the Lovász extensions of the functions F_h^+ and F_h^- used to define it. For the proof of this fact, we will make use of some results in the monograph of Bach (2013).

Lemma 24. *Given a polymatroidal cut function δ_h, we have, for all $\boldsymbol{x} \in \mathbb{R}^h$:*

$$\overline{\delta}_h(\boldsymbol{x}) = \min_{\nu \in \mathbb{R}} \bar{F}_h^-((\boldsymbol{x} - \nu \mathbf{1}_h)_+) + \bar{F}_h^+((\boldsymbol{x} - \nu \mathbf{1}_h)_-).$$

Proof. Since F_h^- and F_h^+ are monotone, for any $\boldsymbol{x} \in \mathbb{R}^n$ there exists a value $z^* = z^*(\boldsymbol{x})$ such that:

$$F_h^-(\{i \in V \mid \boldsymbol{x}(i) \geq z\}) \leq F_h^+(\{i \in V \mid \boldsymbol{x}(i) < z\}),$$

for all $z \geq z^*$, and:

$$F_h^-(\{i \in V \mid \boldsymbol{x}(i) \geq z\}) \geq F_h^+(\{i \in V \mid \boldsymbol{x}(i) < z\}),$$

for all $z < z^*$. We then have, by Proposition 3.1(c) in Bach (2013):

$$\begin{aligned}
\overline{\delta}_h(\boldsymbol{x}) &= \int_{-\infty}^{\infty} \delta(\{i \in V \mid \boldsymbol{x}(i) \geq z\}) dz \\
&= \int_{\infty}^{z^*(\boldsymbol{x})} F_h^+(\{i \in V \mid \boldsymbol{x}(i) < z\}) dz + \int_{z^*(\boldsymbol{x})}^{\infty} F_h^-(\{i \in V \mid \boldsymbol{x}(i) \geq z\}) dz \\
&= \min_{\nu \in \mathbb{R}} \int_{\infty}^{0} F_h^+(\{i \in V \mid \boldsymbol{x}(i) - \nu < z\}) dz + \int_{0}^{\infty} F_h^-(\{i \in V \mid \boldsymbol{x}(i) - \nu \geq z\}) dz \\
&= \min_{\nu \in \mathbb{R}} \bar{F}_h^-((\boldsymbol{x} - \nu \mathbf{1})_+) + \bar{F}_h^+((\boldsymbol{x} - \nu \mathbf{1})_-),
\end{aligned}$$

as needed. □

When a polymatroidal cut function is symmetric, we also have the following fact.

Lemma 25. *Given a symmetric polymatroidal cut function δ_h with $F_h \stackrel{\text{def}}{=} F_h^+ = F_h^-$, we have, for all $\boldsymbol{x} \in \mathbb{R}^h$:*

$$\min_{\nu \in \mathbb{R}} \bar{F}_h(|\boldsymbol{x} - \nu \mathbf{1}_h|) \leq \overline{\delta}_h(\boldsymbol{x}) \leq 2 \cdot \min_{\nu \in \mathbb{R}} \bar{F}_h(|\boldsymbol{x} - \nu \mathbf{1}_h|).$$

Proof. Since F_h is submodular, its Lovász extension $\bar{F}_h$ is convex, and hence, by Lemma 24:

$$\begin{aligned}
\bar{\delta}_h(\boldsymbol{x}) &= \min_{\nu \in \mathbb{R}} \bar{F}_h((\boldsymbol{x} - \nu \mathbf{1}_h)_+) + \bar{F}_h((\boldsymbol{x} - \nu \mathbf{1}_h)_-) = \min_{\nu \in \mathbb{R}} 2 \cdot \left(\frac{1}{2} \bar{F}_h((\boldsymbol{x} - \nu \mathbf{1}_h)_+) + \frac{1}{2} \bar{F}_h((\boldsymbol{x} - \nu \mathbf{1}_h)_-) \right) \\
&\geq \min_{\nu \in \mathbb{R}} 2 \cdot \bar{F}_h \left(\frac{1}{2}(\boldsymbol{x} - \nu \mathbf{1}_h)_+ + \frac{1}{2}(\boldsymbol{x} - \nu \mathbf{1}_h)_- \right) \\
&= \min_{\nu \in \mathbb{R}} 2 \cdot \bar{F}_h \left(\frac{1}{2} |\boldsymbol{x} - \nu \mathbf{1}_h| \right) \\
&= \min_{\nu \in \mathbb{R}} \bar{F}_h \left(|\boldsymbol{x} - \nu \mathbf{1}_h| \right),
\end{aligned}$$

where in the above we applied Jensen's Inequality and the fact that $\bar{F}_h$ is positive homogeneous (Proposition 3.1 (e) from Bach (2013)). On the other hand, since F_h is monotone, we have:

$$\begin{aligned}\bar{\delta}_h(\boldsymbol{x}) &= \min_{\nu\in\mathbb{R}} \bar{F}_h((\boldsymbol{x}-\nu\mathbf{1}_h)_+) + \bar{F}_h((\boldsymbol{x}-\nu\mathbf{1}_h)_-) \leq \min_{\nu\in\mathbb{R}} \bar{F}_h(|\boldsymbol{x}-\nu\mathbf{1}_h|) + \bar{F}_h(|\boldsymbol{x}-\nu\mathbf{1}_h|) \\ &= 2\cdot \min_{\nu\in\mathbb{R}} \bar{F}_h(|\boldsymbol{x}-\nu\mathbf{1}_h|),\end{aligned}$$

where the inequality follows from Lemma 23. This completes the proof. □

3.3 Hypergraph Flows, Base Polytopes and Embeddings

In this section, we discuss hypergraph flows, which will serve as the main dual object to hypergraph cuts. In fact, much like in the case of graphs, one can use the existence of certain hypergraphs flow to lower bound the value of cuts. Note that this needs to be dependent on the structure of the cut functions used. Hence, we will introduce a notion of flow congestion that depends on the base polytope of the cut functions being considered.

We begin by considering properties of the base polytopes of polymatroidal cut functions.

By the definition of base polytope in Equation 3.1, we have

$$\mathcal{B}(\delta_h) = \{\boldsymbol{y}\in\mathbb{R}^h : \langle \boldsymbol{y}, \mathbf{1}_h\rangle = 0 \wedge \forall S\subset h,\, \langle \boldsymbol{y}, \mathbf{1}_S\rangle \leq \min\{F_h^-(S), F_h^+(h\setminus S)\}\}.$$

The first constraint implies that for all $S\subset h$, we must have $\langle \boldsymbol{y}, \mathbf{1}_S\rangle = -\langle \boldsymbol{y}, \mathbf{1}_{h\setminus S}\rangle$. Hence, we can rewrite:

$$\begin{aligned}\mathcal{B}(\delta_h) &= \{\boldsymbol{y}\in\mathbb{R}^h \perp \mathbf{1}_h : \forall S\subset h,\, \langle \boldsymbol{y}, \mathbf{1}_S\rangle \leq F_h^-(S) \wedge -\langle \boldsymbol{y}, \mathbf{1}_{h\setminus S}\rangle \leq F_h^+(h\setminus S)\} \\ &= \{\boldsymbol{y}\in\mathbb{R}^h \perp \mathbf{1}_h : \forall S\subset h,\, -F_h^+(S) \leq \langle \boldsymbol{y}, \mathbf{1}_S\rangle \leq F_h^-(S)\} \\ &= \{\boldsymbol{y}\in\mathbb{R}^h \perp \mathbf{1}_h : (\boldsymbol{y})_+ \in \mathcal{P}_+(F_h^-) \wedge (\boldsymbol{y})_- \in \mathcal{P}_+(F_h^+)\}.\end{aligned} \tag{3.1}$$

Example: Base Polytope for Standard Hypergraph Cut Function Consider the case of the standard hypergraph cut function $\delta_h^{\mathsf{cut}}(S)$ defined in the previous section. Then, it is easy to see that:

$$\mathcal{B}(\delta_h^{\mathsf{cut}}) = \{\boldsymbol{y} \in \mathbb{R}^h \perp \mathbf{1}_h \mid \|\boldsymbol{y}\|_1 \leq 2\}.$$

As discussed above, we can think of each entry $\boldsymbol{y}(i)$ of $\boldsymbol{y}$ for $i \in h$, as the amount of flow flowing into h from vertex i, and the constraint $\|\boldsymbol{y}\|_1 \leq 2$ enforces that the total amount of flow through the hyperedge h is no more than 1. (This is equivalent to a vertex capacity constraint on the factor vertex corresponding to the hyperedge h in the factor graph $\hat{G}$, as defined below)

Example: Base Polytope for Directed Hypergraph Cut Functions Consider the class of directed hypergraph cut functions. Each of these cut functions δ_h has an associated head set $H_h \subseteq h$ and an associated tail set $T_h \subseteq h$ and $\delta_h(S)$ has value 1 if and only if $H_h \cap S \neq \emptyset$ and $\overline{S} \cap \emptyset \neq \emptyset$. Their base polytope is then given by:

$$\mathcal{B}(\delta_h) = \{\boldsymbol{y} \in \mathbb{R}^h \perp \mathbf{1}_h \mid \boldsymbol{y}(H_h) \leq 1 \wedge \forall i \in T_h \setminus H_h : \boldsymbol{y}(i) \leq 0\}.$$

Definition 7 (Hypergraph Flow, Congestion and Demand)**.** Given a weighted submodular hypergraph $G = (V, E, \boldsymbol{w} \in \mathbb{Z}_{>0}^E, \boldsymbol{\mu} \in \mathbb{Z}_{\geq 0}^V)$, a *hyperedge flow* over a hyperedge $h \in E$ is a vector $\boldsymbol{y}_h \in \mathbb{R}^h \perp \mathbf{1}_h$. A *hypergraph flow* $\mathbf{Y} \in \bigoplus_{h \in E} \mathbb{R}^h \perp \mathbf{1}_h$ is a direct sum of hyperedge flows $\mathbf{Y} = (\boldsymbol{y}_h)_{h \in E}$.

Just like in graphs, we can define demands to be the net amount of flow in and out of vertices. In particular, the demand vector $\mathrm{dem}(\mathbf{Y}) \in \mathbb{R}^V \perp \mathbf{1}$ of a hypergraph flow $\mathbf{Y} = (\boldsymbol{y}_h)_{h \in E}$ is

$$\mathrm{dem}_i(\mathbf{Y}) = \sum_{h \in E : h \ni i} \boldsymbol{y}_h(i). \tag{3.2}$$

When a collection of submodular cut functions $\{\delta_h\}_{h \in E}$ is given, we may define a notion

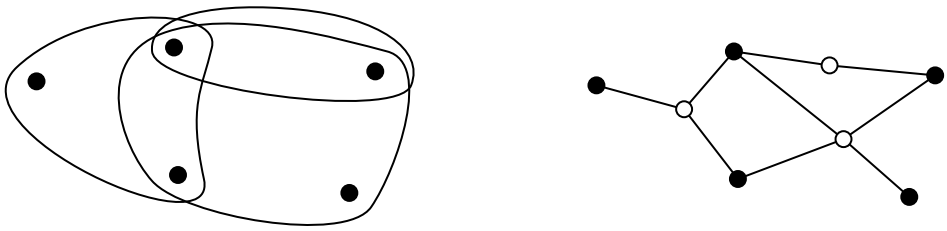

Figure 3.1: A hypergraph (left) and its factor graph (right). In the factor graph image, the variable vertices are shown in solid black, while the factor vertices are shown as empty circles.

of congestion based on the maximum violation of the polymatroidal constraints defining each $\mathcal{B}(\delta_h)$, i.e. the congestion of flow $\mathbf{Y}$ with respect to G and the cut functions $\{\delta_h\}_{h\in E}$ is the quantity:

$$\mathrm{cong}_G(\mathbf{Y}) = \min\{\rho \geq 0 : \forall h \in E_G,\, \boldsymbol{y}_h \in \rho \cdot \boldsymbol{w}_h \cdot \mathcal{B}(\delta_h)\}. \tag{3.3}$$

We will sometimes make use of the vector space structure inherited from $\mathbf{Y} \in \bigoplus \mathbb{R}^h \perp \mathbf{1}_h$ to define linear combinations of hypergraph flows $c\mathbf{Y}_1+\mathbf{Y}_2$ for any $c \in \mathbb{R}$. The demand vector also behaves linearly as $\mathrm{dem}(c\mathbf{Y}_1 + \mathbf{Y}_2) = c \cdot \mathrm{dem}(\mathbf{Y}_1) + \mathrm{dem}(\mathbf{Y}_2)$. Note that hypergraph flows over hyperedges of rank 2 are equivalent to graph flows as, for an edge $\{i, j\} \in E$, $\boldsymbol{y}_{ij}(i)$ represents the flow from i to j, while $\boldsymbol{y}_{ij}(j) = -\boldsymbol{y}_{ij}(j)$ is the flow from j to i.

We now show that one can interpret hypergraph flows as flows in a suitably defined graph.

Definition 8 (Factor Graph (See Figure 3.1))**.** Given a hypergraph $G = (V, E)$ its *factor graph* is an unweighted graph $\hat{G} = (V \cup E_G, \hat{E}_G)$ that is bipartite between the *variable* vertices representing vertices $i \in V$, and *factor* vertices representing hyperedges $h \in E_G$. For $i \in V$ and $h \in E_G$, the edge $\{i, h\}$ belongs to $\hat{E}_G$ if and only if $i \in h$, i.e.:

$$\hat{E}_G = \{\{i, h\} \in V \times E_G : i \in h\}.$$

With this definition, every hypergraph flow $\mathbf{Y}$ can be interpreted as a (graph) flow over

$\hat{G}$, where the flow is preserved at every factor vertex. In this flow, $\boldsymbol{y}_h(i)$ units of flow move from the variable vertex corresponding to i in $\hat{G}$ into the factor vertex corresponding to h.

This also allows us to extend the definition of graph flow embeddings to embedding of graphs into arbitrary hypergraphs as flows.

Definition 9 (Hypergraph Flow Embedding)**.** Let $G = \left(V, E_G, \boldsymbol{w}^G, \boldsymbol{\mu}\right)$ be a weighted submodular hypergraph equipped with a collection of polymatroidal cut functions $\{\delta_h\}_{h \in E_G}$ and let $H = \left(V, E_H, \boldsymbol{w}^H\right)$ be a weighted directed graph on the same vertex set. We say that the graph H embeds into G as a flow with congestion $\rho \geq 0$, denoted $H \preceq_\rho G$ if there exist hypergraph flows $\{\mathbf{Y}^e\}_{e \in E_H}$ such that:

1. For all $e = (i, j) \in E_H$, the flow $\mathbf{Y}^e$ routes w_e^H units of flow from vertex i to vertex j, i.e.,
$$\operatorname{dem}(\mathbf{Y}^e) = w_e^H \cdot (\mathbf{1}_i - \mathbf{1}_j)\,.$$

2. For each $h \in E_G$, the flows ${\mathbf{Y}^e}_{e \in E_H}$ respect the polymatroidal capacity constraints, i.e., for all $h \in E_G$:
$$\sum_{e \in E_h} (\boldsymbol{y}_h^e)_+ \in \rho \cdot w_h \cdot \mathcal{P}_+(F_h^-),$$
$$\sum_{e \in E_h} (\boldsymbol{y}_h^e)_- \in \rho \cdot w_h \cdot \mathcal{P}_+(F_h^+).$$

Just like in the standard graph setting, one can use flow embeddings of graphs into hypergraphs as certificate of lower bounds on the cut functions, as in the following theorem.

Theorem 26. *Let $G = \left(V, E_G, \boldsymbol{w}^G, \boldsymbol{\mu}\right)$ be a weighted hypergraph equipped with a collection of polymatroidal cut functions $\{\delta_h\}_{h \in E_G}$, and $H = \left(V, E_H, \boldsymbol{w}^H\right)$ be a weighted directed graph.*

If $H \preceq_\rho G$, then

$$\forall \boldsymbol{x} \in \mathbb{R}^V\,, \sum_{(i,j)\in E_H} w_{ij}^H \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j)_+ \le \rho \cdot \sum_{h\in E_G} w_h^G \cdot \bar{\delta}_h(\boldsymbol{x}) \;\; \textit{and} \;\; \forall S \subseteq V\,,\, \delta_H(S) \le \rho \cdot \delta_G(S).$$

For symmetric polymatroidal cut functions $\{\delta_h\}_{h\in E_G}$, $H \preceq_\rho G$ implies the following bound on the undirected symmetrization $\tilde{H} = (V, E_{\tilde{H}}, \boldsymbol{w}^{\tilde{H}})$:

$$\forall \boldsymbol{x} \in \mathbb{R}^V\,, \sum_{\{i,j\}\in E_{\tilde{H}}} w_{ij}^{\tilde{H}} \cdot |\boldsymbol{x}_i - \boldsymbol{x}_j| \le 2\cdot\rho\cdot \sum_{h\in E_G} w_h^G \cdot \bar{\delta}_h(\boldsymbol{x}) \;\; \textit{and} \;\; \forall S \subseteq V\,,\, \delta_{\tilde{H}}(S) \le 2\cdot\rho\cdot\delta_G(S).$$

Proof. Consider the hypergraph flows $\{\mathbf{Y}^e\}_{e\in E^H}$ associated with the embedding $H \preceq_\rho G$ and let $\boldsymbol{u} \in \mathbb{R}^{E_G}$ be an arbitrary vector over the hyperedges. As each $\mathbf{Y}^e$ routes demand w_e^H between the endpoints of $e = \{u, v\} \in E_H$, we have

$$\begin{aligned} w_e^H \cdot (\boldsymbol{x}_u - \boldsymbol{x}_v)_+ = (\langle \mathrm{dem}(\mathbf{Y}^e), \boldsymbol{x}\rangle)_+ &= \left(\sum_{h\in E_G} \langle \boldsymbol{y}_h^e, \boldsymbol{x}_h\rangle\right)_+ = \left(\sum_{h\in E_G} \langle \boldsymbol{y}_h^e, \boldsymbol{x}_h - u_h\mathbf{1}_h\rangle\right)_+ \\ &\le \sum_{h\in E_G} \langle (\boldsymbol{y}_h^e)_+, (\boldsymbol{x}_h - u_h\mathbf{1}_h)_+\rangle + \langle (\boldsymbol{y}_h^e)_-, (\boldsymbol{x}_h - u_h\mathbf{1}_h)_-\rangle, \end{aligned}$$

where the last equality relies on the fact that $\boldsymbol{y}_h^e \in \mathbb{R}^h \perp \mathbf{1}_h$. Summing over all edges in E_h yields:

$$\begin{aligned} \sum_{e=\{i,j\}\in E_H} w_{ij}^H \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j)_+ &\le \sum_{e\in E_H}\sum_{h\in E_G} \langle (\boldsymbol{y}_h^e)_+, (\boldsymbol{x}_h - u_h\mathbf{1}_h)_+\rangle + \langle (\boldsymbol{y}_h^e)_-, (\boldsymbol{x}_h - u_h\mathbf{1}_h)_-\rangle, \\ &= \sum_{h\in E_G} \left\langle \sum_{e\in E_H} (\boldsymbol{y}_h^e)_+, (\boldsymbol{x}_h - u_h\mathbf{1}_h)_+ \right\rangle \\ &\quad + \left\langle \sum_{e\in E_H} (\boldsymbol{y}_h^e)_-, (\boldsymbol{x}_h - u_h\mathbf{1}_h)_- \right\rangle \\ &\le \rho \cdot \sum_{h\in E_G} w_h^G \cdot \left(\bar{F}_h^-((\boldsymbol{x}_h - u_h\mathbf{1}_h)_+) + \bar{F}_h^+((\boldsymbol{x}_h - u_h\mathbf{1}_h)_-)\right). \end{aligned}$$

The second inequality follows from Definition 9 and the characterization of Lovász extensions by the positive submodular polytope (see Proposition 3.4 in Bach (2013)). Finally, the first statement in the theorem follows by setting the arbitrary $\boldsymbol{u}$ to be the required minimizer. The specialization to cuts follows by plugging in $\boldsymbol{x} = \mathbf{1}^S$.

In the symmetric case, the capacity constraints in the definition become simply, for all $h \in E_G$:

$$\sum_{e \in E_h} (\boldsymbol{y}_h^e)_+ \,, \sum_{e \in E_h} (\boldsymbol{y}_h^e)_- \in \rho \cdot w_h \cdot \mathcal{P}_+(F_h).$$

As a result, the orientation of each arc $e = (i, j) \in E_H$ can be discarded in this case, as the flow $\mathbf{Y}^e$ can be oriented arbitrarily while satisfying the capacity constraint. Therefore, we can repeat the same analysis for the flow embedding given by $\{-\mathbf{Y}^e\}_{e \in E_H}$. Summing the positive and negative parts, we obtain the required statement. □

Moreover, given a hypergraph flow, one can compute a corresponding demand graph efficiently as shown below.

Theorem 27. *Let $G = \big(V, E_G, \boldsymbol{w}^G, \boldsymbol{\mu}\big)$ be a weighted hypergraph equipped with a collection of polymatroidal cut functions $\{\delta_h\}_{h \in E_G}$. Given a hypergraph flow $\mathbf{Y}$ over G, there is an algorithm that, in time $\tilde{O}\big(\mathsf{sp}(G)\big)$, computes a weighted directed bipartite graph $H = \big(V, E_h, \boldsymbol{w}^H\big)$ such that $H \preceq_{\mathrm{cong}_G(\mathbf{Y})} G$ and $|E_H| = \tilde{O}\big(\mathsf{sp}(G)\big)$. Furthermore, all vertices $i \in V$ with non-negative (resp. negative) demand $\mathrm{dem}_i(\mathbf{Y})$ are source nodes (resp. sink nodes) in H, each with out-degree (resp. in-degree) equal $\mathrm{dem}_i(\mathbf{Y})$.*

Proof. Consider the hypergraph flow as a flow over the factor graph $\hat{G}$. By a standard application of dynamic trees (see for example Lemma 3.8 in the paper of Sherman (2009)) to the computation of flow-path decompositions (see e.g. Ahujia et al. (1993)) over $\hat{G}$, we can compute, in time $\tilde{O}(|\hat{E}|_G) = \tilde{O}(\mathsf{sp}(G))$, a graph $H = (V, E_H, \boldsymbol{w}^H)$ such that $|E_H| = \tilde{O}(\mathsf{sp}(G))$ and such that there exist hypergraph flows $\{\mathbf{Y}^e\}_{e \in E_H}$ in G with the following properties:

- $\sum_{e \in E_H} \mathbf{Y}^e = \mathbf{Y}$,
- For any $e \in E_H$ and any factor graph edge $\{i, h\}$, the flow $(\mathbf{Y}_h^e)_i$ has the same sign as $(\mathbf{Y}_h)_i$. In particular, for all $h \in E_G$:

$$\sum_{e \in E_H} (\boldsymbol{y}_h^e)_+ = (\boldsymbol{y}_h)_+ \text{ and } \sum_{e \in E_H} (\boldsymbol{y}_h^e)_- = (\boldsymbol{y}_h)_-. \tag{3.4}$$

- For all $e = (i, j) \in E^H$, we have $\mathrm{dem}_i(\mathbf{Y}) \geq 0$ and $\mathrm{dem}_j(\mathbf{Y}) < 0$. The flow $\mathbf{Y}^e$ routes w_e^H units of flow between vertex i and vertex j, i.e.,

$$\mathrm{dem}(\mathbf{Y}^e) = w_e^H(\mathbf{1}_i - \mathbf{1}_j).$$

The last point proves the required statements about the bipartiteness of H and its degrees. To complete the proof, we verify that $H \preceq_{\mathrm{cong}_G(Y)} G$. By the definition of congestion in Equation 3.3, we have that for all $h \in E_G$:

$$\boldsymbol{y}_h \in \mathrm{cong}_{G(\mathbf{Y})} \cdot w_h \cdot \mathcal{B}(\delta_h).$$

By the characterization of the base polytope in Equation 3.1, this implies that:

$$(\boldsymbol{y}_h)_+ \in \mathrm{cong}_{G(\mathbf{Y})} \cdot w_h \cdot \mathcal{P}(F_h^-) \text{ and } (\boldsymbol{y}_h)_- \in \mathrm{cong}_{G(\mathbf{Y})} \cdot w_h \cdot \mathcal{P}(F_h^+).$$

Together with Equation 3.4 and Definition 9, this completes the proof. □

3.4 A $O(\sqrt{\log n})$-Approximation Algorithm via Metrics of Negative Type

In this section we prove the following theorem:

Theorem 28. *There exists a randomized polynomial-time $O\big(\sqrt{\log |V|}\big)$-approximation algorithm for solving the minimum ratio-cut problem on weighted submodular hypergraphs equipped with polymatroidal cut functions.*

We begin by providing a relaxation for the symmetric version of the problem, in which $F_h^+ = F_h^-$ for every $h \in E$. We will extend this result to the general (non-symmetric) case later in this section.

Given a weighted submodular hypergraph $G = (V, E, \boldsymbol{w}, \boldsymbol{\mu})$, we construct a vector-program relaxation of the continuous non-convex formulation in Equation (`RC-NonCvx`) by associating to each vertex $i \in V \cup E$ of the factor graph $\hat{G}$ a vector $\boldsymbol{v}_i$. For a hyperedge $h \in E$ and $i \in h$, we can then relax the terms $|\boldsymbol{x}_i - \nu_h \mathbf{1}_h|$, which make up the argument of $\bar{F}_h$ in Fact 25, with the distance $\boldsymbol{d}_i^h \stackrel{\text{def}}{=} \|\boldsymbol{v}_i - \boldsymbol{v}_h\|_2^2$. We obtain the following semidefinite program in its vector embedding form:

$$
\begin{aligned}
\min_{\boldsymbol{v}_i} \quad & \sum_{h \in E} w_h \cdot \bar{F}_h(\boldsymbol{d}^h) \\
\text{s.t.} \quad & \sum_{\{i,j\} \subseteq V} \frac{\mu(i)\mu(j)}{\mu(V)} \cdot \|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2^2 \geq 1 \\
& \|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2^2 \leq \|\boldsymbol{v}_i - \boldsymbol{v}_k\|_2^2 + \|\boldsymbol{v}_k - \boldsymbol{v}_j\|_2^2 && \forall\, i,j,k \in V \cup E \\
& \boldsymbol{d}_i^h = \|\boldsymbol{v}_i - \boldsymbol{v}_h\|_2^2 && \forall h \in E, i \in h \\
& \boldsymbol{v}_i \in \mathbb{R}^n, \boldsymbol{d}^h \in \mathbb{R}^h_{\geq 0} && \forall\, i \in V \cup E, \forall\, h \in E.
\end{aligned}
\tag{RC-Metric}
$$

We first prove that the above is indeed a valid relaxation.

Lemma 29. *The* (`RC-Metric`) *vector program is, up to a constant, a relaxation of the minimum ratio-cut program for the hypergraph $G = (V, E, \boldsymbol{w}, \boldsymbol{\mu})$ with symmetric polymatroidal cut functions $\{\delta_h\}_{h \in E}$.*

Proof. Given any cut $(S, \overline{S})$ with $\mu(S) \leq \mu(\overline{S})$, for each $i \in V$, set the vector embedding of

the vertices to

$$\boldsymbol{v}_i = \begin{cases} \left(\sqrt{\frac{\mu(V)}{\mu(S)\mu(\bar{S})}} \quad 0 \quad \dots \quad 0\right)^\top & \text{if } i \in S \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

And for each hyperedge $h \in E$:

$$\boldsymbol{v}_h = \begin{cases} \mathbf{0} & \text{if } \delta_h(S) = F_h(S), \\ \left(\sqrt{\frac{\mu(V)}{\mu(S)\mu(\overline{S})}}, 0, \cdots, 0\right)^\top & \text{if } \delta_h(S) = F_h(\overline{S}). \end{cases}$$

We then have:

$$\boldsymbol{d}_i^h = \|\boldsymbol{v}_i - \boldsymbol{v}_h\|^2 .$$

It is easy to see that this vector embedding satisfies the triangle inequality constraints. For the variance constraint, we have:

$$\sum_{\{i,j\} \subseteq V} \frac{\mu_i \mu_j}{\mu(V)} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|^2 = \sum_{\substack{i \in S \\ j \in \overline{S}}} \frac{\mu_i \mu_j}{\mu(V)} \cdot \frac{\mu(V)}{\mu(S) \cdot \mu(\overline{S})} = 1.$$

Furthermore, this solution has objective value:

$$\sum_{h \in E} w_h \bar{F}_h(\boldsymbol{d}^h) = \frac{\mu(V)}{\mu(S)\mu(\bar{S})} \cdot \sum_{h \in E} w_h \delta_h(S \cap h) \leq 2 \cdot \frac{\sum_{h \in E} w_h \delta_h(S \cap h)}{\min\{\mu(S), \mu(\bar{S})\}} = 2\Psi_G(S).$$

This shows that the combinatorial solution $(S, \overline{S})$ can be mapped to a feasible solution to the `RC-Metric` while preserving the objective to within a factor of two. □

Rounding via Metric Embedding To round the convex program `RC-Metric`, we apply the following well-known embedding result, which is implicit in the seminal work of Arora et al. (2009).

Theorem 30 (Arora et al. (2009))**.** *Let* $\{\boldsymbol{v}_i \in \mathbb{R}^d\}_{i \in V \cup E}$ *be a feasible solution to the* ℓ_2^2*-metric relaxation* `RC-Metric`*. Then, there exists an efficiently computable* 1*-Lipschitz*[1] *map* $\phi : \mathbb{R}^d \to \mathbb{R}$ *satisfying:*

$$\frac{\sum_{i,j \in V} \mu(i)\mu(j) \cdot |\phi(\boldsymbol{v}_i) - \phi(\boldsymbol{v}_j)|}{\sum_{i,j \in V} \mu(i)\mu(j) \cdot \|\boldsymbol{v}_i - \boldsymbol{v}_j\|^2} \geq \Omega\left(\frac{1}{\sqrt{\log |V|}}\right).$$

To complete the proof of Theorem 28 for the special case of symmetric polymatroidal cut functions, we make use of the embedding in Theorem 30. The proof crucially relies on the monotonicity of the Lovász extension $\bar{F}$ (Fact 23). The ability to perform this step can be taken as a justification for our definition of polymatroidal cut functions.

Lemma 31. *There is a polynomial-time algorithm that given a solution* $\{\boldsymbol{v}_i\}_{i \in V \cup E}$ *to* `RC-Metric` *of objective value* $\kappa := \sum_{h \in E} w_h \cdot \bar{F}_h(\boldsymbol{d}^h)$*, returns a cut* $S \subseteq V$ *such that:*

$$\Psi_G(S) \leq O\big(\kappa \cdot \sqrt{\log |V|}\big).$$

Proof. Apply Theorem 30 to obtain $\boldsymbol{x} \in \mathbb{R}^V$ and $\boldsymbol{\eta} \in R^E$ with $\boldsymbol{x}(i) \stackrel{\text{def}}{=} \phi(\boldsymbol{v}_i)$ for all $i \in V$ and $\eta(h) \stackrel{\text{def}}{=} \phi(\boldsymbol{v}_h)$. The 1-Lipschitzness of ϕ guarantees, for every $h \in E$ and $i \in h$:

$$|\boldsymbol{x}(i) - \eta(h)| = |\phi(\boldsymbol{v}_i) - \phi(\boldsymbol{v}_h)| \leq \|\boldsymbol{v}_i - \boldsymbol{v}_h\|^2 = \boldsymbol{d}_i^h.$$

By the monotonicity of $\bar{F}_h$ in Fact 23, we have:

$$\sum_{h \in E} w_h \cdot \min_{\nu_h \in \mathbb{R}} \bar{F}_h(|\boldsymbol{x}_h - \nu_h \mathbf{1}_h|) \leq \sum_{h \in E} w_h \cdot \bar{F}_h(|\boldsymbol{x}_h - \eta(h)\mathbf{1}_h|) \leq \sum_{h \in E} w_h \cdot \bar{F}_h(\boldsymbol{d}^h) = \kappa. \quad (3.6)$$

It now suffices to obtain a lower bound on the denominator in the continuous formula-

1. Recall that, given a metric space (X, d), a function $f : X \to \mathbb{R}$ is said to be K-Lipschitz for some constant K if $|f(x) - f(y)| \leq K \cdot d(x, y)$ for every $x, y \in X$. Here, the input metric space is $\mathbb{R}^d$ equipped with the ℓ_2^2-metric computed by the relaxation, while the output metric space is $\mathbb{R}$ with the ℓ_1 metric.

tion (RC-NonCvx):

$$\min_{\gamma\in\mathbb{R}} \|\boldsymbol{x}-\gamma\mathbf{1}\|_{1,\boldsymbol{\mu}} \geq \frac{1}{2}\sum_{i,j\in V}\frac{\boldsymbol{\mu}(i)\boldsymbol{\mu}(j)}{\boldsymbol{\mu}(V)}\cdot|\boldsymbol{x}_i-\boldsymbol{x}_j| \tag{3.7}$$

$$\geq \Omega\left(\frac{1}{\sqrt{\log|V|}}\right)\sum_{i,j\in V}\frac{\boldsymbol{\mu}(i)\boldsymbol{\mu}(j)}{\boldsymbol{\mu}(V)}\|\boldsymbol{v}_i-\boldsymbol{v}_j\|_2^2 \geq \Omega\left(\frac{1}{\sqrt{\log|V|}}\right). \tag{3.8}$$

Finally, Equations (3.6) and (3.7) yield, together with Fact 25:

$$\frac{\sum_{h\in E} w_h\bar{\delta}_h(\boldsymbol{x})}{\min_{\gamma\in\mathbb{R}}\|\boldsymbol{x}-\gamma\mathbf{1}\|_{1,\boldsymbol{\mu}}} \leq 2\cdot\frac{\sum_{h\in E} w_h \min_{\nu\in\mathbb{R}}\bar{F}_h(|\boldsymbol{x}-\nu 1|)}{\min_{\gamma\in\mathbb{R}}\|\boldsymbol{x}-\gamma\mathbf{1}\|_{1,\boldsymbol{\mu}}} \leq O(\kappa\cdot\sqrt{\log n}).$$

The vector $\boldsymbol{x}$ can then be efficiently rounded to a subset $S\subseteq V$ by Lemma 21. □

The General Case We now generalize the result to hypergraphs with arbitrary polymatroidal cut functions. For this case, our metric relaxation must be able to capture the signed terms $(\boldsymbol{x}_h-\nu_h\mathbf{1}_h)_+$ and $(\boldsymbol{x}_h-\nu_h\mathbf{1}_h)_-$ that appear in the Lovász extension in Fact 24. For this purpose, we will use the directed ℓ_2^2-semimetrics introduced by Charikar et al. (2006) and, for every $h\in E$ and $i\in h$, relax $(\boldsymbol{x}_i-\nu_h)_+$ to $\boldsymbol{d}_i^{h,-}\stackrel{\text{def}}{=}\|\boldsymbol{v}_i-\boldsymbol{v}_h\|^2+\|\boldsymbol{v}_i\|_2^2-\|\boldsymbol{v}_h\|_2^2$ and $(\boldsymbol{x}_i-\nu_h)_-$ to $\boldsymbol{d}_i^{h,+}\stackrel{\text{def}}{=}\|\boldsymbol{v}_i-\boldsymbol{v}_h\|^2+\|\boldsymbol{v}_h\|_2^2-\|\boldsymbol{v}_i\|_2^2$. We obtain the following semidefinite

program:

$$
\begin{aligned}
\min_{\boldsymbol{v}_i, \boldsymbol{\ell}^h} \quad & \sum_{h \in E} w_h \cdot \left(\bar{F}_h^-(\boldsymbol{d}_h^-) + \bar{F}_h^+(\boldsymbol{d}_h^+) \right) \\
\text{s.t.} \quad & \sum_{\{i,j\} \subseteq V} \frac{\mu(i)\mu(j)}{\mu(V)} \cdot \|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2^2 \geq 1 \\
& \|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2^2 \leq \|\boldsymbol{v}_i - \boldsymbol{v}_k\|_2^2 + \|\boldsymbol{v}_k - \boldsymbol{v}_j\|_2^2 \qquad \forall\, i,j,k \in V \cup E \\
& \boldsymbol{d}_i^{h,-} = \|\boldsymbol{v}_i - \boldsymbol{v}_h\|^2 + \|\boldsymbol{v}_i\|_2^2 - \|\boldsymbol{v}_h\|_2^2 \qquad \forall h \in E, i \in h \\
& \boldsymbol{d}_i^{h,+} = \|\boldsymbol{v}_i - \boldsymbol{v}_h\|^2 + \|\boldsymbol{v}_h\|_2^2 - \|\boldsymbol{v}_i\|_2^2 \qquad \forall h \in E, i \in h \\
& \boldsymbol{v}_i \in \mathbb{R}^n, \boldsymbol{d}^{h,+}, \boldsymbol{d}^{h,-} \in \mathbb{R}^h_{\geq 0} \qquad \forall\, i \in V \cup E, \forall\, h \in E
\end{aligned}
$$

(`RC-Directed-Semimetric`)

We have the following lemma.

Lemma 32. *The `RC-Directed-Semimetric` vector program is, up to a constant, a relaxation of the minimum submodular hypergraph ratio-cut problem for the hypergraph $G = (V, E, \boldsymbol{w}, \boldsymbol{\mu})$ with polymatroidal cut functions $\{\delta_h\}_{h \in E}$.*

Proof. Let $(S, \overline{S})$ be a candidate solution to the minimum hypergraph ratio-cut problem satisfying $\mu(S) \leq \mu(\overline{S})$, with objective value κ, i.e.:

$$\Psi_G(S) = \frac{\sum_{h \in E} w_h \delta_h(S \cap h)}{\mu(S)} = \kappa.$$

We can construct a solution to the `RC-Directed-Semimetric` program with a objective value 4κ as follows. Partition E into $E^+ \cup E^-$ where:

$$E^+ \stackrel{\text{def}}{=} \{h \in E \mid \delta_h(S) = F_h^+(\overline{S})\} \quad \text{and} \quad E^- \stackrel{\text{def}}{=} \{h \in E \mid \delta_h(S) = F_h^-(S)\}.$$

We can consider:

$$\boldsymbol{v}_i = \begin{cases} \left(\sqrt{\frac{\mu(V)}{\mu(S)\mu(\overline{S})}}, 0, \cdots, 0\right)^\top & \text{if } i \in S, \\ \mathbf{0} & \text{if } i \in \overline{S}, \end{cases}$$

and:

$$\boldsymbol{v}_h = \begin{cases} \mathbf{0} & \text{if } h \in E^-, \\ \left(\sqrt{\frac{\mu(V)}{\mu(S)\mu(\overline{S})}}, 0, \cdots, 0\right)^\top & \text{if } h \in E^+, \end{cases}$$

and for any $h \in E$ let $\boldsymbol{d}_h^+$ and $\boldsymbol{d}_h^-$ be defined in terms of $\{\boldsymbol{v}_i\}_{i \in V}$ and $\{\boldsymbol{v}_h\}_{h \in E}$ so as to satisfy the equality constraints in the `RC-Directed-Semimetric` program. One can check that:

$$\boldsymbol{d}_h^+(i) = \begin{cases} 2 \cdot \frac{\mu(V)}{\mu(S)\mu(\overline{S})} & \text{if } i \in \overline{S} \text{ and } h \in E^+, \\ 0 & \text{otherwise,} \end{cases}$$

and:

$$\boldsymbol{d}_h^-(i) = \begin{cases} 2 \cdot \frac{\mu(V)}{\mu(S)\mu(\overline{S})} & \text{if } i \in S \text{ and } h \in E^-, \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that all the triangle inequality constraints are satisfied. We can also verify that the variance constraint is satisfied:

$$\sum_{\{i,j\} \subseteq V} \frac{\mu(i)\mu(j)}{\mu(V)} \cdot \|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2^2 = \sum_{\substack{i \in S \\ j \in \overline{S}}} \frac{\mu(i)\mu(j)}{\mu(V)} \cdot \frac{\mu(V)}{\mu(S) \cdot \mu(\overline{S})} = 1.$$

Hence, the vectors defined above form a feasible solution to the `RC-Directed-Semimetric` program. The objective value of this solution is:

$$\sum_{h \in E^-} w_h \overline{F_h^-}(\boldsymbol{d}_h^-) + \sum_{h \in E^+} w_h \overline{F_h^+}(\boldsymbol{d}_h^+) = 2 \cdot \left(\sum_{h \in E^-} w_h \frac{\mu(V)}{\mu(S)\mu(\overline{S})} \overline{F_h^-}(\mathbf{1}_S) + \sum_{h \in E^+} w_h \frac{\mu(V)}{\mu(S)\mu(\overline{S})} \overline{F_h^+}(\mathbf{1}_{\overline{S}}) \right)$$

$$
\begin{aligned}
&= \frac{2\mu(V)}{\mu(S)\mu(\overline{S})} \cdot \left(\sum_{h \in E^-} w_h F_h^-(S \cap h) + \sum_{h \in E^+} w_h F_h^+(\overline{S} \cap h) \right) \\
&= \frac{2\mu(V)}{\mu(S)\mu(\overline{S})} \cdot \left(\sum_{h \in E} w_h \delta_h(S \cap h) \right) \\
&\leq 4 \cdot \frac{\sum_{h \in E} w_h \delta_h(S \cap h)}{\mu(S)} \\
&= 4\kappa,
\end{aligned}
$$

completing the proof. □

Rounding via Metric Embedding To round a solution of `RC-Directed-Semimetric` to a solution of the original ratio-cut problem `RC-NonCvx`, we require a version of the embedding result of Theorem 30 for directed semimetrics. The following result is a simple consequence of the rounding algorithm of Agarwal et al. (2005) for a similar relaxation of directed expansion. We provide a proof for completeness.

Theorem 33. *Given a feasible solution* $\{\boldsymbol{v}_i \in \mathbb{R}^d\}_{i \in V \cup E}$ *to* `RC-Directed-Semimetric` *and a measure* $\boldsymbol{\mu} \in \mathbb{R}^V_{\geq 0}$*, there exists a polynomial-time computable map* $\phi : \mathbb{R}^d \to \mathbb{R}$ *such that:*

1. $(\phi(\boldsymbol{v}) - \phi(\boldsymbol{w}))_+ \leq \|\boldsymbol{v} - \boldsymbol{w}\|^2 + \|\boldsymbol{v}\|^2 - \|\boldsymbol{w}\|^2$ *for all* $\boldsymbol{v}, \boldsymbol{w} \in \mathbb{R}^d$*,*
2. *and:*

$$
\frac{\sum_{i,j \in V} \mu(i)\mu(j) |\phi(\boldsymbol{v}_i) - \phi(\boldsymbol{v}_j)|}{\sum_{i,j \in V} \mu(i)\mu(j) \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2} \geq \Omega\left(\frac{1}{\sqrt{\log |V|}} \right).
$$

Proof. Since the relaxation is invariant to translations, we will assume, without loss of generality, that:

$$
\sum_{i \in V} \mu(i) \boldsymbol{v}_i = 0, \tag{3.10}
$$

and that:

$$
\sum_{i,j \in V} \frac{\mu(i)\mu(j)}{\mu(V)} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 = 1. \tag{3.11}
$$

We divide the proof in two cases corresponding to the case in which the embedding is balanced and unbalanced respectively. In Case 1, the embedding $\{\boldsymbol{v}_i\}_{i\in V}$ satisfies:

$$\sum_{\{i,j\}\in\binom{R_3}{2}} \frac{\mu(i)\mu(j)}{\mu(V)} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|^2 \geq 1/10,$$

where:

$$R_t \stackrel{\text{def}}{=} \left\{ i \in V \mid \|\boldsymbol{v}_i\|^2 \leq \frac{t}{\mu(V)} \right\}.$$

In this case, it is implicit in the results of Arora et al. (2009), that there exists polynomial-time computable sets $S, T \subseteq V$ satisfying $\mu(S), \mu(T) = \Omega(\mu(V))$, and:

$$\min_{\substack{i\in S\\ j\in T}} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 \geq \Omega\left(\frac{1}{\mu(V)\sqrt{\log n}}\right). \tag{3.12}$$

We construct the map ϕ explicitly as follows. Let S and T be the sets satisfying (3.12). Let $r \in \mathbb{R}$ be a value such that:

$$\mu(\{i \in S \mid \|\boldsymbol{v}_i\|_2 \leq r\}) = \mu(\{i \in S \mid \|\boldsymbol{v}_i\|_2 \geq r\}),$$

and define:

$$S^- \stackrel{\text{def}}{=} \{i \in S \mid \|\boldsymbol{v}_i\|_2 \leq r\}, \qquad S^+ \stackrel{\text{def}}{=} \{i \in S \mid \|\boldsymbol{v}_i\|_2 \leq r\},$$

and:

$$T^- \stackrel{\text{def}}{=} \{i \in S \mid \|\boldsymbol{v}_i\|_2 \leq r\}, \qquad T^+ \stackrel{\text{def}}{=} \{i \in S \mid \|\boldsymbol{v}_i\|_2 \leq r\}.$$

Suppose, without loss of generality, that $\mu(T^+) \geq \mu(T^-)$[2]. Note, in particular, that this

2. If this condition does not hold, the rest of the argument still holds by exchanging the roles of T^+ with T^- and S^+ with S^-.

implies that $\mu(S^-) \geq \mu(S)/2$ and $\mu(T^-) \geq \mu(T)/2$. Let:

$$\phi(\boldsymbol{v}_i) = \min_{j \in T^+} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 + \left\|\boldsymbol{v}_i\right\|_2^2 - \left\|\boldsymbol{v}_j\right\|_2^2.$$

As a simple consequence of the triangle inequality constraints, we have, for any $i, k \in V \cup E$:

$$\phi(\boldsymbol{v}_i) - \phi(\boldsymbol{v}_k) \leq \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 + \left\|\boldsymbol{v}_i\right\|_2^2 - \left\|\boldsymbol{v}_j\right\|_2^2.$$

Let $\boldsymbol{x} \in \mathbb{R}^V$ be the vector given by $\boldsymbol{x}(i) = \phi(\boldsymbol{v}_i)$. Then:

$$\begin{aligned}
\sum_{\{i,j\} \subseteq V} \mu(i)\mu(j)|\boldsymbol{x}(i) - \boldsymbol{x}(j)| &\geq \sum_{\{i,j\} \subseteq V} \mu(i)\mu(j)(\boldsymbol{x}(i) - \boldsymbol{x}(j))_+ \geq \sum_{\substack{i \in S^- \\ j \in T^+}} \mu(i)\mu(j)(\boldsymbol{x}(i) - \boldsymbol{x}(j))_+ \\
&= \sum_{\substack{i \in S^- \\ j \in T^+}} \mu(i)\mu(j)(\boldsymbol{x}(i))_+ = \mu(T^+) \sum_{i \in S^-} \mu(i)(\boldsymbol{x}(i))_+ \\
&= \mu(T^+) \sum_{i \in S^-} \mu(i) \left(\min_{j \in T^+} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 + \left\|\boldsymbol{v}_i\right\|_2^2 - \left\|\boldsymbol{v}_j\right\|_2^2 \right)_+ \\
&\geq \mu(T^+) \sum_{i \in S^-} \mu(i) \left(\min_{j \in T^+} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 + r^2 - r^2 \right)_+ \\
&\geq \mu(T^+) \sum_{i \in S^-} \mu(i) \left(\min_{j \in T^+} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 \right)_+ \\
&\geq \mu(T^+)\mu(S^-) \cdot \min_{\substack{i \in S^- \\ j \in T^+}} \left(\left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 \right)_+ \\
&= \mu(T^+)\mu(S^-) \cdot \min_{\substack{i \in S^- \\ j \in T^+}} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 \\
&\geq \mu(T^+)\mu(S^-) \cdot \min_{\substack{i \in S \\ j \in T}} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 \\
&\geq \frac{1}{4}\mu(T)\mu(S) \cdot \min_{\substack{i \in S \\ j \in T}} \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2 \\
&\geq \Omega(\mu(V)^2) \cdot \Omega\left(\frac{1}{\mu(V)\sqrt{\log n}} \right)
\end{aligned}$$

$$\geq \Omega\left(\frac{\mu(V)}{\sqrt{\log n}}\right).$$

This implies that:

$$\frac{\sum_{\{i,j\}\subseteq V}\mu(i)\mu(j)|\boldsymbol{x}(i)-\boldsymbol{x}(j)|}{\sum_{\{i,j\}\subseteq V}\mu(i)\mu(j)\left\|\boldsymbol{v}_i-\boldsymbol{v}_j\right\|_2^2}=\frac{1}{\mu(V)}\sum_{\{i,j\}\subseteq V}\mu(i)\mu(j)|\boldsymbol{x}(i)-\boldsymbol{x}(j)|\geq\Omega\left(\frac{1}{\sqrt{\log n}}\right).$$

Hence, the map ϕ satisfies conditions 1 and 2 in the statement of the lemma. Moreover, it is easy to see that this map can be computed in polynomial time. This completes the proof for Case 1.

For Case 2, we consider what happens when Case 1 doesn't hold, i.e.:

$$\sum_{\{i,j\}\in\binom{R_3}{2}}\frac{\mu(i)\mu(j)}{\mu(V)}\left\|\boldsymbol{v}_i-\boldsymbol{v}_j\right\|^2<1/10.$$

Here, we have:

$$\frac{\mu(R_{3/2})}{\mu(V)}\geq\frac{1}{3}\qquad\text{and}\qquad\sum_{i\in V\setminus R_3}\mu_i\cdot\|\boldsymbol{v}_i\|^2\geq\frac{3}{5}.$$

Here, the first inequality follows by (3.10), (3.11) and Markov's inequality. Let $T=R_{3/2}$ and $S=V\setminus R_3$. Consider the embedding:

$$\phi(\boldsymbol{v})=\min_{\boldsymbol{u}\in T}\|\boldsymbol{v}-\boldsymbol{u}\|_2^2+\|\boldsymbol{v}\|_2^2-\|\boldsymbol{u}\|_2^2.$$

Again, it is easy to see that the map ϕ satisfies condition 1 of the lemma. Moreover, note that, for any $\boldsymbol{v}_i\in S$:

$$|\phi(\boldsymbol{v}_i)|\geq|\,\|\boldsymbol{v}_i\|_2^2-\|\boldsymbol{u}\|_2^2\,|,$$

for some $\boldsymbol{u}\in T$ and hence:

$$|\phi(\boldsymbol{v}_i)|\geq|\,\|\boldsymbol{v}_i\|_2^2-\|\boldsymbol{u}\|_2^2\,|\geq\frac{1}{2}\|\boldsymbol{v}_i\|_2^2.$$

Let $\boldsymbol{x} \in \mathbb{R}^V$ be the vector given by $\boldsymbol{x}(i) = \phi(\boldsymbol{v}_i)$. We then have:

$$\begin{aligned}\frac{\sum_{\{i,j\}\subseteq V} \mu(i)\mu(j)|\boldsymbol{x}(i) - \boldsymbol{x}(j)|}{\sum_{\{i,j\}\subseteq V} \mu(i)\mu(j) \left\|\boldsymbol{v}_i - \boldsymbol{v}_j\right\|_2^2} &= \frac{1}{\mu(V)} \sum_{\{i,j\}\in\binom{V}{2}} \mu(i)\mu(j)|\boldsymbol{x}(i) - \boldsymbol{x}(j)| = \frac{\mu(T)}{\mu(V)} \sum_{i\in S} \mu(i)|\boldsymbol{x}(i)| \\ &\geq \frac{\mu(T)}{2\mu(V)} \sum_{i\in S} \mu_i \left\|\boldsymbol{v}_i\right\|_2^2 \\ &\geq \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{3}{5} = \Omega(1).\end{aligned}$$

Note that the bound obtained in this case is tighter, i.e. $\Omega(1/\sqrt{\log n})$ has been replaced by a constant.

□

We can now complete the proof of Theorem 28 for the general polymatroidal case by showing that the map ϕ in Theorem 33 yields a $O(\sqrt{\log n})$-approximate solution to the minimum ratio-cut problem.

Lemma 34. *Given a solution* $(\{\boldsymbol{v}_i\}_{i\in V\cup E})$ *to* `RC-Directed-Semimetric` *of value* κ*, we can produce in polynomial time a set* $S \subseteq V$ *such that:*

$$\Psi_G(S) \leq O(\kappa \cdot \sqrt{\log |V|}).$$

Proof. Let ϕ be the map whose existence is guaranteed by Theorem 33. Let $\boldsymbol{x} \in \mathbb{R}^V$ and $\boldsymbol{\eta} \in \mathbb{R}^E$ be the vector given by $\boldsymbol{x}_i = \phi(\boldsymbol{v}_i)$ for all $i \in V$ and $\eta_h = \phi(\boldsymbol{x}_h)$ for all $h \in E$. By the first condition in Theorem 33, we have, for all $h \in E$ and $i \in h$:

$$(\boldsymbol{x}_i - \eta_h)_+ \leq \boldsymbol{d}_i^{h,-} \text{ and } (\boldsymbol{x}_i - \eta_h)_- \leq \boldsymbol{d}_i^{h,+}.$$

We can now exploit the monotonicity of the functions F_h^- and F_h^+ associated with the

polymatroidal cut functions δ_h. By Fact 24 and Fact 23, we have:

$$\begin{aligned}\sum_{h\in E} w_h \bar{\delta}_h(\boldsymbol{x}) &= \sum_{h\in E} w_h \min_{\nu_h\in\mathbb{R}} \left\{\bar{F}_h^-((\boldsymbol{x}_h - \nu_h \mathbf{1}_h)_+) + \bar{F}_h^+((\boldsymbol{x}_h - \nu_h \mathbf{1}_h)_-)\right\} \\ &\leq \sum_{h\in E} w_h \left\{\bar{F}_h^-((\boldsymbol{x}_h - \eta_h \mathbf{1}_h)_+) + \bar{F}_h^+((\boldsymbol{x}_h - \eta_h \mathbf{1}_h)_-)\right\} \\ &\leq \sum_{h\in E} w_h \left(\bar{F}_h^-(\boldsymbol{d}^{h,-}) + \bar{F}_h^+(\boldsymbol{d}^{h,+})\right) = \kappa.\end{aligned}$$

We can now lower bound the denominator in the objective of `RC-NonCvx` as in Equation 3.7. Finally, we have:

$$\frac{\sum_{h\in E} w_h \bar{\delta}_h(\boldsymbol{x})}{\min_{\gamma\in\mathbb{R}} \|\boldsymbol{x} - \gamma\mathbf{1}\|_{1,\boldsymbol{\mu}}} \leq O(\kappa \cdot \sqrt{\log n}).$$

The vector $\boldsymbol{x}$ can then be efficiently rounded to a subset $S \subseteq V$ by Lemma 21. □

This concludes the proof of Theorem 28 for the general case of arbitrary polymatroidal cut functions.

3.5 Local Relaxations and Certificates

While the algorithm given in the previous section runs in polynomial time, this guarantee may be insufficient to run it in practice on large inputs. In this section we begin describing a framework which allows us to obtain a $O(\log n)$-approximation algorithm for the minimum ratio-cut problem in submodular hypergraphs equipped with a polymatroidal cut function, whose runtime is dominated by that of solving a decomposable submodular minimization problem over the cut functions used.

Towards this end, we begin by introducing a family of local convex approximations for this problem. Here, the key observation is that replacing the denominator in $\bar{\Psi}_G$ with a linear term gives us a convex upper bound to the problem. In particular, we can exploit the

dual characterization of the ℓ_1-norm in the lower bound to write:

$$\min_{u\in\mathbb{R}}\|\boldsymbol{x}-u\mathbf{1}\|_{\boldsymbol{\mu},1}=\min_{u\in\mathbb{R}}\max_{\|\boldsymbol{y}\|_\infty\le 1}\langle\boldsymbol{y},\boldsymbol{x}-u\mathbf{1}\rangle_{\boldsymbol{\mu}}=\max_{\substack{\|\boldsymbol{y}\|_\infty\le 1\\ \langle\boldsymbol{y},\mathbf{1}\rangle_{\boldsymbol{\mu}}=0}}\langle\boldsymbol{y},\boldsymbol{x}\rangle_{\boldsymbol{\mu}}. \tag{3.13}$$

This calculation directly leads us to define a novel *localized, convex formulation* of the (`RC-NonCvx`) problem for each vector $\boldsymbol{s}\in\mathbb{R}^V$ with $\|\boldsymbol{s}\|_\infty\le 1$ and $\langle\boldsymbol{s},\mathbf{1}\rangle_{\boldsymbol{\mu}}=0$, which we call a *seed*:

$$\begin{aligned}\bar{\Psi}_{G,\boldsymbol{s}}(\boldsymbol{x})&\stackrel{\text{def}}{=}\frac{\sum_{h\in E}w_h\bar{\delta}_h(\boldsymbol{x})}{\max\{0,\langle\boldsymbol{s},\boldsymbol{x}\rangle_{\boldsymbol{\mu}}\}}\\ \bar{\Psi}^*_{G,\boldsymbol{s}}&\stackrel{\text{def}}{=}\min_{\boldsymbol{x}\in\mathbb{R}^V}\bar{\Psi}_{G,\boldsymbol{s}}(\boldsymbol{x}).\end{aligned}\tag{Local-RC}$$

Intuitively, the program `Local-RC` seeks a distribution over cuts $\boldsymbol{x}$ with small expected cut size and high correlation with the seed $\boldsymbol{s}$.

`Local-RC` is equivalent to the following optimization problem:

$$\begin{aligned}\min_{\boldsymbol{x}\in\mathbb{R}^V}\quad&\sum_{h\in E}w_h\cdot\bar{\delta}_h(\boldsymbol{x})\\ \text{s.t.}\quad&\langle\boldsymbol{s},\boldsymbol{x}\rangle_\mu=1.\end{aligned}\tag{CI-Primal}$$

We can then derive the dual of this program:

$$\begin{aligned}\max\quad&\alpha\\ \text{s.t.}\quad&\operatorname{dem}(\mathbf{Y})=\alpha\cdot\big(\boldsymbol{\mu}\circ\boldsymbol{s}\big)\\ &\operatorname{cong}_G(\mathbf{Y})\le 1\\ &\alpha\in\mathbb{R},\mathbf{Y}=\{\boldsymbol{y}_h\in\mathbb{R}^h\perp\mathbf{1}_h\}_{h\in E}.\end{aligned}\tag{CI-Dual}$$

Every solution to the dual program provides a lower bound to the optimal value of `Local-RC`

and in turn this lower bounds the optimal value of `RC-NonCvx`. Indeed we have the following:

Lemma 35. *The programs `CI-Primal` and `CI-Dual` are dual to each other, and strong duality holds.*

Proof. Let us proceed with taking the conjugate dual. Starting with `CI-Primal`:

$$\begin{aligned}\min_{\substack{\boldsymbol{x}\in\mathbb{R}^V\\ \langle \boldsymbol{s},\boldsymbol{x}\rangle_\mu=1}} \sum_{h\in E} w_h\cdot\bar{\delta}_h(\boldsymbol{x}) &= \min_{\boldsymbol{x}\in\mathbb{R}^V} \sum_{h\in E} w_h\cdot\bar{\delta}_h(\boldsymbol{x}) + \max_{\alpha\in\mathbb{R}}\Big(1-\langle \boldsymbol{s},\boldsymbol{x}\rangle_\mu\Big)\\ &= \min_{\boldsymbol{x}\in\mathbb{R}^V}\max_{\alpha\in\mathbb{R}}\left\{\alpha+\sum_{h\in E} w_h\cdot\bar{\delta}_h(\boldsymbol{x}) - \big\langle \alpha\big(\mu\circ\boldsymbol{s}\big),\boldsymbol{x}\big\rangle\right\}.\end{aligned}$$

Recall that the Lovász extension is given by

$$\bar{\delta}_h(\boldsymbol{x}) = \max_{\boldsymbol{f}\in\mathcal{B}(\delta_h)} \langle \boldsymbol{f},\boldsymbol{x}\rangle\,,$$

and hence:

$$\begin{aligned}&\min_{\boldsymbol{x}\in\mathbb{R}^V}\max_{\alpha\in\mathbb{R}}\left\{\alpha+\sum_{h\in E} w_h\cdot\bar{\delta}_h(\boldsymbol{x}) - \big\langle \alpha\big(\mu\circ\boldsymbol{s}\big),\boldsymbol{x}\big\rangle\right\}\\ =\ &\min_{\boldsymbol{x}\in\mathbb{R}^V}\max_{\alpha\in\mathbb{R}}\left\{\alpha+\sum_{h\in E}\max_{\boldsymbol{f}_h\in w_h\cdot\mathcal{B}(\delta_h)}\langle \boldsymbol{f}_h,\boldsymbol{x}\rangle - \big\langle \alpha\big(\mu\circ\boldsymbol{s}\big),\boldsymbol{x}\big\rangle\right\}\\ =\ &\min_{\boldsymbol{x}\in\mathbb{R}^V}\max_{\substack{\alpha\in\mathbb{R},\boldsymbol{f}_h\in\mathbb{R}^V\\ \boldsymbol{f}_h\in w_h\cdot\mathcal{B}(\delta_h)}}\left\{\alpha+\sum_{h\in E}\langle \boldsymbol{f}_h,\boldsymbol{x}\rangle - \big\langle \alpha\big(\mu\circ\boldsymbol{s}\big),\boldsymbol{x}\big\rangle\right\}\\ =\ &\min_{\boldsymbol{x}\in\mathbb{R}^V}\max_{\substack{\alpha\in\mathbb{R},\boldsymbol{f}_h\in\mathbb{R}^V\\ \boldsymbol{f}_h\in w_h\cdot\mathcal{B}(\delta_h)}}\left\{\alpha+\left\langle \sum_{h\in E}\boldsymbol{f}_h - \alpha\big(\mu\circ\boldsymbol{s}\big),\,\boldsymbol{x}\right\rangle\right\}.\end{aligned}$$

Strong duality always holds when the feasible region is polyhedron. Hence, we have:

$$\min_{\boldsymbol{x}\in\mathbb{R}^V}\max_{\substack{\alpha\in\mathbb{R},\boldsymbol{f}_h\in\mathbb{R}^V\\ \boldsymbol{f}_h\in w_h\cdot\mathcal{B}(\delta_h)}}\left\{\alpha+\left\langle \sum_{h\in E}\boldsymbol{f}_h-\alpha\big(\boldsymbol{\mu}\circ\boldsymbol{s}\big),\,\boldsymbol{x}\right\rangle\right\} = \max_{\substack{\alpha\in\mathbb{R},\boldsymbol{f}_h\in\mathbb{R}^V\\ \boldsymbol{f}_h\in w_h\cdot\mathcal{B}(\delta_h)}}\alpha+\min_{\boldsymbol{x}\in\mathbb{R}^V}\left\langle \sum_{h\in E}\boldsymbol{f}_h-\alpha\big(\boldsymbol{\mu}\circ\boldsymbol{s}\big),\,\boldsymbol{x}\right\rangle,$$

which is equivalent to the required linear program by the definition of congestion (Equation 3.3) and demand (Equation 3.2) for hypergraph flows. □

We will be primarily interested in seeds of the form:

$$\mathbf{1}^{A,B} = \mathbf{1}_A - \frac{\mu(A)}{\mu(B)}\mathbf{1}_B,$$

where $A, B \subseteq V$ are disjoint subsets with $\mu(A) \leq \mu(B)$.

Note that $\mathbf{1}^{A,B}$ is a valid seed for the local formulation of the minimum ratio-cut problem as $\|\mathbf{1}^{A,B}\|_\infty \leq 1$ and $\langle \mathbf{1}^{A,B}, \mathbf{1} \rangle_{\boldsymbol{\mu}} = 0$.

For this kind of seed, the dual problem can be interpreted as a maximum concurrent single-commodity flow over the hypergraph G, in which each vertex $i \in A$ is asked to route demand μ_i to B and each vertex j in B is asked to receive demand $\mu(A)/\mu(B) \cdot \mu_j$ from A. For a solution of value α, each vertex will be able to route an α fraction of its demand and the total demand routed from A to B will be $\alpha\mu(A)$.

We will be interested in making use of fast approximate solvers for the above optimization problem. We define the following:

Definition 10 (Approximate dual solution)**.** Let $G = \left(V, E_G, \boldsymbol{w}^G, \boldsymbol{\mu}\right)$ be a weighted submodular hypergraph equipped with polymatroidal cut functions $\{\delta_h\}_{h \in E}$, and $A, B \subseteq V$ disjoint sets with $\mu(A) \leq \mu(B)$. An *approximate dual solution of value* $\alpha \geq 0$ for $(G, \mathbf{1}^{A,B})$ is a hypergraph flow $\mathbf{Y} = \left(\boldsymbol{y}^h \in \mathbb{R}^h \perp \mathbf{1}_h\right)$ such that the following hold simultaneously:

1. $\mathbf{Y}$ has unit congestion, i.e., $\operatorname{cong}_G(\mathbf{Y}) \leq 1$,
2. $\mathbf{Y}$ routes a $1/2$-fraction of the required flow from A to B, i.e., at least $1/2 \cdot \alpha \cdot \mu(A)$ flow,
3. $\mathbf{Y}$ does not exceed the require demand at every vertex, i.e., $|\operatorname{dem}(\mathbf{Y})| \leq \alpha \cdot |\boldsymbol{\mu} \circ \mathbf{1}^{A,B}|$.

Definition 11. Let $G = \left(V, E_G, \boldsymbol{w}^G, \boldsymbol{\mu}\right)$ be a weighted submodular hypergraph equipped with polymatroidal cut functions $\{\delta_h\}_{h \in E}$, and $A, B \subseteq V$ disjoint sets with $\mu(A) \leq \mu(B)$.

Let $\alpha = \Psi^*_{G,\mathbf{1}^{A,B}}$. An *approximate dual graph certificate of value* $\alpha \geq 0$ for the `Local-RC` problem with seed $\mathbf{1}^{A,B}$ is a directed graph $D = (V, E_D, \boldsymbol{w}^D)$ with the following properties:

1. D is sparse, i.e., $|E_D| \leq \tilde{O}(\mathsf{sp}(G))$,
2. $\alpha \cdot D$ embeds into G with congestion 1, or equivalently $D \preceq_{1/\alpha} G$,
3. D is bipartite from A to B, i.e. all arcs of D go from A to B,
4. for all $i \in A, j \in B$, we have the bounds $\deg_D(i) \leq \mu_i$ and $\deg_D(j) \leq \mu(A)/\mu(B) \cdot \mu_j$,
5. D is large, i.e., $w^D(E(A,B)) \geq 1/2 \cdot \mu(A)$.

Following Theorem 26, we may assume that D is undirected if all cut functions are symmetric.

We note the following result:

Lemma 36. *Under the same assumptions of Definition 11, consider an approximate dual solution* $\mathbf{Y}$ *of value* α *to the ratio-cut improvement problem on* $(G, \mathbf{1}^{A,B})$. *Let* H *be the directed graph obtained by applying the flow-path decomposition algorithm of Theorem 27 to the hypergraph flow* $\mathbf{Y}$. *Then, the scaled graph* $\frac{1}{\alpha} \cdot H$ *is an approximate dual graph certificate of value* α.

Proof. Let $D = \frac{1}{\alpha} \cdot H$. The sparsity of D also follows directly from Theorem 27. By the same theorem, we have $H \preceq_1 G$, as $\mathrm{cong}_G(\mathbf{Y}) \leq 1$ by Property 1 of approximate dual solutions. Hence, the scaled graph D embeds into G with congestion $\frac{1}{\alpha}$, as required. Taking into account the same scaling by $\frac{1}{\alpha}$, the statements on the bipartiteness and degree bounds of D follow directly from Property 2 and 3 of approximate dual solutions via Theorem 27. Similarly, the largeness of D follows from Property 2. □

A key ingredient for the design of algorithms for the hypergraph ratio cut problem is the existence of algorithms for the following problem.

Definition 12. Given a weighted submodular hypergraph $G = (V, E, \boldsymbol{w}, \boldsymbol{\mu})$ with polymatroidal cut functions $\{\delta_h\}_{h \in E}$ and disjoint sets $A, B \subset V$ with $\mu(A) \leq \mu(B)$, an *approximate primal-dual oracle* problem is an algorithm $\mathcal{A}_{\mathtt{CI}}$ that takes as input G and the seed vector $\mathbf{1}^{A,B}$. For some $\alpha \geq 0$, the algorithm $\mathcal{A}_{\mathtt{CI}}$ outputs both:

1. a cut $S \subseteq V$ with $\Psi_{G,\mathbf{1}^{A,B}}(S) \leq 3\alpha$,

2. an approximate dual graph certificate of value α.

In the paper of Chen et al. (2023a), the authors argue the following two theorems. The first gives guarantees for the general case of polymatrodial cut functions, while the second shows that result can be significantly improved for the case of the standard cut functions. The proof of the latter makes use of almost-linear-time maximum flow solvers (see Bernstein et al. (2022); Chen et al. (2022)).

Theorem 37. *For submodular hypergraphs with general polymatroidal cut functions, an approximate primal-dual oracle can be implemented in time* $\tilde{O}\big(|V| \cdot \sum_{h \in E} \Theta_h\big)$, *where* Θ_h *is the running time for a quadratic optimization oracle for* δ_h, *by solving* $O(\log |V|)$*-many decomposable submodular minimization problems.*

Theorem 38. *For a submodular hypergraph G equipped with the directed or standard hypergraph cut functions, an approximate primal-dual oracle can be implemented in almost linear-time by solving* $O(\log |V|)$*-many* $1/2$*-approximate maximum flow problems over a directed, vertex-capacitated version of the factor graph* $\hat{G}$.

3.6 Approximation Algorithms from a New Cut-Matching Game

In this section, we describe a cut-matching game for hypergraphs and show how good strategies for the cut player yield efficient approximation algorithms for the minimum hypergraph ratio cut problem.

As the name suggests, the game is played between two players, a cut player, and a matching player. We first define what constitutes admissible actions played by the cut and matching players.

Definition 13. Let V be a set of vertices with vertex weights $\boldsymbol{\mu} \in \mathbb{Z}^V_{\geq 0}$. A *cut action* is a pair (A, B) of non-empty disjoint subsets $A, B \subseteq V$ such that $\boldsymbol{\mu}(A) \leq \boldsymbol{\mu}(B)$.

Note that, contrarily to most other settings that make use of similar cut-matching games, here the cut player is not restricted to returning a partition of the vertex set, let alone a bisection. The set of valid actions for a matching player is similarly relaxed, as the edges of the response do not need to form a perfect matching. It is only required that enough of the available degree is routed across the cut action.

Definition 14. An *approximate matching response* to a cut action (A, B) is a weighted bipartite graph $D = \big(A \cup B, E_D, \boldsymbol{w}^D, \boldsymbol{\mu}\big)$ satisfying the following conditions.

1. Bounded degree: for all $i \in V$

$$\deg_D(i) \leq \begin{cases} \mu_i & i \in A, \\ \frac{\boldsymbol{\mu}(A)}{\boldsymbol{\mu}(B)} \cdot \mu_i & i \in B. \end{cases}.$$

2. Largeness: $\boldsymbol{w}^D\big(E(A, B)\big) \geq \frac{\boldsymbol{\mu}(A)}{2}$.

We may say D is a *directed approximate matching response* if edges of D are directed from A to B.

This definition is motivated by how an approximate matching response naturally arises from the dual graph certificate obtained by approximately solving the `Local-RC` program above. The matching response will be undirected when we wish to approximate the minimum ratio-cut of a hypergraph with symmetric cut functions and directed when considering asymmetric cut functions. We now define the cut-matching game.

Definition 15. Let $n > 0$, and $\boldsymbol{\mu} \in \mathbb{Z}^V_{\geq 0}$ be a weighting over a set of vertices V where $|V| = n$. A $(n, \boldsymbol{\mu})$-*generalized cut-matching game* is a multi-round, two-player game between a cut player $\mathcal{C}$, and a matching player $\mathcal{M}$ that proceeds as follows. In the t-th round of the game, following definitions 13 and 14:

1. $\mathcal{C}$ plays a cut action (A_t, B_t),

2. $\mathcal{M}$ responds with an approximate matching response D_t to (A_t, B_t).

The *state graph* after t rounds is $H_t = \sum_{s=1}^{t} D_s$ the edge-wise sum of the matching response seen thus far.

For comparison, the original cut-matching game of Khandekar et al. (2009) is captured by the above definition when one specifically considers graph expansion $\boldsymbol{\mu} = \mathbf{1}$, restricts cut actions to be exact bisections $(A, \bar{A})$ where $|A| = |\bar{A}|$, and requires exact matching responses: $\boldsymbol{w}^D\big(E(A, \bar{A})\big) = \boldsymbol{\mu}(A)$.

To approximate minimum ratio-cut using our cut-matching games, we seek a cut player $\mathcal{C}$ that outputs a sequence of cuts which, in as few cuts as possible, forces the matching player $\mathcal{M}$ to place edges that make the minimum ratio-cut objective for the state graph H_T large. We call a cut player *good* if it can achieve this.

Definition 16. A *cut strategy* is a randomized algorithm $\mathcal{A}_{\mathsf{cut}}$ that takes as input the current state graph H, and outputs a cut action (A, B). A cut strategy $\mathcal{A}_{\mathsf{cut}}$ is $\big(f(n), g(n)\big)$-*good* if a cut player $\mathcal{C}$ using $\mathcal{A}_{\mathsf{cut}}$ at every iteration, achieves:

$$\Psi_{H_{g(n)}} \geq f(n)$$

with constant probability, for any (potentially adaptive) sequence of approximate matching responses $D_1, \ldots, D_{g(n)}$.

The key result of this section is then the following.

Theorem 39. *Let $G = (V, E, \boldsymbol{w}, \boldsymbol{\mu})$ be an input submodular hypergraph equipped with polymatroidal cut functions. Consider an execution of the cut-matching game in which the cut player $\mathcal{C}$ plays an $\big(f(n), g(n)\big)$-good cut strategy, while the matching player plays the dual graph certificates output by an approximate primal-dual oracle $\mathcal{A}_{\mathtt{CI}}$ (Definition 12) on G. Let C_t be the cuts output by $\mathcal{A}_{\mathtt{CI}}$. Then, with constant probability, we have the following approximation result:*

$$\min\left\{\Psi_G(C_t)\right\}_{t=1}^{g(n)} \leq O\left(\frac{g(n)}{f(n)}\right)\cdot\Psi_G^*.$$

Proof. First, we check that the dual graph certificates D_t output by $\mathcal{A}_{ci}$ conform to the definition of an approximate matching response. Then, the bipartiteness of D_t across (A_t, B_t) follows from requirement 3 in Definition 11, while the degree and largeness properties follow from requirements 4 and 5 in the same definition. In the symmetric case, we may assume each D_t is undirected.

In the following, let α_t the value of α achieved by the t calls to $\mathcal{A}_{ci}$ and

$$\alpha^* = \min\{\alpha_t\}_{t=1}^{g(n)}.$$

By the definition of approximate primal-dual oracle, we have that

$$\min\left\{\Psi_G(C_t)\right\}_{t=1}^{g(n)} \leq \min\left\{\Psi_{G,\mathbf{1}^{A_t,B_t}}(C_t)\right\}_{t=1}^{g(n)} \leq 3\cdot\min\left\{\alpha_t\right\}_{t=1}^{g(n)} = 3\cdot\alpha^*.$$

By Property 1 in Definition 11, we have that $\alpha_t D_t$ embeds in G with congestion 1 for all t. Averaging this guarantee over all t, we have:

$$\frac{\alpha^*}{g(n)}\cdot H_{g(n)} \preceq_1 \frac{1}{g(n)}\cdot\sum_{t=1}^{g(n)}\alpha_t D_t \preceq_1 G$$

Applying Theorem 26 and the definition of good strategy, we obtain the following lower

bound on cuts of G, for all $S \subseteq V$:

$$\alpha^* \cdot \frac{f(n)}{g(n)} \leq \frac{\alpha^*}{g(n)} \cdot \Psi_{H_{g(n)}}(S) \leq 2 \cdot \Psi_G(S).$$

In particular, this applies to the minimum-ratio cut in G, so that we have:

$$\min \left\{ \Psi_G(C_t) \right\}_{t=1}^{g(n)} \leq 3 \cdot \alpha^* \leq 6 \cdot \frac{g(n)}{f(n)} \cdot \Psi_G^*.$$

This completes the proof. □

In a paper of Chen et al. (2023a), the authors prove the following:

Theorem 40. *Let $H = \big(V, E_H, \boldsymbol{w}^H, \boldsymbol{\mu}\big)$ be the state graph for an $(n, \boldsymbol{\mu})$-generalized cut-matching game. There exists a cut strategy $\mathcal{A}_{\mathsf{cut}}$ satisfying the following:*

1. *If H is undirected, then $\mathcal{A}_{\mathsf{cut}}$ is $\big(\Omega(\log n), O(\log^2 n)\big)$-good with probability $O(1)$.*
2. *If H is directed, then $\mathcal{A}_{\mathsf{cut}}$ is $\big(\Omega(\log^2 n), O(\log^3 n)\big)$-good with probability $O(1)$.*

Furthermore, $\mathcal{A}_{\mathsf{cut}}$ outputs a cut action in time $\tilde{O}\big(\mathsf{sp}(H)\big)$.

This immediately yields the following corollaries, which are the core results of this chapter:

Corollary 41. *There exists an $O(\log n)$-approximation algorithm for the minimum ratio-cut problem over submodular hypergraphs with polymatroidal cut functions whose running time is dominated by the solution of a polylogarithmic number of decomposable submodular minimization problems over the hypergraph cut function.*

Corollary 42. *There exists an $O(\log n)$-approximation algorithm for the minimum ratio-cut problem over submodular hypergraphs equipped with the directed or standard hypergraph cut function whose running time is almost linear in the hypergraph sparsity.*

3.7 Bibliographical Notes

The work in this chapter largely follows the paper by Chen et al. (2023a). Similar results appeared in concurrent work by Veldt (2023).

CHAPTER 4

LEARNING PARTITIONS WITH SAME-CLUSTER ORACLE

In this chapter, we shift gears, away from the perspective of approximation algorithms and towards an interactive setting. We consider the following problem. Suppose a learner is given a finite set of elements V, and that V is partitioned into k clusters according to some unknown partition $\mathcal{C}$. The learner is interested in recovering $\mathcal{C}$, and can obtain information about it by asking questions of the form: "Are elements u and v of V part of the same cluster in $\mathcal{C}$?".

This setup model at least two distinct applications. First, in several scientific domains, particularly bioinformatics, researchers conduct physical experiments to learn whether two objects are part of the same class (Balding et al. (1996); Bouvel et al. (2005); Grebinski and Kucherov (1998); Sorokin et al. (1996)). A second application is learning cluster structure by collecting information from human workers via crowdsourcing services (Chen et al. (2023b); Mazumdar and Saha (2017); Vinayak and Hassibi (2016)). While the classical setup focuses on querying workers for class labels, alternative approaches use simpler *same-cluster* queries. For example, Vinayak and Hassibi (2016) point out that, given images of birds, it may be easier for non-experts to correctly answer the question "Do these two birds belong to the same species?" as opposed to "What species does this bird belong to?" In both of these application domains, the learner typically seeks to minimize the number of queries made, as queries require carrying out potentially expensive measurements or time-consuming experiments.

Several variants of this problem are known. In particular, one key distinguishing factor among the different instantiations of this question lies in whether the answers to the queries are guaranteed to be correct or not. We will consider algorithms that are guaranteed to work only when they are guaranteed to always receive the correct answer, as well as algorithms that are robust to a fixed number of errors. We describe the different error models in Section 4.1.

A central question will be the following: how many same-cluster queries does an algorithm

need to make to guarantee full recovery of the hidden partition when up to ℓ of the answers received may be incorrect? We refer to the problem of recovering the hidden partition in this setting as the *ℓ-bounded error partition learning* (ℓ-PL) problem.

We will primarily be interested in the *query complexity* of the problem, defined as the smallest, worst-case upper bound on the number of queries that any algorithm needs to make to guarantee full recovery of the partition. This will be given in terms of the parameters n, k and possibly ℓ of the problem. Note that the query complexity will often depend on k, even when k is unknown to the learner.

4.1 A Family of Problems

In the course of this chapter we will discuss different versions of the problem of learning partitions with a same-cluster oracle. In particular, in this section we introduce a hierarchy of problems with different degrees of difficulty (see Figure 4.1) based on different error models. In the following sections we will establish upper and lower bounds for these tasks.

At the bottom of the hierarchy, we have the simplest version of the problem, in which the answer to every same-cluster query is guaranteed to be correct.

Definition 17 (Same-cluster Oracle, Learning Partitions Problem)**.** Given a finite set V and a partition $\mathcal{C}$ of V, a same-cluster oracle for $\mathcal{C}$ is an algorithm $\alpha_{\mathcal{C}}$ which on input a pair uv of two elements of V, returns $+1$ if $u \sim_{\mathcal{C}} v$ and -1 otherwise. The (error-free) *learning partition problem* is the problem of learning $\mathcal{C}$ exclusively by querying a same-cluster oracle $\alpha_{\mathcal{C}}$.

Results about this variant are discussed in Section 4.2. There, we will describe optimal algorithms with matching lower bounds, which will allow us to determine the query complexity of the problem down to the exact constant. Since this is the "easiest" version of the problem we will be considering, the lower bounds we discuss for this version immediately imply the same lower bounds for all the other problems in the family.

We then consider two more variants, each only admitting one-sided error. In the first variant, which we refer to as the *ℓ-bounded error partition learning with false positives* (ℓ-PL$^{\mathtt{FP}}$) problem, the oracle might return the answer $+1$ even when two elements are not part of the same cluster. We refer to this kind of fault as a *false positive* answer. Here, we restrict the oracle to return at most ℓ many false positive answers, for some fixed positive integer ℓ that is assumed to be known to the learner, and to always return -1 when the two elements being queried are not part of the same cluster.

In the second variant, which we refer to as the ℓ-PL$^{\mathtt{FN}}$ problem, the oracle behaves the opposite way, and it may return -1 even if the two elements being queried are part of the

same cluster in the hidden partition (a *false negative answer*), but it cannot return a false positive answer. Similarly to the previous variant, we restrict the number of false negative answers to be at most some fixed ℓ known to the learner. We give precise definitions of the ℓ-PL$^{\mathtt{FP}}$ and ℓ-PL$^{\mathtt{FN}}$ problems below.

The ℓ-PL problem introduced at the beginning of the chapter is then a generalization of all of the three variants described above (false positives, false negatives and no error), and hence, it inherits hardness results from all three of these problems. Instead of directly designing an algorithm for the ℓ-PL problem, we consider a yet more general version of the problem, and use results from that generalized version to obtain algorithms for all of the other problems in the class. This final variant is a more fine-grained problem, which allows false positive answers and false negative answers to incur different penalties. In particular we consider a weighted version of ℓ-PL, which is formalized in the following definition:

Definition 18 ($(\ell_{\mathtt{yes}}, \ell_{\mathtt{no}})$-Faulty Oracle and Weighted ℓ-PL Problem)**.** Let V be a finite set, and $\mathcal{C}$ be a partition of V. A $(\ell_{\mathtt{yes}}, \ell_{\mathtt{no}})$-*faulty oracle* for $\mathcal{C}$ is an algorithm $\alpha_{\mathcal{C}}$ which, given as input a pair of elements uv, returns a value $r =\in \{\pm 1\}$ so that for any sequence of queries $\{u_t v_t\}_{t\in[T]}$ the sequence of responses $\{r_t = \alpha_{\mathcal{C}}(u_t v_t)\}_{t\in[T]}$ satisfies:

$$\frac{|\{t \in [T] \mid u_t \not\sim_{\mathcal{C}} v_t \wedge r_t = +1\}|}{\ell_{\mathtt{yes}}} + \frac{|\{t \in [T] \mid u_t \sim_{\mathcal{C}} v_t \wedge r_t = -1\}|}{\ell_{\mathtt{no}}} \leq 1.$$

Note that we assume the oracle can maintain an internal state, so its answers may depend on the query history. The *weighted* ℓ-PL problem asks one to recover an unknown k-partition of V given access to an $(\ell_{\mathtt{yes}}, \ell_{\mathtt{no}})$-faulty oracle for $\mathcal{C}$.

Observe that by setting $\ell_{\mathtt{yes}} < 1$ (resp. $\ell_{\mathtt{no}} < 1$) one would require the oracle to not return any false positive (resp. false negative) answer. As a convention, we will write $\ell_{\mathtt{yes}} = 0$ or $\ell_{\mathtt{no}} = 0$ to indicate any setting that ensures the oracle may not return any false positive or false negative answers respectively. We note that, up to rescaling $\ell_{\mathtt{yes}}$ and $\ell_{\mathtt{no}}$ by a factor

of 2, this problem is equivalent to a version of the problem in which the oracle has separate constraints for the number of false positive answers and false negative answers, a simple fact which we now formalize.

Definition 19 (Asymmetric ℓ-PL Problem)**.** Given a finite set V, and a partition $\mathcal{C}$ of V, an $(\ell_{\texttt{yes}}, \ell_{\texttt{no}})$-asymmetric oracle for $\mathcal{C}$ is an algorithm $\alpha_{\mathcal{C}}$ which, given as input a pair of elements uv, returns a value $r =\in \{\pm 1\}$ so that for any sequence of queries $\{(u_t, v_t)\}_{t\in[T]}$ the sequence of responses $\{r_t = \alpha_{\mathcal{C}}(u_t v_t)\}_{t\in[T]}$ satisfies::

$$|\{t \in [T] \mid r_t = 1 \wedge u_t \not\sim_{\mathcal{C}} v_t\}| \leq \ell_{\texttt{yes}} \quad \text{and} \quad |\{t \in [T] \mid r_t = -1 \wedge u_t \sim_{\mathcal{C}} v_t\}| \leq \ell_{\texttt{no}}.$$

The asymmetric ℓ-PL problem is the problem of recovering $\mathcal{C}$ by making queries to an $(\ell_{\texttt{yes}}, \ell_{\texttt{no}})$-asymmetric oracle.

It is easy to see that any $(\ell_{\texttt{yes}}, \ell_{\texttt{no}})$-faulty oracle is also an $(\ell_{\texttt{yes}}, \ell_{\texttt{no}})$-asymmetric oracle and hence the query complexity of the asymmetric ℓ-PL problem is no larger than that of the weighted ℓ-PL problem for the same choice of parameters. Conversely, we have the following.

Proposition 43. *For any algorithm which solves the weighted ℓ-PL problem by making at most $q(n, k, \ell_{\texttt{yes}}, \ell_{\texttt{no}})$ queries, there exists an algorithm which solves the asymmetric ℓ-PL problem by making at most $q(n, k, 2\ell_{\texttt{yes}}, 2\ell_{\texttt{no}})$ queries.*

Proof. This simply follows from the fact that any $(\ell_{\texttt{yes}}, \ell_{\texttt{no}})$-asymmetric oracle is also an $(\ell_{\texttt{yes}}/2, \ell_{\texttt{no}}/2)$-faulty oracle. □

Given the definition of the weighted ℓ-PL problem above, the (unweighted) ℓ-PL can be cast as a homogeneous version, one in which $\ell_{\texttt{yes}} = \ell_{\texttt{no}}$, as follows:

Definition 20 (ℓ-Faulty Oracle and ℓ-PL Problem)**.** An ℓ-faulty oracle is an (ℓ, ℓ)-faulty oracle. The ℓ-PL problem is the problem of learning a partition with access to an ℓ-faulty oracle.

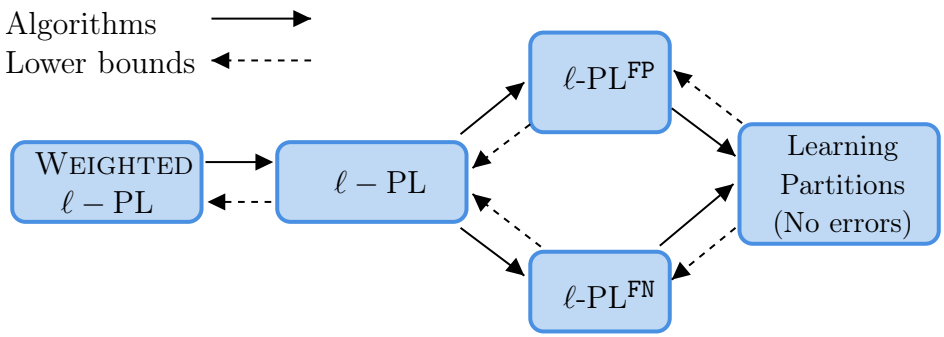

Figure 4.1: The lattice of problems considered in this chapter. Algorithmic results propagate from left to right while lower bound results propagate right to left.

The ℓ-PL$^{\mathtt{FP}}$ and ℓ-PL$^{\mathtt{FN}}$ problems discussed above can now be formally described as special instances of the weighted ℓ-PL problem.

Definition 21 (ℓ-PL$^{\mathtt{FP}}$ Problem and ℓ-PL$^{\mathtt{FN}}$ Problem)**.** The ℓ-PL$^{\mathtt{FP}}$ problem is the problem of learning a partition with access to an $(\ell, 0)$-faulty oracle. Intuitively, this corresponds to the problem of learning partitions subject to at most ℓ false-positive responses and no false negatives. The ℓ-PL$^{\mathtt{FN}}$ problem is the problem of learning a partition with access to an $(0, \ell)$-faulty oracle. This corresponds to learning partitions subject to at most ℓ false-negatives responses and no false positives.

In the rest of this chapter, we will always assume that the set V, its cardinality n, and the parameters $(\ell_{\mathtt{yes}}, \ell_{\mathtt{no}})$ are known to the learner. We will consider both the *k-known* setting, in which the number k of clusters is known to learner, and the *k-unknown* setting, in which it is not. It is easy to see that the query complexity of every problem in the k-unknown setting is no smaller than that of the same problem in the k-known setting, since any learning algorithm could simply ignore the value of k. We design separate algorithms for the k-known and the k-unknown setting, while the lower bounds, which we only prove for the k-known setting, apply directly to the k-unknown setting.

4.2 Deterministic Query Complexity in the Absence of Errors

We now describe an algorithm for the (error-free) Learning Partition Problem. We give two versions of the algorithm: one for the k-known setting, and one for the k-unknown setting. The two variants are very similar and roughly work as follows: the learner picks arbitrary element $v \in V$ and sets v as the key representative of its own cluster. They then proceed to pick a new unassigned element u and compare it to v by querying the oracle. If the oracle returns $+1$ then they assign u to the same cluster as v otherwise they make it the key representative of a new cluster. The algorithm then loops: at each iteration, a new unassigned u is chosen, and it is then compared to all the key representatives found so far until either the oracle returns $+1$, in which case the algorithm found a cluster that u belongs to, or all the key representatives have been compared to u without success, indicating that u is itself the key representative of a previously unseen cluster.

When k is known, the algorithm can always determine the identity of a given element u after having made $k-1$ comparisons, since if u does not belong to any of the first $k-1$ clusters, then it must belong to the remaining one. We give pseudocode for the k-unknown and the k-known versions of this algorithm in Algorithm 1 and Algorithm 2 respectively.

Algorithm 1: $\texttt{RS}(V, \alpha)$

Input: Finite set V, (error-free) same-cluster oracle α for some partition $\mathcal{C}$ of V

Output: $\mathcal{C} \stackrel{\text{def}}{=} \{\mathcal{C}_i\}_{i=1}^{k}$ a partition of V.

$num\_clusters = 0$

// Each iteration of the following loop processes a single vertex

while $V \neq \emptyset$ **do**

  $Is\_Placed =$ False

  Pick $v \in V$ arbitrarily.

  for $a = 1, ..., num\_clusters$ **do**

    Query $\alpha(v, w)$ for some $w \in C_a$

    if $\alpha(v, w) = 1$ **then**

      $C_a = C_a \cup \{v\}$

      $Is\_Placed =$ True

      break

    end

  end

  if $Is\_Placed == False$ **then**

    $num\_clusters = num\_clusters + 1$

    $C_{num\_clusters} = \{v\}$

  end

  $V = V \setminus \{v\}$

end

return $\mathcal{C} \stackrel{\text{def}}{=} \{C_a\}_{a=1}^{k}$;

Algorithm 2: `RS_k`(V, α, k)

Input: Finite set V, same-cluster oracle α, number of clusters k

Output: $\mathcal{C} \stackrel{\text{def}}{=} \{\mathcal{C}_i\}_{i=1}^{k}$ a partition of V.

$num\_clusters = 0$

// Each iteration of the following loop processes a single vertex

while $V \neq \emptyset$ **do**

  Pick $v \in V$ arbitrarily.

  $Is\_Placed =$ False

  for $a = 1, ..., \min\{num\_clusters, k-1\}$ **do**

    Query $\alpha(v, w)$ for some $w \in C_a$

    if $\alpha(v, w) = 1$ **then**

      $C_a = C_a \cup \{v\}$

      $Is\_Placed =$ True

      break

    end

  end

  if $Is\_Placed == False$ **then**

    if $num\_clusters < k$ **then**

      $num\_clusters = num\_clusters + 1$

      $C_{num\_clusters} = \{v\}$

    end

    else

      $C_{num\_clusters} = C_{num\_clusters} \cup \{v\}$

    end

  end

  $V = V \setminus \{v\}$

end

Theorem 44 (Reyzin and Srivastava (2007); Liu and Mukherjee (2022))**.** *Algorithm 1 for*

the problem of learning partition in the k-unknown setting makes at most:

$$RS^u(n,k) := n(k-1) - \binom{k}{2}$$

queries, Algorithm 2 for the problem in the k-known setting makes at most:

$$RS^k(n,k) := n(k-1) - \binom{k}{2},$$

queries.

Remarkably, this simple strategy achieves the best possible worst-case performance guarantees. In fact, in a recent paper by Liu and Mukherjee (2022), the authors prove that the two algorithms given above are optimal for their respective problems. In particular they show the following:

Theorem 45 (Liu and Mukherjee (2022))**.** *Every algorithm for the problem of learning partitions with a same-cluster oracle must make at least $RS^k(n,k)$ queries if k is known to the algorithm and at least $RS^u(n,k)$ otherwise, in the worst case.*

The proof of Liu and Mukherjee highlights the fact that the worst case for Algorithms 1 and 2 occurs when the partition is completely skewed and all the points are in the same cluster. While we do not report the full proof here, the general technique they use is the following: they construct an adversary for the problem, that answers -1 whenever possible, and then show that, at termination, the set of queries which returned -1 must form a uniquely surjectively k-colorable graph. They then make use of a result of Shaoji (1990) to lower bound the number of edges in any such graph and obtain the result.

4.3 Randomized Learner in the Absence of Error

In this section, we analyze the expected performance of a simple randomized version of Algorithm 1. Note that one needs to be careful when defining worst-case expected performance. In general, it is impossible for a randomized algorithm to ensure a better worst-case query complexity than that of Algorithm 1, when worst-case is taken against every possible set of answers the oracle might issue, i.e. against a fully adaptive adversary. In particular, the lower bound described in the section above applies independently of how one picks the queries. Hence, to obtain a reasonable notion of expected query complexity, we must assume that the adversary who chooses the answers to the queries has to commit to a partition $\mathcal{C}$ prior to seeing the realization of the coin flips of the algorithm, and we consider the worst-case expected query complexity over all choices of $\mathcal{C}$.

Consider Algorithm 3. The algorithm is a natural randomized variant of Algorithm 1, in which the vertices are inserted one by one in a uniformly random order.

Algorithm 3: `Randomized_RS`(V, α)

Input: A finite set V, same-cluster oracle α

Output: $\mathcal{C} \stackrel{\text{def}}{=} \{\mathcal{C}_i\}_{i=1}^{k}$ a partition of V.

$num\_clusters = 0$

// Each iteration of the following loop processes a single vertex

while $V \neq \emptyset$ **do**

  $Is\_Placed =$ False Pick $v \in V$ uniformly at random.

  for $a = 1, ..., num\_clusters$ **do**

    Query $\alpha(v, w)$ for some $w \in C_a$

    if $\alpha(v, w) = 1$ **then**

      $C_a = C_a \cup \{v\}$

      $Is\_Placed =$ True

      break

    end

  end

  if $Is\_Placed == False$ **then**

    $num\_clusters = num\_clusters + 1$

    $C_{num\_clusters} = \{v\}$

  end

  $V = V \setminus \{v\}$

end

return $\mathcal{C} \stackrel{\text{def}}{=} \{C_a\}_{a=1}^{k}$;

We can bound the expected number of queries made by the randomized learning strategy above in the k-unknown setting.

Theorem 46. *Let $Q_{\mathcal{C}}$ be the number of queries made by Algorithm 3 when run on a partition*

$\mathcal{C} = \{C_a\}_{a\in[k]}$ *with* $|C_a| = n_a$. *We then have:*

$$\mathbb{E}\left[Q_{\mathcal{C}}\right] = (n-k) + \sum_{\substack{a,b\in[k]\\ a\neq b}} \frac{n_a n_b}{n_a + n_b}. \tag{4.1}$$

Proof. We will assume without loss of generality that $V = [n]$. For any $i \in [n]$, let $a_i \in [k]$ be such that $i \in C_{a_i}$. For any $i \in [n]$ and any $a \in [k] \setminus \{a_i\}$, let $X_{i,a}$ be the indicator random variable of the event that element i is compared to some element in cluster C_a (before it gets compared with some element in the cluster C_{a_i} and gets assigned to it).

Note that, during the run of the algorithm, with the exception of a single representative from each cluster, each element i gets compared to a unique element in its own cluster, amounting to $(n-k)$ many comparisons. The rest of the comparisons are inter-cluster comparisons, in which an element i gets compared to a representative of a different cluster C_a. Note that each element gets compared to a representative of some cluster C_a at most once.

We also have:

$$\mathbb{E}\left[X_{i,a}\right] = \Pr[X_{i,a} = 1] = \frac{n_a}{n_{a_i} + n_a},$$

since the above probability is equal to the probability that the first vertex in C_a is processed before the first vertex in C_{a_i} in a random permutation.

We then have that the number of queries made by `Randomized_RS` is given by:

$$\mathbb{E}\left[Q_{\mathcal{C}}^{\mathtt{RS}}\right] = (n-k) + \sum_{\substack{i\in[n]\\ a\in[k]\setminus\{a_i\}}} \mathbb{E}\left[X_{i,a}\right] = (n-k) + \sum_{\substack{a,b\in[k]\\ a\neq b}} \frac{n_a n_b}{n_a + n_b}$$

as needed. □

We also note the following:

Lemma 47. *The expression on the right-hand side of Equation 4.1 is Schur-concave on the positive orthant.*

Proof. Note that the function is symmetric with respect to permuting the entries of its input. Therefore it is sufficient to show that the function is concave on $\mathbb{R}^k_{>0}$.

We note that it is the sum of a linear term $(n-k)$ and a number of functions of the form:

$$f_{a,b}(n_1, ..., n_k) = \frac{n_a n_b}{n_a + n_b}.$$

We can then simply verify that $f_{a,b}$ is concave. We have:

$$\frac{\partial f_{a,b}}{\partial n_a} = \frac{n_b^2}{(n_a + n_b)^2} \quad \text{and} \quad \frac{\partial f_{a,b}}{\partial n_b} = \frac{n_a^2}{(n_a + n_b)^2}.$$

While:

$$\frac{\partial^2 f_{a,b}}{(\partial n_a)^2} = -\frac{2n_b^2}{(n_a + n_b)^3} \quad \text{and} \quad \frac{\partial^2 f_{a,b}}{(\partial n_b)^2} = -\frac{2n_a^2}{(n_a + n_b)^3},$$

and:

$$\frac{\partial^2 f_{a,b}}{\partial n_a \partial n_b} = \frac{\partial^2 f_{a,b}}{\partial n_b \partial n_a} = -\frac{2(n_a + n_b)^2 n_b - 2n_b^2(n_a + n_b)}{(n_a + n_b)^4} = -\frac{2(n_a + n_b)n_b - 2n_b^2}{(n_a + n_b)^3} = -\frac{2n_a n_b}{(n_a + n_b)^3}.$$

Hence the hessian of $f_{a,b}$ is positive definite at every point $(n_a, n_b) \in \mathbb{R}^2$, and the function is concave as needed. □

Consider two k-partitions $\mathcal{C}$ and $\mathcal{C}'$ of V where the clusters have sizes $(n_1, ..., n_k)$ and $(n'_1, ..., n'_k)$ respectively. A simple consequence of the above lemma is that the expected number of queries made by Algorithm 3 is lower when it is run on partition $\mathcal{C}$ rather than partition $\mathcal{C}'$ if and only if the sequence $(n_1, ..., n_k)$ majorizes $(n'_1, ..., n'_k)$. Interestingly, this points to the fact that even though the worst-case performance of Algorithm 1 is observed when the clusters are completely unbalanced, its randomized counterpart performs better

the more unbalanced the partitions are. We then get the following:

Theorem 48. *Let $\mathcal{C}$, n, k and Q be given as above, then:*

$$\mathbb{E}[Q] \leq \frac{n(k+1)}{2} - k.$$

Proof. From Lemma 47 it follows that the maximum value of the right-hand side of the Equation 4.1 is achieved when, for all $a \in [k]$, $n_a = n/k$, and hence:

$$\mathbb{E}[Q] \leq (n-k) + \sum_{\substack{a,b \in [k] \\ a \neq b}} \frac{n^2}{k^2} \frac{1}{2n/k} = (n-k) + \frac{n(k-1)}{2} = \frac{n(k+1)}{2} - k.$$

□

Note that the number of queries made by the algorithm when k is known is always strictly smaller than the number of queries made by the algorithm when k is unknown, and hence Theorem 48 provides an upper bound to the former too.

It is still unknown whether the above average number of queries matches the randomized query complexity of the problem.

4.4 An Algorithm for the Weighted ℓ-PL Problem in the k-Known Setting

In this section we give an algorithm for the weighted ℓ-PL problem in the k-known setting, which achieves optimal asymptotic query complexity. As an immediate consequence, we obtain algorithms for the (unweighted) ℓ-PL problem as well as its one-sided variants. We begin by briefly providing intuition for the algorithm. The algorithm maintains a refinement $\mathcal{C}'$ of the correct hidden partition $\mathcal{C}$. At the start of the algorithm, $\mathcal{C}'$ is the set of all singleton elements in V. The algorithm repeatedly makes sequences of queries that guarantee progress

towards either (a) learning $\mathcal{C}$, in the form of establishing that a pair of vertices u and v must be part of the same cluster, or (b) certifying that an error has occurred in the oracle's responses.

In particular, the algorithm repeatedly attempts to construct $(k+1)$-cliques of -1 responses. If at any point the oracle responses form such a clique, then a false-negative response must have occurred, as vertices in V belong to at most k distinct clusters (note that k is assumed to be known to the algorithm). In this case, the algorithm has certified a false-negative response. On the other hand, the only way that this process may be interrupted is if the algorithm receives a $+1$ response to some query uv. However, if such a positive response is received, repeatedly querying the pair uv will either quickly determine that u and v must be part of the same cluster or it will reveal that an error has occurred. In the first case, the algorithm makes progress towards learning $\mathcal{C}$, and in the second case it has certified the occurrence of an error.

We provide the pseudocode for the algorithm and describe in prose the main functions of each subroutine. The algorithm is comprised of subroutines `Learn`, `Get_New`, `Insert` and `Compare`:

Learn This is the core component of the algorithm. It maintains a partition $\mathcal{C}'$ for V. Throughout the algorithm, $\mathcal{C}'$ is guaranteed to be a refinement of the true hidden partition $\mathcal{C}$. $\mathcal{C}'$ is initialized as the set of singletons for every element in V. The algorithm then repeatedly tries to construct a clique of negative answers, supported on some set $S \subseteq V$ until $|S| = k+1$. It does so by inserting new vertices v into S by calling the subroutine `Insert`. `Insert` may then either make progress towards constructing the clique (by increasing the size of S) or towards learning the hidden partition by decreasing the size of $\mathcal{C}'$ (merging two clusters together).

Get_New Given $S \subseteq V$ and $\mathcal{C}'$ a partition of V, `Get_New` returns an element $v \in V \setminus S$ representing a cluster in $\mathcal{C}'$ that's not yet represented in S, i.e. $v \notin [u]_{\mathcal{C}'}$ for any $u \in S$.

Insert Given an element v and a subset $S \subseteq V$, `Insert` compares v against every element in S by passing u and v to the function `Compare`. `Compare` returns `result`$\in \{\pm 1\}$. If `Compare` returns a negative result for each $u \in S$, then `Insert` adds v to the subset S. On the other hand, if `Compare` returns a positive result for v and some element $u \in S$, then `Insert` updates the partition $\mathcal{C}'$ to reflect that u and v must be in the same cluster. The latter is done by merging $[u]_{\mathcal{C}'}$ with $[v]_{\mathcal{C}'}$.

Compare Given two vertices u and v, `Compare` repeatedly submits query uv to the oracle. `Compare` maintains `count` to record information about the number of positive and negative responses received for this query. If at any point `Compare` has received strictly more negative responses than positive responses, it returns `result`$= -1$.

For the sake of analysis, we will assume that the algorithm maintains a global history of all the queries made to the oracle and the responses received. We will also let the algorithm assign labels to query-response pairs. This is done in order to charge the queries made to some measure of progress towards either learning the partitions or certifying that errors have occurred. The lines of pseudocode assigning labels to the query history, written in ***bold italic*** characters, are not necessary for the algorithm to work correctly but rather are included only to facilitate the analysis.

Algorithm 4: $\texttt{Learn}(V, \alpha, k, \ell_{\texttt{yes}}, \ell_{\texttt{no}})$

Input: A finite set V, an $(\ell_{\texttt{yes}}, \ell_{\texttt{no}})$-faulty oracle $\alpha_{\mathcal{C}}$, a target number of clusters k, and error parameters $\ell_{\texttt{yes}}$ and $\ell_{\texttt{no}}$.

Output: A partition $\mathcal{C}'$ of V.

$S \leftarrow \{\}$, $\mathcal{C}' \leftarrow \{\{v\} \mid v \in V\}$
while $|\mathcal{C}'| > k$ **do**
  $v \leftarrow \texttt{Get\_New}(S, \mathcal{C}')$
  $\mathcal{C}', S \leftarrow \texttt{Insert}(V, \alpha, S, \mathcal{C}', \ell_{\texttt{yes}}, \ell_{\texttt{no}})$
  if $|S| = k + 1$ **then**
    $S \leftarrow \{\}$
  end
end
return $\mathcal{C}'$

Algorithm 5: $\texttt{Get\_New}(S, \mathcal{C}')$

Input: A partition $\mathcal{C}'$ of some finite set V, and a subset $S \subset V$ such that $|S| \leq |\mathcal{C}'|$

Output: An element $v \in V$ representing a set in $\mathcal{C}'$ which is currently not represented by any element of S.

for $v \in V$ **do**
  if $[v]_{\mathcal{C}'} \cap S = \emptyset$ **then**
    return v
  end
end
return $\perp$

Algorithm 6: `Insert`$(v, \alpha, \mathcal{C}', S, \ell_{\texttt{yes}}, \ell_{\texttt{no}})$

Input: A finite set V, a same-cluster oracle α for V, a partial clique S, a candidate partition $\mathcal{C}'$ of V, and error parameters $\ell_{\texttt{yes}}$ and $\ell_{\texttt{no}}$.

Output: A new candidate cluster $\mathcal{C}'$, partial clique S.

for $u \in S$ **do**
  // Returns either `result` $= -1$ or $+1$
  `result` $\leftarrow$ `Compare`$(u, v, \alpha, \ell_{\texttt{yes}}, \ell_{\texttt{no}})$
  // If $+1$: it is guaranteed that u and v are part of the same cluster and $\mathcal{C}'$ is edited to reflect this
  if *result* $= +1$ **then**
    Re-label all '`clique`' ***queries made during this execution of*** `Insert` ***(and all calls to*** `Compare` ***therein) as*** '`merge-`'
    `new_set` $\leftarrow [v]_{\mathcal{C}'} \cup [u]_{\mathcal{C}'}$
    $\mathcal{C}' \leftarrow (\mathcal{C}' \setminus \{[v]'_{\mathcal{C}}, [u]_{\mathcal{C}'}\}) \cup$ `new_set`
    return $\mathcal{C}', S$
  end
end
// If S is empty or if `result` $= -1$ for every $u \in S$ then a new item is inserted into S
$S \leftarrow S \cup \{v\}$
return $\mathcal{C}', S$

Algorithm 7: `Compare`$(u, v, \alpha, \ell_{\texttt{yes}}, \ell_{\texttt{no}})$

Input: A pair of vertices u and v from some finite set V, a same-cluster oracle α for V, and error parameters $\ell_{\texttt{yes}}$ and $\ell_{\texttt{no}}$.

Output: `result` $\in \{\pm 1\}$.

// Initialize `count` to track "how many more times have we observed $r = +1$ compared to $r = -1$"
`count` $\leftarrow 0$
while *count* ≥ 0 **do**
    $r \leftarrow \alpha(u, v)$
    if $r = -1$ ***and*** *count*> 0 **then**
        `count` $\leftarrow$`count`-1
    else if $r = -1$ ***and*** *count*$= 0$ **then**
        /* If the number of -1 responses exceeds the number of $+1$ responses seen so far, we return -1*/
        Label the last query as '`clique`'
        Label all other queries created on this execution of `Compare` ***as*** '`spurious`'
        return -1
    else
        `count` $\leftarrow$`count`$+1$
    end
    // If `count`$/\ell_{\texttt{yes}} > 1$ then we have verified that $u \sim_{\mathcal{C}*} v$ so return `result` $= +1$
    if *count*$/\ell_{yes} > 1$ **then**
        Label the last `count`***-many queries which received positive responses as*** '`merge+`'
        Label all other queries made during this execution of `Compare` ***as*** '`spurious`'
        return `1`
    end
end

Lemma 49. *Throughout the execution of Algorithm 4, $\mathcal{C}'$ is always a refinement of the true partition.*

Proof of Lemma 49. Note that $\mathcal{C}'$ is essentially a global variable, since any change made to $\mathcal{C}'$ in a subroutine is reflected in its parent subroutine. We begin by noting that $\mathcal{C}'$ is trivially a refinement of the correct partition at the beginning of the algorithm. The only point in the algorithm at which $\mathcal{C}'$ is modified is on Lines 7 and 8 of `Insert`. In that step, the algorithm modifies $\mathcal{C}'$ by merging the cluster containing v with the cluster containing u. These lines are executed only if the value of the variable `result`, output by `Compare`, is $+1$. This happens only if $\texttt{count}/\ell_{\texttt{yes}} > 1$ on Line 16 of `Compare`, in which case the latest call to `Compare` has obtained more than $\ell_{\texttt{yes}}$ many uv queries returning $+1$.

We now argue that if that happens, u and v must have been part of the same cluster in the ground-truth partition $\mathcal{C}$. If u and v were not part of the same cluster in $\mathcal{C}$, we derive a contradiction, as this would imply:

$$\text{FP} \geq \texttt{count} > \ell_{\texttt{yes}}$$

and hence $\text{FP}\,/\ell_{\texttt{yes}} > 1$. Hence, if $\mathcal{C}'$ was a refinement of $\mathcal{C}$ before merging the clusters containing u and v, then it is still a refinement of $\mathcal{C}$ after merging the two clusters. This invariant is then maintained throughout the algorithm, and the lemma holds. □

Lemma 50. *At the end of the algorithm, the history of queries contains at most $(\ell_{\texttt{yes}} + 1)(n-k)$ queries labelled '`merge+`'and at most $k(n-k)$ queries labelled '`merge-`'.*

Proof of Lemma 50. Consider a point in the algorithm in which `Compare` returns $+1$. When that happens, the **if** block starting on Line 5 of `Insert` is executed; we refer to this as a *merge event*.

We begin by arguing that at most $n-k$ merge events can occur during an execution of Algorithm 4. At the start of the algorithm $|\mathcal{C}'| = n$. At each merge event two clusters in $\mathcal{C}'$

are merged and the cardinality of $\mathcal{C}'$ decreases by 1. Since $\mathcal{C}'$ is always a refinement of the hidden k-partition $\mathcal{C}$ (by Lemma 49), this implies that at most $n-k$ merge events can occur during the execution of the algorithm.

Both '`merge-`' and '`merge+`' queries are only created in correspondence with a merge event. We now show that on any merge event, at most k '`merge-`' queries and at most $\ell_{\texttt{yes}}+1$ '`merge+`' queries are created. By the above argument these results imply the statement of the lemma.

We now show that at most k '`merge-`' queries are created during each merge event. '`merge-`' queries are created when some '`clique`' queries are re-labelled as '`merge-`'. Specifically, all the '`clique`' queries created during every execution of `Compare` within the current execution of `Insert` are relabelled as '`merge-`'. On any execution of `Insert`, at most one '`clique`' query is created for every element in the current set S. During any iteration of the **for** loop at `Insert` Line 1, the set S is of size at most k, implying that at most k '`clique`' queries are created during one execution of `Insert`. As a result, at most k '`merge-`' queries are created at every merge event.

Similarly, '`merge+`' queries are only created in correspondence with a merge event. Specifically, merge events occur when `Compare` returns `result`$= +1$. When this happens, '`merge+`' queries are created at `Compare` Line 17. When Line 17 is executed, the number of queries labelled '`merge+`' is equal to the current value of `count`, which is at most at most $\ell_{\texttt{yes}}+1$. Therefore, at most $\ell_{\texttt{yes}}+1$ '`merge+`'queries are created in correspondence with each merge event, concluding the proof. □

Lemma 51. *At the end of the algorithm, the history of queries contains at most* $2\max\{\ell_{\texttt{yes}}, \ell_{\texttt{no}}\}$ *queries labelled '`spurious`'.*

Proof of Lemma 51. Each query uv labelled '`spurious`' can be paired uniquely with another query uv labelled `spurious` which returned the opposite answer. Hence, the number of erroneous answers returned by the oracle is at least the number of `spurious` queries divided

by two. This number is upper bounded by $\max\{\ell_{\texttt{yes}}, \ell_{\texttt{no}}\}$ and the result follows. □

Lemma 52. *At the end of the algorithm, the history of queries contains at most* $(\ell_{\texttt{no}} + 1)\binom{k+1}{2}$ *queries labelled '*`clique`*'.*

Proof of Lemma 52. The algorithm only labels queries as '`clique`' in Line 9 of `Compare`. Thus every '`clique`' query is made between two elements u and v such that both u and v were inserted into some construction of a clique, supported on the set S.

If S reaches cardinality $k+1$, the **if** block on Line 5 of `Insert` is executed. We refer to this occurrence as a *clique event*. When a clique event occurs, queries have made up a new clique of -1 responses of size $k+1$. In order for all of these responses to have been correct, the ground-truth partition would need to have been size at least $k+1$, hence every time a clique event happens, a false negative error must have occurred. Since every clique event can be charged to a false negative error, at most $\ell_{\texttt{no}}$ clique events can occur.

We will charge every '`clique`' query made before a clique event to the first clique event following the query's creation. These queries represent a distinct edge uv of a clique built on S which eventually reaches size $k+1$, and hence exactly $\binom{k+1}{2}$ many of these '`clique`' queries will occur per clique event. Finally, all '`clique`' queries occurring after the last clique event make up the edges of a clique supported on the set S, which never reaches size $k+1$, and hence there are less than $\binom{k+1}{2}$ such queries.

The result then follows. □

Lemma 53. *If Algorithm 4 makes a finite number of queries, then it must terminate.*

Proof of Lemma 53. We show that if Algorithm 4 makes a finite number of queries, then Line 4 of Algorithm 4 is executed a finite number of times. This implies termination of Algorithm 4.

When Line 4 of Algorithm 4 executes, Algorithm 4 calls Algorithm 6, `Insert`. If S is nonempty on the execution of Line 4, then `Insert` calls `Compare` on some pair of elements.

On every call to `Compare`, Line 4 of `Compare` executes at least once, and hence the algorithm makes at least one query. If S is empty on the execution of Line 4, then `Insert` increases the size of S by 1. This implies that on the next execution of Line 4, S will be nonempty. Thus in order for Algorithm 4 to execute Line 4 infinitely many times, the algorithm must make infinitely many queries, yielding the result. □

We now analyze the number of queries made by the algorithm. We begin by noting that when the algorithm terminates, every query made will hold exactly one of the following labels: '`merge-`', '`merge+`', '`clique`', or '`spurious`'. The analysis proceeds by showing how labels of each category contribute to either (a) correctly identifying the true partition, or (b) correctly certifying the occurrence of errors.

Equipped with these lemmata, we now prove the following.

Theorem 54. *There exists an algorithm for the weighted ℓ-PL problem which recovers the full partition $\mathcal{C}$ in the k-known setting with query complexity bounded by*

$$O(RS^k(n,k) + (n-k)\ell_{\texttt{yes}} + k^2\ell_{\texttt{no}}).$$

Proof. Let $|\text{`}\texttt{merge-}'|, |\text{`}\texttt{merge+}'|, |\text{`}\texttt{spurious}'|, |\text{`}\texttt{clique}'|$ be the number of '`merge-`', '`merge+`', '`spurious`' and '`clique`' queries made by the algorithm respectively. By Lemmata 50, 51, and 52, the algorithm makes at most

$$\begin{aligned}
&|\text{`}\texttt{merge-}'|+|\text{`}\texttt{merge+}'| + |\text{`}\texttt{spurious}'| + |\text{`}\texttt{clique}'| \\
&\qquad \leq nk - k^2 + (\ell_{\texttt{yes}}+1)(n-k) + 2\max\{\ell_{\texttt{yes}}, \ell_{\texttt{no}}\} + (\ell_{\texttt{no}}+1)\binom{k+1}{2} \\
&\qquad = O\left(RS^k(n,k) \ + (n-k)\ell_{\texttt{yes}} + k^2\ell_{\texttt{no}}\right)
\end{aligned}$$

many queries. By Lemma 53 this implies that the algorithm must terminate.

When the algorithm terminates, the condition in the **while** loop of `Learn` is not met,

and hence $|\mathcal{C}'| \leq k$. But since, by Lemma 49, $\mathcal{C}'$ is a refinement of $\mathcal{C}$, it must hold that upon termination $|\mathcal{C}'| = k$ and hence $\mathcal{C}' = \mathcal{C}$. The algorithm then returns the correct hidden partition $\mathcal{C}$. □

4.5 Adapting the Algorithm to k-Unknown Setting

In this section, we show that the algorithm in the previous section can be modified to work in the k-unknown setting. We will see that this modification yields optimal asymptotic complexity for the (unweighted) ℓ-PL problem but leaves a small gap between the upper and lower bound for the weighted version of the problem. In particular, this stems from the algorithm's need to make more queries to deal with false negative answers.

The main challenge in adapting Algorithm 4 to the k-unknown setting is that the algorithm crucially relies on knowing the value of k to build a lower bound to the number of observed errors, by counting the number of $(k+1)$-cliques of -1 answers observed. To overcome this problem, the new algorithm will instead simply aim to build the largest possible clique of -1 answers. We now show that, remarkably, when false positive and false negative errors are equivalent ($\ell_{\texttt{yes}} = \ell_{\texttt{no}} = \ell$), this suffices to match the optimal guarantees of Algorithm 4, up to small constant factors.

Algorithm 8: `Learn_k_unknown`$(V, \alpha, \ell_{\texttt{yes}}, \ell_{\texttt{no}})$

Input: A finite set V, a same-cluster oracle α, error parameters $\ell_{\texttt{yes}}$ and $\ell_{\texttt{no}}$.

Output: A partition $\mathcal{C}'$ of V.

$\mathcal{C}' \leftarrow \{\{v\} \mid v \in V\}$, $\texttt{idx} \leftarrow 1$
$S_{\texttt{idx}} \leftarrow \{\}$
while $idx \leq \ell_{no} + 1$ **do**
    $v \leftarrow$`Get_New`$(S, \mathcal{C}')$
    if $v == \perp$ **then**
        $\texttt{idx} += 1$
        $S_{\texttt{idx}} \leftarrow \{\}$
    end
    else
        $\mathcal{C}', S_{\texttt{idx}} \leftarrow$ `Insert`$(V, \alpha, S_{\texttt{idx}}, \mathcal{C}', \ell_{\texttt{yes}}, \ell_{\texttt{no}})$
    end
end
return $\mathcal{C}'$

The intuition behind the success of Algorithm 8 is that, for the algorithm to construct large cliques (significantly larger than the cardinality k of the hidden partition), the oracle must return many incorrect answers. We formalize this intuition in Lemma 55.

Consider the execution of Algorithm 8. The algorithm will repeatedly construct cliques of -1 queries supported on vertex sets S_i, adding elements to S_i until all elements of V belong to an equivalence class $[u]_{\mathcal{C}'}$ for some $u \in S_i$. Once this condition is met, the algorithm increments i and initializes S_{i+1} as the empty set. The following lemma shows that the maximum size of any such set S_i constructed by the algorithm is bounded as a function of the true unknown partition size k and the number of false negative errors that must have occurred in forming S_i.

Consider the set S_i constructed by Algorithm 8 while $\texttt{idx} = i$. We will denote by n_i

the maximum cardinality attained by this set S_i. For convention we will also set $n_0 = n$. Moreover, we let ℓ_i denote the number of false negative responses returned while $\texttt{idx} = i$.

Lemma 55. *For any $i \geq 1$, we have:*

$$n_i \leq k + \sqrt{2k\ell_i}. \tag{4.2}$$

Proof. Note that, if the hidden partition has cardinality k, and the algorithm obtains a clique of -1 responses on n_i vertices, then the number of false negative errors contained in the clique responses is at least the minimum number of within-cluster edges of any k-partition of the complete graph K_{n_i}. I.e. a lower bound to the number of false negative errors that must have occurred is given by:

$$\begin{aligned}
\ell_i &\geq \min\left\{ \sum_{j=1}^{k} \binom{c_j}{2} \middle| c_1, ..., c_k \in \mathbb{N}, \sum_{j=1}^{k} c_j = n_i \right\} \\
&\geq \min\left\{ \sum_{j=1}^{k} \frac{c_j(c_j-1)}{2} \middle| c_1, ..., c_k \in \mathbb{R}_{>0}, \sum_{j=1}^{k} c_j = n_i \right\}.
\end{aligned}$$

Note that the function $f(x) = \frac{x(x-1)}{2}$ is convex, and hence by Karamata's Inequality (Theorem 1 from Kadelburg et al. (2005)) the right-hand side above is lower bounded by:

$$\ell_i \geq \sum_{j=1}^{k} \frac{\frac{n_i}{k}\left(\frac{n_i}{k} - 1\right)}{2} = \frac{1}{2} n_i \left(\frac{n_i}{k} - 1\right).$$

Re-arranging, we find that $n_i^2 - k n_i \leq 2\ell_i k$. Completing the square we obtain:

$$\left(n_i - \frac{1}{2}k\right)^2 \leq 2\ell_i k + \frac{1}{4}k^2$$

giving:

$$n_i - \frac{1}{2}k \leq \sqrt{2\ell_i k + \frac{1}{4}k^2} \leq \sqrt{2\ell_i k} + \frac{1}{2}k,$$

and hence:

$$n_i \leq \sqrt{2\ell_i k} + k,$$

as needed. □

Crucially, we note that as with Algorithm 4, the partition $\mathcal{C}'$ maintained by Algorithm 8 is always a refinement of the ground-truth.

Lemma 56. *Throughout the execution of Algorithm 8, $\mathcal{C}'$ is always a refinement of the true partition.*

The proof of this lemma is identical to the proof of Lemma 49.

Lemma 57. *Algorithm 8 terminates.*

Proof. We show that Lines 6 and 10 in Algorithm 8 execute finitely many times. This implies termination of the algorithm.

Every time Line 6 executes, the value of `idx` is incremented by 1. Once the value of `idx` exceeds $\ell_{\texttt{no}} + 1$, the **while** loop in Algorithm 8 exits, and thus Line 6 can only execute finitely many times.

To show that Line 10 executes finitely many times, we show that every time Line 10 executes, either $\mathcal{C}'$ decreases in cardinality or $S_{\texttt{idx}}$ increases in cardinality. This follows from considering Algorithm 6, `Insert`. On each call to `Insert`, either `result` $= +1$ for some $u \in S$, in which case the cardinality of $\mathcal{C}'$ is reduced in Line 8, or `result` $= -1$ for every $u \in S$. In the latter case, the size of S is increased in Line 13.

By Lemma 56, $\mathcal{C}'$ is always a refinement of $\mathcal{C}$ and thus its cardinality cannot decrease more than $(n - k)$ times. Similarly, for each value `idx` $= i$, $|S_i| \leq n$ so S_i can increase in

size finitely many times for each value of $\texttt{idx} \in [1, \ell_{\texttt{no}} + 1]$. This implies that Line 10 can only execute finitely many times, completing the proof. □

Lemma 58. *When Algorithm 8 terminates $\mathcal{C}'$ is equal to the ground-truth partition $\mathcal{C}$.*

Proof. By Lemma 56, $\mathcal{C}'$ is always a refinement of $\mathcal{C}$. Therefore to show that the partition $\mathcal{C}'$ returned is equal to $\mathcal{C}$, it remains to show that the cardinality of $\mathcal{C}'$ upon termination is the true unknown value k.

Assume for the sake of contradiction that this does not hold. Once initialized, the only place where $\mathcal{C}'$ is modified during the execution of Algorithm 8 is in Lines 7 and 8 of `Insert`. There, the algorithm modifies $\mathcal{C}'$ by merging two previously disjoint equivalence classes. This reduces the number of disjoint equivalence classes in $\mathcal{C}'$, and so in particular the cardinality of $\mathcal{C}'$ montonically decreases during the execution of Algorithm 8. Thus if upon termination the cardinality of $\mathcal{C}'$ is strictly greater than k, then this was true at all points during the algorithm's execution.

Recall that n_i is defined to be the maximum cardinality attained by set S_i. In particular, S_i attains maximum size when Line 6 of Algorithm 8 is executed. When this happens, S_i contains exactly one representative from every cluster in $\mathcal{C}'$ at the time of execution. If the size of $\mathcal{C}'$ is strictly greater than k at all points during the execution of Algorithm 8 then $n_i > k \ \forall i \in [0, \ell_{\texttt{no}} + 1]$.

In particular for every value $i \in [1, \ell_{\texttt{no}} + 1]$, for every distinct $u, v \in S_i$ Algorithm 8 receives at least one response -1 for query (u, v). This response is consistent with the scenario in which u and v belong to disjoint sets in the ground truth partition $\mathcal{C}$. If $n_i > |\mathcal{C}|$ then at least one of these responses must have been a false negative error. As this holds for every value $i \in [1, \ell_{\texttt{no}} + 1]$, this implies least $\ell_{\texttt{no}} + 1$ false negative responses occur during the execution of Algorithm 8. In particular this violates the assumption that α is an $(\ell_{\texttt{yes}}, \ell_{\texttt{no}})$-faulty oracle, and we thus conclude that the size of $\mathcal{C}'$ cannot be greater than k upon termination, implying that the partition returned by the algorithm satisfies $\mathcal{C}' = \mathcal{C}$. □

As with the analysis of Algorithm 4, bounding the query complexity of Algorithm 8 reduces to bounding the number of queries with labels '`merge-`', '`merge+`', '`clique`', or '`spurious`' upon termination of the algorithm.

Lemma 59. *When Algorithm 8 terminates the history of queries contains at most:*

$$\left(1+\sqrt{2}\right)(\ell_{no}+1)k^2$$

queries labelled '`clique`'.

Proof. All queries labeled '`clique`' upon termination must have been created during the construction of some set S_i while `idx`$= i$. Given n_i the maximum cardinality attained by set S_i, the number of '`clique`' queries present upon termination that were constructed while `idx`$= i$ is exactly

$$\binom{n_i}{2} = \frac{1}{2}(n_i^2 - n_i) \leq k^2 + 2k\sqrt{2k\ell_i} + 2k\ell_i - n_i$$

where the upperbound follows from Lemma 55 and ℓ_i denotes the number of false negative responses returned by the oracle while `idx`$= i$. Summing the total number of such queries over all iterates `idx`$= 1, \ldots, \ell_{\texttt{no}} + 1$, we obtain:

$$\begin{aligned}
\sum_{i=1}^{\ell_{\texttt{no}}+1}\binom{n_i}{2} &\leq \frac{1}{2}\sum_{i=1}^{\ell_{\texttt{no}}+1} k^2 + 2k\sqrt{2k\ell_i} + 2k\ell_i - n_i \\
&= \frac{1}{2}(\ell_{\texttt{no}}+1)k^2 + \sqrt{2}\sum_{i=1}^{\ell_{\texttt{no}}+1} k\sqrt{k\ell_i} + k\sum_{i=1}^{\ell_{\texttt{no}}+1}\ell_i - \frac{1}{2}\sum_{i=1}^{\ell_{\texttt{no}}+1} n_i \\
&\leq \frac{1}{2}(\ell_{\texttt{no}}+1)k^2 + \sqrt{2}k\sum_{i=1}^{\ell_{\texttt{no}}+1}\frac{\ell_i+k}{2} + k\sum_{i=1}^{\ell_{\texttt{no}}+1}\ell_i - \frac{1}{2}\sum_{i=1}^{\ell_{\texttt{no}}+1} n_i \\
&= \frac{1}{2}(\ell_{\texttt{no}}+1)k^2 + \frac{k\ell_{\texttt{no}}}{\sqrt{2}} + \frac{(\ell_{\texttt{no}}+1)k^2}{\sqrt{2}} + k\ell_{\texttt{no}} - \frac{1}{2}k(\ell_{\texttt{no}}+1).
\end{aligned}$$

The last line follows by observing that by definition of ℓ_i, $\sum_{i=1}^{\ell+1}\ell_i \leq \ell_{\texttt{no}}$.

We can simplify the above expression using the fact that for integer k it holds that $k \leq k^2$, to obtain the bound:

$$\frac{1}{2}(\ell_{\texttt{no}}+1)k^2 + \frac{k\ell_{\texttt{no}}}{\sqrt{2}} + \frac{(\ell_{\texttt{no}}+1)k^2}{\sqrt{2}} + k\ell - \frac{1}{2}k(\ell_{\texttt{no}}+1) = \left(\frac{1}{2}+\frac{1}{\sqrt{2}}\right)\left((\ell_{\texttt{no}}+1)k^2 + k\ell_{\texttt{no}}\right)$$
$$\leq \left(1+\sqrt{2}\right)(\ell_{\texttt{no}}+1)k^2.$$

□

Lemma 60. *Upon termination of Algorithm 8, the history of queries contains at most* $2\max\{\ell_{\texttt{yes}}, \ell_{\texttt{no}}\}$ *queries labelled '*`spurious`*'.*

The proof follows analogously to that of Lemma 51.

Lemma 61. *Upon termination of Algorithm 8, the history of queries contains at most* $(\ell_{\texttt{yes}}+1)(n-k)$ *queries labelled '*`merge+`*', and at most*

$$(n-k)\max\left\{(1+\sqrt{2})k, k+\ell_{\texttt{no}}\sqrt{2}\right\}$$

queries labelled '`merge-`*'.*

Proof. Similarly to the analysis of Algorithm 4, we analyze the numbers of '`merge+`' and '`merge-`' queries, by considering *merge events*. A merge event is defined as the execution of Line 5 in `Insert`.

Recall that n_i is defined to be the maximum cardinality attained by set S_i. In particular, S_i attains maximum size when Line 6 of Algorithm 8 is executed. When this happens, S_i contains exactly one representative from every cluster in $\mathcal{C}'$ at the time of execution. Thus $n_i - n_{i+1}$ is equal to the number of clusters merged while $\texttt{idx} = (i+1)$, i.e. the number of merge events that occurred while $\texttt{idx} = (i+1)$. Moreover the size of $\mathcal{C}'$ upon termination

of Algorithm 8 is equal to $n_{\ell_{\texttt{no}}+1}$, and hence by Lemma 58 $n_{\ell_{\texttt{no}}+1} = k$. This implies

$$\sum_{i=0}^{\ell_{\texttt{no}}} n_i - n_{i+1} = n_0 - n_{\ell_{\texttt{no}}+1} = n - k.$$

As with the proof of Lemma 50, we note that each '`merge+`' query can be charged to a unique merge event in such a way that exactly $\ell_{\texttt{yes}} + 1$ '`merge+`' queries are charged to each merge event. As a result, the number $|\text{`}\texttt{merge+}'|$ of '`merge+`' queries satisfies:

$$|\text{`}\texttt{merge+}'| = \sum_{i=0}^{\ell_{\texttt{no}}} (\ell_{\texttt{yes}} + 1)(n_i - n_{i+1}) = (\ell + 1)(n - k).$$

Similarly, '`merge-`' queries are only created in correspondence with a merge event. Specifically, '`merge-`' are created when '`clique`' queries are re-labelled as '`merge-`'. When `idx` $= (i + 1)$ on any single execution of `Insert`, at most one '`clique`' query is created for every element in the current set $S_{(i+1)}$. Hence at most n_{i+1} '`merge-`' queries correspond to each merge event that occurs while `idx` $= (i + 1)$. Summing over all values attained by `idx`, this implies the total number of queries labeled '`merge-`' is bounded by

$$\begin{aligned}
|\text{`}\texttt{merge-}'| &\leq \sum_{i=0}^{\ell_{\texttt{no}}} n_{i+1}(n_i - n_{i+1}) \\
&\leq \sum_{i=0}^{\ell_{\texttt{no}}} (k + \sqrt{2k\ell_{i+1}})(n_i - n_{i+1}) \\
&= k \sum_{i=0}^{\ell_{\texttt{no}}} (n_i - n_{i+1}) + \sqrt{2k} \sum_{i=0}^{\ell_{\texttt{no}}} (n_i - n_{i+1})\sqrt{\ell_{i+1}},
\end{aligned}$$

where the second inequality follows from the bound in Lemma 55. Invoking the fact that $\sum_{i=0}^{\ell_{\texttt{no}}} n_i - n_{i+1} = n - k$, we can bound the above and obtain:

$$|\text{`}\texttt{merge-}'| \leq k \cdot (n - k) + \sqrt{2k} \cdot \max_i \sqrt{\ell_{i+1}} \cdot \sum_{i=0}^{\ell_{\texttt{no}}} (n_i - n_{i+1})$$

$$\leq \left(k + \sqrt{2k\ell_{\texttt{no}}}\right)(n-k)$$
$$\leq (n-k) \cdot \max\left\{(1+\sqrt{2})k, k + \ell_{\texttt{no}}\sqrt{2}\right\}$$

yielding the result. □

Equipped with the above lemmata, we proceed to prove the following theorem.

Theorem 62. *There exists an algorithm for the weighted ℓ-PL problem which recovers the full partition $\mathcal{C}$ in the k-unknown setting with query complexity bounded by*

$$O\left(RS^u(n,k) + (n-k)\max\{\ell_{\textit{yes}}, \ell_{\textit{no}}\} + k^2\ell_{\textit{no}}\right).$$

Proof. By Lemmata 57 and 58, Algorithm 8 terminates and returns the correct hidden partition. It thus remains to bound the number of queries made during execution.

Let $|\text{`}\texttt{merge-}'|, |\text{`}\texttt{merge+}'|, |\text{`}\texttt{spurious}'|, |\text{`}\texttt{clique}'|$ be the number of '`merge-`', '`merge+`', '`spurious`' and '`clique`' queries made by the algorithm respectively. By Lemmata 59, 60, and 61, the algorithm makes at most

$$\begin{aligned}
&|\text{`}\texttt{merge-}'| + |\text{`}\texttt{merge+}'| + |\text{`}\texttt{spurious}'| + |\text{`}\texttt{clique}'| \\
&\qquad \leq (n-k)\max\left\{(1+\sqrt{2})k, k + \ell_{\texttt{no}}\sqrt{2}\right\} + (\ell_{\texttt{yes}} + 1)(n-k) \\
&\qquad\quad + 2\max\{\ell_{\texttt{yes}}, \ell_{\texttt{no}}\} + (1+\sqrt{2})(\ell_{\texttt{no}} + 1)k^2 \\
&\qquad = O\left(RS^u(n,k) + (n-k)\max\{\ell_{\texttt{yes}}, \ell_{\texttt{no}}\} + k^2\ell_{\texttt{no}}\right)
\end{aligned}$$

many queries. □

4.6 Lower Bound Techniques: Rényi-Ulam Games and Correlation Clustering

In the sections that follow, we will provide lower bounds for the $\ell_{\texttt{yes}}$ and $\ell_{\texttt{no}}$ problems. Underlying our lower bound analysis is a connection between partition learning problems, Rényi-Ulam and Chip-Liar games, and correlation clustering. The Rényi-Ulam game, as defined in the background is equivalent to the following Chip-Liar game (see e.g. Chapter 15, in the book of Alon and Spencer (2016)). In the game, N chips, numbered 1 to N are placed on a game board that has $\ell + 1$ positions, labeled $0, \ldots, \ell$. At the start of the game, all of the chips begin at position 0 on the board. The game takes place in rounds. On each round the questioner player Q selects a subset S of the chips, and the responder player R decides on one of the following moves: they can either increase the position of every chip in S by 1, or increase the position of every chip in $\overline{S}$ by 1. If a chip at position ℓ is in a group whose position is advanced, the chip is said to *have fallen off the board.* After this point such chips will no longer advance. The rules of the game constrain R's responses; R must ensure that on every round, at least one chip remains on the board at position $i \leq \ell$. The game terminates when there is a unique chip remaining on the board.

Note that one can think of the ℓ-PL problem as a constrained version of this Chip-Liar game. In the k-known setting[1], the chips are equivalent to k-partitions of the finite set V, so that the number N of chips is the Stirling Number of the second kind $N = \left\{ {n \atop k} \right\}$. Moreover, unlike the general Chip-Liar game, in our setup the questioner may only select specific subsets S: for any pair of elements $u, v \in V$, the questioner can select S to be the set of all partitions in which u and v are part of the same cluster.

This allows one to adopt the following perspective. One may think of the queries $\{u_t v_t\}_{t \in [T]}$ made by the questioner together with the signs given by the responses to the

1. Note that an analogous game can be defined for the k-unknown setting, where chips are equivalent to any partition of V, and N is the Bell number B_n.

queries as making up an instance of the correlation clustering problem. The position of a chip on the board will then be equal to the cost of the corresponding partition as a solution of the correlation clustering instance constructed. In the next two sections, we formalize this intuition by defining *Rényi-Ulam Correlation Clustering* (RUCC) games, which are the key tools we use to lower bound query complexity.

4.7 Lower Bounds for ℓ-bounded False Negatives

In this section we prove a lower bound on the number of questions needed to learn partitions when up to ℓ of the -1 answer could be incorrect, but all of the $+1$ answers must be correct. The key result of this section is that solving this problem requires at least $\max\{RS(n,k) + \Omega(\ell k^2)\}$. This will imply the same lower bound for more general versions of the problem.

We define the following game, which we call the RUCC$^{\texttt{FN}}$ game. The game is played by two players, the questioner (Q) and the responder (R) and it is parametrized by a finite set V of cardinality n, a positive integer $k \in \{2, ..., n-1\}$, and a non-negative integer ℓ. The goal of the questioner is to infer a k-partition of V given input from the responder. At the start of the game, we are given a complete undirected graph G_0 on vertex set V. $G_0 = (V, E, w_0^+, w_0^-)$ has two associated edge weight functions w_0^+ and w_0^- which are set to zero at the beginning of the game.

At each iteration t, the questioner Q submits a query $\{u, v\}$ for distinct u and v in V. The responder R then issues a response $r_t \in \{\pm 1\}$. If $r_t = +1$, then w^+ is updated by setting $w_t^+(uv) = w_{t-1}^+(uv) + 1$. Otherwise, if $r_t = -1$, then w^- is updated by setting $w_t^-(uv) = w_{t-1}^-(uv) + 1$. We then set $G_t = (V, E, w_t^+, w_t^-)$.

For any partition $\mathcal{C}$ of V, we define the cost:

$$\text{cost}_{G_t}^{\texttt{FN}}(\mathcal{C}) \stackrel{\text{def}}{=} \sum_{\substack{uv \in E \\ u \sim_{\mathcal{C}} v}} w_t^-(uv) + \mathbb{I}\left(\sum_{\substack{uv \in E \\ u \not\sim_{\mathcal{C}} v}} w_t^+(uv) = 0 \right)$$

for $\mathbb{I}$ the convex indicator function, i.e. for any predicate $\mathcal{S}$:

$$\mathbb{I}(\mathcal{S}) = \begin{cases} 0 & \text{if } \mathcal{S} \text{ holds,} \\ \infty & \text{otherwise.} \end{cases}$$

. Intuitively, the value of $\text{cost}^{\texttt{FN}}_{G_t}(\mathcal{C})$ is the number of incorrect negative (-1) answers that the responder would have to have given if $\mathcal{C}$ was the correct partition, or infinity if the responder would have had to give incorrect positive $(+1)$ answers.

We require that the responder must, at each iteration t, reply in a way that guarantees that there exists some k-partition $\mathcal{C}^*$ of V such that:

$$\text{cost}^{\texttt{FN}}_{G_t}(\mathcal{C}^*) \leq \ell.$$

The game terminates when $\text{cost}^{\texttt{FN}}_{G_t}(\mathcal{C})$ is strictly more than ℓ for all but one k-partitions $\mathcal{C}^*$ of V, indicating that the responder is left with a single feasible partition, and hence they have learned the ground truth.

Given a questioner strategy Q and a responder strategy R, we let $\text{Game}^{\texttt{FN}}(Q, R)$ be the number of iterations the game lasts when questioner Q plays against responder R. The query complexity of the problem of learning partitions with ℓ false negatives is the value:

$$\min_{Q \in \mathcal{Q}} \max_{R \in \mathcal{R}} \text{Game}^{\texttt{FN}}(Q, R),$$

where $\mathcal{Q}$ is the set of all possible questioner strategies and $\mathcal{R}$ is the set of all possible responder strategies. A simple argument then shows:

Lemma 63. *If there exists a responder strategy R^* such that:*

$$\min_{q \in \mathcal{Q}} \text{Game}^{\texttt{FN}}(Q, R^*) \geq f(n, k, \ell),$$

then the query complexity of the problem of learning a partition with ℓ false negatives is at least $f(n, k, \ell)$.

Proof. This follows from:

$$\min_{Q \in \mathcal{Q}} \max_{R \in \mathcal{R}} \text{Game}^{\texttt{FN}}(Q, R) \geq \min_{Q \in \mathcal{Q}} \text{Game}^{\texttt{FN}}(Q, R^*) \geq f(n, k, \ell).$$

□

We now show the following:

Theorem 64. *There exists a responder strategy R^* satisfying*

$$\min_{Q \in \mathcal{Q}} \text{Game}^{\texttt{FN}}(Q, R^*) \geq (\ell + 1)\left(\binom{k+1}{2} - 1\right) = \Omega(\ell k^2)$$

queries.

This immediately implies a lower bound on the query complexity:

Corollary 65. *Every algorithm for the problem of learning a partition with ℓ false negatives which guarantees full recovery of the correct partitions requires* $\max\{RS^k(n, k), \Omega(k^2\ell)\}$ *queries in the worst case.*

We consider a responder strategy which we call the $(k+1)$*-groups responder strategy*, denoted $R_{(k+1)}$. In this strategy, the responder fixes an arbitrary $(k+1)$-partition of V, denoted $\mathcal{C}^* = \{C_i^*\}_{i=1}^{k+1}$. We define $R_{(k+1)}$ as the responder strategy in which the responder replies consistently with $\mathcal{C}^*$ whenever the rules of the RUCC$^{\texttt{FN}}$ game allow. We lowerbound the number of iterations needed for any questioner strategy to terminate the RUCC$^{\texttt{FN}}$ game when playing against $R_{(k+1)}$:

Theorem 66. *The $(k+1)$-groups responder strategy $R_{(k+1)}$ satisfies*

$$\min_{Q \in \mathcal{Q}} \text{Game}^{\texttt{FN}}(Q, R_{(k+1)}) \geq (\ell + 1)\left(\binom{k+1}{2} - 1\right) = \Omega(\ell k^2)$$

queries.

Proof. For $i, j \in [k+1]$, consider the k-partitions $\mathcal{C}_{ij}$ obtained by starting with $\mathcal{C}^*$ and merging C_i^* and C_j^* i.e.:

$$\mathcal{C}_{ij} = \{C_a^*\}_{a \in [k+1] \setminus \{i,j\}} \cup \{C_i^* \cup C_j^*\}.$$

We show that the questioner cannot increase the cost of more than one of these partitions $\mathcal{C}_{ij}$ on any single iteration:

Claim 1. *Suppose that* $\text{cost}_{G_t}^{\texttt{FN}}(\mathcal{C}_{ij}) \leq \ell$. *Then on iteration* $t+1$ *the cost of* $\mathcal{C}_{ij}$ *increases only if the questionner submits a query* $\{u, v\}$ *for* $u \in C_i^*$ *and* $v \in C_j^*$.

We now prove Theorem 66 using Claim 1. Consider the potential function:

$$\phi^{(t)} := \sum_{\substack{i,j \in [k+1] \\ i \neq j}} \min\{\text{cost}_{G_t}^{\texttt{FN}}(\mathcal{C}_{ij}), \ell + 1\}. \tag{4.3}$$

Note that, $\phi^{(0)} = 0$ and, by Claim 1, at every step t of the game we have:

$$\phi^{(t+1)} \leq \phi^{(t)} + 1. \tag{4.4}$$

Suppose that the game terminates at iteration T. Then by the rule of the RUCC$^{\texttt{FN}}$ game, for all but one k-partitions $\mathcal{C}$, it must be the case that $\text{cost}_{G_T}^{\texttt{FN}}(\mathcal{C}) > \ell$. In particular, all but at most one of the partitions $\mathcal{C}_{ij}$ must have cost at least $\ell + 1$, implying:

$$\phi^{(T)} \geq (\ell + 1)\left(\binom{k+1}{2} - 1\right), \tag{4.5}$$

and hence:

$$T \geq (\ell + 1)\left(\binom{k+1}{2} - 1\right) = \Omega(\ell k^2),$$

as needed. □

We conclude this section by proving Claim 1.

Proof of Claim 1. Consider G_t such that $\text{cost}^{\texttt{FN}}_{G_t}(\mathcal{C}_{ij}) \leq \ell$ and such that the RUCC$^{\texttt{FN}}$ Game has not yet terminated. Assume the query submitted by the questioner is not of the form $\{u, v\}$ for any $u \in C^*_i$, $v \in C^*_j$. For any such $\{u, v\}$, by the definition of $\mathcal{C}_{ij}$ the pair u, v is in the same cluster with respect to partition $\mathcal{C}_{ij}$ if and only if the pair is in the same cluster with respect to partition $\mathcal{C}^*$. Thus if the responder is able to return an answer consistent with $\mathcal{C}^*$, this response will also be consistent with $\mathcal{C}_{ij}$, and as a result $\text{cost}^{\texttt{FN}}_{G_t}(\mathcal{C}_{ij}) = \text{cost}^{\texttt{FN}}_{G_{t+1}}(\mathcal{C}_{ij})$.

It remains to show that the responder is able to reply consistently with $\mathcal{C}^*$ on such a query. By assumption $\text{cost}^{\texttt{FN}}_{G_t}(\mathcal{C}_{ij}) \leq \ell$, and as shown above responding to the query $\{u, v\}$ in a manner consistent with $\mathcal{C}^*$ will ensure $\text{cost}^{\texttt{FN}}_{G_t}(\mathcal{C}_{ij}) = \text{cost}^{\texttt{FN}}_{G_{t+1}}(\mathcal{C}_{ij}) \leq \ell$. Thus if the responder replies to query $\{u, v\}$ consistently with $\mathcal{C}^*$, there will still exist a k-partition–namely, $\mathcal{C}_{ij}$–such that $\text{cost}^{\texttt{FN}}_{G_{t+1}}(\mathcal{C}_{ij}) \leq \ell$. Thus the rules of the RUCC$^{\texttt{FN}}$ game allow the responder to reply to query $\{u, v\}$ consistently with $\mathcal{C}^*$, and by definition of strategy $R_{(k+1)}$ the responder will do so. Thus the cost of $\mathcal{C}_{ij}$ at iteration $(t+1)$ does not increase when the questioner plays such a pair $\{u, v\}$. □

4.8 Lower Bounds for ℓ-bounded False Positives

In this section, we prove a lower bound on the number of questions needed to learn partitions when up to ℓ of the $+1$ answer could be incorrect, but all of the -1 answers must be correct. The key result from this section is that solving this problem requires $\max\{RS(n, k), \Omega(n\ell)\}$ queries. This will imply the same lower bound for more general versions of the problem.

The RUCC$^{\texttt{FP}}$ game is defined with the same setup as the RUCC$^{\texttt{FN}}$ game.

The questioner Q, responder R, finite set V, and gameplay graph $G_t = (V, E, w^+_t, w^-_t)$ are all defined as in Section 4.7. The core distinction between the RUCC$^{\texttt{FP}}$ game and the

RUCC$^{\texttt{FN}}$ game is the notion of cost employed. For the RUCC$^{\texttt{FP}}$ game, for any partition $\mathcal{C}$ of V, we define the cost

$$\text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}) \stackrel{\text{def}}{=} \sum_{\substack{uv \in E \\ u \not\sim_{\mathcal{C}} v}} w_t^+(uv) + \mathbb{I}\left(\sum_{\substack{uv \in E \\ u \sim_{\mathcal{C}} v}} w_t^-(uv) \right).$$

Intuitively, the value of $\text{cost}^{\texttt{FP}}(\mathcal{C})$ is either the number of incorrect positive answers $(+1)$ that the responder would have made if $\mathcal{C}$ were the correct partition, or the value is infinity if the responder would have given incorrect negative (-1) answers.

The rules of the game require that at each iteration t, the responder must reply in such a way that guarantees the existence of some k-partition $\mathcal{C}^*$ of V such that

$$\text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}^*) \leq \ell.$$

The game terminates when there exists a unique k-partition $\mathcal{C}^*$ such that $\text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}^*) \leq \ell$.

The query complexity of the problem of learning partitions with ℓ false positives can be characterized as

$$\min_{Q \in \mathcal{Q}} \max_{R \in \mathcal{R}} \text{Game}^{\texttt{FP}}(Q, R),$$

where $\text{Game}^{\texttt{FP}}(Q, R)$ indicates the number of iterations in the RUCC$^{\texttt{FP}}$ game when questioner strategy Q plays against responder R. As in Section 4.7, this characterization allows us to lower bound the query complexity of the problem by analyzing the RUCC$^{\texttt{FP}}$ game:

Lemma 67. *If there exists a responder strategy R^* such that*

$$\min_{Q \in \mathcal{Q}} \text{Game}^{\texttt{FP}}(Q, R^*) \geq f(n, k, \ell),$$

then the query complexity of learning partitions with ℓ false positives is at least $f(n, k, \ell)$.

Theorem 68. *There exists a responder strategy R^* which satisfies*

$$\min_{Q \in \mathcal{Q}} \text{Game}^{\texttt{FP}}(Q, R^*) \geq \frac{(\ell+1)(n-k+1))}{2} = \Omega(\ell(n-k)).$$

Corollary 69. *Every algorithm for the problem of learning a partition with ℓ false positives which guarantees exact recovery of the correct partition requires $RS^k(n,k) + \Omega(\ell(n-k))$ queries in the worst case.*

We consider a responder strategy which we call the $(k-1)$-*groups responder strategy*, denoted $R_{(k-1)}$. This strategy fixes $k-2$ arbitrary elements $v_1^*, ..., v_{k-2}^*$ and then fixes the following $(k-1)$-partition:

$$\mathcal{C}^* := \{V \setminus \{v_1^*, ..., v_{k-2}^*\}, \{v_1^*\}, ..., \{v_{k-2}^*\}\}.$$

Note that when $k = 2$, the set $\{v_i^*\}$ is just empty and the partition $\mathcal{C}^*$ consists of a single set equal to V. We define the $(k-1)$-groups strategy as the responder strategy in which the responder replies consistently with $\mathcal{C}^*$ whenever the rules of the RUCC$^{\texttt{FP}}$ game allow them to.

Theorem 70. *The $(k-1)$-groups responder strategy $R_{(k-1)}$ satisfies:*

$$\min_{Q \in \mathcal{Q}} \text{Game}^{\texttt{FP}}(Q, R_{(k-1)}) \geq \frac{(\ell+1)(n-k+1))}{2} = \Omega(\ell(n-k)).$$

Proof. We will prove the theorem by considering a subset of the partitions which are hard for the questioner to differentiate between when playing against this responder strategy. Let $S^* := V \setminus \{v_1^*, ..., v_{k-2}^*\}$, and for any $v \in S^*$ let $\mathcal{C}_v$ be the partition:

$$\mathcal{C}_v := \{\{v\}, S^* \setminus \{v\}, \{v_1^*\}, ..., \{v_{k-2}^*\}\}.$$

By definition of the RUCC$^{\mathtt{FP}}$ game, at the end of the final iteration T, we must have $\mathrm{cost}^{\mathtt{FP}}_{G_T}(\mathcal{C}_v) > \ell$ for all but one of the partitions $\{\mathcal{C}_v\}_{v \in S^*}$. In particular, consider the potential function:

$$\phi^{(t)} \stackrel{\text{def}}{=} \sum_{v \in S^*} \min\{\mathrm{cost}^{\mathtt{FP}}_{G_t}(C_v), \ell + 1\}.$$

Then, if the game ends at iteration T, we must have:

$$\phi^{(T)} \geq (\ell + 1)(|S^*| - 1) = (n - k + 1)(\ell + 1). \tag{4.6}$$

On the other hand, we have:

$$\phi^{(0)} = 0. \tag{4.7}$$

We show that this potential $\phi^{(t)}$ changes by a small amount on any given iteration of the RUCC$^{\mathtt{FP}}$ game:

Claim 2. *If $\phi^{(t)} < (n - k + 1)(\ell + 1)$, then $\phi^{(t+1)} \leq \phi^{(t)} + 2$.*

An immediate consequence of Claim 2 and Equations (4.6) and (4.7) is that the number of iterations required for the game to terminate is at least $T \geq (n - k + 1)(\ell + 1)/2$, yielding the statement of Theorem 70. □

We conclude by proving Claim 2.

Proof of Claim 2. Let $\{u, v\}$ be the query submitted by the questioner on iteration $t + 1$. We will split the proof of the theorem into cases.

Case 1 Suppose $\{u, v\} \not\subseteq S^*$. Then, if the responder answers in a way that's consistent with $\mathcal{C}^*$, their answer will also be consistent with $\mathcal{C}_w$ for every $w \in S^*$. By consequence of the assumption that $\phi^{(t)} < (n - k + 1)(\ell + 1)$, a simple counting argument implies that there exists an element $w_t \in S^*$ such that:

$$\mathrm{cost}^{\mathtt{FP}}_{G_t}(\mathcal{C}_{w_t}) \leq \ell.$$

Since w_t as defined above satisfies $\text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}_{w_t}) \leq \ell$, this implies that the responder can answer in a way consistent with $\mathcal{C}^*$ without violating the rules of the game. By the definition of the strategy, the responder then will answer in a way consistent with $\mathcal{C}^*$ (and with all partitions $\mathcal{C}_w$) and hence, for all $w \in S^*$: $\text{cost}^{\texttt{FP}}_{G_{t+1}}(\mathcal{C}_w) = \text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}_w)$, giving $\phi^{(t+1)} = \phi^{(t)}$.

Case 2.a Suppose that $\{u, v\} \subseteq S^*$, and that the responder replies in a way that is consistent with $\mathcal{C}^*$. In this scenario, for any $w \in S^* \setminus \{u, v\}$ we have $\text{cost}^{\texttt{FP}}_{G_{t+1}}(\mathcal{C}_w) = \text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}_w)$, while $\text{cost}^{\texttt{FP}}_{G_{t+1}}(\mathcal{C}_u) \leq \text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}_u) + 1$ and $\text{cost}^{\texttt{FP}}_{G_{t+1}}(\mathcal{C}_v) \leq \text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}_v) + 1$. This immediately implies $\phi^{(t+1)} \leq \phi^{(t)} + 2$.

Case 2.b Suppose that $\{u, v\} \subseteq S^*$, and that the responder replies in a way that is **not** consistent with $\mathcal{C}^*$, i.e the responder replies to the query $\{u, v\}$ with -1. Then by the definition of the responder strategy, it must be the case that answering $+1$ to the query would violate the rules of the game, i.e. it would cause every k-partition $\mathcal{C}$ of V to have $\text{cost}^{\texttt{FP}}_{G_{t+1}}(\mathcal{C}) \geq \ell + 1$. But, for every $w \in S^* \setminus \{u, v\}$, the cost of the partition $\mathcal{C}_w$ does not increase if the response to the query $\{u, v\}$ is $+1$. Hence, it must be the case that $\text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}_w) \geq \ell + 1$ for all such w. Hence, for any $w \in S^* \setminus \{u, v\}$:

$$\min\{\ell + 1, \text{cost}^{\texttt{FP}}_{G_{t+1}}(\mathcal{C}_w)\} = \ell + 1 = \min\{\ell + 1, \text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}_w)\}. \tag{4.8}$$

On the other hand, since the responder is replying to the query $\{u, v\}$ with -1, their answer is simultaneously consistent with both $\mathcal{C}_u$ and $\mathcal{C}_v$. We then have:

$$\text{cost}^{\texttt{FP}}_{G_{t+1}}(\mathcal{C}_u) = \text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}_u) \text{ and } \text{cost}^{\texttt{FP}}_{G_{t+1}}(\mathcal{C}_v) = \text{cost}^{\texttt{FP}}_{G_t}(\mathcal{C}_v). \tag{4.9}$$

In particular, Equations (4.8) and (4.9) directly imply $\phi^{(t+1)} = \phi^{(t)}$.

This concludes the proof of Claim 2 (and in turn the proof of Theorem 70). □

4.9 Conclusions and Summary of Results for Learning Partition with Same-Cluster Oracles

In this chapter, we have discussed several results on the problem of learning partitions with a same-cluster oracle. These suffice to characterize the asymptotic query complexity of most problems discussed in Section 4.1 both for the k-known and for the k-unknown setting. We summarize our results in Table 4.1.

In the k-known setting, we are able to resolve the complexity of all problems in Section 4.1. The lower bound of $RS^k(n,k)$ for the error-free variant, implies the same lower bound for all problems in the class. Similarly, the lower bound of $\Omega(\ell(n-k))$ proved in Section 4.8 for the ℓ-PL$^{\texttt{FP}}$ problem immediately yields the same lower bound for the unweighted ℓ-PL problem and a lower bound of $\Omega(\ell_{\texttt{yes}}(n-k))$ for the weighted ℓ-PL problem. Finally, the lower bound of $\Omega(\ell k^2)$ proved in Section 4.7 for the ℓ-PL$^{\texttt{FN}}$ problem implies the same lower bound for the unweighted ℓ-PL problem, and a lower bound of $\Omega(\ell_{\texttt{no}}k^2)$ for the weighted ℓ-PL problem. On the upper bound side, the algorithm for the weighted ℓ-PL problem given in Section 4.4 which has a query complexity of $O(RS^k(n,k) + (n-k)\ell_{\texttt{yes}} + k^2\ell_{\texttt{no}})$ immediately yields: (1) an algorithm for the unweighted ℓ-PL problem with query complexity $O(RS^k(n,k)+(n-k)\ell+k^2\ell)$, (2) an algorithm with query complexity $O(RS^k(n,k)+(n-k)\ell)$ for the ℓ-PL$^{\texttt{FP}}$ problem, and (3) an algorithm with query complexity $O(RS^k(n,k) + k^2\ell)$ for the ℓ-PL$^{\texttt{FN}}$ problem. Hence all the upper bounds match their respective lower bounds modulo constant factors.

The situation is slightly different for the k-unknown setting. All the lower bounds proved for the k-known setting directly apply in the k-unknown setting as well. On the other hand, the algorithm given in Section 4.5 for the weighted ℓ-PL problem, with query complexity:

$$O\left(RS^u(n,k) + (n-k)\max\{\ell_{\texttt{yes}}, \ell_{\texttt{no}}\} + k^2\ell_{\texttt{no}}\right)$$

directly yields: (1) an algorithm for the unweighted ℓ-PL problem with query complexity:

$$O\left(RS^u(n,k) + (n-k)\ell + k^2\ell\right),$$

(2) an algorithm with query complexity $O(RS^u(n,k) + (n-k)\ell)$ for the ℓ-PL$^{\texttt{FP}}$ problem, and (3) an algorithm with query complexity:

$$O\left(RS^u(n,k) + (n-k)\ell + k^2\ell\right)$$

for the ℓ-PL$^{\texttt{FN}}$ problem. Note that this leaves an additive gap of $O(\ell_{\texttt{no}}(n-k))$ queries between the upper and lower bounds for the weighted ℓ-PL problem and the ℓ-PL$^{\texttt{FN}}$ problem in the k-unknown setting. We leave the question of resolving this gap as an open problem.

4.10 Bibliographical Notes

Much of the work in this chapter is due to DePavia et al. (2023). Rényi-Ulam liar games were inspired by a question first asked by Ulam (1991) in his autobiography. The chip-liar game was first introduced by Spencer (1992). Since then, a lot of work has been done in fault-tolerant search. We refer the reader to the book by Cicalese (2013) for a comprehensive introduction to the subject.

.

	Query Complexity	
	k-known	k-unknown
Weighted ℓ-PL	$\Theta\big(RS^k(n,k) + (n-k)\ell_{\texttt{YES}} + k^2\ell_{\texttt{NO}}\big)$	$O\big(RS^u(n,k) + (n-k)\max\{\ell_{\texttt{YES}}, \ell_{\texttt{NO}}\} + k^2\ell_{\texttt{NO}}\big)$
ℓ-PL	$\Theta\left(RS^k(n,k) + (n-k)\ell + k^2\ell\right)$	$\Theta\left(RS^u(n,k) + (n-k)\ell + k^2\ell\right)$
ℓ-PL$^{\texttt{FN}}$	$\Theta\left(RS^k(n,k) + k^2\ell\right)$	$\Theta\left(RS^u(n,k) + (n-k)\ell + k^2\ell\right)$
ℓ-PL$^{\texttt{FP}}$	$\Theta\left(RS^k(n,k) + (n-k)\ell\right)$	$\Theta\left(RS^u(n,k) + (n-k)\ell\right)$
Error-Free	$RS^k(n,k)$	$RS^u(n,k)$

Table 4.1: The query complexity of the different variants of the ℓ-PL problem, for both the k-known and the k-unknown settings. We obtain matching upper and lower bounds for all the variants, with the exception of the weighted ℓ-PL problem in k-unknown setting, for which the upper bound does not match the lower bound (given for the k-unknown case) exactly.

REFERENCES

Amit Agarwal, Moses Charikar, Konstantin Makarychev, and Yury Makarychev. $\mathcal{O}(\sqrt{\log n})$-approximation algorithms for Min UnCut, Min 2CNF Deletion, and directed cut problems. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 573–581, 2005.

Ravindra K Ahujia, Thomas L Magnanti, and James B Orlin. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, 1993.

Noga Alon. Eigenvalues and expanders. *Combinatorica*, 6(2):83–96, 1986.

Noga Alon and Vitali D Milman. $\lambda 1$, isoperimetric inequalities for graphs, and superconcentrators. *Journal of Combinatorial Theory, Series B*, 38(1):73–88, 1985.

Noga Alon and Joel H Spencer. *The probabilistic method*. John Wiley & Sons, 2016.

Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):1–37, 2009.

Francis Bach. Learning with submodular functions: A convex optimization perspective. *Foundations and Trends® in machine learning*, 6(2-3):145–373, 2013.

DJ Balding, WJ Bruno, DC Torney, and E Knill. A comparative survey of non-adaptive pooling designs. In *Genetic mapping and DNA sequencing*, pages 133–154. Springer, 1996.

Aaron Bernstein, Maximilian Probst Gutenberg, and Thatchaphol Saranurak. Deterministic decremental sssp and approximate min-cost flow in almost-linear time. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1000–1008. IEEE, 2022.

Sergey Bobkov, Christian Houdré, and Prasad Tetali. λ_∞, vertex isoperimetry and concentration. *COMBINATORICA-BUDAPEST-*, 20(2):153–172, 2000.

Mathilde Bouvel, Vladimir Grebinski, and Gregory Kucherov. Combinatorial search on graphs motivated by bioinformatics applications: A brief survey. In *Graph-Theoretic Concepts in Computer Science: 31st International Workshop, WG 2005, Metz, France, June 23-25, 2005, Revised Selected Papers 31*, pages 16–27. Springer, 2005.

Stephen Boyd, Persi Diaconis, Jun Sun, and Lin Xiao. Fastest mixing markov chain on a path. *The American Mathematical Monthly*, 113(1):70–74, 2006.

Thang Nguyen Bui and Curt Jones. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42(3):153–159, 1992.

Moses Charikar, Konstantin Makarychev, and Yury Makarychev. Directed metrics and directed graph partitioning problems. In *SODA*, volume 6, pages 51–60, 2006.

Antares Chen, Lorenzo Orecchia, and Erasmo Tani. Submodular hypergraph partitioning: Metric relaxations and fast algorithms via an improved cut-matching game. *arXiv preprint arXiv:2301.08920*, 2023a.

Li Chen, Rasmus Kyng, Yang P Liu, Richard Peng, Maximilian Probst Gutenberg, and Sushant Sachdeva. Maximum flow and minimum-cost flow in almost-linear time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 612–623. IEEE, 2022.

Yi Chen, Ramya Korlakai Vinayak, and Babak Hassibi. Crowdsourced clustering via active querying: Practical algorithm with theoretical guarantees. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 11, pages 27–37, 2023b.

Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

Julia Chuzhoy. A distanced matching game, decremental apsp in expanders, and faster deterministic algorithms for graph cut problems. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2122–2213. SIAM, 2023.

Julia Chuzhoy and Sanjeev Khanna. A new algorithm for decremental single-source shortest paths with applications to vertex-capacitated flow and cut problems. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, pages 389–400, 2019.

Julia Chuzhoy, Yu Gao, Jason Li, Danupon Nanongkai, Richard Peng, and Thatchaphol Saranurak. A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1158–1167. IEEE, 2020.

Ferdinando Cicalese. *Fault-Tolerant Search Algorithms*. Springer, 2013.

Yves Colin de Verdiere. Sur un nouvel invariant des graphes et un critere de planarité. *Journal of Combinatorial Theory, Series B*, 50(1):11–21, 1990.

Adela Frances DePavia, Olga Medrano Martín del Campo, and Erasmo Tani. Error-tolerant exact query learning of finite set partitions with same-cluster oracle. *arXiv preprint arXiv:2305.13402*, 2023.

Uriel Feige, Mohammad Taghi Hajiaghayi, and James R Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2): 629–657, 2008.

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

Zoran Kadelburg, Dusan Dukic, Milivoje Lukic, and Ivan Matic. Inequalities of karamata, schur and muirhead, and some applications. *The Teaching of Mathematics*, 8(1):31–45, 2005.

Rohit Khandekar, Satish Rao, and Umesh Vazirani. Graph partitioning using single commodity flows. *Journal of the ACM (JACM)*, 56(4):1–15, 2009.

Tsz Chiu Kwok, Lap Chi Lau, and Kam Chuen Tung. Cheeger inequalities for vertex expansion and reweighted eigenvalues. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 366–377. IEEE, 2022.

Tom Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM (JACM)*, 46(6):787–832, 1999.

Pan Li and Olgica Milenkovic. Submodular hypergraphs: p-laplacians, cheeger inequalities and spectral clustering. In *International Conference on Machine Learning*, pages 3014–3023. PMLR, 2018.

Xizhi Liu and Sayan Mukherjee. Tight query complexity bounds for learning graph partitions. In *Conference on Learning Theory*, pages 167–181. PMLR, 2022.

Anand Louis. Cut-matching games on directed graphs. *arXiv preprint arXiv:1010.1047*, 2010.

Anand Louis and Yury Makarychev. Approximation algorithms for hypergraph small-set expansion and small-set vertex expansion. *Theory of Computing*, 12(1):1–25, 2016.

Anand Louis, Prasad Raghavendra, and Santosh Vempala. The complexity of approximating vertex expansion. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 360–369. IEEE, 2013.

Arya Mazumdar and Barna Saha. Clustering with noisy queries. *Advances in Neural Information Processing Systems*, 30, 2017.

Danupon Nanongkai and Thatchaphol Saranurak. Dynamic spanning forest with worst-case update time: adaptive, las vegas, and o (n1/2-ε)-time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1122–1129, 2017.

Sam Olesker-Taylor and Luca Zanetti. Geometric bounds on the fastest mixing markov chain. *Probability Theory and Related Fields*, pages 1–46, 2024.

Lorenzo Orecchia. *Fast approximation algorithms for graph partitioning using spectral and semidefinite-programming techniques*. University of California, Berkeley, 2011.

Lorenzo Orecchia and Nisheeth K Vishnoi. Towards an sdp-based approach to spectral methods: A nearly-linear-time algorithm for graph partitioning and decomposition. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 532–545. SIAM, 2011.

Lorenzo Orecchia, Leonard J Schulman, Umesh V Vazirani, and Nisheeth K Vishnoi. On partitioning graphs via single commodity flows. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 461–470, 2008.

Lev Reyzin and Nikhil Srivastava. Learning and verifying graphs using queries with a focus on edge counting. In *Algorithmic Learning Theory: 18th International Conference, ALT 2007, Sendai, Japan, October 1-4, 2007. Proceedings 18*, pages 285–297. Springer, 2007.

Xu Shaoji. The size of uniquely colorable graphs. *Journal of combinatorial theory. Series B*, 50(2):319–320, 1990.

Jonah Sherman. Breaking the multicommodity flow barrier for $\mathcal{O}(\sqrt{\log n})$-approximations to sparsest cut. In *2009 50th Annual IEEE Symposium on Foundations of Computer Science*, pages 363–372. IEEE, 2009.

Jiří Šíma and Satu Elisa Schaeffer. On the np-completeness of some graph cluster measures. In *International Conference on Current Trends in Theory and Practice of Computer Science*, pages 530–537. Springer, 2006.

Alexei Sorokin, Alla Lapidus, Veronique Capuano, Nathalie Galleron, Petar Pujic, and S Dusko Ehrlich. A new approach using multiplex long accurate pcr and yeast artificial chromosomes for bacterial chromosome mapping and sequencing. *Genome research*, 6 (5):448–453, 1996.

Joel Spencer. Ulam's searching game with a fixed number of lies. *Theoretical Computer Science*, 95(2):307–321, 1992.

J Michael Steele. *The Cauchy-Schwarz master class: an introduction to the art of mathematical inequalities*. Cambridge University Press, 2004.

Stanislaw M Ulam. *Adventures of a Mathematician*. Univ of California Press, 1991.

Hein van der Holst. A short proof of the planarity characterization of colin de verdiere. *Journal of Combinatorial Theory, Series B*, 65(2):269–272, 1995.

Nate Veldt. Cut-matching games for generalized hypergraph ratio cuts. In *Proceedings of the ACM Web Conference 2023*, pages 694–704, 2023.

Nate Veldt, Austin R Benson, and Jon Kleinberg. Hypergraph cuts with general splitting functions. *SIAM Review*, 64(3):650–685, 2022.

Ramya Korlakai Vinayak and Babak Hassibi. Crowdsourced clustering: Querying edges vs triangles. *Advances in Neural Information Processing Systems*, 29, 2016.